A Quick Introduction to MATLAB/Octave

Kenny Marino, Nupur Chatterji

Basics

- MATLAB (and it's free cousin Octave) is an interpreted language
  - Two basic kinds of files
    - Scripts
    - Functions
- MATLAB is optimized for matrix and vector operations
  - All numeric data types are actually matrices
    - Stored as a 2d array
    - Vectors are a special case of matrices that contain one column or row
    - Scalars are matrices with only one row and column
  - Writing for loops will make code much slower!

Basic Mathematical Operations

- Assignment
  - a = b
  - a = 5
  - a = [1, 2, 3]
- Addition
  - a + b
- Subtraction
  - a - b

More Mathematical Operations

- Warning - these operations behave differently if a and b are matrices
- Using .* and ./ will force these operations to be element-wise

- Multiplication
  - a * b or a .*b
- Division
  - a / b or a ./ b
- Power
  - a^b or a.^b

Creating matrices, vectors

- Crucial since exploiting vectorization (instead of loops) is the crux of Matlab/Octave

Creating a row vector

```
>> x = [1, 0, 5]

x =

     1     0     5
```

Creating a column vector

```
>> x = [1; 0; 5]

x =

    1

    0

    5
```

- Semicolon (;) acts as a placeholder for vertical concatenation

Creating a matrix

Can think of it as rows stacked on each other

```
>> x = [1, 0, 1; 2, 4, 5; 1, 4, 5]

x =

     1     0     1

     2     4     5

     1     4     5
```

Indexing operations

- Indexing starts at 1
- Parentheses are used for indexing (rather than the usual square brackets)
- Vectors are indexed by x(index)
- Matrices are indexed by x(row, column)
    - (Note: indexing matrices with a single index will do what is called “vectorizing” the matrix. Will basically flatten and then index)
- Multiple rows or columns can be indexed with the colon operator

Indexing examples

```
>> x = [1, 2, 3];

>> x(1)

ans = 1

>> x(3)

ans = 3
```

Indexing examples

```
>> x = [1, 0, 1; 2, 4, 5; 1, 4, 5]

>> x(2, 1)

ans = 2

>> x(3, 2)

ans = 4
```

The colon operator

- Syntax is start_ind:step:end_ind
- Can also do start_ind:end_ind, which will default step to 1
- Can use “end” keyword to refer to the last index along that dimension

Examples

```
>> x = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];

>> x(1:5)

ans = [1, 2, 3, 4, 5]

>> x(1:2:8)

ans = [1, 3, 5, 7]

>> x(8:end)

ans = [8, 9, 10]
```

Examples

```
>> x = [1, 2, 3; 4, 5, 6; 7, 8, 9];

>> x(:, 1)

ans = [1; 4; 7]

>> x(2, :)

ans = [4, 5, 6]

>> x(2:3, 1:2:end)

ans = [4, 6; 7, 9]
```

Concatenating matrices

- Simply concatenate with commas and/or semicolons
- Dimensions need to be correct

Examples

```
>> [[1, 2, 3], [2, 3]]

ans =

    1     2     3     2     3

>> [[2, 3]; [1, 2]]

ans =

    2     3

    1     2
```

Some useful matrix functions

- Create matrix of all ones
  - X = ones(r, c)
- Create matrix of all zeros
  - X= zeros(r, c)
- Create diagonal matrix with vector of diagonal values
  - A = diag(x)
- Get size of matrix
  - [r, c] = size(A)
- Sum along rows
  - y = sum(A)
- Create identity matrix
  - A = eye(len)

More useful matrix functions

- Matrix transpose
  - X = A’
- Invert a matrix
  - X = inv(A)
- Get pseudo inverse
  - X = pinv(A)
- Get the determinant
  - d = det(A)

Matrix mathematical operations

- Matrix operations built into MATLAB
- Matrix multiplication done with * operator
  - If you want element-wise operation, use .* operator

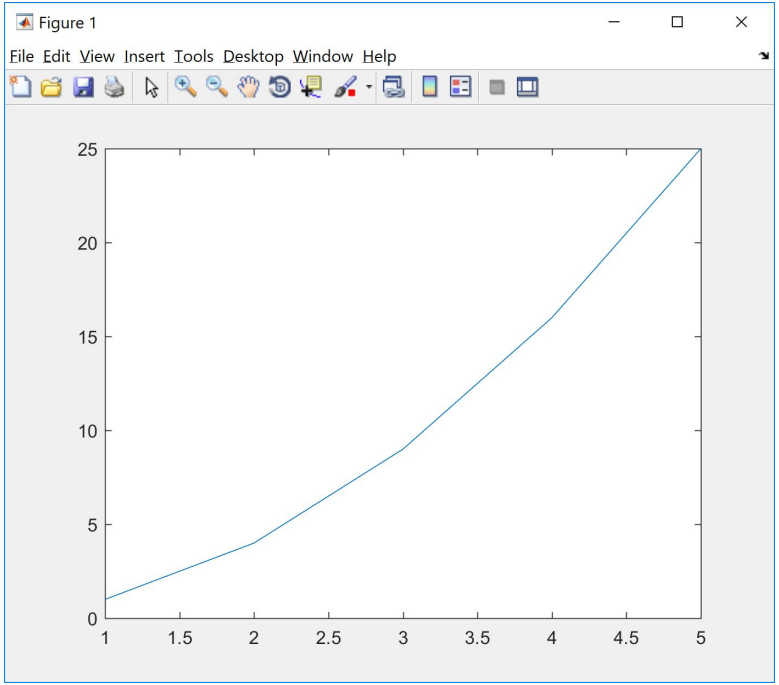
```
>> X = [2, 0, 0; 0, 2, 0; 0, 0, 2]

>> b = [2; 4; 5]

>> X * b

ans =

    4

    8

    10
```

Plotting

- Can make simple plot with plot(X, Y)

```
>> plot([1, 2, 3, 4, 5], [1, 4, 9, 16, 25])
```

- See matlab documentation for more plotting options



Scripts

- All end in .m extension
- Ending statements with ; will suppress the output
  - Otherwise every line will print to the console
- Comments begin with % character
  - Use mod function to do modulus operations

Functions

function [o1, o2, … on] = foobar(i1, i2, ...in)

Line1;

Line2;

…

LineN;

end

Conditional operations and Loops

```
while(condition)
    Line1;
    …
end
```

Conditional operations and Loops

```
for x = 1:10 % x is assigned to 1, then 2, then 3 ...
        Line1;
        …
end
```

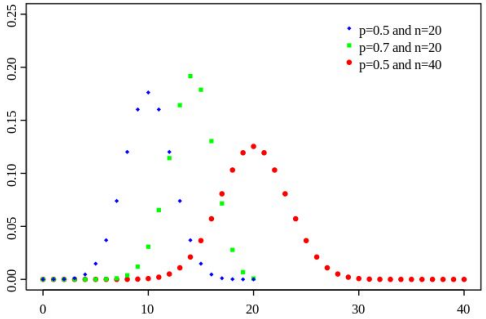
More MATLAB/Octave Resources

1. First of all, you can download Octave here.
2. http://www.cyclismo.org/tutorial/matlab/index.html
3. http://www.mathworks.com/help/pdf_doc/matlab/getstart.pdf
4. http://www.csc.kth.se/utbildning/kth/kurser/DN2255/ndiff13/matopt.pdf
5. https://class.coursera.org/ml-005/lecture/preview (Part V is an Octave tutorial)

Distributions

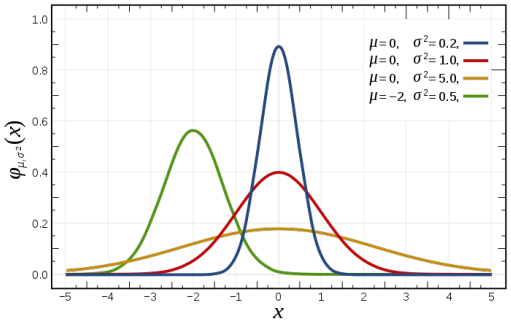
- Univariate
  - Binomial

$$\binom{n}{k} p^k (1-p)^{n-k}$$



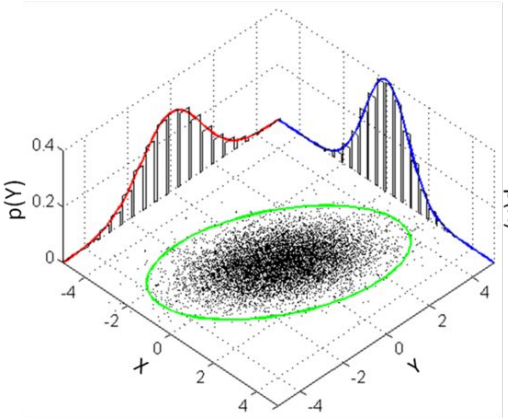
  - Normal

$$\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$



- Multivariate
  - 2-d Gaussian Distribution

$$f_{\mathbf{X}}(x_1, \ldots, x_k) = \frac{\exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right)}{\sqrt{(2\pi)^k|\boldsymbol{\Sigma}|}}$$



Programming Question

```
% Part a

mu = [0, 0];

sigma = eye(2);

r = mvnrnd(mu,sigma,100);

plot(r(:,1),r(:,2),'o');
```

```
% Part b

mu = [-1, 1];

sigma = eye(2);

r = mvnrnd(mu, sigma, 100);

plot(r(:,1),r(:,2),'o');
```

```
% Part c

mu = [-1, 1];

sigma = 2 * sigma;

r = mvnrnd(mu, sigma, 100);

plot(r(:,1),r(:,2),'o');
```

```
% Part d

mu = [-1, 1];

sigma = [1, 0.5; 0.5, 1];

r = mvnrnd(mu, sigma, 100);

plot(r(:,1),r(:,2),'o')
```

```
% Part e

mu = [-1, 1];

sigma = [1, -0.5; -0.5, 1];

r = mvnrnd(mu, sigma, 100);

plot(r(:,1),r(:,2),'o');
```