Grassmannian as Continuous Abstract Data Type with Computable Semantics

Seokbin Lee, Donghyun Lim, Sewon Park, and Martin Ziegler

Korea Advanced Institute of Science and Technology

1st July 2020

We can think of a Grassmannian as a subspace of a given vector space.

Grassmannians

We can think of a Grassmannian as a subspace of a given vector space.
More formally, we denote $Gr(k, V)$ to be the set of all k-dimensional linear subspaces of a vector space V.

Grassmannians

We can think of a Grassmannian as a subspace of a given vector space.
More formally, we denote $Gr(k, V)$ to be the set of all k-dimensional linear subspaces of a vector space V.

Example.

The Grassmannian $Gr(1, \mathbb{R}^3)$ is the set of all lines in $\mathbb{R}^3$ passing through the origin. The Grassmannian $Gr(n-1, \mathbb{R}^n)$ is the set of hyperplanes in $\mathbb{R}^n$, passing through the origin.

Representation of Grassmannian elements

An intuitive way of representing Grassmannian elements is to consider a basis for the subspace, and encoding the basis as column vectors for a $d \times m$ matrix, where m is the subspace dimension and d is the ambient dimension.

Representation of Grassmannian elements

An intuitive way of representing Grassmannian elements is to consider a basis for the subspace, and encoding the basis as column vectors for a $d \times m$ matrix, where m is the subspace dimension and d is the ambient dimension.
Computation of operations on the Grassmannian is defined with respect to such a basis representation.

Operations on Grassmannian elements (1/3)

We are interested in operations on Grassmannian elements.

Operations on Grassmannian elements (1/3)

We are interested in operations on Grassmannian elements.
In particular, given a subspaces A and B of a d-dimensional Euclidean space, we are interested in the *complement* of A, the *join* and *meet* of A and B, and the *projection* of B onto A.

Definition.

Given a Grassmannian element A, its orthogonal complement is the set $A^{\perp} = \{x : \forall a \in A, x \perp a\}$.

Operations on Grassmannian elements (2/3)

Definition.

Given Grassmannian elements A and B, the set $A + B = \{a + b : a \in A, b \in B\}$ is the join (or Minkowski sum) of A and B.

Operations on Grassmannian elements (2/3)

Definition.

Given Grassmannian elements A and B, the set $A + B = \{a + b : a \in A, b \in B\}$ is the join (or Minkowski sum) of A and B.

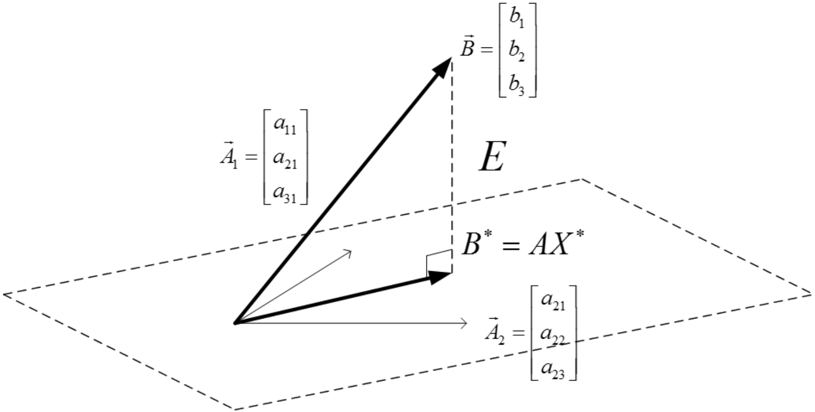
Definition.

Given Grassmannian elements A and B, the set $A \cap B$ is the meet (or intersection) of A and B.

Operations on Grassmannian elements (3/3)

Definition.

Given Grassmannian elements A and B, the set $\text{proj}_A B$ is the projection of B onto A.



Specifications (1/3)

We will be utilizing Gaussian Elimination for the input matrices.

Specifications (1/3)

We will be utilizing Gaussian Elimination for the input matrices.
The specification on operations has a non-triviality:

Fact.

Testing inequality in Exact Real Computation is equivalent to the Halting problem, so in undecidable (yet semidecidable).

We will be utilizing Gaussian Elimination for the input matrices. The specification on operations has a non-triviality:

Fact.

Testing inequality in Exact Real Computation is equivalent to the Halting problem, so in undecidable (yet semidecidable).

Since we are representing subspaces as matrices with elements in $\mathbb{R}$, we have the following:

Corollary.

Testing equality "$x = 0$" is undecidable.

Specifications (2/3)

Fortunately, we have the following multi-valued "select" operator, which is computable:

Fortunately, we have the following multi-valued "select" operator, which is computable:

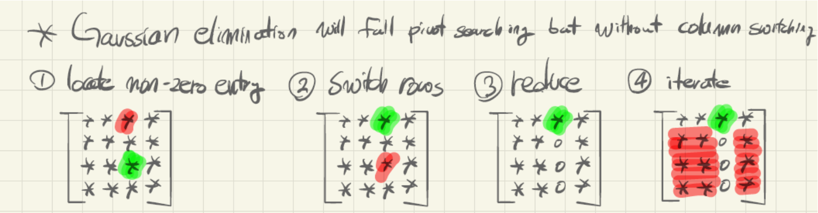
Definition.

The multi-valued `select` function takes two inputs b and c from $\{0, 1, \bot\}$, and computes as follows:

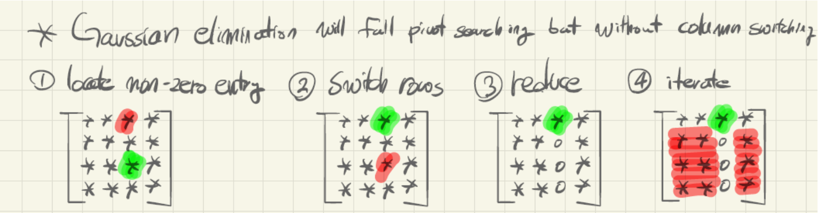
$$\texttt{select}(b, c) = \begin{cases} 0 & \text{if } b \text{ is defined} \\ 1 & \text{if } c \text{ is defined} \\ 0/1 & \text{if both are defined} \\ \bot & \text{if neither are defined} \end{cases}$$

We will thus use a modified version of Gaussian Elimination that instead does not check for equality.

We will thus use a modified version of Gaussian Elimination that instead does not check for equality.



We will thus use a modified version of Gaussian Elimination that instead does not check for equality.



Also, we will *specify* that the dimension of the output space is given as part of the input.

The algorithm for the orthogonal complement of a subspace A is given in the following pseudocode.

The algorithm for the orthogonal complement of a subspace A is given in the following pseudocode.

procedure GAUSSIAN1(A, k) ▷ A has dimensions $m \times n$
    for i from 1 to k **do** ▷ k is the number of iterations
        $j \leftarrow$ **select**$(|A_{i,i}| > 0, \ldots, |A_{i,n}| > 0)$
        Swap c_i and c_j of A ▷ c_i denotes the i-th column of A
        for p from $i+1$ to n **do**
            $c_p \leftarrow c_p - \frac{A_{i,p}}{A_{i,i}} c_i$
    return A
procedure COMPLEMENT(A, m, d)
    $M \leftarrow \begin{bmatrix} A^T \\ I_d \end{bmatrix}$ ▷ I_d is the $d \times d$ identity matrix
    $M \leftarrow \text{Gaussian}(M, m)$
    return $M[m+1 : m+d, m+1 : d]$ ▷ return the last d rows and $d - m$ columns of M

Algorithms for operations (1/4)

The algorithm for the orthogonal complement of a subspace A is given in the following pseudocode.

procedure GAUSSIAN1(A, k) ▷ A has dimensions $m \times n$
  for i from 1 to k **do** ▷ k is the number of iterations
    $j \leftarrow$ **select**$(|A_{i,i}| > 0, \ldots, |A_{i,n}| > 0)$
    Swap c_i and c_j of A ▷ c_i denotes the i-th column of A
    for p from $i+1$ to n **do**
      $c_p \leftarrow c_p - \frac{A_{i,p}}{A_{i,i}} c_i$
  return A
procedure COMPLEMENT(A, m, d)
  $M \leftarrow \begin{bmatrix} A^T \\ I_d \end{bmatrix}$ ▷ I_d is the $d \times d$ identity matrix
  $M \leftarrow \text{Gaussian}(M, m)$
  return $M[m+1 : m+d, m+1 : d]$ ▷ return the last d rows and $d-m$ columns of M

This follows from the fact that $\text{col}(A)^{\perp} = \ker(A^T)$.

Algorithms for operations (2/4)

The following describe the algorithm for the join, meet, and projection of two subspaces.

Algorithms for operations (2/4)

The following describe the algorithm for the join, meet, and projection of two subspaces.

For join and meet, we want to reduce the matrix $\left[\begin{array}{c|c} A & B \\ A & 0 \end{array}\right]$ to column-reduced form $\left[\begin{array}{c|c} C & 0 \\ * & D \end{array}\right]$ via Zassenhaus' Algorithm.

Algorithms for operations (2/4)

The following describe the algorithm for the join, meet, and projection of two subspaces.

For join and meet, we want to reduce the matrix $\left[\begin{array}{c|c} A & B \\ A & 0 \end{array}\right]$ to column-reduced form $\left[\begin{array}{c|c} C & 0 \\ * & D \end{array}\right]$ via Zassenhaus' Algorithm.

procedure GAUSSIAN2(A, r) ▷ A has dimensions $m \times n$
  for i from 1 to k **do** ▷ r is the number of recursions
    if $r = 0$ **then**
      return A
    $j, k \leftarrow$ **select**$(|A_{1,1}| > 0, \ldots, |A_{2d,m+n}| > 0)$
    Swap c_1 and c_k of A ▷ c_k denotes the k-th column of A
    for p from 2 to n **do**
      $c_p \leftarrow c_p - \frac{A_{j,p}}{A_{j,1}} c_1$
  $A[1 : j-1 \cup j+1 : m, 2 : n] \leftarrow \text{Gaussian2}(A[1 : j-1 \cup j+1 : m, 2 : n], r-1)$
  return A
procedure JOIN(A, B, l)
  $M \leftarrow \left[\begin{array}{c|c} A & B \\ A & 0 \end{array}\right]$
  $M \leftarrow \text{Gaussian2}(M, l)$
  return $M[1 : d, 1 : l]$

Whereas the submatrix C has the information of the join, the submatrix D has the meet:

Algorithms for operations (3/4)

Whereas the submatrix C has the information of the join, the submatrix D has the meet:

procedure MEET(A, B, l)

12: $M \leftarrow \left[\begin{array}{c|c} A & B \\ A & 0 \end{array}\right]$

$M \leftarrow \text{Gaussian2}(M, l)$

return $M[d+1:2d, m+n-k+1:m+n]$

Finally, the algorithm for projection is given as below; recall that a projection matrix is of the form $A(A^T A)^{-1} A^T$ for the underlying subspace A.

Algorithms for operations (4/4)

Finally, the algorithm for projection is given as below; recall that a projection matrix is of the form $A(A^TA)^{-1}A^T$ for the underlying subspace A.

```
procedure PROJECTION(A, B, l)
    P ← A(A^T A)^{-1} A^T      ▷ We are guaranteed the existence of (A^T A)^{-1} because A^T A is regular
    return Gaussian2(PB, l)[1 : d, 1 : l]
```

Implementation in iRRAM

The source code can be viewed at
https://github.com/realcomputation/irramplus/tree/master/GRASSMANN.

Thank you!