Vectors in Julia

Keegan Go    David Zeng    Stephen Boyd

EE103
Stanford University

August 24, 2014
Vectors in Julia

main topics:

- how to create and manipulate vectors in Julia
- how Julia notation differs from math notation
Scalars

- represented by two types, Int64 and Float64
  
a = 1
  
b = 0.5

- usually the types work together correctly, for example
  
1 + 0.5

produces a float
Vectors

Vector operations

Norm and distance
Vectors

- vectors are represented by arrays in Julia
- to create the 3-vector

\[ x = (8, -4, 3.5) = \begin{bmatrix} 8 \\ -4 \\ 3.5 \end{bmatrix} \]

use

\[ x = [8, -4, 3.5] \]

\( (x = [8; -4; 3.5] \text{ also works}) \)

- watch out for similar looking expressions
  - \((8,-4,3.5)\) and \(\{8,-4,3.5\}\) mean something else
  - \([8 \ -4 \ 3.5]\) is a row vector (later)

- length of an array: \texttt{length}(x)
Indexing and slicing

- indexes run from 1 to $n$: $x_2$ is $x[2]$
- can also set an element, e.g., $x[3] = 10.5$
- use a range to select more than one element
- $x[2:3]$ selects the second and third elements
- to select every other element use $x[1:2:end]$
Block vectors

to form a stacked vector like

\[ a = (b, c) = \begin{bmatrix} b \\ c \end{bmatrix} \]

(with \( b \) and \( c \) vectors)

\( a = [b, c] \)

(a = \([b; c]\) also works)

can mix vectors and scalars:

\( a = [b, 2, c, -6] \)
Basic functions for arrays

- sum of (the entries of) a vector: \texttt{sum}(x)
- mean of the entries (\texttt{avg}(x)): \texttt{mean}(x)
- \(0_n\) is \texttt{zeros}(n)
- \(1_n\) is \texttt{ones}(n)
Creating unit vectors

- form $e_3$ with length 10
- create a zero vector of size 10 then set the third element to 1
  
  ```
  e_3 = zeros(10); e_3[3] = 1;
  ```
Julia array types

- an array's type is the most specific given its elements
- consider \(\text{arr1} = [100, 7, -83]\) and \(\text{arr2} = [4.5, -10, 13]\)
- \(\text{arr1}\) is an Int array while \(\text{arr2}\) is a Float array
- \(\text{arr1}[2] = 0.1\) will error because \(\text{arr1}\) can only store Ints
- to make \(\text{arr1}\) a Float array, give one entry a decimal point

\(\text{arr1} = [100., 7, -83]\)
List of vectors

- to form a list with vectors a, b, and c:
  \[
  \text{vector\_list} = \{a, b, c\}
  \]
- the second vector in this list is \text{vector\_list}[2]
- to access an element in a vector: \text{vector\_list}[2][3]
do not mix mathematical notation with Julia notation
notations are not compatible, for example
\[ v = (0, 1, 1) \]
produces a tuple, not an array (vector)
similarly,
\[ v = [1, 10, 7] \]
defines an array (vector) in Julia, but isn’t mathematically correct
Outline

Vectors

Vector operations

Norm and distance
Vector addition and subtraction

- vector addition uses $+$, for example
  \[
  \begin{bmatrix}
  1 \\
  2 \\
  3 \\
  \end{bmatrix}
  +
  \begin{bmatrix}
  4 \\
  5 \\
  6 \\
  \end{bmatrix}
  \]

  is written
  \[
  [1, 2, 3] + [4, 5, 6]
  \]

- subtraction uses $-$
- the arrays must have the same length (unless one is scalar)
Scalar-vector addition

- In Julia, a scalar and a vector can be added.
- The scalar is added to each entry of the vector.

\[ [2, 4, 8] + 3 \]

Gives (in mathematical notation)

\[
\begin{bmatrix}
2 \\
4 \\
8 \\
\end{bmatrix}
\]

\[ + 31 = \begin{bmatrix}
5 \\
7 \\
11 \\
\end{bmatrix}
\]
Scalar-vector multiplication

- scalar-vector multiplication uses *
- for example,

\[ (-2) \begin{bmatrix} 1 \\ 9 \\ 6 \end{bmatrix} \]

is written

\[ -2 * [1, 9, 6] \]

- the other order gives the same result:

\[ [1, 9, 6] * -2 \]
Inner product

- inner product $a^T b$ is written as `dot(a, b)` which returns a scalar (Int or Float)
- $a$ and $b$ must have the same length
Outline

Vectors

Vector operations

Norm and distance
Norm and distance

- The norm $\|x\| = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2}$ is written $\text{norm}(x)$.
- $\text{dist}(x, y) = \|x - y\|$ is written $\text{norm}(x-y)$. 
RMS value

- \textbf{rms}(x) \text{ is defined as}

\[
\text{rms}(x) = \sqrt{\frac{1}{n} (x_1^2 + \cdots + x_n^2)} = \frac{\|x\|}{\sqrt{n}}.
\]

- can be expressed as

\[
\text{rms}_x = \text{norm}(x) / \sqrt{\text{length}(x)}
\]
standard deviation is defined as

$$\text{std}(x) = \frac{\|x - \text{avg}(x)1\|}{\sqrt{n}}$$

which can be expressed as

$$\text{std}_{of\_x} = \frac{\text{norm}(x - \text{mean}(x))}{\sqrt{\text{length}(x)}}$$

warning: the Julia function \text{std} uses the slightly different definition

$$\text{std}(x) = \frac{\|x - \text{avg}(x)1\|}{\sqrt{n - 1}}$$
**Angle**

- the angle between two vectors $a$ and $b$ is
  \[
  \angle(a, b) = \arccos \left( \frac{a^T b}{\|a\| \|b\|} \right)
  \]

- can be expressed as
  \[
  \text{angle}_a_b = \cos(\text{dot}(a, b)/(\text{norm}(a)*\text{norm}(b)))
  \]
Nearest neighbor example

# Compares vectors in vector_list against a_vector
# and returns the index of the one which is closest
function nearest_neighbor(vector_list, a_vector)
    closest_distance = Inf
    closest_index = 0
    for i in 1:length(vector_list)
        ith_distance = norm(vector_list[i] - a_vector)
        if (ith_distance < closest_distance)
            closest_distance = ith_distance
            closest_index = i
        end
    end
    return closest_index
end