

Heuristic for the Bilinear Rank Problem

Hamlil Mohamed

November 20, 2024

1 Introduction

Let G be the set of rank 1 bilinear maps.

Let K be a field.

Let T be a bilinear map defined as follows:

$$K^n \times K^m \rightarrow K^l$$

$\text{Dim}(T)$ = the number of bilinear maps that compose T , denoted T_i , defined as follows:

$$K^n \times K^m \rightarrow K$$

$\text{rank}(T) =$

$$\sum \text{rank}(T_i)$$

T' is the tensor that generates T with

$$\arg \min \left(\sum \text{rank}(T'_i) \right)$$

$\text{Brk}(T) = \text{rank}(T')$

1.1 Reformulating the Problem Using Tensors

Consider a bilinear map $f : \mathbb{R}^2 \times \mathbb{R}^2 \rightarrow \mathbb{R}$ defined by:

$$f((x_1, x_2), (y_1, y_2)) = x_1y_1 + 2x_1y_2 + 3x_2y_1 + 4x_2y_2$$

Matrix Representation

$$f((x_1, x_2), (y_1, y_2)) = \begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$

Tensor Representation (2 covariant)

$$f((x_1, x_2), (y_1, y_2)) = [[[1, 2], [3, 4]]] \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$

We will take polynomial multiplication as an example.

Let $A(x)$ and $B(x)$ be polynomials with coefficients from a field K of degrees n and m respectively.

Multiplying the two polynomials can be represented as a bilinear map:

$$K^n \times K^m \rightarrow K^l$$

$$((a_0, \dots, a_{n-1}), (b_0, \dots, b_{m-1})) \mapsto (c_0, \dots, c_{l-1})$$

The bilinear map can be represented as a tensor T_2^1 with 0 or 1 as coefficients. For example, let $A(x) = a_0 + a_1x$ and $B(x) = b_0 + b_1x + b_2x^2$. It can be represented as:

$$\begin{bmatrix} [1, 0, 0] & [0, 0, 0] \\ [0, 1, 0] & [1, 0, 0] \\ [0, 0, 1] & [0, 1, 0] \\ [0, 0, 0] & [0, 0, 1] \end{bmatrix}$$

This tensor will be called the associated tensor (similar to how linear maps have associated matrices), where:

$$\begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} [1, 0, 0] & [0, 0, 0] \\ [0, 1, 0] & [1, 0, 0] \\ [0, 0, 1] & [0, 1, 0] \\ [0, 0, 0] & [0, 0, 1] \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \end{bmatrix}$$

We can see that each row represents the multiplications used to calculate each term c .

1.1.1 Problem Identification

We will now try to find a decomposition where $T = \phi \cdot T'$. We will continue with the same example. The tensor T can be decomposed as follows:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ -1 & 0 & -1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} [1, 0, 0] & [0, 0, 0] \\ [0, 0, 1] & [0, 0, 0] \\ [0, 0, 0] & [0, 1, 0] \\ [0, 0, 0] & [0, 0, 1] \\ [1, 1, 0] & [1, 1, 0] \end{bmatrix}$$

Let $A(x)$ and $B(x)$ be polynomials with coefficients from a field K of degrees n and m respectively.

Multiplying the two polynomials can be represented as a bilinear map:

$$K^n \times K^m \rightarrow K^l$$

$$((a_0, \dots, a_{n-1}), (b_0, \dots, b_{m-1})) \mapsto (c_0, \dots, c_{l-1})$$

The bilinear map can be represented as a tensor T_2^1 with 0 or 1 as coefficients. For example, let $A(x) = a_0 + a_1x$ and $B(x) = b_0 + b_1x + b_2x^2$. It can be

represented as:

$$\begin{bmatrix} [1, 0, 0] & [0, 0, 0] \\ [0, 1, 0] & [1, 0, 0] \\ [0, 0, 1] & [0, 1, 0] \\ [0, 0, 0] & [0, 0, 1] \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \end{bmatrix}$$

This configuration is inspired by the Karatsuba method:

$$\begin{aligned} C(x) &= (a_1 \cdot x + a_0) \times (b_2 \cdot x^2 + b_1 \cdot x + b_0) \\ &= a_1 b_2 \cdot x^3 + (a_1 b_1 + a_0 b_2) \cdot x^2 + (a_0 b_1 + a_1 b_0) \cdot x + a_0 b_0 \end{aligned}$$

Five products are required instead of six. Using the Karatsuba method:

$$C(x) = a_1 b_2 \cdot x^3 + (a_1 b_1 + a_0 b_2) \cdot x^2 + ((a_0 + a_1)(b_0 + b_1) - a_1 b_1 - a_0 b_0) \cdot x + a_0 b_0$$

Calculate the products:

$$\begin{aligned} g_0 &= a_0 \cdot b_0, \\ g_1 &= a_0 \cdot b_2, \\ g_2 &= a_1 \cdot b_1, \\ g_3 &= a_1 \cdot b_2, \\ g_4 &= (a_0 + a_1) \cdot (b_0 + b_1). \end{aligned}$$

Reconstruct the result:

$$C(x) = g_3 \cdot x^3 + (g_1 + g_2) \cdot x^2 + (g_4 - g_2 - g_0) \cdot x + g_0$$

The tensor T' requires only five multiplications instead of six.

The key observation is that, in the last row of T' , the first and second components are identical, which means they can be taken as a common factor, reducing the number of multiplications to one. From this, we conclude that the number of multiplications in a given row is the number of linearly independent components.

For an all-zero vector, no calculation is necessary (the definition is linearly dependent). Linearly dependent vectors can be expressed by other vectors, thus reducing the number of multiplications in a row.

In simpler terms, the number of multiplications in a given row is the rank of the matrix formed by the components (covectors) of that row. We will call this matrix the row matrix and denote for a given tensor A that A_i is the i -th row matrix.

Therefore, the total number of multiplications is equal to the sum of the ranks of the row matrices. Our goal is to find a decomposition $T = \phi \cdot T'$ where:

$$\arg \min \left(\sum \text{rank}(T'_i) \right)$$

Conclusion

We need a generating family T' where

$$\arg \min \left(\sum \text{rank}(T'_i) \right)$$

and that generates the same vector space as T .

Thus, minimizing the number of multiplications in a bilinear map T is equivalent to finding a decomposition T' such that it generates T and:

$$\arg \min \left(\sum \text{rank}(T'_i) \right)$$

2 The Algorithm

Our algorithm will iterate over the set G and apply an oracle that will identify the rank 1 bilinear maps to add to T to obtain T' .

2.1 Properties of the Oracle

The oracle, when given T as input, returns a T'' such that $\text{rank}(T'') = \text{Brk}(T)$ under the constraint that $\text{Dim}(T) = \text{Dim}(T'')$. From this definition, the properties follow:

First Property: Suppose $T = T'$. For any v belonging to G , we have $\text{rank}(T) \leq \text{rank}(T + v)$.

Second Property: For any v belonging to G such that $\text{rank}(T + v) < \text{rank}(T)$, then v belongs to the span of T' .

Third Property: If there exist v_1, v_2 belonging to G such that $\text{rank}(T + v_1) \geq \text{rank}(T)$ and $\text{rank}(T + v_2) \geq \text{rank}(T)$, and that $\text{rank}(T + v_1 + v_2) \leq \text{rank}(T)$, then there exists v_3 belonging to G such that $\text{rank}(T + v_3) \leq \text{rank}(T)$.

2.2 The Oracle

The question arises: is there an oracle that respects the definition and characteristics mentioned above?

We propose a heuristic that respects these properties but does not always find the tensor T'' with the minimum number of multiplications. The heuristic models the oracle as a constrained optimization problem, then applies a *greedy* approach similar to the knapsack problem.

Steps of the heuristic

1. **First Step:** Generate the span of T .
2. **Second Step:** Sort the bilinear maps belonging to the span of T in ascending order according to their rank.

3. **Third Step:** Select the first linearly independent bilinear maps belonging to the span of T .

For better understanding, here is the pseudocode of our algorithm assuming the span of T will be sorted:

```

Function Greedy(T)
  s := empty list
  for each i in span_of(T)
    if is_linearly_independent(s, i)
      s := vcat(s, [i])
    end if
  end for
  return s
end function

Function rank_minimizer(T, G)
  for i in 1:length(G)
    if is_linearly_independent(T, G[i])
      v = smallest_base(vcat(T, [G[i]]))
      if Number_of_multiplications(v) < Number_of_multiplications(T)
        T = v
      else
        G_new = [j for j in G[i+1:end] if is_linearly_independent(v, j)]
        return rank_minimizer(T, G_new)
      end
    end
  end
  return T
end

```

3 Results and Discussion

Before passing T through rank minimizer:

We first try to find the smallest base in multiplications while keeping the same dimension.

Then, we noticed that before iterating over G , we could start with the set of rank 1 bilinear maps that compose the bilinear maps that compose T , as some bilinear maps that compose T have multiplications (rank 1 bilinear maps) in common.

We will use the rank minimizer algorithm.

- Step 1: apply Greedy on T .

- Step 2: Then, apply rank minimizer on T resulting from Step 1 and the set of rank 1 bilinear maps that compose the bilinear maps that compose T .
- Step 3: Finally, apply rank minimizer on T resulting from Step 2.

The Set	Polynomial Multiplication	Original Rank	Step 1		Step 2		Step 3	
			Rank	Time	Rank	Time	Rank	Time
\mathbb{F}_2	5x5	25	16	0.2	14	4.38	14	586.22
\mathbb{F}_2	3x8	24	19	0.15	16	21.39	15	3460.54
\mathbb{F}_2	4x7	28	19	0.19	16	24	16	5044.06
\mathbb{F}_3	3x6	18	12	0.69	11	18.05		more than 4

Table 1: Table on tests done with Julia language. The reported times correspond to the total execution time on a single core of a 12th Gen Intel(R) Core(TM) i5-12450H at 2.2 GHz.

4 Conclusion

In terms of computation time, our heuristic gives very good results, and these results can still be improved. For example:

- The heuristic finds the bilinear maps that belong to T' , which means that the solution T'' can still be minimized (it does not lead to a bad path).
- Relate the oracle problem to the matrix rank minimization problem and use heuristics based on the nuclear norm or NP-complete algorithms.
- Minimize the set G using automorphisms RP .