Octave for Machine Learning

Thursday, August 2, 2018 6:50 PM

Octave Basics

Description	Command	result
To make code readable *PS1('>> ') takes out default 'octave:1' and replaces with '>>' to make code readable*	PS1('>> ')	>>
To Exit or Quit	exit quit	Either one of them exit the octave instance
>> % Basic operations		
addition	>> 3+5	8
subtraction	>> 3-5	-2
multiplication	>> 3*5	15
division	>> 3/5	0.60000
power	>> 2^6	64
>> % Logical operations		
false	>> 1 ==2	1
True	>> 1 ~=2	0
AND	>> 1 && 0	0
OR	>> 1 || 0	1
??? Not sure yet	>> xor(1,0)	1
>> % Variables		
Assigning	>> a = 3	3
semicolon suppress the printing output if we need output we need to specifically check	>> a = 3; a	3
	Display(a)	a = 3
	>> b = 'hi'	b = hi
evaluate if it is true and if it is it will prin	>> c = (3>=1);	c = 1
	>> a = pi; % display() or disp() both are say. It will display or provide output >> display(a) >> disp(a)	a = 3.1416 3.1416
	>> disp(sprintf('only two decimals: %0.2f',a))	only two decimals: 3.14
	>> disp(sprintf('display six decimals:%0.6f', a))	display six decimals:3.141593
	>> a a = 3.1416 >> format short	a = 3.1416
	>> format long >> a	a = 3.141592653589793

>> % Matrices and Vectors		
Matrix	>> A = [1,2;3,4;5,6] % semicolon ';' indicates next row of the matrix	A = 1 2 3 4 5 6

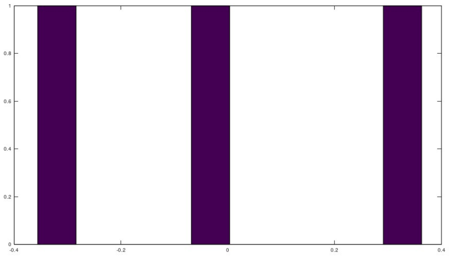
	>> A = [1,2;
	> 3,4;
	> 5,6]
		A =
		1 2
		3 4
		5 6

vector	>> v = [1;2;3]	v = 1 2 3
	>> v1 = [1 2 3]	v1 = 1 2 3

Size - Get dimension of the matrix A were m = rows and ne = columns	>> [m,n] = size(A)	m = 3
		n = 2

% You could also store it this way	>> szA = size(A)	szA = 3 2
	>> size(szA)	ans = 1 2
	>> szv = size(v)	szv = 3 1
	>> size(szv)	ans = 1 2
Index - % Now let's index into the 3nd row 2nd column of matrix A	>> A = [1,2;3,4;5,6] >> A(3,2)	A = 1 2 3 4 5 6 ans = 6
Pick entire row : means fetch entire row / column	>> A(2,:)	ans = 3 4
Pick entire column	>> A(:,2)	ans = 2 4 6
Picking specific rows - pick row 1 and 3	>> A([1 3], :)	ans = 1 2 5 6
Assigning new values to rows or columns	>> A(:, 2) = [10;11;12]	A = 1 10 3 11

		5 12
Append a new column vector to the write	>> A = [A, [100;101;102]]	A = 1 10 100 3 11 101 5 12 102
Put all elements of a matrix into a single vector	>> A(:)	ans = 1 3 5 2 4 6 100 101 102
Concatenating two matrices	>> A = [1 2;3 4;5 6] >> B = [11,12;13,14;15,16] >> C = [A B] (OR) >> C = [A, B] >> C = [A; B]	A = 1 2 3 4 5 6 B = 11 12 13 14 15 16 C = 1 2 11 12 3 4 13 14 5 6 15 16 C = 1 2 3 4 5 6 11 12 13 14 15 16
giving range to vector with increments	>> v = 1:2:10	v = 1 3 5 7 9
	>> v = 1:6; >> v	v = 1 2 3 4 5 6
Length- usually used in vectors. It gives longest dimension	>> length(v)	ans = 3
same number matrix / vector	>> ones(2,3)	1 1 1 1 1 1
	>> 2*ones(2,3)	2 2 2 2 2 2
	>> v = zeros(1,4)	v = 0 0 0 0
random number generation	>> v = rand(1,3)	v = 4.036444337557339e-001 6.236177331018921e-001 2.862962755786958e-001
Gaussian random number	>> v = randn(1,3)	v =

		-3.558743052605708e-001 3.629663240380411e-001 -3.789169559342462e-002
histogram of v with hist command	hist(v)	
Identical matrix	>> I = eye(3)	I = Diagonal Matrix 1 0 0 0 1 0 0 0 1
Help	Help commond Help eye Help help	
Addition and Scalar Multiplication	>> A = [1,2,4;5,1,3]; >> B = [1,3,4;1,1,1]; >> s = 2; % contstant 's' A B s	A = 1 2 4 5 1 3 B = 1 3 4 1 1 1 s=2
Addition	add_AB = A+B >> A+1	add_AB = 2 5 8 6 2 4 ans = 2 3 5 6 2 4
Subtraction	sub_AB = A-B	sub_AB = 0 -1 0 4 0 2
multiplication	>> Mult_As = A*s	Mult_As = 2 4 8 10 2 6
division	>> div_As = A /s	div_As = 5.000000000000000e-001 1.000000000000000e+000 2.000000000000000e+000 2.500000000000000e+000 5.000000000000000e-001 1.500000000000000e+000
	add_As = A+s	add_As = 3 4 6 7 3 5
Matrix Vector multiplication	>> A = [1, 2, 3; 4, 5, 6;7, 8, 9]	A = 1 2 3 4 5 6 7 8 9

	>> v = [1;1;1] A*v	v = 1 1 1 6 15 24
Matrix Matrix multiplication Element wise multiplication Element wise squaring Reciprocal Log Exponentiation Absolute value: gives non negative values Negative	>> A = [1,2;3,4;5,6]; >> B = [11,12;13,14;15,16] >> C = [1 1;2 2] >> A*C >> A.*B >> A.^2 >> 1./A >> Log(A) >> Exp(A) >> abs(A) >> -A % -1 * A	A = 1 2 3 4 5 6 B = 11 12 13 14 15 16 C = 1 1 2 2 ans = 5 5 11 11 17 17 ans = 11 24 39 56 75 96 ans = 1 4 9 16 25 36 ans = 1.000000000000000e+000 5.000000000000000e-001 3.333333333333333e-001 2.500000000000000e-001 2.000000000000000e-001 1.666666666666667e-001 ans = 0.000000000000000e+000 6.931471805599453e-001 1.098612288668110e+000 1.386294361119891e+000 1.609437912434100e+000 1.791759469228055e+000 ans = 2.718281828459045e+000 7.389056098930650e+000 2.008553692318767e+001 5.459815003314424e+001 1.484131591025766e+002 4.034287934927351e+002 ans = 1 2 3 4 5 6 ans = -1 -2

		-3 -4 -5 -6
Matrix multiplication	>> A = [1,2;4,5] >> B = [1,1;0,2] >> I = eye(2)	A = 1 2 4 5 B = 1 1 0 2 I = Diagonal Matrix 1 0 0 1
	AI = IA	AI = 1 2 4 5 IA = 1 2 4 5
	AB is not equal to BA	AB = 1 5 4 14 BA = 5 7 8 10
Inverse and Transpose	>> A = [1,2;4,5] A' >> inv(A) (OR) >> pinv(A) Inv(A) *A	A = 1 2 4 5 trans = 1 4 2 5 A_inv = -1.666666666666667e+000 6.666666666666666e-001 1.333333333333333e+ 000 -3.333333333333333e-001 1.000000000000000e+000 1.110223024625157e-016 0.000000000000000e+000 1.000000000000000e+000
True or false	>> a = [1 15 2 0.5] >> a<3	a = 1.00000 15.00000 2.00000 0.50000 ans = 1 0 1 1
Find - it will find the index where equation holds good	>> find(a<3)	ans = 1 3 4
Sum(a)	>> sum(a)	ans = 18.500
prod(a)	>>prod(a)	ans = 15
Floor(a) - rounds down	>> floor(a)	ans = 1 15 2 0

Ceil(a) - rounds up	>> ceil(a)	ans = 1 15 2 1

Rand	>> rand(3)	ans =
		  0.124186 0.964188 0.085185
		  0.544257 0.214253 0.409160
		  0.121604 0.974376 0.081557

Max per column	>> max(A, [], 1) (OR) >>max(A)	ans = 8 9 7

Max per row	>> max(A, [], 2)	ans =
		  8
		  7
		  9

Max value in the matrix	>> max(max(A)) OR >> max(A(:))	ans = 9

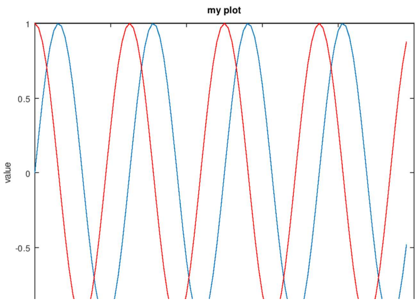
Magic() - to create n by n magic square where sum of each row/ column/ diagonal sum up to same value	>> magic(3)	ans =
		  8 1 6
		  3 5 7
		  4 9 2

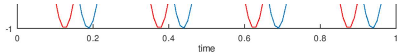
Moving data		
current directory	>> pwd	ans = C:\Users\HOME
Change directory	>> cd 'C:\Users\HOME\octave_test' >> pwd	ans = C:\Users\HOME\octave_test
List directory	ls	Volume in drive C is OS Volume Serial Number is 6AEA-E7CE Directory of C:\Users\HOME\octave_test [.] [..] 0 File(s) 0 bytes 2 Dir(s) 410,966,540,288 bytes free
Add path to the Octave	>>addpath('C:\Users\Home\octave_test') >>cd 'C:\'	
Remove path to octave	>> r**mpath** ('C:\Users\Home\octave_test')	
Load data into octave	>> load featuresX.dat >> load ('priceY.dat') >> load ex1data1.txt	
Who - shows what all variables that are in memory in octave	>> who	Variables in the current scope: A AI A_inv BA IA a add_As b dimA div_As n sub_AB szv v w AB A_21 B I Mult_As add_AB ans c dimv m s szA trans v1

Whos - gives the details of the variables	>> whos	Variables in the current scope:
		
		  Attr Name Size Bytes Class
		  ==== ==== ==== ===== =====
		       A 3x2 48 double

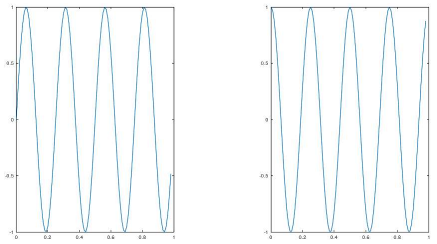
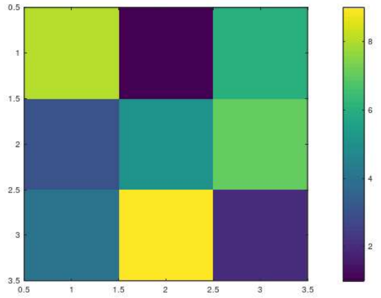
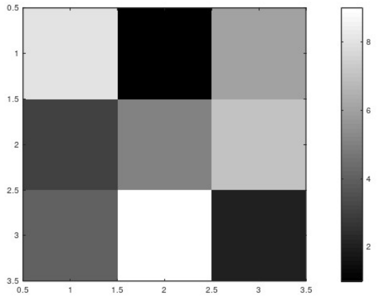
		AB 2x2 32 double AI 2x2 32 double A_21 1x1 8 double A_inv 2x2 32 double B 2x2 32 double BA 2x2 32 double I 2x2 16 double IA 2x2 32 double Mult_As 2x3 48 double a 1x1 8 double add_AB 2x3 48 double add_As 2x3 48 double ans 1x25 25 char b 1x2 2 char c 1x1 1 logical dimA 1x2 16 double dimv 1x2 16 double div_As 2x3 48 double m 1x1 8 double n 1x1 8 double s 1x1 8 double sub_AB 2x3 48 double szA 1x2 16 double szv 1x2 16 double trans 2x2 32 double v 3x1 24 double v1 1x3 24 double w 1x10 80 double
Clear - will delete the variable from the memory	>> clear A >> who	Variables in the current scope: AB A_21 B I Mult_As add_AB ans c dimv m s szA trans v1 AI A_inv BA IA a add_As b dimA div_As n sub_AB szv v w
	>> clear >> who	% will show nothing, as everything is cleared
Load file to octave	>> load ex1data1.txt >> who	Variables in the current scope: ex1data1
	>> whos	Variables in the current scope: Attr Name Size Bytes Class ==== ==== ==== ===== ===== ex1data1 97x2 1552 double Total is 194 elements using 1552 bytes

	>> vector1 = ex1data1(:,1); >> v = vector1(1:5)	v = 6.110100000000000 5.527700000000000 8.518599999999999 7.003200000000000 5.859800000000000
	whos	Variables in the current scope: Attr Name Size Bytes Class ==== ==== ==== ===== ===== A 3x2 48 double ans 3x1 24 double ex1data1 97x2 1552 double s 10x1 80 double v 5x1 40 double vector1 97x1 776 double Total is 315 elements using 2520 bytes
Save to file	>> save hello.mat v; >> ls	Volume in drive C is OS Volume Serial Number is 6AEA-E7CE Directory of C:\Users\HOME\octave_test [.] [..] ex1data1.txt hello.mat 2 File(s) 1,603 bytes 2 Dir(s) 410,971,734,016 bytes free
	>> save hello.txt v -ascii % save as text (ASCII) >> ls	Volume in drive C is OS Volume Serial Number is 6AEA-E7CE Directory of C:\Users\HOME\octave_test [.] [..] ex1data1.txt hello.mat hello.txt 3 File(s) 1,693 bytes 2 Dir(s) 410,972,196,864 bytes free
Plotting Data		

Use vector range and equations and plot the graphs	>> t = [0 : 0.01 : 0.98];
	>> y1 = sin(2*pi*4*t);
	>> y2 = cos(2*pi*4*t);
	>> plot(t, y1);
	>> hold on;
	>> plot(t, y2, 'r')
	>> xlabel('time')
	>> ylabel('value')
	>> title('my plot')
		

	>> cd 'C:\Users\ang\Desktop'; print -dpng 'myplot.png' >>close	

Different figures for each equation	>> figure(1); plot(t, y1);
	>> figure(2); plot(t, y2);	

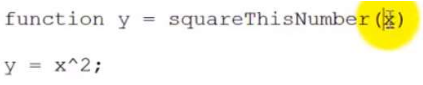
Subplot	>> subplot(1,2,1); % divides plot a 1x2 grid, access first element >> plot(t,y1) >> subplot(1,2,2); >> plot(t,y2) >> axis([0.5 1 0.5 1]) >> axis([0 1 -1 1]) Help axis	
Clear the figure	>> clf	
Colorbar	>> A = magic(3) Imagesc(A), colorbar Imagesc(A), colorbar, colormap gray;	A = 8 1 6 3 5 7 4 9 2  
Control statements: for, while, if statements		

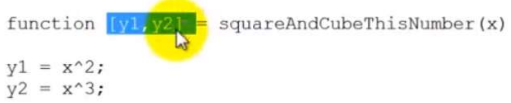
For	v = zeros(10,1);
	>> for i = 1:10,
	> v(i)=2^i;
	> end;
	>> v
		v =
		2
		4
		8
		16
		32
		64
		128
		256
		512
		1024

While	>>i = 1;
	>>while i<5,
	> v(i) = 100;
	> i = i+1;
	> end;
		v =
		100
		100
		100
		100
		32
		64
		128
		256
		512
		1024

	>>i = 1;
	>>while true,
	> v(i) = 999;
	> i=i+1;
	> if i ==5,
	> break;
	> end;
	> end;
		v =
		999
		999
		999
		999
		32
		64
		128
		256
		512
		1024

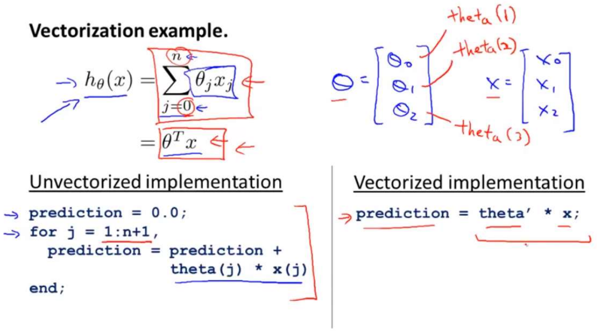
If	>>if v(1) ==1, > disp('the value is one'); > elseif v(1) = 999, > disp('the value is 999'); > else > disp('the value is something else'); > end;	The value is 999

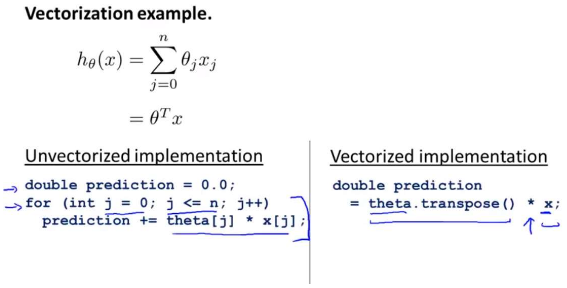
Function	>>pwd
	>>cd ' C:\Users\Home\octave_test'
	>>pwd
	%make sure the file is created as this
	
	function y = squareThisNumber(x)
	y = x^2;
	>>square(5)
		ans = C:\Users\Home
		
		ans = C:\Users\Home\octave_test
		
		ans = 25

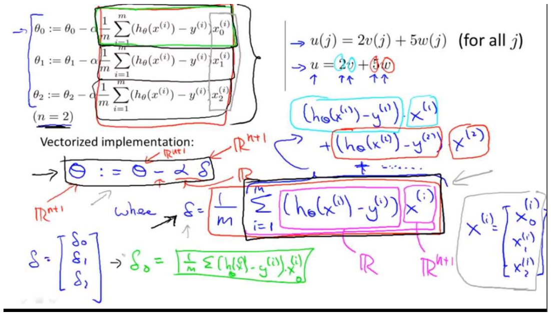
	>>[a,b] = squareAndCube(5)
	
	function [y1,y2] = squareAndCubeThisNumber(x)
	y1 = x^2;
	y2 = x^3;
		a = 25
		b = 125

Cost function	>>X = [1 1;1 2; 1 3] >>y = [1;2;3] >>theta = [0;1] >>j = costFunctionJ(X, y, theta)	X = 1 1 1 2 1 3 y = 1 2 3 theta = 0 1 m = 3

		predictions = 1 2 3 j = 0
Vectorization		

In Octave	
	Vectorization example.
	$h_\theta(x) = \sum_{j=0}^{n} \theta_j x_j$
	$= \theta^T x$
	$\Theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \end{bmatrix}$ $x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix}$
	Unvectorized implementation
	`prediction = 0.0;`
	`for j = 1:n+1,`
	`  prediction = prediction +`
	`      theta(j) * x(j)`
	`end;`
	Vectorized implementation
	`prediction = theta' * x;`	

In other programming languages (like C++)	
	Vectorization example.
	$h_\theta(x) = \sum_{j=0}^{n} \theta_j x_j$
	$= \theta^T x$
	Unvectorized implementation
	`double prediction = 0.0;`
	`for (int j = 0; j <= n; j++)`
	`  prediction += theta[j] * x[j];`
	Vectorized implementation
	`double prediction`
	`  = theta.transpose() * x;`	

Gradient descent	 $\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_0^{(i)}$ $\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_1^{(i)}$ $\theta_2 := \theta_2 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_2^{(i)}$ $(n = 2)$ $u(j) = 2v(j) + 5w(j)$ (for all j) $u = 2v + 5w$ Vectorized implementation: $\Theta := \Theta - \alpha \delta$ where $\delta = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x^{(i)}$ $(h_\theta(x^{(1)}) - y^{(1)}) x^{(1)} + (h_\theta(x^{(2)}) - y^{(2)}) x^{(2)} + \ldots$ $x^{(i)} = \begin{bmatrix} x_0^{(i)} \\ x_1^{(i)} \\ x_2^{(i)} \end{bmatrix}$ $\delta = \begin{bmatrix} \delta_0 \\ \delta_1 \\ \delta_2 \end{bmatrix} \rightarrow \delta_0 = \frac{1}{m} \sum (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_0^{(i)}$ $\mathbb{R}^{n+1}$ $\mathbb{R}^{n+1}$ $\mathbb{R}^{n+1}$ $\mathbb{R}$ $\mathbb{R}$ $\mathbb{R}^{n+1}$	