Machine Learning by Andrew Ng
pdf by CarolusRex

MALAB语法

Basic Operations

优先级

优先级	运算符
1	圆括号 ( )
2	矩阵转置和乘方：转置（.'）、共轭转置（'）、乘方（.^）、矩阵乘方（^）
3	一元加法（+）、一元减法（-）、取反（~）
4	乘法（.*）、*矩阵乘法*（*）、右除（./）、左除（.\）、矩阵右除（/）、矩阵左除（\）
5	加法（+）、减法（-）、逻辑非（~）
6	冒号运算符（:）
7	小于（<）、小于等于（<=）、大于（>）、大于等于（>=）、等于（==）、不等于（~=）

优先级	运算符
8	逐元素逻辑与（&）
9	逐元素逻辑或（|）
10	避绕式逻辑与，或者捷径逻辑与（&&）
11	避绕式逻辑或，或者捷径逻辑或（||）

考虑到 ^ 用于指数，故异或为函数$xor()$。
;表示不显示运行结果。
π is expressed by pi

Basic example

```
>> a = pi;
>> a

a =

    3.1416

>> disp(a);
    3.1416

>> disp(sprintf('2 decimals: %0.2f', a))
2 decimals: 3.14
>> disp(sprintf('6 decimals: %0.6f', a))
6 decimals: 3.141593
>> a

a =

    3.1416

>> format long
>> a

a =

   3.141592653589793

>> format short
>> a

a =

    3.1416
```

vectors & matrices example

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

A =

     1     2
     3     4
     5     6

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

A =

     1     2
     3     4
     5     6

>> v = [1 2 3]

v =

     1     2     3

>> v = [1; 2; 3]

v =

     1
     2
     3

>>
>> v = 1 : 0.1 : 2

v =

    1.0000    1.1000    1.2000    1.3000    1.4000    1.5000    1.6000    1.7000    1.8000    1.

>> v = 1 : 6

v =

     1     2     3     4     5     6

>> ones(2, 3)

ans =

     1     1     1
     1     1     1
```

```
>> C = 2 * ones(2, 3)

C =

     2     2     2
     2     2     2

>> C = [2 2 2; 2 2 2]

C =

     2     2     2
     2     2     2

>> w = ones(1, 3)

w =

     1     1     1

>> w = zeros(1, 3)

w =

     0     0     0

>> rand(3, 3)

ans =

    0.7757    0.6158    0.3686
    0.7339    0.7982    0.8099
    0.1452    0.8202    0.0607

>> w = randn(1, 3)

w =

   -1.9734    1.0101    0.1092

>> hist(w)
>> hist(w, 50)
>> I = eye(4)

I =

     1     0     0     0
     0     1     0     0
     0     0     1     0
```

0 0 0 1

Moving Data Around

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

A =

     1     2
     3     4
     5     6

>> size(A)

ans =

     3     2

>> sz = size(A)

sz =

     3     2

>> size(A, 1)

ans =

     3

>> size(A, 2)

ans =

     2

>> v = [1 2 3 4]

v =

     1     2     3     4

>> length(v)

ans =

     4

>> length(A)

ans =

     3

>> pwd
```

```
ans =

    '/MATLAB Drive'

>> who

Your variables are:

A    C    I    a    ans  sz   v    w

>> load("ex1/ex1data1.txt")
>> size(ex1data1)

ans =

    97     2

>> load("ex1/ex1data2.txt")
>> size(ex1data2)

ans =

    47     3

>> whos
  Name           Size            Bytes  Class     Attributes

  A              3x2                48  double
  C              2x3                48  double
  I              4x4               128  double
  a              1x1                 8  double
  ans            1x2                16  double
  ex1data1      97x2              1552  double
  ex1data2      47x3              1128  double
  sz             1x2                16  double
  v              1x4                32  double
  w              1x3                24  double

>> clear ex1data1
>> whos
  Name           Size            Bytes  Class     Attributes

  A              3x2                48  double
  C              2x3                48  double
  I              4x4               128  double
  a              1x1                 8  double
  ans            1x2                16  double
  ex1data2      47x3              1128  double
  sz             1x2                16  double
  v              1x4                32  double
```

```
  w              1x3                24  double

>> v = ex1data2(1:10)

v =

        2104        1600        2400        1416        3000        1985        1534        1427

>> save hello.mat v; % mat is compressed
>> clear
>> who
>> load hello.mat
>> who

Your variables are:

v

>> save hello.txt v -ascii   % save as text(ASCII)
```

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

A =

     1     2
     3     4
     5     6

>> A(3, 2)

ans =

     6

>> A(2, :)

ans =

     3     4

>> A(:, 2)

ans =

     2
     4
     6

>> A([1, 3], :)

ans =

     1     2
     5     6

>> A(:, 2) = [10, 11, 12]

A =

     1    10
     3    11
     5    12

>> A = [A, [100; 101; 102]]

A =

     1    10   100
     3    11   101
     5    12   102
```

```
>> size(A)

ans =

     3     3

>> A(:) % put all elements of A into a single vector

ans =

     1
     3
     5
    10
    11
    12
   100
   101
   102

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

A =

     1     2
     3     4
     5     6

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

B =

    11    12
    13    14
    15    16

>> C = [A B]

C =

     1     2    11    12
     3     4    13    14
     5     6    15    16

>> C = [A; B]

C =

     1     2
     3     4
     5     6
```

```
    11    12
    13    14
    15    16

>> size(C)

ans =

     6     2
```

Computing on Data

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

A =

     1     2
     3     4
     5     6

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

B =

    11    12
    13    14
    15    16

>> C = [1 1; 2 2]

C =

     1     1
     2     2

>> A * C

ans =

     5     5
    11    11
    17    17

>> A .* B

ans =

    11    24
    39    56
    75    96

>> A .^ 2

ans =

     1     4
     9    16
    25    36

>> v = [1; 2; 3]

v =
```

```
     1
     2
     3

>> 1 ./ v

ans =

    1.0000
    0.5000
    0.3333

>> 1 ./ A

ans =

    1.0000    0.5000
    0.3333    0.2500
    0.2000    0.1667

>> log(v)

ans =

         0
    0.6931
    1.0986

>> exp(v)

ans =

    2.7183
    7.3891
   20.0855

>> abs([-1; 2; -3])

ans =

     1
     2
     3

>> -v

ans =

    -1
    -2
    -3
```

```
>> v + ones(length(v), 1)

ans =

     2
     3
     4

>> v + 1

ans =

     2
     3
     4

>> A

A =

     1     2
     3     4
     5     6

>> A'

ans =

     1     3     5
     2     4     6

>> (A')'

ans =

     1     2
     3     4
     5     6

>> a = [1 15 2 0.5]

a =

    1.0000   15.0000    2.0000    0.5000

>> val = max(A)

val =

     5     6
```

```
>> val = max(a)

val =

    15

>> [val ind] = max(a)

val =

    15


ind =

     2

>> a < 3

ans =

  1×4 logical array

   1   0   1   1

>> find(a < 3)

ans =

     1     3     4

>> A = magic(3)

A =

     8     1     6
     3     5     7
     4     9     2

>> [r, c] = find(A >= 7)

r =

     1
     3
     2


c =
```

```
     1
     2
     3

>> sum(a)

ans =

   18.5000

>> prod(a)

ans =

    15

>> floor(a)

ans =

     1    15     2     0

>> ceil(a)

ans =

     1    15     2     1

>> rand(3)

ans =

    0.8147    0.9134    0.2785
    0.9058    0.6324    0.5469
    0.1270    0.0975    0.9575

>> max(rand(3), rand(3))

ans =

    0.9649    0.9572    0.6787
    0.9595    0.8491    0.7577
    0.9706    0.9340    0.9157

>> A

A =

     8     1     6
     3     5     7
     4     9     2
```

```
>> max(A, [], 1)

ans =

     8     9     7

>> max(A, [], 2)

ans =

     8
     7
     9

>> max(A)

ans =

     8     9     7

>> max(max(A))

ans =

     9

>> A(:)

ans =

     8
     3
     4
     1
     5
     9
     6
     7
     2

>> max(A(:))

ans =

     9

>> A = magic(9)

A =
```

```
    47    58    69    80     1    12    23    34    45
    57    68    79     9    11    22    33    44    46
    67    78     8    10    21    32    43    54    56
    77     7    18    20    31    42    53    55    66
     6    17    19    30    41    52    63    65    76
    16    27    29    40    51    62    64    75     5
    26    28    39    50    61    72    74     4    15
    36    38    49    60    71    73     3    14    25
    37    48    59    70    81     2    13    24    35

>> sum(A, 1)

ans =

   369   369   369   369   369   369   369   369   369

>> sum(A, 2)

ans =

   369
   369
   369
   369
   369
   369
   369
   369
   369

>> eye(9)

ans =

     1     0     0     0     0     0     0     0     0
     0     1     0     0     0     0     0     0     0
     0     0     1     0     0     0     0     0     0
     0     0     0     1     0     0     0     0     0
     0     0     0     0     1     0     0     0     0
     0     0     0     0     0     1     0     0     0
     0     0     0     0     0     0     1     0     0
     0     0     0     0     0     0     0     1     0
     0     0     0     0     0     0     0     0     1

>> A .* eye(9)

ans =

    47     0     0     0     0     0     0     0     0
     0    68     0     0     0     0     0     0     0
     0     0     8     0     0     0     0     0     0
```

```
     0     0     0    20     0     0     0     0     0
     0     0     0     0    41     0     0     0     0
     0     0     0     0     0    62     0     0     0
     0     0     0     0     0     0    74     0     0
     0     0     0     0     0     0     0    14     0
     0     0     0     0     0     0     0     0    35

>> sum(sum(A .* eye(9)))

ans =

   369

>> sum(sum(A .* flipud(eye(9))))

ans =

   369

>> flipud(eye(9))

ans =

     0     0     0     0     0     0     0     0     1
     0     0     0     0     0     0     0     1     0
     0     0     0     0     0     0     1     0     0
     0     0     0     0     0     1     0     0     0
     0     0     0     0     1     0     0     0     0
     0     0     0     1     0     0     0     0     0
     0     0     1     0     0     0     0     0     0
     0     1     0     0     0     0     0     0     0
     1     0     0     0     0     0     0     0     0

>> A = magic(3)

A =

     8     1     6
     3     5     7
     4     9     2

>> temp = pinv(A)

temp =

    0.1472   -0.1444    0.0639
   -0.0611    0.0222    0.1056
   -0.0194    0.1889   -0.1028

>> temp * A
```

```
ans =

    1.0000    0.0000   -0.0000
   -0.0000    1.0000    0.0000
    0.0000   -0.0000    1.0000
```

Plotting 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;
>> plot(t, y2, 'r')
>> xlabel('time')
>> ylabel('value')
>> legend('sin', 'cos')
>> title('my plot')
>> print -dpng 'myPlot.png'
>> close
>> figure(1); plot(t, y1);
>> figure(2); plot(t, y2);
>> subplot(1, 2, 1); % Divides plot a 1 * 2 grid, access first element
>> plot(t, y1);
>> subplot(1, 2, 2);
>> plot(t, y2);
>> axis([0.5 1 -1 1])
>> clf;
>>
>> A = magic(5)

A =

    17    24     1     8    15
    23     5     7    14    16
     4     6    13    20    22
    10    12    19    21     3
    11    18    25     2     9

>> imagesc(A), colorbar, colormap gray;
>> imagesc(magic(15)), colorbar, colormap gray;
>> a = 1, b = 2, c = 3

a =

     1


b =

     2


c =

     3
```

```
>> a = 1; b = 2; c = 3;
```

Control Statements:

for

```
>> v = zeros(10, 1)

v =

     0
     0
     0
     0
     0
     0
     0
     0
     0
     0

>> for i = 1: 10,
v(i) = 2^i;
>> end;
>> v

v =

           2
           4
           8
          16
          32
          64
         128
         256
         512
        1024

>> indices = 1: 10;
>> indices

indices =

     1     2     3     4     5     6     7     8     9    10

>> for i = indices,
disp(i);
end;
     1

     2

     3

     4
```

```
5
6
7
8
9
10
```

while

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

v =

         100
         100
         100
         100
         100
          64
         128
         256
         512
        1024

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

v =

         999
         999
         999
         999
         999
         999
          64
         128
         256
         512
        1024
```

if statement

```
>> 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;
The value is two
>>
```

Functions

```
>> squareThisNumber(5)

ans =

    25

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

a =

    25

>> b

b =

   125

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

X =

     1     1
     1     2
     1     3

>> y = [1; 2; 3]

y =

     1
     2
     3

>> theta = [0; 1];
>> j = costFunctionJ(X, y, theta)

j =

     0

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

j =

    2.3333
```