Octave Tutorial

Machine Learning – WS 12/13

Umer Khan

Information Systems and Machine Learning Lab (ISMLL)

University of Hildesheim, Germany

Basic Commands

- Try Elementary arithmetic operations: 5+6, 3-2, 5*8, 1/2, 2^6 etc …
- Logical Operations: 1==2 % false, 1 ~= 2, 1 && 0 % AND, 1 || 0 % OR, xor(1,0)
- To change your octave prompt: PS1('>> ');
- Octave Variables: >> a=3;

```
>> b = 'hi';
>> b      % print the value of b
>> disp(b);  % will print 'hi'
>> a=pi;
>> disp(sprintf('2 decimals: %0.2f', a))  %  2 decimals: 3.14
```

- Matrices and Vectors

```
>> A = [1 2; 3 4; 5 6]     % prints a 3x2 matrix
    1   2
    3   4
    5   6                  % ; marks the next row
>> v = [1  2  3]           % a 1x3 row vector
>> v = [1; 2; 3]           % a 3x1 column vector
>> v = 1:0.1:2             % assigns 'v' a row vector with values starting from
                             1, incrementing by 0.1 up-till 2
>> v = 1:6                 % v = 1  2  3  4  5  6
>> ones(2,3)               % generates a 2x3 matrix of 1
>> C = 2*ones(2,3)
>> w = zeros(1,3)
>> w = rand(1,3)           % generates 1x3 matrix of random numbers from uniform
                             distribution between 0 and 1. Use 'randn' to get random numbers from
                             Gaussian distribution.
>>  w = -6 + sqrt(10)*(randn(1,10000));
>> hist(w)                 % octave creates a histogram and show in a new window.
>> hist (w, 50)            % increase the number of bins to 50
>> I = eye (4)             % generates a 4x4 identity matrix.
```

```
>> help                    % for getting help on any command
```

Moving Data Around

```
>>  size (A)          % gives you size of a matrix like 3     2
>> b = size (A)       % creates a 1x2 matrix with values  3   2
>> size(A, 1)         % size of dimension 1 i.e. 3
>> length(A)          % returns size of longest dimension. length is usually applied to vectors
% Go to desired directory, where your data file is present.
>> load [filename]
>> who   % shows what variables are there in out octave workspace
% data filename is also a variable. Just type that variable, to see whole data on octave terminal
>> size(filename)  % returns the rows x cols of data.
>> whos   % shows you detail view of variables in workspace
>> clear [variable]   % will remove the variable from workspace
>>v = datafilename(1:10)  % saves first 10 elements of datafilename
>> save hello.mat v;     // saves data in binary format
>> load hello.mat
>> who  % you can see variable 'v' back in your workspace
>> save hello.txt v –ascii  % save as text (ASCII)
>> A(3,2)    % accessing an index in matrix A, at 3rd row and 2nd col.
>> A(2, :)   % ":" means every element along that row/column
>> A([1 3], :)  % get all the elements of A from 1st and 3rd row, and every column.
>> A(: , 2) = [10; 11; 12]  % Assigning every element in 2nd col of A, to the new values 10, 11 & 12
>> A = [A, [100; 101; 102]] % adds another column vector to A with values …..
>> A(:)  % put all elements of A into a single column vector.
>> C = [A  B]  % concatenating matrix A and B into C. [A  B] is same as [A, B]
>> C = [A; B]  % putting matrices on top of each other. Try  size(C)
```

Computing on Data

```
>> A=[1 2; 3 4; 5 6]
>> B=[11 12; 13 14; 15 16]
>> C = [1  1; 2  2]
>> V = [1; 2; 3]
>> A*C;      % Multiply
>> A .*B     % Element-wise Multiply
>> A . ^2   % Element-wise squaring ex. A^2
>> 1 ./ C    %  Element-wise reciprocal of C
>> log(C)   % Element wise logarithm
>> exp(C)  % base e exponentiation of C
>> abs([-1; 2; -3])  % gives the element-wise absolute value
>> -V   % gives -1*V
>> V + ones(length(V),1)  % just as V+1
>> A'  % A transpose
>> val = max (A)  % gives column-wise max
>> [val, ind] = max(A)  % gives max value and its index
>> V < 3;  % returns element-wise comparison truth value
>> find(V<3)  % returns elements < 3
>> A = magic (3)  % try it to find what is interesting ??
>> [r,c]=find(A >= 7)  % gives index of an element which is >= 7
>> % useful functions  sum(A) , prod(A) , floor(A), ceil(A), rand(3), max(rand(3), rand(3))
>> max(A, [ ], 1) % takes column-wise max  and max(A, [ ], 2 ) takes row-wise max : default is col-wise
>>max(max(A))  % gives the maximum value in whole matrix
>>sum(A,1)  % column-wise sum  And  sum(A,2) gives row-wise sum
>> A= magic(9); %  then do like → A .* eye(9)  → b= sum(A .* eye(9)) → sum(b)
>> flipud(eye(9));  % flip up-side down the matrix
>> pinv(A) % gives the inverse of matrix    try → pinv(A) *A
```

Plotting the Data

```
>> t=[0:0.01:0.98];
>> y1=sin(2*pi*4*t);
>>plot(t,y1);
>>y2=cos(2*pi*4*t);
>>plot(t,y2);
>>plot(t,y1);
>>hold on;  % plotting one function plot over another
>>plot(t,y2,'r');
>>xlabel('time'); % giving a lable to x-axis
>>ylabel('value'); % giving a lable to Y-axis
>>legend('sin', 'cos'); % giving a legend
>>title('My Plot');  % giving title of your plot
>> print -dpng 'myplot.png';  % saving the plot as png image file. For other file formats use help
>> figure(1); plot(t,y1);  %  also try → figure(2);plot(t,y2);  % save two plots in your current dir.
>> subplot(1,2,1) % divides a plot into 1x2 grid, and access the first element
>> plot(t,y1);
>>subplot(1,2,2) % access 2nd element
>> plot(t,y2);
>> axis([0.5 1 -1 1]) % sets the scale of axis.
>> clf;  % clears a figure
>> A = magic(5);
>> imagesc(A);  % assigns each element of matrix a color. Also try → imagesc(A), colorbar, colormap gray;
>>a=1, b=2, c=3 % carries three commands and executes one after another. Comma chaining of commands
```

Control Statements

```
>> v = zeros(10,1);
>> for i=1:10,          % a for loop iterating from i=1 to 10
      v(i)=2^I;
   end;
>>indices = 1:10
>> for i= indices ……. % using predefined indices
>>  i = 1;
>> while i<5,              % try this while loop and its output
     v(i) = 100;
      i=i+1;
    end;
>> i=1;
>> while true,
     v(i)= 999;
     i= i+1;
     if i == 6,
       break;
     end;
   end;
>> % also try using if, elseif and else. % use disp( ) function if you want to display some string.
```

Defining Functions

>> addpath('C:/Users/Umer/Desktop'); % adds a search path for Octave to search for function/Data files

>> % Create Function files with .m extension and execute them. Just follow me on screen.

>> X = [1 1; 1 2; 1 3]

>> y= [1; 2; 3]

>> theta = [0;1];

>> create a costFunctionJ.m function file and code as following me on screen.

>> J = costFunctionJ(X, y, theta);