Octave

May 28, 2012

1 Octave Tutorial

1.1 Basic operations

- octave - fast prototyping, free
- Matlab - fast prototyping, expensive
- Python/NumPy - slow prototyping
- R - slow prototyping

%	comment
~=	not equal
&&	logical and
||	logical or
xor	xor
PS1('>>');	changes prompt to >>

1.2 variables

a = 3
a = 3; % semicolon supresses output
disp(a); - more complex display
disp(sprintf('2 decimals: %0.2f', a)); %displays string, 0.2f-> two decimal places
format long - long floating point printing format short

A = [1 2; 3 4; 5 6] = $\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}$

v = [1 2 3] - 1x3 matrix
v = [1;2;3] - 3x1 matrix
v=1:0.1:2 - 1x11 matrix, start at 1, increment by 0.1 until 2

$$v = \begin{bmatrix} 1 & 1.1 & 1.2 & 1.3 & 1.4 & 1.5 & 1.6 & 1.7 & 1.8 & 1.9 & 2 \end{bmatrix}$$

ones(2, 3) - 2x3 matrix of all ones

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

2 * ones(2, 3)

$$\begin{bmatrix} 2 & 2 & 2 \\ 2 & 2 & 2 \end{bmatrix}$$

rand(3, 3)	3x3 random matrix
randn(1, 3)	gausian distributed random 1x3 matrix
w = -6 + sqrt(10) * (randn(1, 100000));	
hist(w)	prints histogram of w
hist(w, 50)	histogram with more bins
eye(4)	4x4 identity matrix

Documentation:

help eye
help rand

1.3 Moving data around

A = [1 2; 3 4; 5 6]	
size(A)	3 2←answer returns a 1x2 matrix answer
size(A, 1)	3←rows
size(A, 2)	2←columns
v = [1 2 3 4]	
length(v)	4

1.4 fiding data on file system

pwd	current directory
cd	change directory cd 'c:\users\jz\Desktop'
ls	list files
load <filename>	loads file
load('<filename>')	loads file, string filename
who	shows what variables are available in memory data loaded from memory will
whos	gives details about variables in memory
clear <varname>	clears variable varname
v = priceY(1:10)	sets v to be first 10 elements of priceY vector
save hello.mat variablename;	saves data in variablename to a binary file hello.mat
save hello.txt v -ascii	saves as ascii formatted text

1.5 manipulating data

A=[1 2; 3 4; 5 6]	
A(3, 2)	6 element in 3rd row, 2nd column
A(2,:)	means every element along that row or column
A(:,2)	everything in second column
A([1 3], :)	get all first and third rows of A and all columns
A(:, 2) = [10; 11; 12]	assignment, assign that vector to second column
A = [A, [100; 101; 102]]	append another column vector to the right
A(:)	put all elements of A into a single column vector

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

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}$$

B = [11 12; 13 14; 15 16]

$$\begin{bmatrix} 11 & 12 \\ 13 & 14 \\ 15 & 16 \end{bmatrix}$$

C = [A B] - horizontally concatinating matrices

$$\begin{bmatrix} 1 & 2 & 11 & 12 \\ 2 & 4 & 13 & 14 \\ 5 & 6 & 15 & 16 \end{bmatrix}$$

C = [A; B] - puts the matrices on top of each other

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \\ 11 & 12 \\ 13 & 14 \\ 15 & 16 \end{bmatrix}$$

1.6 Computing on Data

A = [1 2; 3 4; 5 6]
B = [11 12; 13 14; 15 16]
C = [1 1; 2 2]

A * C	matrix multiplication
A .* B	.* Multiply by corresponding element
A .^ 2	squaring of elements
v = [1; 2; 3]	[1/1; 1/2; 1/3]
1 ./ v	
log(v)	element wise logarithm
exp(v)	exponentiation
abs(v)	element wise absolute value
-v	negative
v + ones(length(v), 1)	increments elements of v
v+1	increment elements of v
A'	transpose

a = [1 15 2 05]

max(a)	maximum value of a, 15
[val, ind] = max(a)	value, index
max(A)	column wise maximum
a < 3	[1 0 1 1] element wise comparison
find(a < 3)	[1 3 4]

A = magic(3)	3x3 matrix where each row and colum and diagonal add up to the same thing, proba
[r, c] = find(A >= 7)	finds elements in condition, returns to r, c

sum(a)	adds elements of a
floor(a)	floor
ceil(a)	ceiling
prod(a)	product
max(rand(3), rand(3))	element maximum
max(A, [], 1)	column wise maximum, 1 take max along first dimension
max(A, [], 2)	per row
max(A(:))	max in matrix max(max(A))

A = magic(9)	
sum(A, 1)	sum each column, same number, since magic square
sum(A, 2)	sum each row, same number, since magic square
sum(sum(A .* eye(9)))	will give single sum, 369 along diagonal
flipud(eye(9))	flips the diagonals
pinv(A)	pseudo inverse

1.7 Plotting Data

t=[0:0.01:0.98];	
y1 = sin(2 * pi * 4 * t);	
plot(t, y1);	plots the sin function with y1 axis
y2 = cos(2 * pi * 4 * t);	
plot(t, y2);	takes the sin plot and replaces it with cos

plot(t, y1);	
hold on;	will plot another figure on top of the previous one
plot(t, y2, 'r');	plots on the same plot as the one before hold on, 'r' is red color
xlabel('time')	x label
ylabel('value')	y label
legend('sin', 'cos')	legend
title('my plot')	title on top of figure
print -dpng 'myPlot.png'	save as a png file, may need to cd first, change dirclose - gets rid of the plot

figure(1); plot(t, y1); - can specify figure number
figure(2); plot(t, y2); - another figure number

subplot(1, 2, 1)	subdivides the plot into a 1x2 grid, access first element
plot(t, y1);	starts to access first element subplot(1, 2, 2) - second element plot(t, y2); -
axis([0.5 1 -1 1])	access second element xrange, yrange for last plot, on the right in this case
clf;	clears figure

A = magic(5)
imagesc(A) - plots a 5x5 grid of colors, where diff colors correspond to values in A
imagesc(A), colorbar, colormap gray; - sets grey colormap and sets colorbar showing shade corresponding to value
command chaining with ','

1.8 Control statements and functions

v = zeros(10, 1)

for loop

```
for i=1:10,
  v(i) = 2^i; % spacing doesn't matter
end;
```

for loop ext range

```
indices = 1:10;
for i=indices, disp(i); end;
```

while loop

```
i = 1;
while i <= 5,
  v(i)=100;
  i = i+1;
end;
```

while loop with break

```
i = 1;
while true,
  v(i) = 999;
  i = i + 1;
  if i == 6,
    break,
```

```
  end;
end;
```

if, elseif, else

```
v(1) = 2;
if v(1) == 1,
  disp('the value is one');
elseif v(1) == 2,
  disp('the value is two');
else
  disp('the value is not one or two.');
end;
```

1.9 functions

You create function by creating a file with functionname.m file name

```
function y = squareThisNumber(x)
% function declaration, return parameter y, name of function - squareThisNumber, x accepted param
y = x^2;
```

cd to dir where you hold the file use function OR modify search path by addpath('c:\users\<user>\desktop')

1.10 functions that return multiple values

```
function [y1, y2] = squareAndCubeThisNumber(x)
y1 = x^2;
y2 = x^3;
```

run and return values to a, b

```
[a, b] = squareAndCubeThisNumber(5);
```

define function to compute cost function J(Theta)

```
X = [1 1; 1 2; 1 3];
y = [1; 2; 3] ;
theta = [0, 1];
```

```
function J = costFunction(X, y, theta)
% X is the 'design matrix' containing our training examples % y is the class labels
m = size(X, 1); % number of training examples
predictions = X * theta; % predictions of hypothesis on all m examples
sqrErrors = (predictions - y) .^ 2; % squred errors
J = 1/(2 * m) * sum(sqrErrors);
```

```
>> j = costFunctionJ(X, y, theta)
j = 0
theta = [0, 0];
```

```
>> j = costFunctionJ(X, y, theta)
j = 2.3333
```

1.11 Vectorization

Readily available numerical algos

$$h_\Theta(x) = \sum_{j=0}^{n} \Theta_j x_j = \Theta^\intercal x$$

Unvectorized Implementation

```
prediction = 0.0;
for j = 1:n+1,
  prediction = prediction + theta(j) * x(j)
end;
```

Vectors in matlab are 1-indexed, so Θs may end up 1 indexed

Vectorized Implementation

```
prediction = theta' * x;
```

Unvectorized C++:

```
double prediction = 0.0;
for (int j = 0; j <= n; j++) {
    prediction += theta[j] * x[j];
}
```

Vectorized C++:

```
double prediction = theta.transpose() * x;
```

Gradient Descent

Recall:

$$\Theta_j := \Theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\Theta(x^{(i)}) - y^{(i)}) x_j^{(i)}, \ \forall_j$$

Simultaneous updates:

$$\Theta_0 := \Theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\Theta(x^{(i)}) - y^{(i)}) x_0^{(i)}$$

$$\Theta_1 := \Theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\Theta(x^{(i)}) - y^{(i)}) x_1^{(i)}$$

$$\Theta_2 := \Theta_2 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\Theta(x^{(i)}) - y^{(i)}) x_2^{(i)}$$

Vectorized Implementation

$$\Theta := \Theta - \alpha\delta$$

$$where\, \delta = \frac{1}{m}\sum_{i=1}^{m}(h_\Theta(x^{(i)}) - y^{(i)})x^{(i)}$$

$$\Theta \in \mathbb{R}^{n+1}$$

$$\alpha \in \mathbb{R}$$

$$\delta \in \mathbb{R}^{n+1}$$

$$x^{(i)} \in \mathbb{R}^{n+1}$$

Vector δ is the term after $\alpha\forall_\Theta$

$$\delta = \begin{bmatrix} \delta_0 \\ \delta_1 \\ \delta_2 \end{bmatrix},\, x^{(i)} = \begin{bmatrix} x_0^{(i)} \\ x_1^{(i)} \\ x_2^{(i)} \end{bmatrix}$$