Algorithms and Data Structures (CSci 115)

California State University Fresno
College of Science and Mathematics
Department of Computer Science
H. Cecotti

Learning outcomes

- Vectors, Matrices, and Tensors
- Class Matrix

Introduction

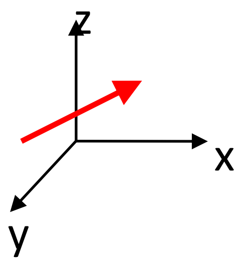
- Most of the data structures that we saw
  - Mainly relevant for single-thread applications
    - Doesn't mean you have to throw everything away!!
  - State of the art algorithms
    - Some defined in the 70s and adapted to the hardware of the 70s
      - B-tree on disk storage
- Data structures
  - A use for a particular type of function
  - Need to transfer data between data structures
    - Fill a data structure from another data structure
      - Array to List
      - Graph
        - Example: Ajacency list ← → Ajacency matrix
    - Possible need of **several** data structures to represent a **unique** dataset
  - Why
    - Pseudo code suggests one data structure but
    - Real-world/the implementation can lead you to a different choice
    - Because you may want to take advantage of parallel processing

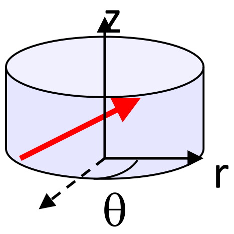
Rationale

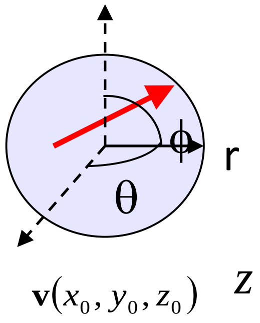
- **Question 1**
  - How to represent:
    - $y = \sum_{i=0}^{n-1} w(i)^2$
    - where w is a vector
- **Question 2**
  - We consider 2 vectors: A and B
    - What is the pseudo-code to obtain the cross product of A and B?
  - Use loops
  - Consider A and B as “vectors”
- → Difference between
    - vectors (linear algebra) (meaning of the representation vertical/horizontal)
    - arrays (sequence of elements of a fixed size)

Rationale

- Representation of data along multiple dimensions
  - X, Y (2D) : image
  - X, Y, Time (3D) : frames in a video
  - X, Y, N (3D) : stack of images
  - X, Y, Z, Time (4D)
- Extraction of statistical information along dimension(s)
  - Projection in 1 dimension
  - Example: mean, standard deviation, … across a particular dimension
- Change the way you will observe data
  - Linear algebra
    - Example: Principal Component Analysis





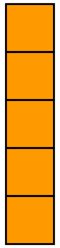


Tensors

- **Definition**
  - An n^{th}-rank tensor in m-dimensional space is a mathematical object that has n indices and m^n components
  - Geometric objects that describe linear relations between geometric vectors, scalars, and other tensors
    - Each index of a tensor ranges over the number of dimensions of space
- Example 3-rank tensor
  - A_{ijk} = access the cell at position (i,j,k) in A
- Big topic in Linear Algebra
  - We keep it only for simple functions

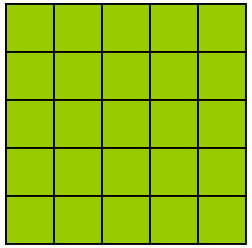
Tensors

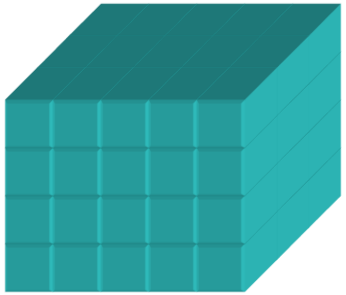
- Example
  - Vector= **1 column** = 1 matrix of size n x 1
  - From 1D to 3D



Vector: order-1 tensor

Matrix: order-2 tensor

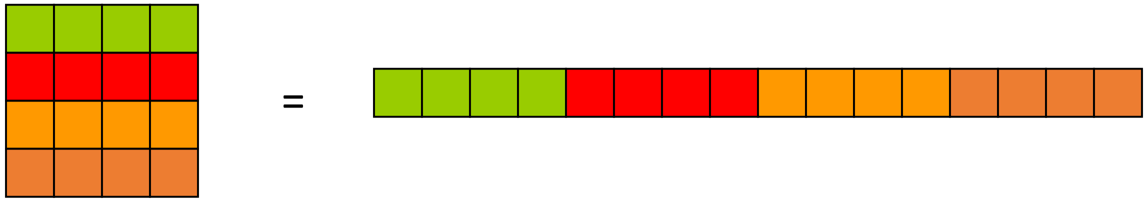




Order-3 tensor

Reshape function

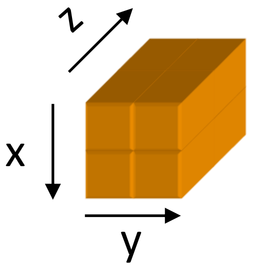
- To reshape
  - To change the dimensions
    - Example:
      - From a Matrix (n x m) to a vector (n*m x 1)



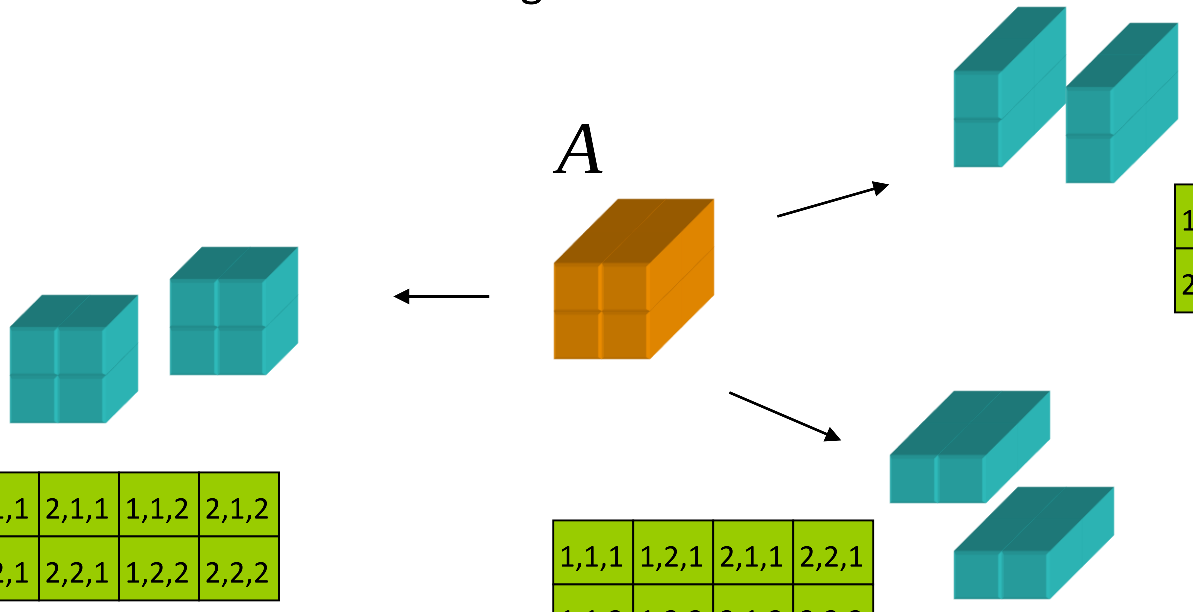
The idea: Parallel processing
Towards a **single** loop: for each block b, do f(b)
→ Cut the vector of blocks into p blocks (p threads/processors)

Reshape function

- Example
  - ➢Order-3 tensor → Matrix



“Flattening” an order-3 tensor



$A_{(1)}$

1,1,1	1,1,2	1,2,1	1,2,2
2,1,1	2,1,2	2,2,1	2,2,2

$A_{(2)}$

1,1,1	2,1,1	1,1,2	2,1,2
1,2,1	2,2,1	1,2,2	2,2,2

$A_{(3)}$

1,1,1	1,2,1	2,1,1	2,2,1
1,1,2	1,2,2	2,1,2	2,2,2

Repmat

- Repmat
  - B = repmat(A,n)
    - It returns an array containing n copies of A in the row and column dimensions
    - The size of B is size(A)*n when A is a matrix.
  - Need to extend/replicate existing data
- Example
  - $y = \sum_{i=0}^{n-1} w(i) - x$
  - x is a constant in this example:
    - Need to create a new vector of size n that contains x everywhere to y=w-**x**

Tensor product and direct sum

- Matrix product A (p x q) * B (q x r) = C (p x r)
- Tensor sum (Kronecker sum) $\dim(V \otimes W) = \dim V \times \dim W.$

the Kronecker sum of two 2×2 matrices $(a)_{ij}$ and $(b)_{ij}$ is given by

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \oplus \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix}$$

$$= \begin{bmatrix} a_{11}+b_{11} & b_{12} & a_{12} & 0 \\ b_{21} & a_{11}+b_{22} & 0 & a_{12} \\ a_{21} & 0 & a_{22}+b_{11} & b_{12} \\ 0 & a_{21} & b_{21} & a_{22}+b_{22} \end{bmatrix}.$$

$$A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}_V , \qquad B = \begin{pmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{pmatrix}_W ;$$

$$A \oplus B = \left(\begin{array}{cc|ccc} a_{11} & a_{12} & 0 & 0 & 0 \\ a_{21} & a_{22} & 0 & 0 & 0 \\ \hline 0 & 0 & b_{11} & b_{12} & b_{13} \\ 0 & 0 & b_{21} & b_{22} & b_{23} \\ 0 & 0 & b_{31} & b_{32} & b_{33} \end{array}\right)$$

- Tensor direct product $\dim(V \otimes W) = \dim V \times \dim W.$

the matrix direct product of the 2×2 matrix A and the 3×2 matrix B is given by the following 6×4 matrix

$$A \otimes B = \begin{bmatrix} a_{11}\,B & a_{12}\,B \\ a_{21}\,B & a_{22}\,B \end{bmatrix}$$

$$= \begin{bmatrix} a_{11}\,b_{11} & a_{11}\,b_{12} & a_{12}\,b_{11} & a_{12}\,b_{12} \\ a_{11}\,b_{21} & a_{11}\,b_{22} & a_{12}\,b_{21} & a_{12}\,b_{22} \\ a_{11}\,b_{31} & a_{11}\,b_{32} & a_{12}\,b_{31} & a_{12}\,b_{32} \\ a_{21}\,b_{11} & a_{21}\,b_{12} & a_{22}\,b_{11} & a_{22}\,b_{12} \\ a_{21}\,b_{21} & a_{21}\,b_{22} & a_{22}\,b_{21} & a_{22}\,b_{22} \\ a_{21}\,b_{31} & a_{21}\,b_{32} & a_{22}\,b_{31} & a_{22}\,b_{32} \end{bmatrix}.$$

$$A \otimes B = \left(\begin{array}{ccc|ccc} a_{11}b_{11} & a_{11}b_{12} & a_{11}b_{13} & a_{12}b_{11} & a_{12}b_{12} & a_{12}b_{13} \\ a_{11}b_{21} & a_{11}b_{22} & a_{11}b_{23} & a_{12}b_{21} & a_{12}b_{22} & a_{12}b_{23} \\ a_{11}b_{31} & a_{11}b_{32} & a_{11}b_{33} & a_{12}b_{31} & a_{12}b_{32} & a_{12}b_{33} \\ \hline a_{21}b_{11} & a_{21}b_{12} & a_{21}b_{13} & a_{22}b_{11} & a_{22}b_{12} & a_{22}b_{13} \\ a_{21}b_{21} & a_{21}b_{22} & a_{21}b_{23} & a_{22}b_{21} & a_{22}b_{22} & a_{22}b_{23} \\ a_{21}b_{31} & a_{21}b_{32} & a_{21}b_{33} & a_{22}b_{31} & a_{22}b_{32} & a_{22}b_{33} \end{array}\right)$$

C++ STL

- You don’t need to reinvent the wheel
  - **But** you need to know how it works
- You can use libraries
  - Standard Template Library (STL) (Architecture: Alexander Stepanov)
    - Toward generic programming
    - 4 key components
      - Algorithms
      - Containers
      - Functions
      - Iterators
    - Some issues before C++ v11.
  - Libraries from other developers
    - Warning
      - Interoperability between other different components of your application
      - → to port an application from one type of machine to another, with hardware dependent libraries

C++ STL

- Sequence Containers (ordered collections):
    - vector
        - Dynamic array: **double** the size when you need space $\frac{nO(1)+O(n)}{n+1} = O(1)$
    - list
    - deque (double-ended queue)
    - arrays
    - forward_list
- Container Adaptors: They provide a different interface for sequential containers.
    - queue (FIFO queue)
    - priority_queue
    - stack (LIFO stack)
- Associative Containers (unordered collections):
  - ➢ They implement sorted data structures that can be quickly searched (O(log n) complexity).
    - set
    - multiset
    - map
    - multimap

Class Matrix

- No matrix or tensor in STL
  - You have to use other libraries
- See example on blackboard
  - Matrix.h
  - Matrix.cpp

Conclusion

- **Think** Tensors because
  - Some programming languages are primarily based on vectors/matrices
    - Matlab
  - Used for image processing, machine learning, big data, engineering
    - Problems in Physics (elasticity, fluid mechanics, and general relativity)
  - Important for multithreaded applications
    - Decomposition of the data structure into multiple blocks
      - That can be processed in parallel
- Use well established Libraries
  - uBLAD
    - https://www.boost.org/doc/libs/1_67_0/libs/numeric/ublas/doc/index.html
  - arma
    - http://arma.sourceforge.net/
  - Vector c++ stl libraries

Questions ?

- Reading
  - See link on Canvas