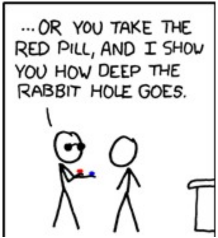
[2] The Vector

What is a vector?

- This is a 4-vector over R:

$$[3.14159, 2.718281828, -1.0, 2.0]$$

- We will often use Python's lists to represent vectors.
- Set of all 4-vectors over R is written R^4.
- This notation might remind you of the notation R^D : the set of functions from D to R.



Vectors are functions

Think of our 4-vector $[3.14159, 2.718281828, -1.0, 2.0]$ as the function

$$0 \mapsto 3.14159, \quad 1 \mapsto 2.718281828, \quad 2 \mapsto -1.0, \quad 3 \mapsto 2.0$$

$\mathbb{F}^d$ is notation for set of functions from $\{0, 1, 2, \ldots, d-1\}$ to $\mathbb{F}$.

Example: $GF(2)^5$ is set of 5-element bit sequences, e.g. [0,0,0,0,0], [0,0,0,0,1], ...

Let WORDS = set of all English words

In information retrieval, a document is represented ("bag of words" model) by a function $f : WORDS \longrightarrow \mathbb{R}$ specifying, for each word, how many times it appears in the document.

We would refer to such a function as a WORDS-vector over $\mathbb{R}$

Definition: For a field $\mathbb{F}$ and a set D, a *D-vector over* $\mathbb{F}$ is a function from D to $\mathbb{F}$.
The set of such functions is written $\mathbb{F}^D$

For example, $\mathbb{R}^{WORDS}$

Representation of vectors using Python dictionaries

We often use Python's dictionaries to represent such functions, e.g.

```
{0:3.14159, 1:2.718281828, 2:-1.0, 3:2.0}
```

What about representing a WORDS-vector over R?

For any single document, most words are *not* represented. They should be mapped to zero.

Our convention for representing vectors by dictionaries: we are allowed to omit key-value pairs when value is zero.

Example: "The rain in Spain falls mainly on the plain" would be represented by the dictionary

```
{'on': 1, 'Spain': 1, 'in': 1, 'plain': 1, 'the': 2,
 'mainly': 1, 'rain': 1, 'falls': 1}
```

Sparsity

A vector most of whose values are zero is called a *sparse* vector.
If no more than k of the entries are nonzero, we say the vector is *k-sparse*.

A k-sparse vector can be represented using space proportional to k.

Example: when we represent a corpus of documents by WORD-vectors, the storage required is proportional to the total number of words in all documents.

Most signals acquired via physical sensors (images, sound, ...) are not exactly sparse.

Later we study *lossy compression*: making them sparse while preserving perceptual similarity.

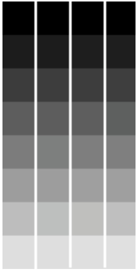
What can we represent with a vector?

- Document (for information retrieval)
- Binary string (for cryptography/information theory)
- Collection of attributes
  - Senate voting record
  - demographic record of a consumer
  - characteristics of cancer cells
- State of a system
  - Population distribution in the world
  - number of copies of a virus in a computer network
  - state of a pseudorandom generator
  - state of *Lights Out*
- Probability distribution, e.g. {`1:1/6, 2:1/6, 3:1/6, 4:1/6, 5:1/6, 6:1/6`}

What can we represent with a vector?

- Image

```
{(0,0) 0,      (0,1): 0,     (0,2): 0,     (0,3): 0,
      :
(1,0): 32,     (1,1): 32,    (1,2): 32,    (1,3): 32,
(2,0): 64,     (2,1): 64,    (2,2): 64,    (2,3): 64,
(3,0): 96,     (3,1): 96,    (3,2): 96,    (3,3): 96,
(4,0): 128,    (4,1): 128,   (4,2): 128,   (4,3): 128,
(5,0): 160,    (5,1): 160,   (5,2): 160,   (5,3): 160,
(6,0): 192,    (6,1): 192,   (6,2): 192,   (6,3): 192,
(7,0): 224,    (7,1): 224,   (7,2): 224,   (7,3): 224 }
```



What can we represent with a vector?

- Points
  - Can interpret the 2-vector $[x, y]$ as a point in the plane.
  - Can interpret 3-vectors as points in space, and so on.

Vector addition: Translation and vector addition

With complex numbers, translation achieved by adding a complex number, e.g. $f(z) = z + (1 + 2\mathbf{i})$

Let's do the same thing with vectors...

Definition of vector addition:

$$[u_1, u_2, \ldots, u_n] + [v_1, v_2, \ldots, v_n] = [u_1 + v_1, u_2 + v_2, \ldots, u_n + v_n]$$

For 2-vectors represented in Python as 2-element lists, addition procedure is

```
def add2(v,w): return [v[0]+w[0], v[1]+w[1]]
```

Vector addition: Translation and vector addition

Quiz: Suppose we represent n-vectors by n-element lists. Write a procedure `addn(v, w)` to compute the sum of two vectors so represented.

Answer:

```
def addn(v, w): return [v[i]+w[i] for i in range(len(v))]
```

Vector addition: The zero vector

The D-vector whose entries are all zero is the *zero vector*, written $\mathbf{0}_D$ or just $\mathbf{0}$

$$\mathbf{v} + \mathbf{0} = \mathbf{v}$$

Vector addition: Vector addition is associative and commutative

- *Associativity*

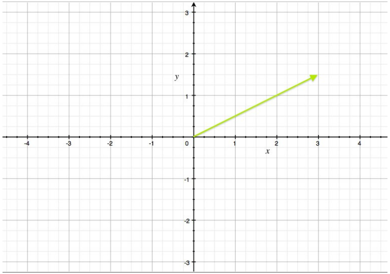
$$(\mathbf{x} + \mathbf{y}) + \mathbf{z} = \mathbf{x} + (\mathbf{y} + \mathbf{z})$$

- *Commutativity*

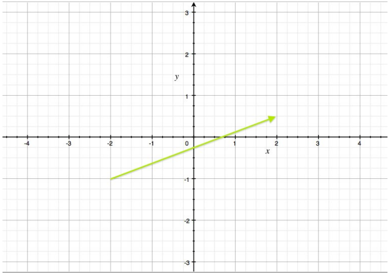
$$\mathbf{x} + \mathbf{y} = \mathbf{y} + \mathbf{x}$$

Vector addition: Vectors as arrows

Like complex numbers in the plane, n-vectors over R can be visualized as *arrows* in 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).



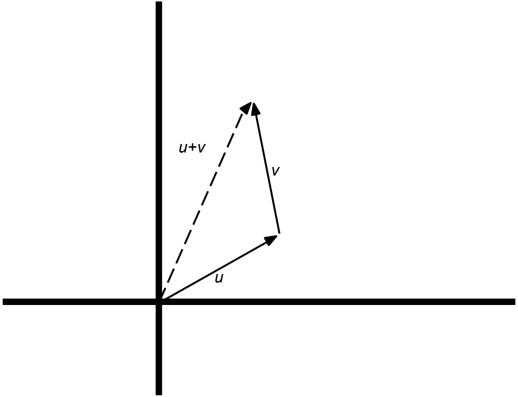
or, equivalently, by an arrow whose tail is at $(-2, -1)$ and whose head is at $(1, 0.5)$.

Vector addition: Vectors as arrows

Like complex numbers, addition of vectors over R can be visualized using arrows.

To add **u** and **v**:

- place tail of **v**'s arrow on head of **u**'s arrow;
- draw a new arrow from tail of **u** to head of **v**.



The vector

[2]Vectors: Multiplication

Scalar-vector multiplication

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

For vectors,

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

$$\alpha \mathbf{v}$$

Greek letters (e.g. α, β, γ) denote scalars.

Scalar-vector multiplication

Definition: Multiplying a vector **v** by a scalar α is defined as multiplying each entry of **v** by α:

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

Example: $2\,[5, 4, 10] = [2 \cdot 5, 2 \cdot 4, 2 \cdot 10] = [10, 8, 20]$

Scalar-vector multiplication

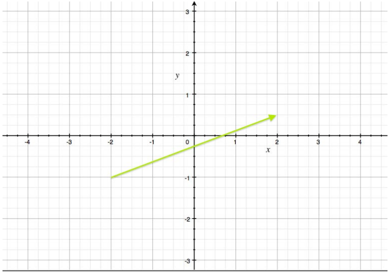
Quiz: Suppose we represent *n*-vectors by *n*-element lists. Write a procedure `scalar_vector_mult(alpha, v)` that multiplies the vector `v` by the scalar `alpha`.

Answer:

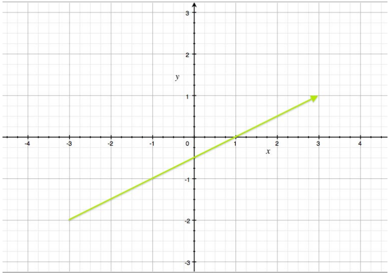
```
def scalar_vector_mult(alpha, v): return [alpha*x for x in v]
```

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:



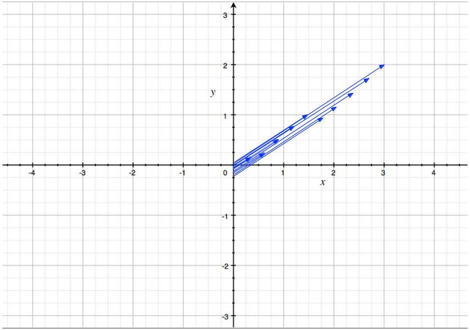
Scalar-vector multiplication: Associativity of scalar-vector multiplication

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 α in this set,
$\alpha\,\mathbf{v}$ is shorter than $\mathbf{v}$ but in same direction.



Scalar-vector multiplication: Line segments through the origin

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 α range over all real numbers?

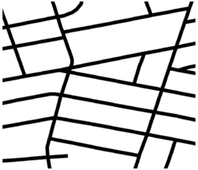
- Scalars bigger than 1 give rise to somewhat larger copies
- Negative scalars give rise to vectors pointing in the opposite direction

The set of points

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

forms the line through the origin and **v**

Combining vector addition and scalar multiplication

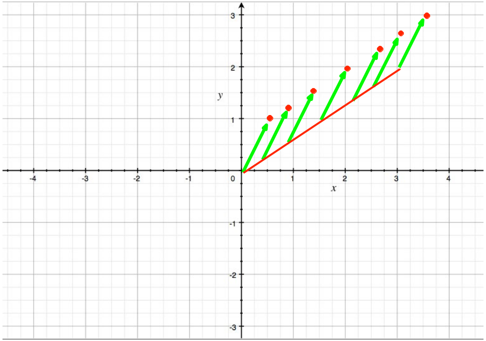


We want to describe the set of points forming an arbitrary line segment (not necessarily through the origin).

Idea: Use the idea of 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:

$$\alpha(\mathbf{u} + \mathbf{v}) = \alpha\mathbf{u} + \alpha\mathbf{v}$$

Example:

- On the one hand,

$$2\,([1, 2, 3] + [3, 4, 4]) = 2\,[4, 6, 7] = [8, 12, 14]$$

- On the other hand,

$$2\,([1, 2, 3] + [3, 4, 4]) = 2\,[1, 2, 3] + 2\,[3, 4, 4] = [2, 4, 6] + [6, 8, 8] = [8, 12, 14]$$

Combining vector addition and scalar multiplication: First look at convex combinations

Set of points making up the 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 Q

Use distributivity:

$$\begin{aligned}
\alpha\,[3, 2] + [0.5, 1] &= \alpha\,([3.5, 3] - [0.5, 1]) + [0.5, 1] \\
&= \alpha\,[3.5, 3] - \alpha\,[0.5, 1] + [0.5, 1] \\
&= \alpha\,[3.5, 3] + (1 - \alpha)\,[0.5, 1] \\
&= \alpha\,[3.5, 3] + \beta\,[0.5, 1]
\end{aligned}$$

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 $\mathbb{Q}$

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 **u** and **v**

The **u**-to-**v** line segment consists of the set of convex combinations of **u** and **v**.

Combining vector addition and scalar multiplication: First look at convex combinations

$\mathbf{u} =$  and $\mathbf{v} =$ 

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

$\frac{1}{2}$  $+$ $\frac{1}{2}$  $=$ 

"Line segment" between two faces:


$1\mathbf{u} + 0\mathbf{v}$


$\frac{7}{8}\mathbf{u} + \frac{1}{8}\mathbf{v}$


$\frac{6}{8}\mathbf{u} + \frac{2}{8}\mathbf{v}$


$\frac{5}{8}\mathbf{u} + \frac{3}{8}\mathbf{v}$


$\frac{4}{8}\mathbf{u} + \frac{4}{8}\mathbf{v}$


$\frac{3}{8}\mathbf{u} + \frac{5}{8}\mathbf{v}$


$\frac{2}{8}\mathbf{u} + \frac{6}{8}\mathbf{v}$


$\frac{1}{8}\mathbf{u} + \frac{7}{8}\mathbf{v}$


$1\mathbf{u} + 0\mathbf{v}$

Combining vector addition and scalar multiplication: First look at convex combinations

Combining vector addition and scalar multiplication: First look at convex combinations

Combining vector addition and scalar multiplication: First look at convex combinations

Combining vector addition and scalar multiplication: First look at convex combinations

Combining vector addition and scalar multiplication: First look at convex combinations

Combining vector addition and scalar multiplication: First look at convex combinations

Combining vector addition and scalar multiplication: First look at convex combinations

Combining vector addition and scalar multiplication: First look at convex combinations

Combining vector addition and scalar multiplication: First look at convex combinations

Combining vector addition and scalar multiplication: First look at convex combinations

Combining vector addition and scalar multiplication: First look at convex combinations

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 $\mathbb{Q}$

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

Nicer formulation $\mathbb{Q}$:

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

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

The line through **u** and **v** consists of the set of affine combinations of **u** and **v**.

Vectors over *GF* (2)

Addition of vectors over *GF* (2):

$$\begin{array}{rccccc} & 1 & 1 & 1 & 1 & 1 \\ + & 1 & 0 & 1 & 0 & 1 \\ \hline & 0 & 1 & 0 & 1 & 0 \end{array}$$

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:

$$\mathbf{k} = [0, 1, 1, 0, 1, 0, 0, 0, 0, 1]$$

Alice wants to send this message to Bob:

$$\mathbf{p} = [0, 0, 0, 1, 1, 1, 0, 1, 0, 1]$$

She encrypts it by adding $\mathbf{p}$ to $\mathbf{k}$:

$$\mathbf{c} = \mathbf{k} + \mathbf{p} = [0, 1, 1, 0, 1, 0, 0, 0, 0, 1] + [0, 0, 0, 1, 1, 1, 0, 1, 0, 1] = [0, 1, 1, 1, 0, 1, 0, 1, 0, 0]$$

When Bob receives $\mathbf{c}$, he decrypts it by adding $\mathbf{k}$:

$$\mathbf{c} + \mathbf{k} = [0, 1, 1, 1, 0, 1, 0, 1, 0, 0] + [0, 1, 1, 0, 1, 0, 0, 0, 0, 1] = [0, 0, 0, 1, 1, 1, 0, 1, 0, 1]$$

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

$$\mathbf{k} \mapsto \mathbf{k} + \mathbf{p}$$

is invertible

Since the key **k** has the uniform distribution,
the cyphertext **c** also has the uniform distribution.