Math operations are same as C++

Logic operations are same as C++

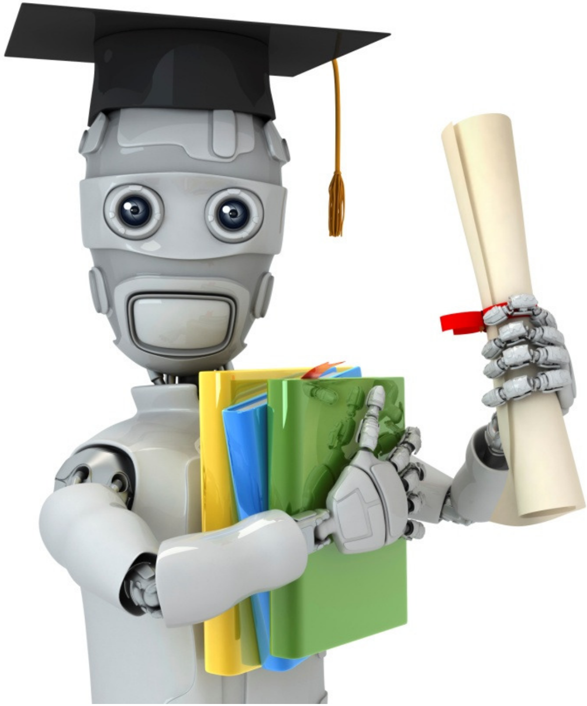
Semicolon supresses the print output

disp() for printing

sprintf() for string printing like printf in C

Matrices: A = [x1,x2,x3 ; y1,y2,y3] ==> 3*2 Matrix & semicolon for row separation

Vector: A = [x1;x2;x3] ==> 3d vector



Machine Learning

Octave Tutorial

Basic operations

v = 1:0.1:2 ==> starting from 1 & incrementing by 0.1 till reaching 2

v = 1:6 ==> starting from 1 to 6

ones(n,m) ==> generates matrix of ones(n:rows *m:cols)

also zeros(n,m)

rand(n,m) ==> generates matrix of random no.s between 0 & 1

randn(n,m) ==> generates matrix of gaussian distribution(Normal) with u=0 & stdDeviation = 1

hist(var,no. of pins) ==> draws histogram of var

eye(n) ==> generates n*n identity matrix

help command

size(A) ==> returns n,m. size(A,1) ==> returns size of 1st row

length(v) ==> returns maxSize in a matrix

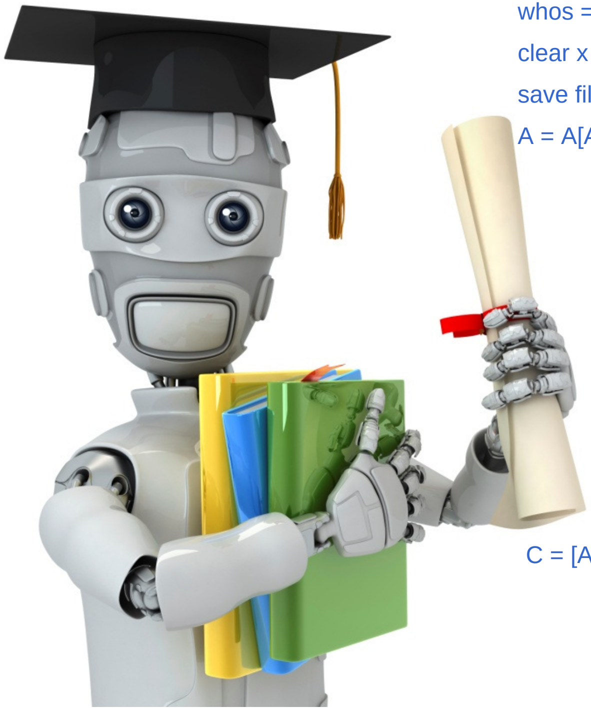
load('file name') ==> loads data

whos ==> list details about current variables

clear x ==> clears a variable, clear ==> clears all vars

save file_name var_name ==> saves a file with content of var_name. save file_name var_name -ascii ==> text

A = A[A, [y1;y2;y3]] ==> append another column. A(:) ==> converts A to vector



Octave Tutorial

Moving data around

C = [A,B] ==> concatenates A & B to C. C = [A;B] ==> puts A on top of B on C

Machine Learning

A .*B ==> multiplies each element of A to each element of B

1 ./ A ==> inverses each element in A

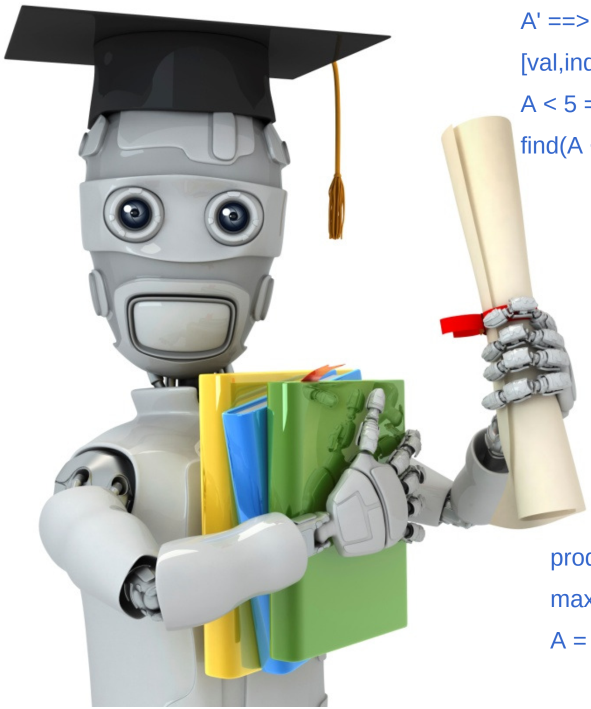
log(A), exp(A), abs(A)

A' ==> A transpose

[val,ind] = max(A) ==> returns max element in matrix A & its index

A < 5 ==> element wise operation

find(A < 3) ==> returns all elements that less than 3. [r,c] = find(A > 5) == returns indexes of elements



Octave Tutorial

Computing on data

prod(A), sum(A), floor(A)

max(A, [], 1) ==> returns the max of each column. max(A, [], 2) ==> returns max of each row

A = magic(n) ==> generates n*n magic matrix where sum(A,1) is all the same for all columns

Machine Learning

plot(x,y,'event_color'). hold on ==> to plot figures on top of each other

xlabel('x-axis label'), ylabel(). legend('', '',) ==> labeling graphs. title('') ==> for naming window

cd 'directory' print -dpng 'image_name' ==> to save the plot at specific directory

close ==> closes the plotted figure

figure(n); plot(t,y1) ==> for getting a specific figure

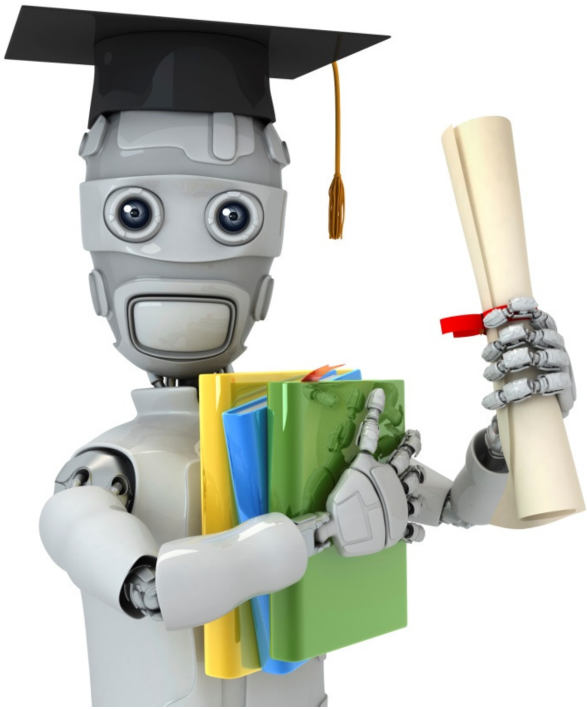
subplot(1,2,1); plot(t,y1) ==> dividing the plot into 1*2 grid. where the 1st is plot of (t,y1)

axis([x1 x2 y1 y2]) ==> ranges the plot x-axis & y-axis

Octave Tutorial

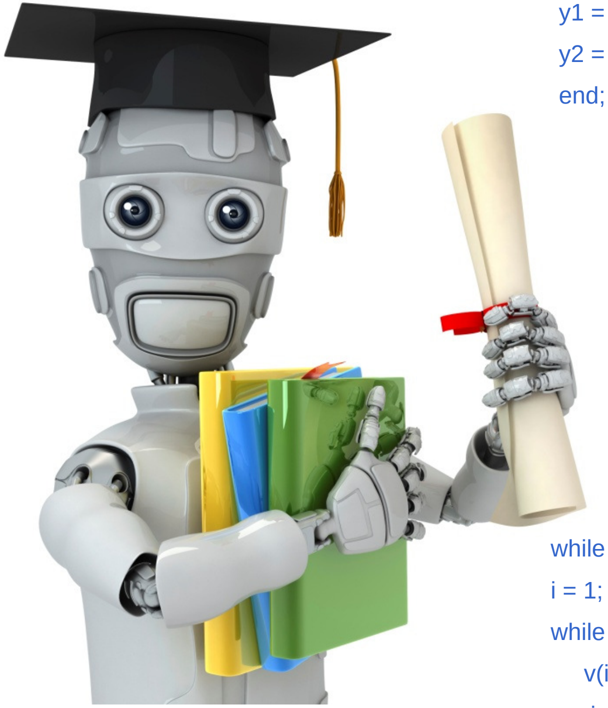
Plotting data

imagesc(A), color bar==> plotting matrix as grid of colored bars.



Machine Learning

```
function y = squareThisNumber(x),
y = x^2; ==> returing y from function called squareThisNumber(x) that takes x
fucntion [y1,y2] = getCubeSquare(x),
y1 = x^2;
y2 = x^3;
end; ==> returning multiple values
```

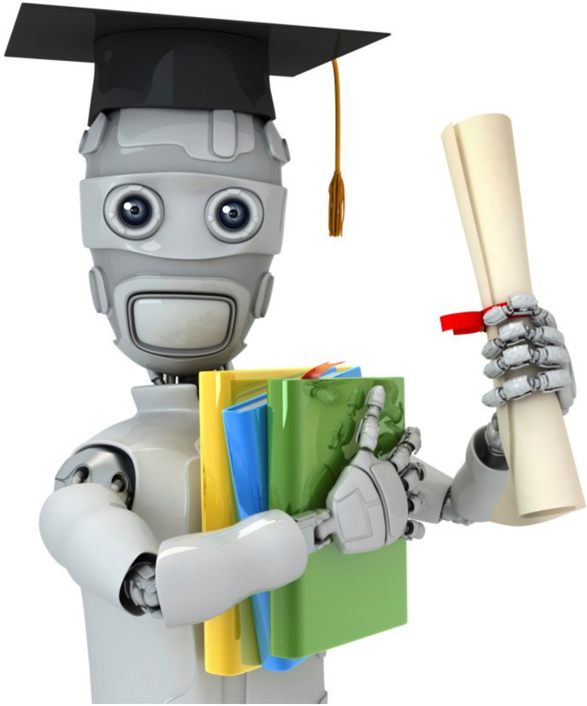


Machine Learning

Octave Tutorial

Control statements: for, while, if statements

```
while loop ex:
i = 1;
while true,
    v(i) = 99;
     i = i+1;
     if i == 5,
         break;
      end;
end;
```



Machine Learning

Octave Tutorial

Vectorial implementation

Vectorization example.

$$h_\theta(x) = \sum_{j=\theta}^{n} \theta_j x_j$$

$$= \theta^T x$$

<u>Unvectorized implementation</u>

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

<u>Vectorized implementation</u>

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

Vectorization example.

Implemetation in C++

$$h_\theta(x) = \sum_{j=0}^{n} \theta_j x_j$$

$$= \theta^T x$$

Unvectorized implementation

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

Vectorized implementation

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

Gradient descent

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

(for all j)

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_0^{(i)}$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_1^{(i)}$$

$$\theta_2 := \theta_2 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_2^{(i)}$$

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_0^{(i)}$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_1^{(i)}$$

$$\theta_2 := \theta_2 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_2^{(i)}$$

$(n = 2)$

$u(j) = 2v(j) + 5w(j)$ (for all j)

$u = 2v + 5w$