Introduction to MATLAB

CS 229 Machine Learning Session

10/14/2016
MATLAB is mandatory for class assignments.
Alternatives for projects include Python, R, Julia, Java, C++.
How to get MATLAB (GUI, Corn)
What version of MATLAB

Helpful Links
- https://web.stanford.edu/group/farmshare/cgi-bin/wiki/index.php/MATLAB
Use Matlab at Stanford

- Linux, OSX: ssh + X11 forwarding (need XQuartz on OSX)
- Windows: putty + Xming, cygwin...

- ssh -X sunetid@corn
- Type *password and two-step authentification*
- module load matlab
- matlab

- [http://www.geo.mtu.edu/geoschem/docs/putty_install.html](http://www.geo.mtu.edu/geoschem/docs/putty_install.html)
- [http://petrkout.com/linux/x11-forwarding-over-cygwin-on-windows/](http://petrkout.com/linux/x11-forwarding-over-cygwin-on-windows/)
Octave

- Open-source alternative to Matlab (also available on corn)
- Similar syntax, similar function names
- Octave has a more flexible syntax, so beware of code compatibility
  - "abc" vs. 'abc'
  - A**2 vs. A^2
  - a != b vs. a~=b
- Matlab has a few advantages (speed, debugger, support, community, toolboxes, plotting)

Today’s Agenda

- Overview the fundamentals of MATLAB
- Basic Operations
- Vectors and Matrices
- Useful Functions
- Flow Control
- Plotting
- Data Input and Output
Basic Operations

- $5 + 6$
- $3 - 2$
- $5 \times 8$
- $1 / 2 \% 0.5$
- $2 ^ 6$
- $1 == 2 \%$ false
- $1 ~= 2 \%$ true
- $1 && 0 \%$
- $1 || 0 \%$
- xor(1, 0)
- i, j, 1i, 1j \% imaginary number
- pi \% predefined value

- a = 3, \% semicolon suppresses output
- b = ‘hello’, b(1)
- c = 3>=1,

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

- clear \% clear specified variable or all

- help roots
- doc roots
Vectors and Matrices

- \( V = [1 \ 2 \ 3] \)
- \( V', \ V.' \) % conjugate transpose and 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
- \( V = \text{linspace}(1, 6, 10) \) % from 1 to 6, with 10 elements total spaced linearly
- \( A = [1 \ 2; \ 3 \ 4; \ 5 \ 6] \)

- \( B = \text{ones}(2, 3) \)
- \( B = \text{zeros}(2, 3) \)
- \( B = \text{nan}(2, 3) \)
- \( B = \text{eye}(3) \)
- \( B = \text{rand}(1, 3) \) % 1x3 random matrix, uniform distribution on \([0,1]\)
- \( B = \text{randn}(1,3) \) % 1x3 random matrix, normal distribution \(\text{N}(0,1)\)
Vectors and Matrices - Continued

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

- $A(3, 2)$ % (row, column), 1-based
- $A(2, :)$ % get second row
- $A(:, 2)$ % get second column
- $A(1, \text{end})$ % first row, last element
- $A(\text{end}, :)$ % last row
- $A(2:end,:)$ % get all but first row
- $A(:)$ % returns all the elements of $A$ as column
Vectors and Matrices - Continued

- \( A \times B \)  % matrix multiplication, matrices must be compatible
- \( A \cdot \times B \)  % element-wise multiplication, matrices must have same dimensions
- \( A^2 \)  % \( A \times A \)
- \( A^\wedge 2 \)
- \( 1 \cdot / A \)

Advanced:
- \( A / B \)  % multiplication by pseudo-inverse of \( B \), matrices must be compatible
- \( A \backslash B \)  % multiplication by pseudo-inverse of \( A \), matrices must be compatible
- \( A \wedge B \)  % different from \( A \& \& B \), \( A \) and \( B \) can be matrices of same dimensions
- \( A \mid B \)  % different from \( A \| \| B \), \( A \) and \( B \) can be matrices of same dimensions
- // n * n square cell
- C = cell(n);

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

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

- Cell{1:2} vs. Cell(1:2)

```matlab
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()` % pseudoinverse, `inv(A'*A)*A'`
- `reshape(A, [2 3])`
- `tic` `toc`
- WARNING: don’t overwrite function names.
Control Flow

```plaintext
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

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

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');
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()

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

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

- sigmoid.m, logistic_grad_ascent.m, matlab_session.m

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

- Matlab is pretty well documented => use that ("help", "doc" or online)

- Matlab has a large community => Google your questions!

- If you need something you feel is relatively common for your project, someone has probably needed the same code and published it
  › Google it!
  › Check on https://www.mathworks.com/matlabcentral/fileexchange