Vector addition: The zero vector

The $D$-vector whose entries are all zero is the zero vector, written $0_D$ or just $0$

$$v + 0 = v$$
Vector addition: Vector addition is associative and commutative

- **Associativity**
  \[(x + y) + z = x + (y + z)\]

- **Commutativity**
  \[x + y = y + x\]
Vector addition: Vectors as arrows

Like complex numbers in the plane, $n$-vectors over $\mathbb{R}$ can be visualized as arrows in $\mathbb{R}^n$.

The 2-vector $[3, 1.5]$ can be represented by an arrow with its tail at the origin and its head at $(3, 1.5)$.

or, equivalently, by an arrow whose tail is at $(-2, -1)$ and whose head is at $(1, 0.5)$. 
Like complex numbers, addition of vectors over $\mathbb{R}$ can be visualized using arrows.

To add $\mathbf{u}$ and $\mathbf{v}$:
- place tail of $\mathbf{v}$’s arrow on head of $\mathbf{u}$’s arrow;
- draw a new arrow from tail of $\mathbf{u}$ to head of $\mathbf{v}$.
Scalar-vector multiplication

With complex numbers, *scaling* was multiplication by a real number $f(z) = rz$

For vectors,

- we refer to field elements as *scalars*;
- we use them to scale vectors:

$$\alpha \mathbf{v}$$

Greek letters (e.g. $\alpha, \beta, \gamma$) denote scalars.
Scalar-vector multiplication

**Definition:** Multiplying a vector \( \mathbf{v} \) by a scalar \( \alpha \) is defined as multiplying each entry of \( \mathbf{v} \) by \( \alpha \):

\[
\alpha [v_1, v_2, \ldots, v_n] = [\alpha v_1, \alpha v_2, \ldots, \alpha v_n]
\]

**Example:** \( 2 [5, 4, 10] = [2 \cdot 5, 2 \cdot 4, 2 \cdot 10] = [10, 8, 20] \)
Quiz: Suppose we represent \( n \)-vectors by \( n \)-element lists. Write a procedure `scalar_vector_mult(alpha, v)` that multiplies the vector \( v \) by the scalar \( \alpha \).

Answer:
def scalar_vector_mult(alpha, v):
    return [alpha*x for x in v]
Scalar-vector multiplication: Scaling arrows

An arrow representing the vector \([3, 1.5]\) is this:

and an arrow representing two times this vector is this:
Scalar-vector multiplication: Associativity of scalar-vector multiplication

Associativity: $\alpha(\beta \mathbf{v}) = (\alpha \beta)\mathbf{v}$
Scalar-vector multiplication: Line segments through the origin

Consider scalar multiples of \( \mathbf{v} = [3, 2] \):
\{0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}

For each value of \( \alpha \) in this set, 
\( \alpha \mathbf{v} \) is shorter than \( \mathbf{v} \) but in same direction.
Conclusion: The set of points

\[ \{ \alpha \mathbf{v} : \alpha \in \mathbb{R}, 0 \leq \alpha \leq 1 \} \]

forms the line segment between the origin and \( \mathbf{v} \)
Scalar-vector multiplication: Lines through the origin

What if we let $\alpha$ range over all real numbers?
- Scalars bigger than 1 give rise to somewhat larger copies
- Negative scalars give rise to vectors pointing in the opposite direction

The set of points

$$\{\alpha \mathbf{v} : \alpha \in \mathbb{R}\}$$

forms the line through the origin and $\mathbf{v}$.
Combining vector addition and scalar multiplication

We want to describe the set of points forming an arbitrary line segment (not necessarily through the origin).

Idea: Use translation.

Start with line segment from \([0, 0]\) to \([3, 2]\):

\[
\{ \alpha [3, 2] : 0 \leq \alpha \leq 1 \}
\]

Translate it by adding \([0.5, 1]\) to every point:

\[
\{ [0.5, 1] + \alpha [3, 2] : 0 \leq \alpha \leq 1 \}
\]

Get line segment from \([0, 0] + [0.5, 1]\) to \([3, 2] + [0.5, 1]\)
Combining vector addition and scalar multiplication: Distributive laws for scalar-vector multiplication and vector addition

Scalar-vector multiplication distributes over vector addition:

\[ \alpha(u + v) = \alpha u + \alpha v \]

Example:

- On the one hand,
  
  \[ 2 \left( \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} + \begin{bmatrix} 3 \\ 4 \\ 4 \end{bmatrix} \right) = 2 \begin{bmatrix} 4 \\ 6 \\ 7 \end{bmatrix} = \begin{bmatrix} 8 \\ 12 \\ 14 \end{bmatrix} \]

- On the other hand,
  
  \[ 2 \left( \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} + \begin{bmatrix} 3 \\ 4 \\ 4 \end{bmatrix} \right) = 2 \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} + 2 \begin{bmatrix} 3 \\ 4 \\ 4 \end{bmatrix} = \begin{bmatrix} 2 \\ 4 \\ 6 \end{bmatrix} + \begin{bmatrix} 6 \\ 8 \\ 8 \end{bmatrix} = \begin{bmatrix} 8 \\ 12 \\ 14 \end{bmatrix} \]
Combining vector addition and scalar multiplication: First look at convex combinations

Set of points making up the [0.5, 1]-to-[3.5, 3] segment:

\[
\{ \alpha [3, 2] + [0.5, 1] : \alpha \in \mathbb{R}, 0 \leq \alpha \leq 1 \}
\]

Not symmetric with respect to endpoints 😞

Use distributivity:

\[
\alpha [3, 2] + [0.5, 1] = \alpha ([3.5, 3] - [0.5, 1]) + [0.5, 1]
\]

\[
= \alpha [3.5, 3] - \alpha [0.5, 1] + [0.5, 1]
\]

\[
= \alpha [3.5, 3] + (1 - \alpha) [0.5, 1]
\]

\[
= \alpha [3.5, 3] + \beta [0.5, 1]
\]

where \( \beta = 1 - \alpha \)

New formulation:

\[
\{ \alpha [3.5, 3] + \beta [0.5, 1] : \alpha, \beta \in \mathbb{R}, \alpha, \beta \geq 0, \alpha + \beta = 1 \}
\]

Symmetric with respect to endpoints 😊
Combining vector addition and scalar multiplication: First look at convex combinations

New formulation:

\[ \{ \alpha [3.5, 3] + \beta [0.5, 1] : \alpha, \beta \in \mathbb{R}, \alpha, \beta \geq 0, \alpha + \beta = 1 \} \]

Symmetric with respect to endpoints 😊

An expression of the form

\[ \alpha \mathbf{u} + \beta \mathbf{v} \]

where \(0 \leq \alpha \leq 1, 0 \leq \beta \leq 1, \) and \(\alpha + \beta = 1\) is called a convex combination of \(\mathbf{u}\) and \(\mathbf{v}\).

The \(\mathbf{u}\)-to-\(\mathbf{v}\) line segment consists of the set of convex combinations of \(\mathbf{u}\) and \(\mathbf{v}\).
Combining vector addition and scalar multiplication: First look at convex combinations

\[ \mathbf{u} = \quad \text{and} \quad \mathbf{v} = \]

Use scalars \( \alpha = \frac{1}{2} \) and \( \beta = \frac{1}{2} \):

\[ \frac{1}{2} \mathbf{u} + \frac{1}{2} \mathbf{v} = \]

“Line segment” between two faces:

\[ 1\mathbf{u} + 0\mathbf{v} \quad \frac{7}{8}\mathbf{u} + \frac{1}{8}\mathbf{v} \quad \frac{6}{8}\mathbf{u} + \frac{2}{8}\mathbf{v} \quad \frac{5}{8}\mathbf{u} + \frac{3}{8}\mathbf{v} \quad \frac{4}{8}\mathbf{u} + \frac{4}{8}\mathbf{v} \quad \frac{3}{8}\mathbf{u} + \frac{5}{8}\mathbf{v} \quad \frac{2}{8}\mathbf{u} + \frac{6}{8}\mathbf{v} \quad \frac{1}{8}\mathbf{u} + \frac{7}{8}\mathbf{v} \quad 0\mathbf{u} + 1\mathbf{v} \]
Combining vector addition and scalar multiplication: First look at convex combinations
Line segments not necessarily through the origin

How to write the (infinite) line through $[0.5, 1]$ and $[3.5, 3]$?

Start with the line through the origin and $[3, 2]$, and translate it by adding $[0.5, 1]$ to each point.

The untranslated line is $\{\alpha [3, 2] : \alpha \in \mathbb{R}\}$.

so the translated line is $\{[0.5, 1] + \alpha [3, 2] : \alpha \in \mathbb{R}\}$
Combining vector addition and scalar multiplication: First look at affine combinations

Infinite line through $[0.5, 1]$ and $[3.5, 3]$?

Our formulation so far 😞

$$\{[0.5, 1] + \alpha [3, 2] : \alpha \in \mathbb{R}\}$$

Nicer formulation 😊:

$$\{\alpha [3.5, 3] + \beta [0.5, 1] : \alpha \in \mathbb{R}, \beta \in \mathbb{R}, \alpha + \beta = 1\}$$

An expression of the form $\alpha \mathbf{u} + \beta \mathbf{v}$ where $\alpha + \beta = 1$ is called an affine combination of $\mathbf{u}$ and $\mathbf{v}$.

The line through $\mathbf{u}$ and $\mathbf{v}$ consists of the set of affine combinations of $\mathbf{u}$ and $\mathbf{v}$. 
Vectors over $GF(2)$

Addition of vectors over $GF(2)$:

\[
\begin{array}{cccccc}
1 & 1 & 1 & 1 & 1 & 1 \\
+ & 1 & 0 & 1 & 0 & 1 \\
\hline
0 & 1 & 0 & 1 & 0 & 0
\end{array}
\]

For brevity, in doing $GF(2)$, we often write 1101 instead of [1,1,0,1].

**Example:** Over $GF(2)$, what is 1101 + 0111?

**Answer:** 1010
Vectors over $GF(2)$: Perfect secrecy

Represent encryption of $n$ bits by addition of $n$-vectors over $GF(2)$.

**Example:**
Alice and Bob agree on the following 10-vector as a key:

$$k = [0, 1, 1, 0, 1, 0, 0, 0, 1]$$

Alice wants to send this message to Bob:

$$p = [0, 0, 0, 1, 1, 1, 0, 1, 0, 1]$$

She encrypts it by adding $p$ to $k$:

$$c = k + p = [0, 1, 1, 0, 1, 0, 0, 0, 1] + [0, 0, 0, 1, 1, 1, 0, 1, 0, 1] = [0, 1, 1, 1, 0, 1, 0, 1, 0, 0]$$

When Bob receives $c$, he decrypts it by adding $k$:

$$c + k = [0, 1, 1, 1, 0, 1, 0, 1, 0, 0] + [0, 1, 1, 0, 1, 0, 0, 0, 1, 1] = [0, 0, 0, 1, 1, 1, 0, 1, 0, 1]$$

which is the original message.
Vectors over $GF(2)$: Perfect secrecy

If the key is chosen according to the uniform distribution, encryption by addition of vectors over $GF(2)$ achieves *perfect secrecy*. For each plaintext $p$, the function that maps the key to the cyphertext

$$k \mapsto k + p$$

is invertible.

Since the key $k$ has the uniform distribution, the cyphertext $c$ also has the uniform distribution.
Vectors over $GF(2)$: All-or-nothing secret-sharing using $GF(2)$

- I have a secret: the midterm exam.
- I’ve represented it as an $n$-vector $\mathbf{v}$ over $GF(2)$.
- I want to provide it to my TAs Alice and Bob (A and B) so they can administer the midterm while I take vacation.
- One TA might be bribed by a student into giving out the exam ahead of time, so I don’t want to simply provide each TA with the exam.

**Idea:** Provide pieces to the TAs:
- the two TAs can jointly reconstruct the secret, but
- neither of the TAs all alone gains any information whatsoever.

**Here’s how:**
- I choose a random $n$-vector $\mathbf{v}_A$ over $GF(2)$ randomly according to the uniform distribution.
- I then compute

\[ \mathbf{v}_B := \mathbf{v} - \mathbf{v}_A \]

- I provide Alice with $\mathbf{v}_A$ and Bob with $\mathbf{v}_B$, and I leave for vacation.
Vectors over $GF(2)$: All-or-nothing secret-sharing using $GF(2)$

- What can Alice learn without Bob?
  - All she receives is a random $n$-vector.

- What about Bob?
  - He receives the output of $f(x) = v - x$ where the input is random and uniform.
  - Since $f(x)$ is invertible, the output is also random and uniform.
Vectors over $GF(2)$: All-or-nothing secret-sharing using $GF(2)$

Too simple to be useful, right? RSA just introduced a product based on this idea:

>> Split each password into two parts.
>> Store the two parts on two separate servers.
Vectors over $GF(2)$: *Lights Out*

- **input**: Configuration of lights
- **output**: Which buttons to press in order to turn off all lights?

**Computational Problem:** Solve an instance of *Lights Out*

Represent state using $\text{range}(5) \times \text{range}(5)$-vector over $GF(2)$.

**Example state vector:**

Represent each button as a vector (with ones in positions that the button toggles)

**Example button vector:**
Vectors over $GF(2)$: *Lights Out*

Look at $3 \times 3$ case.

- State $\begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$ move $\begin{bmatrix} 0 & 1 & 0 \end{bmatrix}$ results in new state $\begin{bmatrix} 0 & 1 & 1 \end{bmatrix}$
- State $\begin{bmatrix} 1 & 1 & 0 \end{bmatrix}$ move $\begin{bmatrix} 1 & 1 & 1 \end{bmatrix}$ results in new state $\begin{bmatrix} 0 & 0 & 1 \end{bmatrix}$
- State $\begin{bmatrix} 0 & 0 & 0 \end{bmatrix}$ move $\begin{bmatrix} 1 & 0 & 1 \end{bmatrix}$ results in new state $\begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$
Vectors over $GF(2)$: $3 \times 3$ Lights Out button vectors

Computational Problem: Which sequence of button vectors plus $s$ sums to $0$?

$\implies$ Which sequence of button vectors sum to $s$?
Vectors over $GF(2)$: *Lights Out*

**Computational Problem:** Which sequence of button vectors sums to $s$?

**Observations:**
- By commutative property of vector addition, order doesn’t matter.
- A button vector occurring twice cancels out.

Replace Computational Problem with: **Which set of button vectors sums to $s$?**
Vectors over $GF(2)$: *Lights Out*

Replace our original Computational Problem with a more general one:

Solve an instance of *Lights Out* $\Rightarrow$ Which set of button vectors sum to $s$?

$\Rightarrow$ Find subset of $GF(2)$ vectors $\mathbf{v}_1, \ldots, \mathbf{v}_n$ whose sum equals $s$.
Vectors over $GF(2)$: *Lights Out*

Button vectors for $2 \times 2$ version:

\[
\begin{array}{cccc}
\bullet & \bullet & \bullet & \bullet \\
\bullet & \bullet & \bullet & \bullet \\
\end{array}
\]

where the black dots represent ones.

**Quiz:** Find the subset of the button vectors whose sum is

\[
\begin{array}{ccc}
\bullet & \bullet & \\
\bullet & \bullet & \\
\end{array}
\]

**Answer:**

\[
\begin{array}{ccc}
\bullet & \bullet & \\
\bullet & \bullet & \\
\end{array} = \begin{array}{ccc}
\bullet & \bullet & \\
\bullet & \bullet & \\
\end{array} + \begin{array}{ccc}
\bullet & \bullet & \\
\bullet & \bullet & \\
\end{array}
\]

Dot-product of two $D$-vectors is sum of product of corresponding entries:

$$\mathbf{u} \cdot \mathbf{v} = \sum_{k \in D} \mathbf{u}[k] \mathbf{v}[k]$$

**Example:** For traditional vectors $\mathbf{u} = [u_1, \ldots, u_n]$ and $\mathbf{v} = [v_1, \ldots, v_n]$, 

$$\mathbf{u} \cdot \mathbf{v} = u_1 v_1 + u_2 v_2 + \cdots + u_n v_n$$

Output is a scalar, not a vector.

Dot-product sometimes called *scalar product.*
Example: Dot-product of \([1, 1, 1, 1, 1]\) and \([10, 20, 0, 40, -100]\):

\[
1 \quad 1 \quad 1 \quad 1 \quad 1
\]
\[
\cdot \quad 10 \quad 20 \quad 0 \quad 40 \quad -100
\]
\[
10 \quad + \quad 20 \quad + \quad 0 \quad + \quad 40 \quad + \quad (-100) \quad = \quad -30
\]
Dot-product: Total cost or benefit

Suppose $D$ consists of four main ingredients of beer:

$$D = \{\text{malt, hops, yeast, water}\}$$

A cost vector maps each food to a price per unit amount:

$$\text{cost} = \{\text{hops :$2.50/ounce}, \text{malt :$1.50/pound}, \text{water :$0.06/gallon}, \text{yeast :$0.45/g}\}$$

A quantity vector maps each food to an amount (e.g. measured in pounds).

$$\text{quantity} = \{\text{hops:6 oz}, \text{malt:14 pounds}, \text{water:7 gallons}, \text{yeast:11 grams}\}$$

The total cost is the dot-product of cost with quantity:

$$\text{cost} \cdot \text{quantity} = 2.50 \cdot 6 + 1.50 \cdot 14 + 0.06 \cdot 7 + 0.45 \cdot 11 = 40.992$$

A value vector maps each food to its caloric content per pound:

$$\text{value} = \{\text{hops : 0}, \text{malt : 960}, \text{water : 0}, \text{yeast : 3.25}\}$$

The total calories represented by a pint is the dot-product of value with quantity:

$$\text{value} \cdot \text{quantity} = 0 \cdot 6 + 960 \cdot 14 + 0 \cdot 7 + 3.25 \cdot 11 = 13475.75$$
Dot-product: Linear equations

**Example:** A sensor node consists of hardware components, e.g.

- CPU
- radio
- temperature sensor
- memory

Battery-driven and remotely located so we care about energy usage.

Suppose we know the power consumption for each hardware component. Represent it as a $D$-vector with $D = \{\text{radio}, \text{sensor}, \text{memory}, \text{CPU}\}$

$$\text{rate} = \text{Vec}(D, \{\text{memory} : 0.06W, \text{radio} : 0.06W, \text{sensor} : 0.004W, \text{CPU} : 0.0025W\})$$

Have a test period during which we know how long each component was working. Represent as another $D$ vector:

$$\text{duration} = \text{Vec}(D, \{\text{memory} : 1.0s, \text{radio} : 0.2s, \text{sensor} : 0.5s, \text{CPU} : 1.0s\})$$

Total energy consumed (in Joules): $\text{duration} \cdot \text{rate}$