Matrices in Julia

David Zeng   Keegan Go   Stephen Boyd

EE263
Stanford University

October 1, 2015

Outline

Matrices

Matrix operations

Matrices

- matrices in Julia are repersented by 2D arrays
- to create the 2×3 matrix

  $$A = \left[\begin{array}{ccc} 2 & -4 & 8.2 \\ -5.5 & 3.5 & 63 \end{array} \right]$$

  use

  ```
  A = [2 -4 8.2; -5.5 3.5 63]
  ```

- semicolons delimit rows; spaces delimit entries in a row
- `size(A)` returns the size of `A` as a pair, *i.e.*,

  ```
  A_rows, A_cols = size(A) # or
  A_size = size(A)
  # A_rows is A_size[1], A_cols is A_size[2]
  ```

- row vectors are $1 \times n$ matrices, *e.g.*, `[4 8.7 -9]`

Indexing and slicing

- A_{13} is found with `A[1,3]`
- ranges can also be used: `A[2,1:2:end]`
- `:` selects all elements along that dimension
  - `A[:,3]` selects the third column
  - `A[2,:]` selects the second row
  - `A[:,end:-1:1]` reverses the order of columns
- `A[:]` returns the columns of A stacked as a vector, *i.e.*, if

  `A = [2 7; 8 1]`

  then `A[:]` returns

  `[2, 8, 7, 1]`

Block matrices

- the block matrix
$$X = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$$
(with `A`, `B`, `C`, and `D` matrices) is formed with
`X = [A B; C D]`
- all matrices in a row must have the same height
- the total number of columns in each row be consistent
(c.f. standard math notation, in which A and C must have the same number of columns)

Common matrices

- $\mathbf{0}_{m\times n}$ is `zeros(m,n)`
- $m \times n$ matrix with all entries 1 is `ones(m,n)`
- $I_{n\times n}$ is `eye(n)`
- $\mathbf{diag}(x)$ is `diagm(x)` (where x is a vector)
- random $m \times n$ matrix with entries from standard normal distribution: `randn(m,n)`
- random $m \times n$ matrix with entries from uniform distribution on $[0, 1]$: `rand(m,n)`

Outline

Matrices

Matrix operations

Transpose and matrix addition

- A^T is written `A'` (single quote mark)
- $+$/- are overloaded for matrix addition/substraction
- for example,

$$\begin{bmatrix} 4.0 & 7 \\ -10.6 & 89.8 \end{bmatrix} + \begin{bmatrix} 19 & -34.7 \\ 20 & 1 \end{bmatrix}$$

  is written

  `[4.0 7; -10.6 89.8] + [19 -34.7; 20 1]`

  matrices must have the same size (unless one is a scalar)

Matrix-scalar operations

- all matrix-scalar operations (+,-,*) apply elementwise
- for example, matrix-scalar addition:

  ```
  [1 2; 3 4] + 10
  ```

  gives

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} + 10 \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 11 & 12 \\ 13 & 14 \end{bmatrix}$$

- scalar-multiplication:

  ```
  [1 2; 3 4] * 10
  ```

  gives

$$10 \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} = \begin{bmatrix} 10 & 20 \\ 30 & 40 \end{bmatrix}$$

Matrix-vector multiplication

- the * operator is used for matrix-vector multiplication
- for example,

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \begin{bmatrix} 5 \\ 6 \end{bmatrix}$$

  is written

  ```
  [1 2; 3 4] * [5, 6]
  ```

- for vectors x and y, `x'*y` finds their inner product
  - unlike `dot(x,y)`, `x'*y` returns a 1×1 array, not a scalar

Matrix multiplication

- * is overloaded for matrix-matrix multiplication:

$$\begin{bmatrix} 2 & 4 & 3 \\ 3 & 1 & 5 \end{bmatrix} \begin{bmatrix} 3 & 10 \\ 4 & 2 \\ 1 & 7 \end{bmatrix}$$

  is written

  `[2 4 3; 3 1 5] * [3 10; 4 2; 1 7]`

- A^k is `A^k` for square matrix A and nonnegative integer k

Other functions

- sum of entries of a matrix: `sum(A)`
- average of entries of a matrix: `mean(A)`
- `max(A,B)` and `min(A,B)` finds the element-wise $\max$ and $\min$ respectively
  - the arguments must have the same size unless one is a scalar
- `maximum(A)` and `minimum(A)` return the largest and smallest entries of `A`
- `norm(A)` is not what you might think
  - to find $\left(\sum_{i,j} A_{ij}^2\right)^{1/2}$ use `norm(A[:])` or `vecnorm(A)`

Computing regression model RMS error

the math:

- X is an $n \times N$ matrix whose N columns are feature n-vectors
- y is the N-vector of associated outcomes
- regression model is $\hat{y} = X^T\beta + v$ (β is n-vector, v is scalar)
- RMS error is $\mathbf{rms}(\hat{y} - y)$

in Julia:

```
y_hat = X'*beta + v
rms_error = norm(y_hat-y)/sqrt(length(y))
```