Matrix-vector products

The operational way: row-by-column $y_i = \sum_j A_{ij} x_j$.

$$\begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \\ A_{41} & A_{42} & A_{43} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix}.$$

The smart way: linear combinations of columns of A

$$\underbrace{\begin{bmatrix} A_{11} \\ A_{21} \\ A_{31} \\ A_{41} \end{bmatrix}}_{\mathbf{v}_1} x_1 + \underbrace{\begin{bmatrix} A_{12} \\ A_{22} \\ A_{32} \\ A_{42} \end{bmatrix}}_{\mathbf{v}_2} x_2 + \underbrace{\begin{bmatrix} A_{31} \\ A_{32} \\ A_{33} \\ A_{43} \end{bmatrix}}_{\mathbf{v}_3} x_3 = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix}.$$

The entries of $\mathbf{x}$ are coordinates used to write $\mathbf{y}$ as a linear combination of $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$.

Bases

A basis is a tuple of vectors $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n$ such that we can write each vector $\mathbf{y}$ of a certain space, uniquely, as a linear combination of them.

Uniquely = coordinates for each vector are unique / well defined.

Canonical basis: vectors with only one 1; e.g. for $n = 4$

$$\mathbf{e}_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad \mathbf{e}_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \quad \mathbf{e}_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \quad \mathbf{e}_4 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}.$$

Coordinates wrt this basis $\iff$ vector entries

$\mathbf{y} = \mathbf{e}_1 y_1 + \mathbf{e}_2 y_2 + \mathbf{e}_3 y_3 + \mathbf{e}_4 y_4$.

(I like to put scalars on the right.)

(In real life, vectors are not always boldfaced/underlined for your convenience.)

Linear systems

Problem: find coordinates $x_1, \ldots, x_n$ needed to write $\mathbf{y}$ as linear combinations of the columns of $A \in \mathbb{R}^{m \times n}$, or

$$A\mathbf{x} = \mathbf{y}.$$

Sometimes there are multiple solutions, or none, e.g.,

$$A = \begin{bmatrix} 2 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad \mathbf{y}_1 = \begin{bmatrix} 4 \\ 4 \\ 0 \\ 0 \end{bmatrix}, \quad \mathbf{y}_2 = \begin{bmatrix} 4 \\ 4 \\ 1 \\ 2 \end{bmatrix}.$$

Im A: the set of vectors $\mathbf{y}$ that we can obtain.
ker A: possible choices of $\mathbf{x}$ that produce $A\mathbf{x} = \mathbf{0}$.

Main problem in this part of our course: find $\mathbf{x}$ that reaches a given $\mathbf{y}$ exactly, or gets as close as possible.

Square linear systems

A is called invertible if $A\mathbf{x} = \mathbf{y}$ has a unique solution; i.e., its columns are a basis or $\mathbb{R}^n$. (It must be square for this to hold.)

In this case, the solution is given by another matrix: $\mathbf{x} = A^{-1}\mathbf{y}$

$$AA^{-1} = A^{-1}A = I = \begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & & \ddots & \\ & & & & 1 \end{bmatrix}$$

(Convention: omitted elements are zero.)

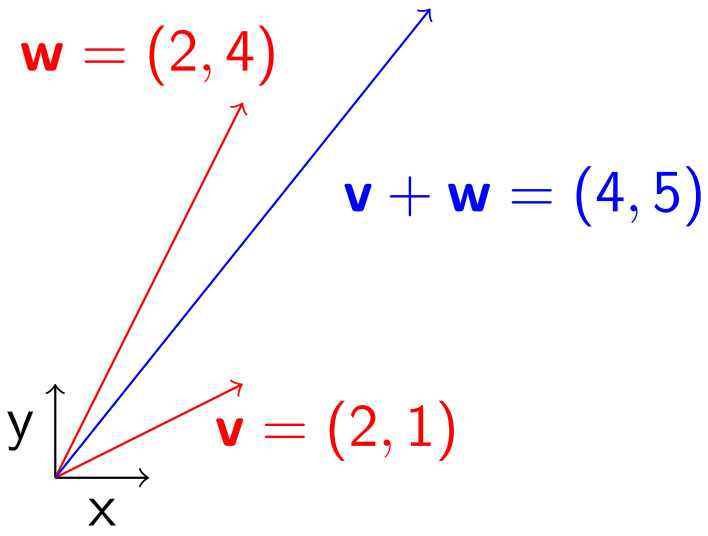
Warning `inv(A)*y` is not the best way to solve $A\mathbf{x} = \mathbf{y}$, numerically.
Most languages have a specialized instruction, e.g., Matlab's `x = A \ y` or Python's `scipy.linalg.solve(A, y)`. Use it!

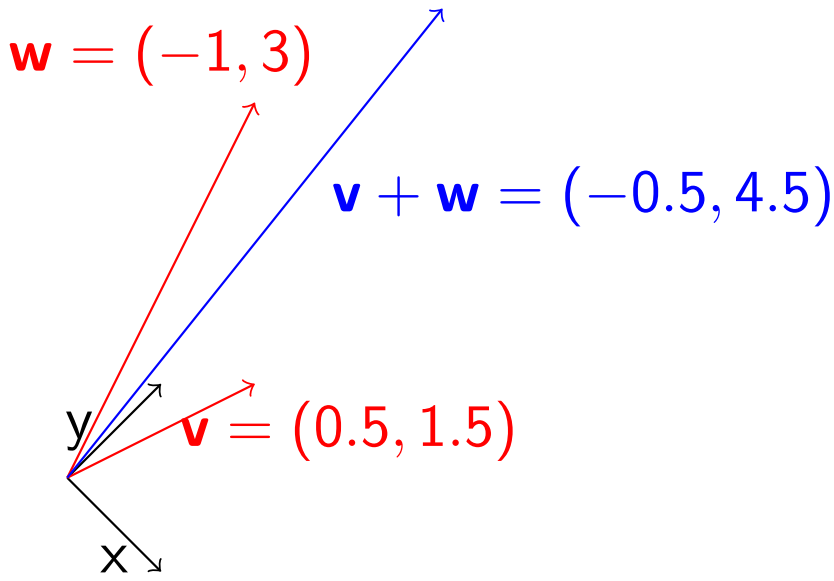
Linear algebra in a slide

(You have already studied linear algebra, right?)

The powerful idea behind linear algebra

- there is a 'space' of vectors as abstract geometrical objects.
- we can represent them as coordinates w.r.t a given basis.
- operations on vectors $\iff$ operations on coordinates.
- many relations are true regardless of the choice of coordinates.

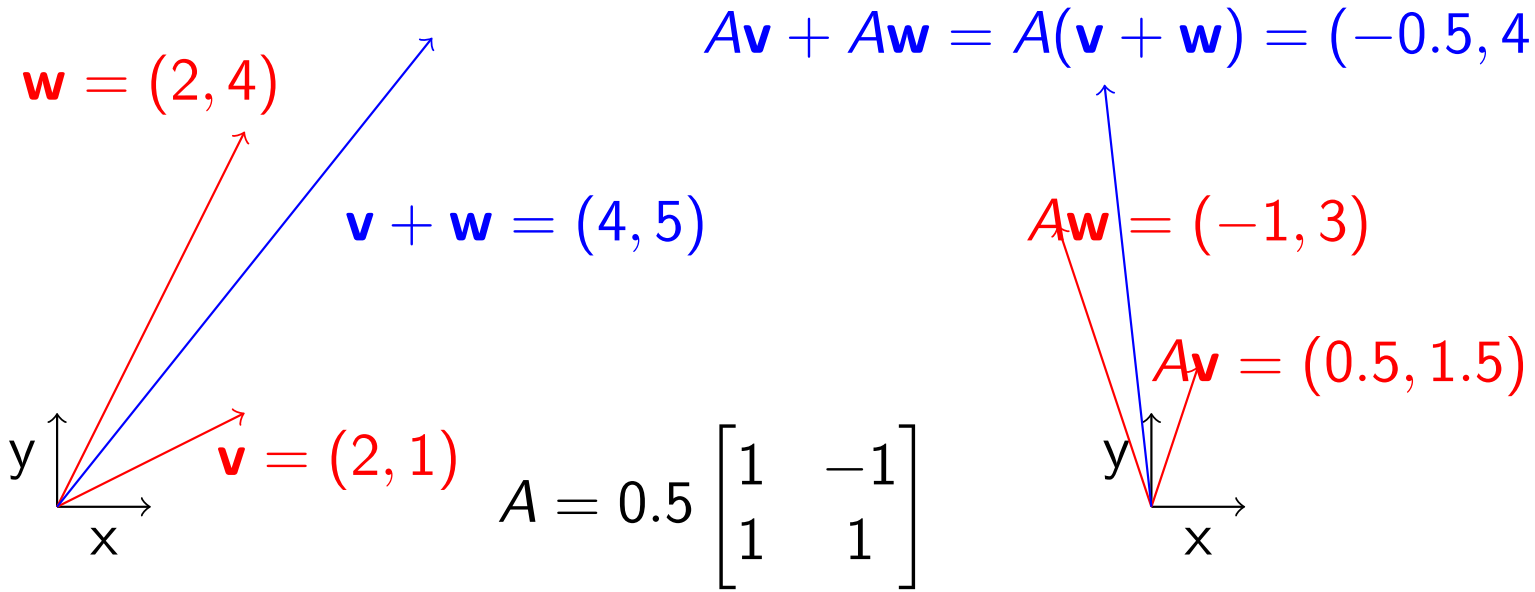




Matrices in linear algebra

Matrices also represent linear transformations: if you plot $\mathbf{v}$ and $A\mathbf{v}$ on the same axes, the second is a transformation (rotation, or scaling, shearing, ...) of the first.

Example: the change-of-basis matrix



$$A = 0.5 \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}$$

I represents the identity: $I\mathbf{v} = \mathbf{v}$ for each $\mathbf{v}$.

$A(B\mathbf{v})$ = "apply B first, then A" is another transformation, represented by the matrix AB; so $A(B\mathbf{v}) = (AB)\mathbf{v}$.

Matrix-matrix product

$$\begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \\ A_{41} & A_{42} & A_{43} \end{bmatrix} \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \\ B_{31} & B_{32} \end{bmatrix}$$

$A \in \mathbb{R}^{4\times 3}$, $B \in \mathbb{R}^{3\times 2}$. $AB \in \mathbb{R}^{4\times 2}$.

Mnemonic: if the inner dimensions agree, the product is well-defined and removes them.

We can identify vectors with columns ($n \times 1$ matrices).

Cost: multiplying $m \times n$ and $n \times p$ requires $m(2n-1)p = O(mnp)$ floating point operations (flops). Fancier algorithms (e.g. Strassen) typically do not perform better in practice.

Order of operations

Usual manipulations work, e.g.: $A(B + C) = AB + AC$, $A(BC) = (AB)C$, etc.

Warning: Parenthesization matters a lot: if $A, B \in \mathbb{R}^{n \times n}$, $v \in \mathbb{R}^n$, then $(AB)v$ costs $O(n^3)$, but $A(Bv)$ costs $O(n^2)$.

Warning: programming languages usually do not rearrange parentheses to help you.

Matlab example:

```
n = 2000;
A = randn(n, n);
B = randn(n, n);
v = randn(n, 1);
tic, A * (B * v); toc
tic, (A * B) * v; toc
```

Matrix algebra

$$(A+B)^2 = (A+B)(A+B) = A^2 + AB + BA + B^2.$$

What doesn't work

$AB \neq BA$: one of them might not even make sense dimension-wise.
Exception: We can move around numbers (scalars): $3AB = A(3B)$.

$AB = AC$ does not imply $B = C$ (example: $A = B = \left[\begin{smallmatrix} 1 & 0 \\ 0 & 0 \end{smallmatrix}\right], C = I$).

However, if there is a matrix M such that $MA = I$, I can multiply by M:

$$(MA)B = (MA)C \iff B = C.$$

Warning: multiplying 'on the left' and 'on the right' differ.

Row and column vectors

It is useful to keep the concepts of row and column vectors separate.

$$\mathbf{c} = \begin{bmatrix} 4 \\ 5 \\ 6 \end{bmatrix}, \mathbf{r} = \mathbf{c}^T = \begin{bmatrix} 4 & 5 & 6 \end{bmatrix}.$$

$\mathbf{c}$ is a vector in $\mathbb{R}^3$; you can identify it with a matrix in $\mathbb{R}^{3\times 1}$ for most purposes. $\mathbf{r}$ is a matrix in $\mathbb{R}^{1\times 3}$ (or row vector).
Matlab keeps these concepts separate also in programming:

```
>> c = [4;5;6];
>> r = [1 2 3];
>> r * c
ans =
    32
```

This operation is the scalar product of vectors in $\mathbb{R}^n$:
$\mathbf{v}^\top \mathbf{w} = \langle \mathbf{v}, \mathbf{w} \rangle$.

Row and column vectors

```
>> c * r
ans =
     4  8 12
     5 10 15
     6 12 18
>> c'
ans =
     4  5  6
>> r * c'
Error using *
Inner matrix dimensions must agree.
```

Useful convention: "bare" letters are for columns, $^\top$ for rows.

Some people (even other professors) write **vw** for the scalar product $\mathbf{v}^\top\mathbf{w}$. This will be confusing: what is **uvw**?

$\mathbf{v} \cdot \mathbf{w}$: more acceptable; at least it's clear it is a different operation.

Rank

The word rank has a precise linear-algebra meaning. (For some computer scientists, rank = number of indices of an array; don't confuse the two concepts.)

Definition

Rank of a matrix A = minimum r so that it is possible to find vectors $v_1, \ldots, v_r$ such that all the columns of A are linear combinations of these vectors.

Example:

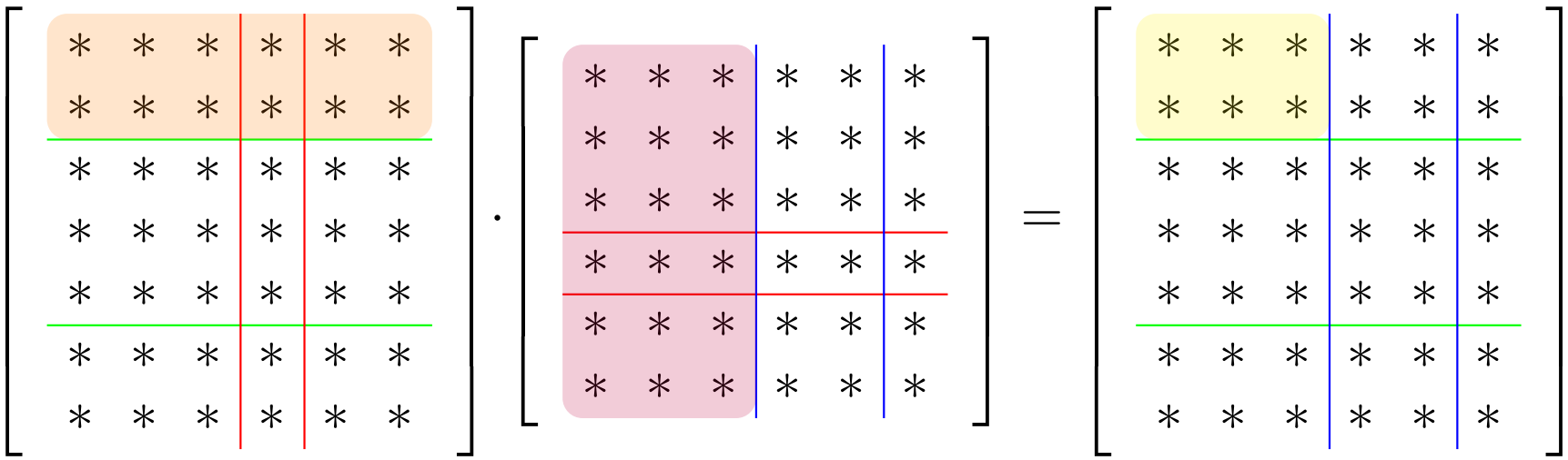
$$\mathbf{v}\mathbf{w}^T = \begin{bmatrix} v_1 w_1 & v_1 w_2 & v_1 w_3 \\ v_2 w_1 & v_2 w_2 & v_2 w_3 \\ v_3 w_1 & v_3 w_2 & v_3 w_3 \end{bmatrix}$$

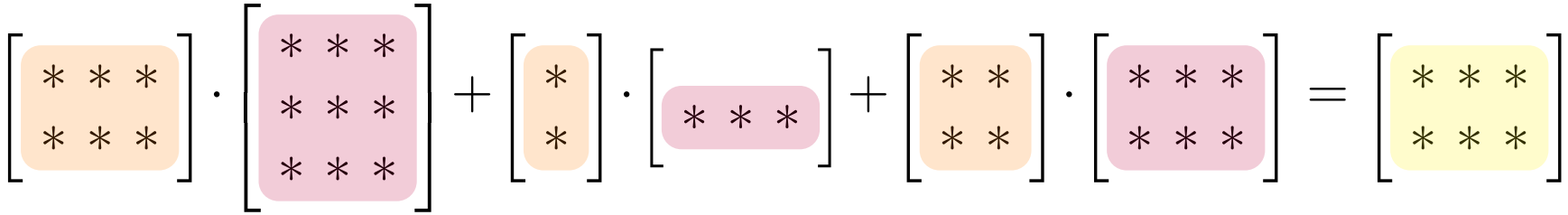
has rank $r = 1$: all columns are multiples of $\mathbf{v}$.

(Theorem: column rank = row rank: if you replace "columns" with "rows" in the definition, you get the same value r. For instance, in the example above, all rows are multiples of $\mathbf{w}^T$).

Block operations

When computing a matrix product, we get the same result if we use the row-by-column rule block-wise.





"Inner" dimensions (in red) must be partitioned in the same way, for the second line to make sense.
(Matlab example — syntax `A(1:2, 1:3)`.)

Block operations

When implementing linear algebra on a computer, usually chopping up matrices into large blocks (e.g., 64×64) gives better performance, even with an equal number of floating point operations, because of caching/locality reasons.

This is one of the reasons why libraries usually perform better than hand-coded loops.

Triangular linear systems and substitution

Idea If A is lower triangular (i.e., square with all zeros above the main diagonal), then we can solve $A\mathbf{x} = \mathbf{y}$ one entry at a time by forward-substitution.

$$\begin{bmatrix} A_{11} & 0 & 0 & 0 \\ A_{21} & A_{22} & 0 & 0 \\ A_{31} & A_{32} & A_{33} & 0 \\ A_{41} & A_{42} & A_{43} & A_{44} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} \iff \begin{cases} A_{11}x_1 = y_1 \\ A_{21}x_1 + A_{22}x_2 = y_2 \\ \vdots \end{cases}$$

Cost: $O(n^2)$ (check!)

Cheaper than computing A^{-1}, which costs $O(n^3)$.

Another instance of an important principle: never form inverses explicitly.

The same computations hold if the above quantities are blocks (careful with the order!): $x_1 = A_{11}^{-1}y$; $x_2 = A_{22}^{-1}(y_2 - A_{21}x_1), \ldots$.

Exercises

1. What is the computational cost of solving a linear system with diagonal A?
2. What is the computational cost of solving a linear system with an upper triangular matrix by back-substitution, i.e., starting from the last equation and working your way up?
3. Let $A = I + \mathbf{u}\mathbf{u}^T$, where I is the $n \times n$ identity matrix and $\mathbf{u}$ is a vector. How can one compute the product $A\mathbf{v}$ (for a vector $\mathbf{v}$) in $O(n)$ flops?

Exercises

1. Compute (by giving expressions in the blocks A_{ij}, B_{ij}) the product of two 3×3 block lower triangular matrices, i.e., of the form
$$\begin{bmatrix} A_{11} & 0 & 0 \\ A_{21} & A_{22} & 0 \\ A_{31} & A_{32} & A_{33} \end{bmatrix}$$
(all A_{ij} here are square matrices, not numbers.) Be careful with the order of the factors.
2. Simplify the expression $A^{-1}(A-B)B^{-1}(A-B)$.
3. What is the inverse of a matrix of the form $\begin{bmatrix} 0 & A \\ B & C \end{bmatrix}$ (all blocks square of the same size)? Is the product of two matrices in this form still in the same form? (Suppose all blocks are square, and help yourself with Matlab or Numpy to formulate a conjecture before diving into computations.)

References

Trefethen-Bau book, Lecture 1 (matrix-vector product).

Other exercises (also more challenging) on the Trefethen-Bau and Demmel books.