신경망 네트워크와 수학적 기반

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, β,*
- other conventions: $\vec{a}$
- i-th element of n-vector a is denoted a_i where i is the index
- for an n-vector, indexes run from $i = 1$ to $i = n$
- 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 $a = b$

Vectors

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

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

- a is 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

- zero vector : n-vector with all entries 0 is denoted 0_n or just 0
- one vector : n-vector with all entries 1 is denoted 1_n or just 1
- unit vector : a *unit vector* has one entry 1 and all others 0
- unit vector is denoted by e_i where i is entry that is 1
- unit vectors of length 3:

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

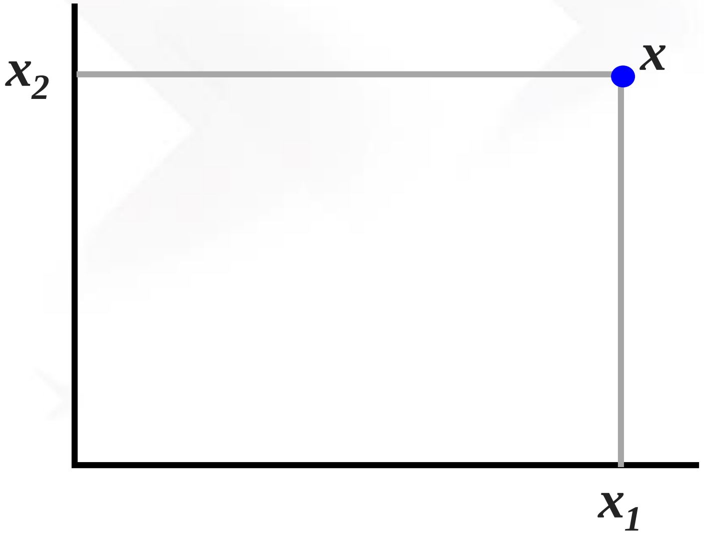
Sparsity

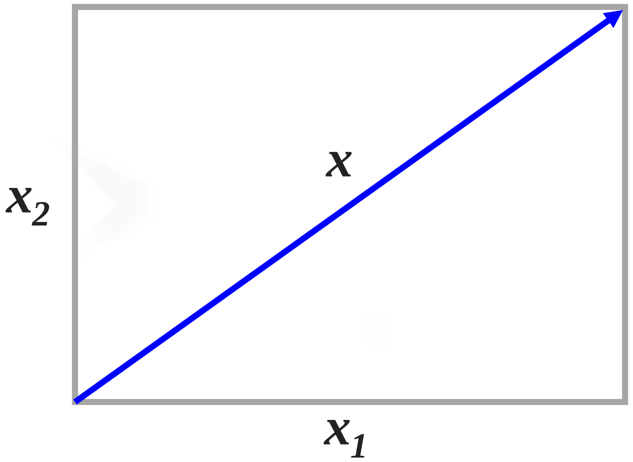
- a vector is *sparse* if many of its entries are 0
- A sparse vector can be stored and manipulated efficiently on a computer
- **nnz**(x) is number of entries that are nonzero

Vectors

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)
- audio : x_i is the acoustic pressure at sample time i
- features : x_i is the value of i-th *feature* or *attribute* of an entity
- word count : x_i is the number of times word i appears in a document

Vectors

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)

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

- dictionaries used in practice are much larger

Vectors

Vector addition

- n-vectors a and b can be added, $a + b$
- 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

Vectors

Properties of vector addition

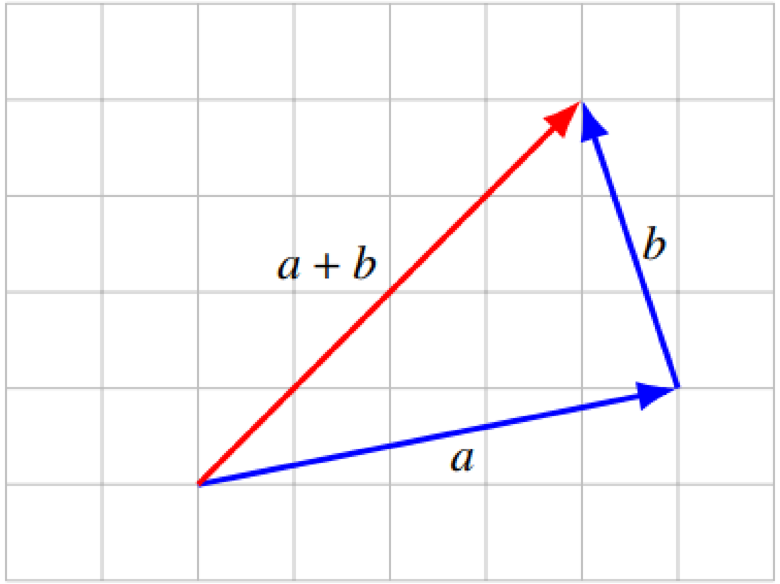
- *commutative*: $a + b = b + a$
- *associative*: $(a + b) + c = a + (b + c)$

  (so we can write both as $a + b + c$)
- $a + 0 = 0 + a = a$
- $a - a = 0$

Vectors

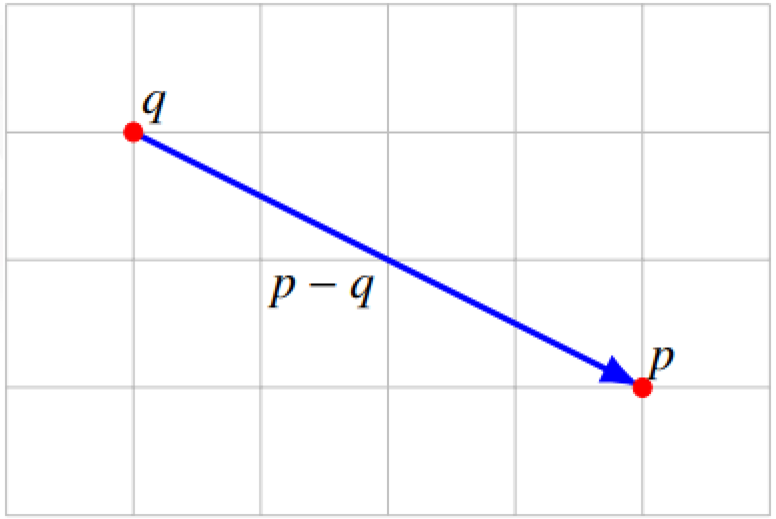
Adding displacements

- 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 $p - q$



Vectors

Scalar-vector multiplication

- scalar β and n-vector a can be multiplied

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

- 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

- Let a and b be vectors and β *and* γ be scalars
- associative: $(\beta\gamma)a = \beta(\gamma a)$
- left distributive: $(\beta + \gamma)a = \beta a + \gamma a$
- right distributive: $\beta(a + b) = \beta a + \beta b$

Linear combinations

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

$$\beta_1 a_1 + \cdots + \beta_m a_m$$

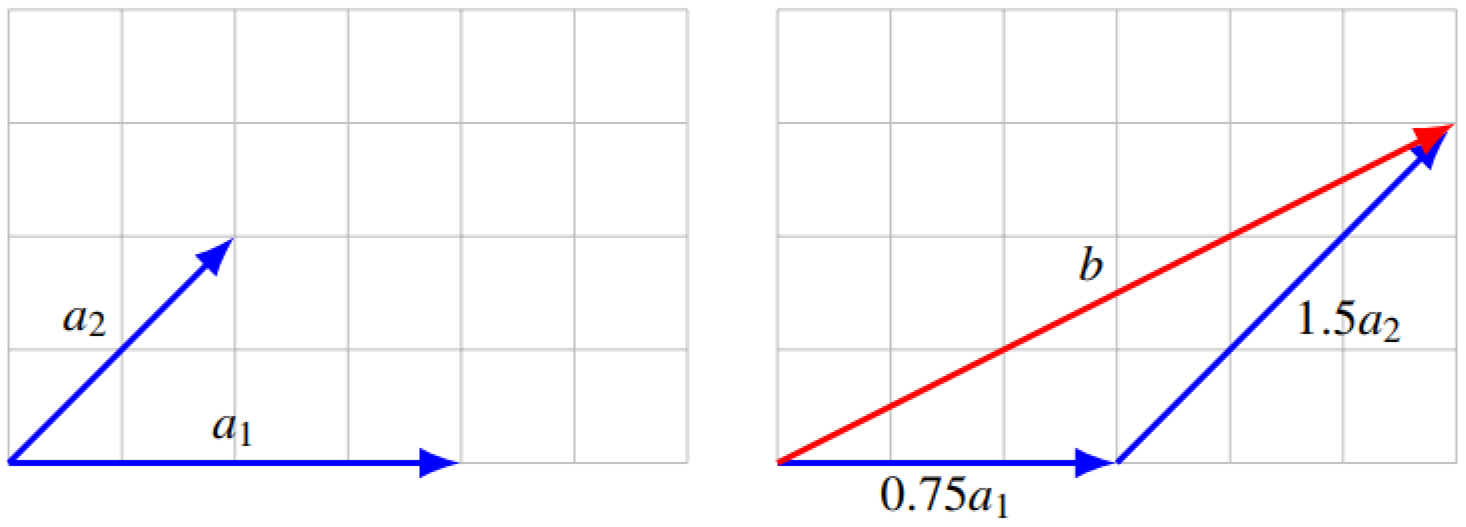
  is a ***linear combination*** of the vectors

- $\beta_1, \ldots, \beta_m$ are the coefficients
- a simple identity: for any n-vector b,

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

Vectors
Example

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



Vectors

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: (a, b), $(a|b)$, (a, b), $a \cdot b$
- example:

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

Vectors

Properties of inner product

- Let a, b and c be vectors and γ be a scalar
- $a^T b = b^T a$
- $(\gamma a)^T b = \gamma(a^T b)$
- $(a + b)^T c = a^T c + b^T c$

- can combine $(a + b)^T(c + d) = a^Tc + a^Td + b^Tc + b^Td$

Vectors

General examples

- $e_i^T a = a_i$ (picks out i-th entry)
- $1^T a = a_1 + \cdots + a_n$ (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

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)

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 $x^T y$
- flops are much less when x or y is sparse