Matricized tensor times Khatri-Rao product computation

Suraj Kumar

Inria & ENS Lyon

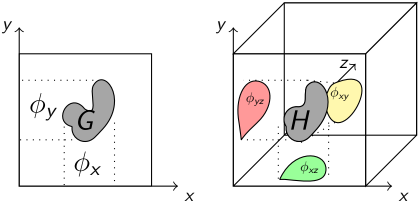
Email:*suraj.kumar@inria.fr*

CR12: October 2023

https://surakuma.github.io/courses/daamtc.html

Loomis-Whitney inequality

- Relates volume of a d-dimensional object with its all $d-1$ dimensional projections
  - For the 2d object G, $Area(G) \leq \phi_x \phi_y$
  - For the 3d object H, $Volume(H) \leq \sqrt{\phi_{xy}\phi_{yz}\phi_{xz}}$



- Similarly, for a 4d object I, $Volume(I) \leq \phi_{xyz}^{\frac{1}{3}} \phi_{xyw}^{\frac{1}{3}} \phi_{xzw}^{\frac{1}{3}} \phi_{yzw}^{\frac{1}{3}}$

- How to work with arbitrary dimensional projections?

Hölder-Brascamp-Lieb (HBL) inequality

- Generalize Loomis-Whitney inequality for arbitrary dimensional projections
- Provide exponent for each projection

Lemma

Consider any positive integers ℓ and m and any m projections $\phi_j : \mathbb{Z}^\ell \to \mathbb{Z}^{\ell_j}$ ($\ell_j \leq \ell$), each of which extracts ℓ_j coordinates $S_j \subseteq [\ell]$ and forgets the $\ell - \ell_j$ others. Define $\mathcal{C} = \left\{ \mathsf{s} \in [0,1]^m : \Delta \cdot \mathsf{s} \geq \mathsf{1} \right\}$, where the $\ell \times m$ matrix Δ has entries $\Delta_{i,j} = 1$ if $i \in S_j$ and $\Delta_{i,j} = 0$ otherwise. If $[s_1 \ \cdots \ s_m]^\mathsf{T} \in \mathcal{C}$, then for all $F \subseteq \mathbb{Z}^\ell$,

$$|F| \leq \prod_{j \in [m]} |\phi_j(F)|^{s_j}.$$

- For tighter bound, we usually work with $\Delta \cdot \mathsf{s} = \mathsf{1}$
  - Possible that $\Delta \cdot \mathsf{s} = \mathsf{1}$ does not have solution, then consider s such that $\Delta \cdot \mathsf{s}$ is not very far from $\mathsf{1}$

Notation: $\mathsf{1}$ represents a vector of all ones. $[m]$ denotes $\{1, 2, \cdots, m\}$ throughout the slides.

HBL inequality

Lemma

Consider any positive integers ℓ and m and any m projections $\phi_j : \mathbb{Z}^\ell \to \mathbb{Z}^{\ell_j}$ ($\ell_j \leq \ell$), each of which extracts ℓ_j coordinates $S_j \subseteq [\ell]$ and forgets the $\ell - \ell_j$ others. Define $\mathcal{C} = \left\{ \mathrm{s} \in [0,1]^m : \Delta \cdot \mathrm{s} \geq 1 \right\}$, where the $\ell \times m$ matrix Δ has entries $\Delta_{i,j} = 1$ if $i \in S_j$ and $\Delta_{i,j} = 0$ otherwise. If $[s_1 \ \cdots \ s_m]^\mathsf{T} \in \mathcal{C}$, then for all $F \subseteq \mathbb{Z}^\ell$,

$$|F| \leq \prod_{j \in [m]} |\phi_j(F)|^{s_j}.$$

Matrix multiplication ($C = AB$) example

Here $A \in \mathbb{R}^{n_1 \times n_2}$, $B \in \mathbb{R}^{n_2 \times n_3}$, and $C \in \mathbb{R}^{n_1 \times n_3}$.

for $i = 1{:}n_1$, for $k = 1{:}n_2$, for $j = 1{:}n_3$

$C[i][j] += A[i][k] * B[k][j]$

$$\Delta = \begin{array}{c} \\ i \\ j \\ k \end{array} \begin{array}{c} \begin{array}{ccc} A & B & C \end{array} \\ \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \end{pmatrix} \end{array}$$

- Find $\mathrm{s} = \begin{bmatrix} s_1 & s_2 & s_3 \end{bmatrix}^\mathsf{T}$ such that $\Delta \cdot \mathrm{s} = 1$
- ϕ_A, ϕ_B, ϕ_C: projections of computations on arrays A, B, C
- HBL inequality: amount of computations $\leq |\phi_A|^{s_1} |\phi_B|^{s_2} |\phi_C|^{s_3}$

HBL inequality

It can be used to obtain sequential or parallel communication lower bound.

Sequential lower bound formulation for matrix multiplication:

- Determine maximum amount of computations under segment size constraint: *Maximize* $|\phi_A|^{s_1}|\phi_B|^{s_2}|\phi_C|^{s_3}$ s.t. $|\phi_A| + |\phi_B| + |\phi_C| <= Constt$
- Calculate total data transfers for all the segments

Parallel lower bound formulation for matrix multiplication:

- Determine the sum of array accesses to perform the required computations
  - *Minimize* $|\phi_A| + |\phi_B| + |\phi_C|$ s.t. amount of computations $\leq |\phi_A|^{s_1}|\phi_B|^{s_2}|\phi_C|^{s_3}$

Optimization problems [Ballard et al., IPDPS 2017]

Lemma

Given $s_i > 0$, the optimization problem

$$\max_{x_i \geq 0} \prod_{i \in [m]} x_i^{s_i} \text{ subject to } \sum_{i \in [m]} x_i \leq c$$

yields the maximum value

$$c^{\sum_i s_i} \prod_{j \in [m]} \left(\frac{s_j}{\sum_i s_i} \right)^{s_j}.$$

Lemma

Given $s_i > 0$, the optimization problem

$$\min_{x_i \geq 0} \sum_{i \in [m]} x_i \text{ subject to } \prod_{i \in [m]} x_i^{s_i} \geq c$$

yields the maximum value

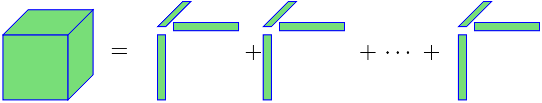
$$\left(\frac{c}{\prod_i {s_i}^{s_i}} \right)^{\frac{1}{\sum_i s_i}} \sum_{j \in [m]} s_j.$$

Both lemmas can be proved with the Lagrange multipliers.

Table of Contents

CP decomposition of $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$

It factorizes a tensor into a sum of rank one tensors.



CP decomposition of a 3-dimensional tensor.

$$\mathcal{A} = \sum_{\alpha=1}^{r} U_1(:,\alpha) \circ U_2(:,\alpha) \circ \cdots \circ U_d(:,\alpha)$$

It can be concisely expressed as $\mathcal{A} = [\![U_1, U_2, \cdots, U_d]\!]$. CP decomposition for a 3-dimensional tensor in matricized form can be written as:

$$A_{(1)} = U_1(U_3 \odot U_2)^T,\ A_{(2)} = U_2(U_3 \odot U_1)^T\ A_{(3)} = U_3(U_2 \odot U_1)^T.$$

It is useful to assume that $U_1, U_2 \cdots U_d$ are normalized to length one with the weights given in a vector $\lambda \in \mathbb{R}^r$.

CP-ALS algorithm for a 3-dimensional tensor $\mathcal{A}$

Repeat until maximum iterations reached or no further improvement obtained

1. Solve $U_1(U_3 \odot U_2)^T = A_{(1)}$ for $U_1 \Rightarrow U_1 = A_{(1)}(U_3 \odot U_2)(U_3^T U_3 * U_2^T U_2)^\dagger$
2. Normalize columns of U_1
3. Solve $U_2(U_3 \odot U_1)^T = A_{(2)}$ for $U_2 \Rightarrow U_2 = A_{(2)}(U_3 \odot U_1)(U_3^T U_3 * U_1^T U_1)^\dagger$
4. Normalize columns of U_2
5. Solve $U_3(U_2 \odot U_1)^T = A_{(3)}$ for $U_3 \Rightarrow U_3 = A_{(3)}(U_2 \odot U_1)(U_2^T U_2 * U_1^T U_1)^\dagger$
6. Normalize columns of U_3

Here $A^\dagger$ denotes the Moore–Penrose pseudoinverse of A. We use the following identity to get expressions for U_1, U_2 and U_3:

$$(A \odot B)^T (A \odot B) = A^T A * B^T B$$

ALS for computing a CP decomposition

Algorithm 1 CP-ALS method to compute CP decomposition

Require: input tensor $\mathcal{A} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$, desired rank k, initial factor matrices $U_j \in \mathbb{R}^{n_j \times k}$ for $1 \leq j \leq d$

Ensure: $[\![\lambda; U_1, \cdots, U_d]\!]$: a rank-k CP decomposition of $\mathcal{A}$

repeat

  for $i = 1$ to d **do**

    $V \leftarrow U_1^\mathsf{T} U_1 * \cdots * U_{i-1}^\mathsf{T} U_{i-1} U_{i+1}^\mathsf{T} U_{i+1} * \cdots * U_d^\mathsf{T} U_d$

    $U_i \leftarrow A_{(i)}(U_d \odot \cdots \odot U_{i+1} \odot U_{i-1} \odot U_1)$

    $U_i \leftarrow U_i V^\dagger$

    $\lambda \leftarrow$ normalize colums of U_i

  end for

until converge or the maximum number of iterations

- The collective operation $A_{(i)}(U_d \odot \cdots \odot U_{i+1} \odot U_{i-1} \odot U_1)$ is known as Matricized tensor times Khatri-Rao product (MTTKRP) computation

Gradient based CP decomposition

$$F = \min_{U_1, U_2 U_3} ||\mathcal{A} - [\![U_1, U_2, U_3]\!]||_F^2$$

Gradients:

$$\mathcal{G} = 2(\mathcal{A} - [\![U_1, U_2, U_3]\!])$$

$$\frac{\partial F}{\partial U_1} = -G_{(1)}(U_3 \odot U_2)$$

$$\frac{\partial F}{\partial U_2} = -G_{(2)}(U_3 \odot U_1)$$

$$\frac{\partial F}{\partial U_3} = -G_{(3)}(U_2 \odot U_1)$$

Update U_1, U_2 and U_3 based on gradients until convergence or for the fixed number of iterations

Gradient based algorithm also employs MTTKRP computations.

Table of Contents

MTTKRP

We want to find R-rank CP decomposition of $\mathcal{A} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$. The corresponding MTTKRP operation is

$$U_i \leftarrow A_{(i)}(U_d \odot \cdots \odot U_{i+1} \odot U_{i-1} \odot U_1).$$

Two approaches to compute this operation:

- Conventional approach
  - Compute Khatri-Rao products in a temporary T
  - Multiply $A_{(i)}$ with the temporary T, $U_i = A_{(i)}T$
  - Total arithmetic cost $= \mathcal{O}(NR)$
- All-at-once approach

$$U_i(j_i, r) = \sum_{j_1,\cdots,j_{i-1},j_{i+1},\cdots j_d} \mathcal{A}(j_1,\cdots j_d) \prod_{k \in [d]-\{i\}} U_k(j_k, r)$$

  - Total arithmetic cost $= \mathcal{O}(dNR)$
  - No intermediate is formed (may limit the partial reuse)
  - Very useful to work with sparse tensor

$n_1 n_2 \cdots n_d$ is denoted by N through out the slides. We will mainly focus on all-at-once approach. This approach reduces communication.

MTTKRP all-at-once pseudo code

For $\{j_1 = 1 \text{ to } n_1\}$

$\ddots$

For $\{j_d = 1 \text{ to } n_d\}$

For $\{r = 1 \text{ to } R\}$

$$U_i(j_i, r) += \mathcal{A}(j_1, \cdots j_d) \cdot \prod_{k \in [d]-\{i\}} U_k(j_k, r)$$

Total number of loop iterations $= NR$

We assume that the innermost computation is performed atomically. This is required for the communication lower bounds.

- *Sequential case* : all the inputs are present in the memory when the single output value is updated
- *Parallel case*: all the multiplications of this computation are performed on only one processor

Table of Contents

Δ matrix for MTTKRP

$$
\begin{array}{cc}
& \begin{array}{cccccc} \mathcal{A} & U_1 & \cdots & U_i & \cdots & U_d \end{array} \\
\Delta = \begin{array}{c} j_1 \\ \vdots \\ j_i \\ \vdots \\ j_d \\ r \end{array} &
\begin{pmatrix}
1 & 1 & & & & \\
\vdots & & \ddots & & & \\
1 & & & 1 & & \\
\vdots & & & & \ddots & \\
1 & & & & & 1 \\
 & 1 & \cdots & 1 & \cdots & 1
\end{pmatrix}
\end{array}
$$

- To obtain tight lower bound, find $\mathrm{s} = [s_1, \cdots, s_d]^\mathsf{T}$ such that $\Delta \cdot \mathrm{s} = 1$

$$\mathrm{s}^\mathsf{T} = \left[1 - \frac{1}{d}, \frac{1}{d}, \cdots, \frac{1}{d}\right]$$

Analysis of a segment

We consider a segment of M loads and stores. Any algorithm in the segment can access at most $3M$ elements.

- *Output*: at most M elements can be live after each segment & $M - L$ elements written to the slow memory
- *Inputs*: at most M elements are available at the start of the segment & L elements loaded to the fast memory

Let F be the subset of iteration space evaluated during the segment. $\phi_i(F)$ denotes the projection of F on the i-th variable.

Optimization problem:

$$\text{Maximize } |F| \text{ subject to}$$

$$|F| \leq \prod_{i \in [d+1]} |\phi_i(F)|^{s_i}$$

$$\sum_{i \in [d+1]} |\phi_i(F)| \leq 3M$$

Communication lower bound

After solving the optimization problem, we get

$$|F| \leq \frac{1}{d}\left(\frac{1}{2-1/d}\right)^{2-1/d}(1-1/d)^{1-1/d}(3M)^{2-1/d} \leq \frac{1}{d}(3M)^{2-1/d}.$$

Theorem

Any sequential MTTKRP algorithm performs at least $\frac{1}{3^{2-1/d}}\frac{dNR}{M^{1-1/d}} - M$ *loads and stores.*

Proof: Data transfer lower bound $= \left\lfloor \frac{NR}{|F|} \right\rfloor M \geq \left(\frac{NR}{|F|} - 1\right) M = \frac{1}{3^{2-1/d}}\frac{dNR}{M^{1-1/d}} - M$

Corollary

Any parallel MTTKRP algorithm performs at least $\frac{1}{3^{2-1/d}}\frac{dNR}{PM^{1-1/d}} - M$ *sends and receives.*

Proof: There must be a processor which performs at least $\frac{NR}{P}$ loop iterations, applying the previous theorem for this processor yields the mentioned bound.

Generalized size of a segment

We are interested to know how many loop iterations we can perform by accessing A elements.

Optimization problem:

$$
\begin{aligned}
\text{Maximize } & |F_{M+A}| \text{ subject to} \\
& |F_{M+A}| \leq \prod_{i \in [d+1]} |\phi_i(F)|^{s_i} \\
\sum_{i \in [d+1]} |\phi_i(F)| & \leq M + A
\end{aligned}
$$

Data transfer lower bound $= \left\lfloor \frac{NR}{|F_{M+A}|} \right\rfloor A \geq \left(\frac{NR}{|F_{M+A}|} - 1 \right) A$

We select A such that the bound is maximum.

Communication optimal sequential algorithm

We select a block size b such that $b^d + db \leq M$.

1. Loop over $b \times \cdots \times b$ blocks of the tensor
2. With block in memory, loop over subcolumns of input factor matrices and update corresponding subcolumn of output matrix

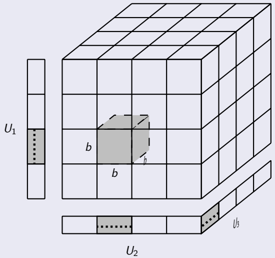
Amount of data transfer is bounded by

$$N + \left\lceil \frac{n_1}{b} \right\rceil \cdots \left\lceil \frac{n_d}{b} \right\rceil \cdot R(d+1)b.$$

With $b \approx M^{1/N}$, data transfer cost $=$

$$\mathcal{O}\left(N + \frac{dNR}{M^{1-1/d}}\right)$$

Sequential block algorithm for $d = 3$:



Comparisons

	Lower Bound	**All-at-once**	**Conventional (MM)**
Flops	-	dNR	$2NR$
Words	$\Omega\left(\frac{dNR}{M^{1-1/d}}\right)$	$\mathcal{O}\left(N + \frac{dNR}{M^{1-1/d}}\right)$	$O\left(N + \frac{NR}{M^{1/2}}\right)$
Temp Mem	-	-	$\frac{NR}{n_i}$

- All-at-once approach performs $\frac{d}{2}$ more flops than the conventional approach
- For relatively small R, N term dominates communication
  - This is the typical case in practice
- For relatively large R, all-at-once approach based algorithm communicates less
  - better exponent on M

Table of Contents

Settings to compute parallel communication lower bound

- The algorithm load balances the computation – each processor performs NR/P number of loop iterations

- One copy of data is in the system
  - There exists a processor whose input data at the start plus output data at the end must be at most $\frac{N+\sum_{i=1}^{d} n_i R}{P}$ words – will analyze data transfers for this processor

Communication lower bound

Let F be the subset of iteration space evaluated on a processor. $\phi_i(F)$ denotes the projection of F on the i-th variable. We recall that $\mathrm{s}^\mathsf{T} = \left[1-\frac{1}{d}, \frac{1}{d}, \cdots, \frac{1}{d}\right]$.

Optimization problem:

$$\text{Minimize} \sum_{i\in[d+1]} |\phi_i(F)| \text{ subject to}$$

$$\frac{NR}{P} \leq \prod_{i\in[d+1]} |\phi_i(F)|^{s_i}$$

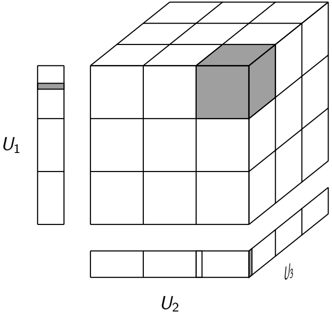
After solving the above optimization we obtain,

$$\sum_{i\in[d+1]} |\phi_i(F)| = \left(\sum_i s_i\right)\left(\frac{NR/P}{\prod_i s_i^{s_i}}\right)^{1/\sum_i s_i} = (2-1/d)\left(\frac{NR/P}{\prod_i s_i^{s_i}}\right)^{\frac{d}{2d-1}} \geq 2\left(\frac{dNR}{P}\right)^{\frac{d}{2d-1}}.$$

$$\begin{aligned}\text{Communication lower bound } &= \sum_{i\in[d+1]} |\phi_i(F)| - \text{data owned by the processor}\\ &\geq 2\left(\frac{dNR}{P}\right)^{\frac{d}{2d-1}} - \frac{N+\sum_{i=1}^{d} n_i R}{P}\end{aligned}$$

Sketch of communication optimal algorithm for $d = 3$

Assume that the required rank (R) is small. We do not need to communicate tensor in this setting. Suppose we want to update U_2.

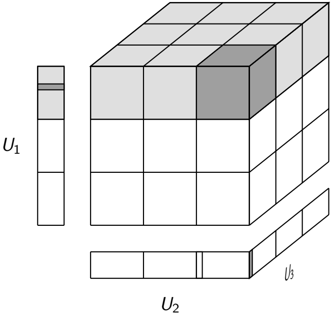


Each processor

1. Starts with one subtensor and subset of rows of each input factor matrix

Sketch of communication optimal algorithm for $d = 3$

Assume that the required rank (R) is small. We do not need to communicate tensor in this setting. Suppose we want to update U_2.

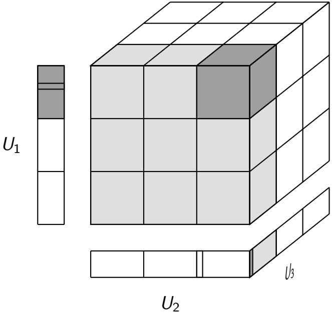


Each processor

1. Starts with one subtensor and subset of rows of each input factor matrix
2. All-Gathers all the rows needed from U_1

Sketch of communication optimal algorithm for $d = 3$

Assume that the required rank (R) is small. We do not need to communicate tensor in this setting. Suppose we want to update U_2.

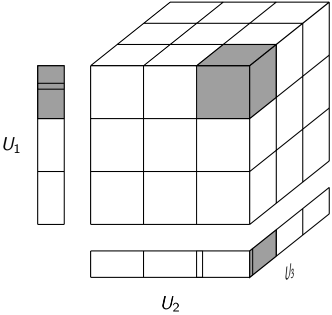


Each processor

1. Starts with one subtensor and subset of rows of each input factor matrix
2. All-Gathers all the rows needed from U_1
3. All-Gathers all the rows needed from U_3

Sketch of communication optimal algorithm for $d = 3$

Assume that the required rank (R) is small. We do not need to communicate tensor in this setting. Suppose we want to update U_2.

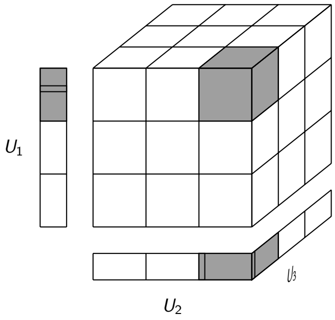


Each processor

1. Starts with one subtensor and subset of rows of each input factor matrix
2. All-Gathers all the rows needed from U_1
3. All-Gathers all the rows needed from U_3
4. Computes its contribution to rows of U_2 (local MTTKRP)

Sketch of communication optimal algorithm for $d = 3$

Assume that the required rank (R) is small. We do not need to communicate tensor in this setting. Suppose we want to update U_2.

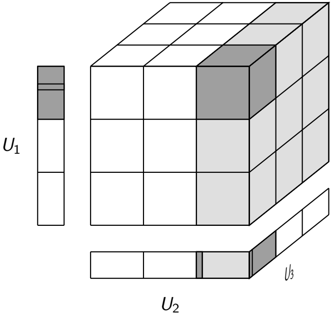


Each processor

1. Starts with one subtensor and subset of rows of each input factor matrix
2. All-Gathers all the rows needed from U_1
3. All-Gathers all the rows needed from U_3
4. Computes its contribution to rows of U_2 (local MTTKRP)

Sketch of communication optimal algorithm for $d = 3$

Assume that the required rank (R) is small. We do not need to communicate tensor in this setting. Suppose we want to update U_2.



Each processor

1. Starts with one subtensor and subset of rows of each input factor matrix
2. All-Gathers all the rows needed from U_1
3. All-Gathers all the rows needed from U_3
4. Computes its contribution to rows of U_2 (local MTTKRP)
5. Reduce-Scatters to compute and distribute U_2 evenly

Parallel communication optimal MTTKRP algorithm

Algorithm 2 Parallel MTTKRP algorithm

Require: input tensor $\mathcal{A} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$, factor matrices $U_j \in \mathbb{R}^{n_j \times R}$ for $1 \leq j \leq d$, mode j, P processors are logically arranged in $p_0 \times p_1 \times \cdots \times p_d$ logical processor grid

Ensure: Updated U_j

```
(p'_0, p'_1, ..., p'_d) is my processor id
//All-gather input tensor
A_{p'_1,...,p'_d} = All-Gather(A, (*, p'_1, ..., p'_d))
//All-gather factor matrices except U_j
for k ∈ [d] − {j} do
    (U_k)_{p'_0,p'_k} = All-Gather(U_k, (p'_0, *, ..., *, p'_k, *, ..., *))
end for
//Compute local MTTKRP
T = Local-MTTKRP (A_{p'_1,...,p'_d}, (U_k)_{p'_0,p'_k}, j)
//Reduce scatter along the processors which have same p'_0 and p'_j
Reduce-Scatter((U_j)_{p'_0,p'_j}, T, (p'_0, *, ..., *, p'_j, *, ..., *))
```

Communication cost

We set $p_0 \approx \frac{(dR)^{\frac{d}{2d-1}}}{(N/P)^{\frac{d}{2d-1}}}$ and $p_k \approx \frac{n_k}{(Np_0/P)^{\frac{1}{d}}}$ for $k \in [d]$.

Communication cost of the algorithm with the above processor grid is

$$\mathcal{O}\left(\frac{dNR}{P}\right)^{\frac{d}{2d-1}}.$$