Introduction to MATLAB

CS 229 MACHINE LEARNING SESSION

4/15/2016

- MATLAB is recommended but not required for this class
- Alternatives are Python, R, Julia, Java
- How to get MATLAB (GUI, Corn)
- What version of MATLAB

Helpful Links

- http://www.mathworks.com/help/matlab/index.html
- http://cs229.stanford.edu/materials.html
- https://web.stanford.edu/group/farmshare/cgi-bin/wiki/index.php/MATLAB

Today's Agenda

- Overview the fundamentals of MATLAB

- Basic Operations
- Vectors and Matrices
- Useful Functions
- Flow Control
- Plotting
- Data Input and Output

Basic Operations

- 5 + 6
- 3 - 2
- 5 * 8
- 1 / 2 % 0.5
- 2 ^ 6
- 1 == 2 % false
- 1 ~= 2 % true
- 1 && 0
- 1 || 0
- xor(1, 0)
- i, j % imaginary number

- a = 3; % semicolon suppresses output
- a = 'hello';

- who
- whos % name, size, bytes, class, attributes

- clear % clear specified variable or all

- **help** roots
- **doc** roots

Vectors and Matrices

- V = [1 2 3]
- V' = [1; 2; 3] % transpose
- V = [1 : 0.1 : 2] % from 1 to 2, with a step size of 0.1
- V = 1 : 6 % from 1 to 6, with a default step size of 1
- A = [1 2; 3 4; 5 6]

- B = ones(2, 3)
- B = zeros(2, 3)
- B = nan(2, 3)
- B = eye(3)
- B = rand(1, 3) % uniform distribution
- B = randn(1,3) % normal distribution (mean = 0, var = 1)

Vectors and Matrices - Continued

- A = [1 2; 3 4; 5 6]
- sz = size(A)
- size(A, 1) % number of rows
- size(A, 2) % number of columns
- length(A) % size of the longest dimension
- numel(v) % number of elements

- A(3, 2) % (row, column), 1-based
- A(2, :) % get second row
- A(:, 2) % get second column
- A(1, end) % first row, last element
- A(end, :) % last row

Vectors and Matrices - Continued

- A * B % matrix multiplication, matrices must be compatible
- A .* B % element-wise multiplication, matrices must have same dimensions
- A ^ 2 % A * A
- A .^ 2
- 1 ./ A

Cell

- // n * n square cell
- C = cell(n);

- // cell of size sz1 * sz2 * … * szN
- C = cell(sz1, sz2, … szN);

- Cell{1, 2, 3} = [];

```
close all; clear all; clc;
numImg = 100;
images = cell(1, numImg);
for i = 1 : numImg
    images{i} = Imread(sprintf('image%d', i));
end
save('images.mat', 'images');

% Some time later ...
numImg = 100;
load images;
for i = 1 : numImg
    image = images{i};
    % do something on image
end
```

Useful Functions

- log() % natural logarithm, element-wise operation
- exp() % exponential
- abs()
- max() min() % returns [value, index]
- find() % A = [2 3 4]; find(A < 3);
- sum(B, 1) % sum columns (default)
- sum(B, 2) % sum rows
- inv() % inverse
- pinv() % psedoinverse, inv(A'*A)*A'
- reshape(A, [2 3])
- tic toc

Control Flow

```
sum = 0;
for i = 1 : 100
    i
    sum = sum + i;
    if (i == 99)
        break;
    elseif(i == 98)
        continue;
    else
        continue;
    end
end
sum
% Same as sum(1 : 99)
```

```
A = 1 : 100;
i = 1;
sum = 0;
while (i <= numel(A))
    sum = sum + A(i);
    i = i + 1;
end
sum
% Same as sum(1 : 100)
```

Prefer Matrix Operation over For-Loop

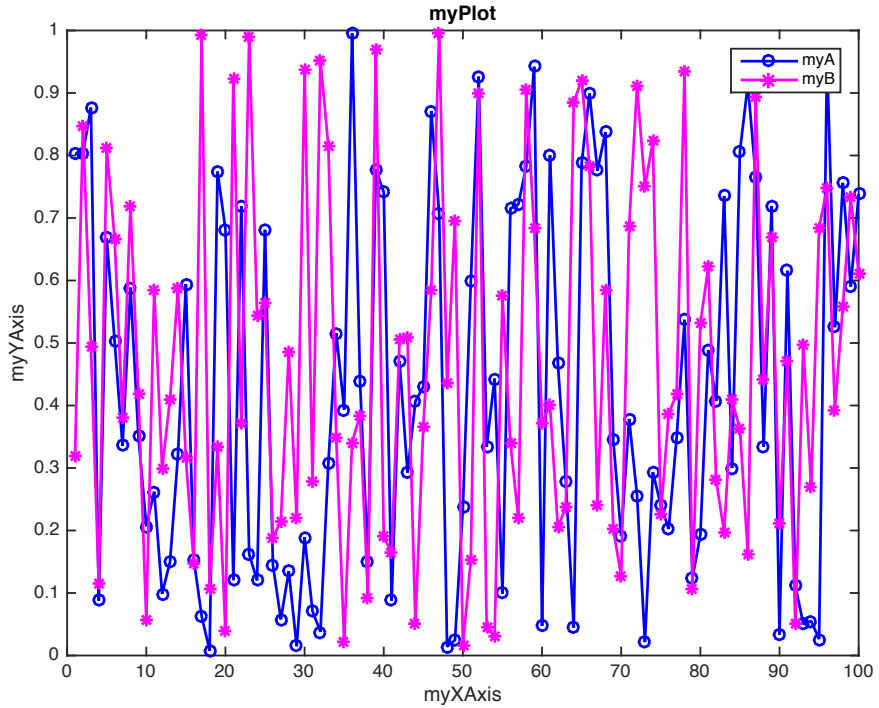
- find()
- ones()
- zeros()
- sum()
- ……
- Softmax Regression

$$\phi_i = \frac{e^{\eta_i}}{\sum_{j=1}^{k} e^{\eta_j}}$$

- https://www.quora.com/What-are-good-ways-to-write-matlab-code-in-matrix-way

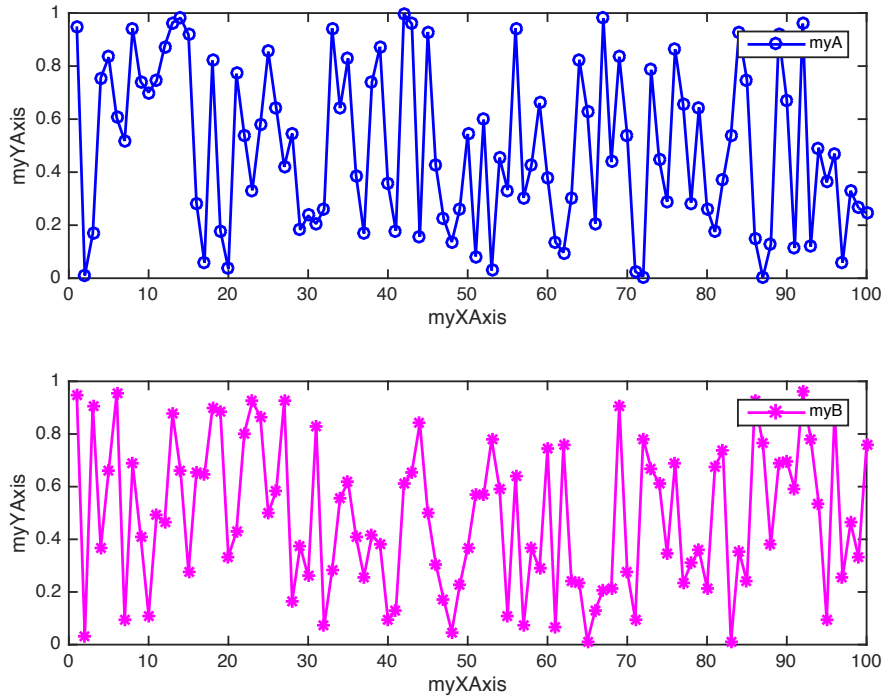
Plotting

```
close all; clear all; clc;
A = 1 : 100;
B = rand(1, 100);
C = rand(1, 100);
figure();
plot(A, B, 'b-o','linewidth', 1.5);
hold on;
plot(A, C, 'm-*', 'linewidth', 1.5);
xlabel('myXAxis'); ylabel('myYAxis');
legend('myA', 'myB');
title('myPlot');
saveas(gcf, 'myPlot', 'epsc');
```



Plotting – subplot

```
close all; clear all; clc;
A = 1 : 100;
B = rand(1, 100);
C = rand(1, 100);
figure();
subplot(2, 1, 1);
plot(A, B, 'b-o','linewidth', 1.5);
xlabel('myXAxis'); ylabel('myYAxis');
legend('myA');
subplot(2, 1, 2);
plot(A, C, 'm-*', 'linewidth', 1.5);
xlabel('myXAxis'); ylabel('myYAxis');
legend('myB');
saveas(gcf, 'myPlot', 'epsc');
```



Plotting – other plot functions

- plot()
- plot3()
- scatter()
- scatter3()
- loglog()
- semilogx()
- semilogy()
- histogram()

- http://www.mathworks.com/help/matlab/ref/plot.html

Data Input and Output

- save('myWorkspace') % save the whole workspace
- save('myA', 'A') % save the specified variable
- load('myWorkspace')
- load('myA')

- csvread() % read a comma-separated value file into a matrix
- dlmread() % read an ASCII-delimited numeric data file into a matrix
- textscan() % manual input processing

Data Input and Output – Continued

- csvwrite() % write numeric data in a matrix into file as comma-separated values
- dlmwrite() % write numeric data in a matrix to an ASCII format file
- fprintf() % manual output processing
- saveas(gcf, ‘myPlot’, ‘epsc’)

Output to Command Window

- fprintf()
- e.g. fprintf('I scored %d in %s!\n', 100, 'CS 229')
- I scored 100 in CS 229!

- disp()

Common Bugs

- Improper Matrix Operation (A .* B vs A * B)
- Access Incorrect Vector / Matrix Element (1-based)
- Overwrite Iteration Variable
- Gradient Ascent v.s. Gradient Descent

```
for i = 1 : 100
    % ....................
    % ....................
    % Calculate Derivatives
    for j = 1 : 50
        for i = 1 : 50
            % Do Something
        end
    end
    % ....................
    % ....................
    % Calculate Cost
    for j = 1 : 50
        for i = 1 : 50
            % Do Something
        end
    end
    % ....................
    % ....................
end
```

Useful References

- http://www.mathworks.com/help/matlab/index.html
- http://cs229.stanford.edu/materials.html
- sigmoid.m, logistic_grad_ascent.m, matlab_session.m

- Load the data ➔ Process the data ➔ Gradient Descent / Ascent ➔ Plot the Data