

# 1. Vectors

# Outline

Notation

Examples

Addition and scalar multiplication

Inner product

Complexity

## Vectors

- ▶ a *vector* is an ordered list of numbers
- ▶ written as

$$\begin{bmatrix} -1.1 \\ 0.0 \\ 3.6 \\ -7.2 \end{bmatrix} \quad \text{or} \quad \begin{pmatrix} -1.1 \\ 0.0 \\ 3.6 \\ -7.2 \end{pmatrix}$$

or  $(-1.1, 0, 3.6, -7.2)$

- ▶ numbers in the list are the *elements* (*entries*, *coefficients*, *components*)
- ▶ number of elements is the *size* (*dimension*, *length*) of the vector
- ▶ vector above has dimension 4; its third entry is 3.6
- ▶ vector of size  $n$  is called an *n-vector*
- ▶ numbers are called *scalars*

## Vectors via symbols

- ▶ we'll use symbols to denote vectors, e.g.,  $a$ ,  $X$ ,  $p$ ,  $\beta$ ,  $E^{\text{aut}}$
- ▶ other conventions:  $\mathbf{g}$ ,  $\vec{a}$
- ▶  $i$ th element of  $n$ -vector  $a$  is denoted  $a_i$
- ▶ if  $a$  is vector above,  $a_3 = 3.6$
- ▶ in  $a_i$ ,  $i$  is the *index*
- ▶ for an  $n$ -vector, indexes run from  $i = 1$  to  $i = n$
- ▶ *warning*: sometimes  $a_i$  refers to the  $i$ th vector in a list of vectors
- ▶ two vectors  $a$  and  $b$  of the same size are equal if  $a_i = b_i$  for all  $i$
- ▶ we overload  $=$  and write this as  $a = b$

## Block vectors

- ▶ suppose  $b$ ,  $c$ , and  $d$  are vectors with sizes  $m$ ,  $n$ ,  $p$
- ▶ the *stacked vector* or *concatenation* (of  $b$ ,  $c$ , and  $d$ ) is

$$a = \begin{bmatrix} b \\ c \\ d \end{bmatrix}$$

- ▶ also called a *block vector*, with (block) entries  $b$ ,  $c$ ,  $d$
- ▶  $a$  has size  $m + n + p$

$$a = (b_1, b_2, \dots, b_m, c_1, c_2, \dots, c_n, d_1, d_2, \dots, d_p)$$

## Zero, ones, and unit vectors

- ▶  $n$ -vector with all entries 0 is denoted  $0_n$  or just 0
- ▶  $n$ -vector with all entries 1 is denoted  $\mathbf{1}_n$  or just  $\mathbf{1}$
- ▶ a *unit vector* has one entry 1 and all others 0
- ▶ denoted  $e_i$  where  $i$  is entry that is 1
- ▶ unit vectors of length 3:

$$e_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \quad e_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \quad e_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

# Sparsity

- ▶ a vector is *sparse* if many of its entries are 0
- ▶ can be stored and manipulated efficiently on a computer
- ▶ **nnz**( $x$ ) is number of entries that are nonzero
- ▶ examples: zero vectors, unit vectors

# Outline

Notation

**Examples**

Addition and scalar multiplication

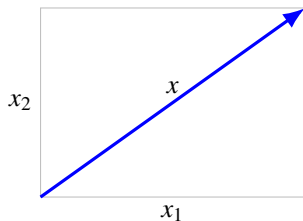
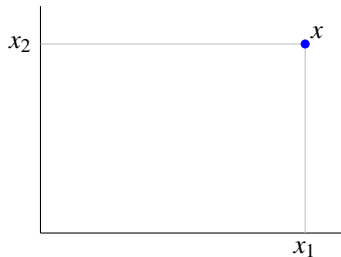
Inner product

Complexity



## Location or displacement in 2-D or 3-D

2-vector  $(x_1, x_2)$  can represent a location or a displacement in 2-D



## More examples

- ▶ color: (R,G,B)
- ▶ quantities of  $n$  different commodities (or resources), *e.g.*, bill of materials
- ▶ portfolio: entries give shares (or \$ value or fraction) held in each of  $n$  assets, with negative meaning short positions
- ▶ cash flow:  $x_i$  is payment in period  $i$  to us
- ▶ audio:  $x_i$  is the acoustic pressure at sample time  $i$  (sample times are spaced  $1/44100$  seconds apart)
- ▶ features:  $x_i$  is the value of  $i$ th *feature* or *attribute* of an entity
- ▶ customer purchase:  $x_i$  is the total \$ purchase of product  $i$  by a customer over some period
- ▶ word count:  $x_i$  is the number of times word  $i$  appears in a document

## Word count vectors

- ▶ a short document:

**Word** count vectors are used **in** computer based **document** analysis. Each entry of the **word** count vector is the **number** of times the associated dictionary **word** appears **in** the **document**.

- ▶ a small dictionary (left) and word count vector (right)

word	$\begin{bmatrix} 3 \\ 2 \\ 1 \\ 0 \\ 4 \\ 2 \end{bmatrix}$
in	
number	
horse	
the	
document	

- ▶ dictionaries used in practice are much larger

# Outline

Notation

Examples

**Addition and scalar multiplication**

Inner product

Complexity

## Vector addition

- ▶  $n$ -vectors  $a$  and  $b$  can be added, with sum denoted  $a + b$
- ▶ to get sum, add corresponding entries:

$$\begin{bmatrix} 0 \\ 7 \\ 3 \end{bmatrix} + \begin{bmatrix} 1 \\ 2 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 9 \\ 3 \end{bmatrix}$$

- ▶ subtraction is similar

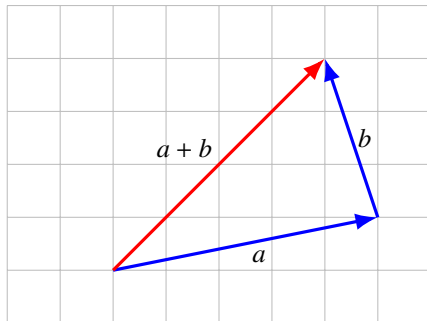
## Properties of vector addition

- ▶ *commutative*:  $a + b = b + a$
- ▶ *associative*:  $(a + b) + c = a + (b + c)$   
(so we can write both as  $a + b + c$ )
- ▶  $a + 0 = 0 + a = a$
- ▶  $a - a = 0$

these are easy and boring to verify

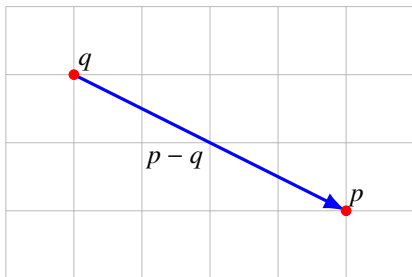
## Adding displacements

if 3-vectors  $a$  and  $b$  are displacements,  $a + b$  is the sum displacement



## Displacement from one point to another

displacement from point  $q$  to point  $p$  is  $p - q$





## Scalar-vector multiplication

- ▶ scalar  $\beta$  and  $n$ -vector  $a$  can be multiplied

$$\beta a = (\beta a_1, \dots, \beta a_n)$$

- ▶ also denoted  $a\beta$
- ▶ example:

$$(-2) \begin{bmatrix} 1 \\ 9 \\ 6 \end{bmatrix} = \begin{bmatrix} -2 \\ -18 \\ -12 \end{bmatrix}$$

## Properties of scalar-vector multiplication

- ▶ associative:  $(\beta\gamma)a = \beta(\gamma a)$
- ▶ left distributive:  $(\beta + \gamma)a = \beta a + \gamma a$
- ▶ right distributive:  $\beta(a + b) = \beta a + \beta b$

these equations look innocent, but be sure you understand them perfectly

## Linear combinations

- ▶ for vectors  $a_1, \dots, a_m$  and scalars  $\beta_1, \dots, \beta_m$ ,

$$\beta_1 a_1 + \dots + \beta_m a_m$$

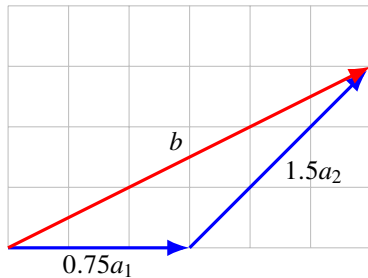
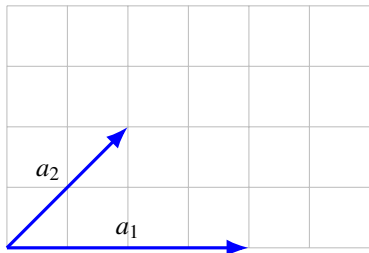
is a *linear combination* of the vectors

- ▶  $\beta_1, \dots, \beta_m$  are the *coefficients*
- ▶ a *very* important concept
- ▶ a simple identity: for any  $n$ -vector  $b$ ,

$$b = b_1 e_1 + \dots + b_n e_n$$

## Example

two vectors  $a_1$  and  $a_2$ , and linear combination  $b = 0.75a_1 + 1.5a_2$



## Replicating a cash flow

- ▶  $c_1 = (1, -1.1, 0)$  is a \$1 loan from period 1 to 2 with 10% interest
- ▶  $c_2 = (0, 1, -1.1)$  is a \$1 loan from period 2 to 3 with 10% interest
- ▶ linear combination

$$d = c_1 + 1.1c_2 = (1, 0, -1.21)$$

is a two period loan with 10% compounded interest rate

- ▶ we have *replicated* a two period loan from two one period loans

# Outline

Notation

Examples

Addition and scalar multiplication

**Inner product**

Complexity

## Inner product

- ▶ *inner product* (or *dot product*) of  $n$ -vectors  $a$  and  $b$  is

$$a^T b = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n$$

- ▶ other notation used:  $\langle a, b \rangle$ ,  $\langle a|b \rangle$ ,  $(a, b)$ ,  $a \cdot b$

- ▶ example:

$$\begin{bmatrix} -1 \\ 2 \\ 2 \end{bmatrix}^T \begin{bmatrix} 1 \\ 0 \\ -3 \end{bmatrix} = (-1)(1) + (2)(0) + (2)(-3) = -7$$

## Properties of inner product

- ▶  $a^T b = b^T a$
- ▶  $(\gamma a)^T b = \gamma(a^T b)$
- ▶  $(a + b)^T c = a^T c + b^T c$

can combine these to get, for example,

$$(a + b)^T (c + d) = a^T c + a^T d + b^T c + b^T d$$



## General examples

- ▶  $e_i^T a = a_i$  (picks out  $i$ th entry)
- ▶  $\mathbf{1}^T a = a_1 + \cdots + a_n$  (sum of entries)
- ▶  $a^T a = a_1^2 + \cdots + a_n^2$  (sum of squares of entries)

## Examples

- ▶  $w$  is weight vector,  $f$  is feature vector;  $w^T f$  is score
- ▶  $p$  is vector of prices,  $q$  is vector of quantities;  $p^T q$  is total cost
- ▶  $c$  is cash flow,  $d$  is discount vector (with interest rate  $r$ ):

$$d = (1, 1/(1+r), \dots, 1/(1+r)^{n-1})$$

$d^T c$  is net present value (NPV) of cash flow

- ▶  $s$  gives portfolio holdings (in shares),  $p$  gives asset prices;  $p^T s$  is total portfolio value

# Outline

Notation

Examples

Addition and scalar multiplication

Inner product

**Complexity**

## Flop counts

- ▶ computers store (real) numbers in *floating-point format*
- ▶ basic arithmetic operations (addition, multiplication, ...) are called *floating point operations* or flops
- ▶ complexity of an algorithm or operation: total number of flops needed, as function of the input dimension(s)
- ▶ this can be *very grossly approximated*
- ▶ crude approximation of time to execute: (flops needed)/(computer speed)
- ▶ current computers are around 1Gflop/sec ( $10^9$  flops/sec)
- ▶ but this can vary by factor of 100

## Complexity of vector addition, inner product

- ▶  $x + y$  needs  $n$  additions, so:  $n$  flops
- ▶  $x^T y$  needs  $n$  multiplications,  $n - 1$  additions so:  $2n - 1$  flops
- ▶ we simplify this to  $2n$  (or even  $n$ ) flops for  $x^T y$
- ▶ and much less when  $x$  or  $y$  is sparse