Matrices in Julia

Reese Pathak David Zeng Keegan Go Stephen Boyd

EE103
Stanford University

November 4, 2017

Matrices

- matrices in Julia are repersented by 2D arrays
- `[2 -4 8.2; -5.5 3.5 63]` creates the 2×3 matrix

$$A = \begin{bmatrix} 2 & -4 & 8.2 \\ -5.5 & 3.5 & 63 \end{bmatrix}$$

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

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

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

Indexing and slicing

- A_{ij} is found with `A[i,j]`
- can use ranges: `A[1:2,1:3]` is 2×3 submatrix or slice $A_{1:2,1:3}$
- `:` selects all elements along that dimension
  - `A[:,3]` is third column
  - `A[2,:]` is second row
- `A[:]` stacks the columns of A as a vector (column-major order)
- `A'[:]` stacks the rows of A as a vector (row-major order)

Block matrices

- block matrix
$$X = \left[\begin{array}{cc} A & B \\ C & D \end{array} \right]$$
(with `A`, `B`, `C`, and `D` matrices) is formed with
`X = [A B; C D]`
- usual rules governing dimensions of A, B, C, and D apply

Useful matrices in Julia

- $\mathbf{0}_{m\times n}$ 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)

Transpose and matrix addition

- A^T is written `A'` (single quote mark)
- $+$/- are used for matrix addition/substraction (matrices must have the same size)
- 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]`

Matrix-scalar operations

- matrix-scalar operations (`+`,`-`,`*`,`\`) apply elementwise
- scalar-matrix multiplication:

  `10 * [1 2; 3 4]`

  gives

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

  (scalar can also appear on right of matrix)
- matrix-scalar addition:

  `[1 2; 3 4] + 10`

  gives

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

  (which is not standard mathematical notation)

Matrix-vector multiplication

- * 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]
  ```

Matrix multiplication

- `*` is also used 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)

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
- `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))
```