Introduction to Matlab

CSE555 Introduction to Pattern Recognition
Spring 2011 recitation
Introduction to Matlab

• Matlab stands for “Matrix Laboratory”. It was originally designed for solving linear algebra problems using matrices.

• Matlab is also a programming language that is widely used as a platform for developing Image Processing, Machine Learning programs.
Introduction to Matlab

Matlab is a very useful prototyping language

• A lot of libraries/toolboxes and tons of functions
• Easy to visualize data
• Quick prototype development
• Can cooperate with C/C++
• Can be slow, especially with bad programming practice
Introduction to Matlab

- Working directory
- Workspace
- Command Window
- Command History
Introduction to Matlab

• Variables
  – Does not require to be declared before use
  – Have not been defined previously (otherwise will overload the variable)
  – Variable name must start with a letter (e.g. _ind, 1st are not valid variable names)
  – Variable names are case sensitive (e.g. Test and test are two different variables)
Introduction to Matlab

• Operators
  – Assignment: \( x = y \)
  – Addition/subtraction: \( x + y, x - y \)
  – Multiplication (scalar or matrix): \( x * y \)
  – Multiplication (by element): \( x .* y \)
  – Division (scalar or matrix): \( x / y \)
  – Division (by element): \( x ./ y \)
  – Power: \( x ^ y, x.^ y \)
Introduction to Matlab

```matlab
>> x
x =
    1  2  3  4  5

>> y
y =
    1  2
    3  4

>> x = y
x =
    1  2
    3  4
```
Introduction to Matlab

\[
x = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}
\]

\[
x + 1
\]

\[
\text{ans} = \begin{pmatrix} 2 & 3 \\ 4 & 5 \end{pmatrix}
\]

\[
x + x
\]

\[
\text{ans} = \begin{pmatrix} 2 & 4 \\ 6 & 8 \end{pmatrix}
\]

\[
y = \begin{pmatrix} 8 & 1 & 6 \\ 3 & 5 & 7 \\ 4 & 9 & 2 \end{pmatrix}
\]

\[
\text{>> } x + y
\]

??? Error using ==> plus
Matrix dimensions must agree.
Introduction to Matlab

\[ x = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \]

\[ \gg x*x \]
\[ ans = \begin{bmatrix} 7 & 10 \\ 15 & 22 \end{bmatrix} \]

\[ \gg x.*x \]
\[ ans = \begin{bmatrix} 1 & 4 \\ 9 & 16 \end{bmatrix} \]

\[ \gg x^2 \]
\[ ans = \begin{bmatrix} 7 & 10 \\ 15 & 22 \end{bmatrix} \]

\[ \gg x.^x \]
\[ ans = \begin{bmatrix} 1 & 4 \\ 27 & 256 \end{bmatrix} \]

\[ \gg x*2 \]
\[ ans = \begin{bmatrix} 2 & 4 \\ 6 & 8 \end{bmatrix} \]

\[ \gg x.*2 \]
\[ ans = \begin{bmatrix} 2 & 4 \\ 6 & 8 \end{bmatrix} \]

\[ \gg x.^2 \]
\[ ans = \begin{bmatrix} 1 & 4 \\ 9 & 16 \end{bmatrix} \]
Introduction to Matlab

- Keep in mind that all variables in Matlab are treated as matrices.
- When not using element by element operation such as `.*`, `./` the operations are really the same as matrix operations.
Introduction to Matlab

- Define a matrix

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

  \[
  \gg x = [1,2,3;4,5,6]
  \]

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

  \[
  \gg x = [1,2,3;4,5,6];
  \]

- Apostrophe operator (’) makes the transpose operation

  \[
  \gg x' 
  \]

  \[
  \text{ans =}
  \begin{bmatrix}
  1 & 4 \\
  2 & 5 \\
  3 & 6
  \end{bmatrix}
  \]
Introduction to Matlab

• Matrix operations
  – matrix(r, c) will return the element that at row r and column c

```matlab
x = 

1   2   3
4   5   6

>> x(1,3)

ans =

3
```
**Introduction to Matlab**

- **Matrix operations**
  - `matrix(r1 : rstep : r2, c1 : cstep : c2)` will return a portion of the matrix, where `r1`, `r2` specifies the beginning and ending row of the matrix, and `c1`, `c2` specifies the beginning and ending column of the matrix, `rstep` and `cstep` denotes the step size to increment from `r1` to `r2` and `c1` to `c2` respectively. `r1:1:r2` is equivalent to `r1:r2`.
  - If we want whole row or column, we could use `:` to replace the corresponding position.
Introduction to Matlab

\[ x = \begin{bmatrix}
92 & 99 & 1 & 8 & 15 & 67 & 74 & 51 & 58 & 40 \\
98 & 80 & 7 & 14 & 16 & 73 & 55 & 57 & 64 & 41 \\
4 & 81 & 88 & 20 & 22 & 54 & 56 & 63 & 70 & 47 \\
85 & 87 & 19 & 21 & 3 & 60 & 62 & 69 & 71 & 28 \\
86 & 93 & 25 & 2 & 9 & 61 & 68 & 75 & 52 & 34 \\
17 & 24 & 76 & 83 & 90 & 42 & 49 & 26 & 33 & 65 \\
23 & 5 & 82 & 89 & 91 & 48 & 30 & 32 & 39 & 66 \\
79 & 6 & 13 & 95 & 97 & 29 & 31 & 38 & 45 & 72 \\
10 & 12 & 94 & 96 & 78 & 35 & 37 & 44 & 46 & 53 \\
11 & 18 & 100 & 77 & 84 & 36 & 43 & 50 & 27 & 59
\end{bmatrix} \]

\[ \text{ans =} \begin{bmatrix}
98 & 80 & 7 \\
4 & 81 & 88 \\
85 & 87 & 19 \\
86 & 93 & 25
\end{bmatrix} \]

\[ \text{ans =} \begin{bmatrix}
92 & 1 & 15 \\
4 & 88 & 22 \\
86 & 25 & 9 \\
23 & 82 & 91
\end{bmatrix} \]
Introduction to Matlab

```
>> x(:,1)
ans =
   92
   98
    4
   85
   86
   17
   23
   79
   10
   11

>> x(1,:)
ans =
   92
   99
   1
   8
  15
  67
  74
  51
  58
  40
```
## Introduction to Matlab

<table>
<thead>
<tr>
<th>i : j</th>
<th>Denotes the number from i to j, i.e. [i, i+1, i+2, ..., j], and is empty if j &lt; i</th>
</tr>
</thead>
<tbody>
<tr>
<td>i : s : j</td>
<td>Denotes the number from i to j take on step value s, i.e. [i, i+s, i+2s, ..., j]</td>
</tr>
<tr>
<td>A(i, :)</td>
<td>The ith row of A</td>
</tr>
<tr>
<td>A(:, i)</td>
<td>The ith column of A</td>
</tr>
<tr>
<td>A(i:s:j, :)</td>
<td>The same as A(i, :), A(i+s, :), ..., A(j, :)</td>
</tr>
<tr>
<td>A(vecA, vecB)</td>
<td>The matrix that contain the rows specified in vector vecA, and columns specified in vector vecB</td>
</tr>
<tr>
<td>A(:)</td>
<td>Convert all the elements in A to a single column vector.</td>
</tr>
</tbody>
</table>
Introduction to Matlab

• Matrix operation
  – If we want to access to the last row/column of a matrix, we can use keyword ‘end’ in the corresponding position.
  – Matrix can also be concatenated using ‘,’ or ‘;’
    
    $z = [x, y];$ or $z = [x; y]$

    The corresponding dimensions must fit while we concatenate the matrices/vectors.
Introduction to Matlab

• Some matrix functions
  – \( x = \text{ones}(\text{number of rows, number of columns}) \)
    Constructs a full matrix with ones.
  – \( x = \text{zeros}(\text{number of rows, number of columns}) \)
    Constructs a full matrix with zeros.
  – \( x = \text{diag}(y) \)
    Constructs a diagonal matrix with \( y \) be the diagonal elements. If \( y \) is a matrix, \( x \) will be the diagonal elements in matrix \( y \).
Introduction to Matlab

• Some matrix functions
  – x = mean(y) calculate the mean value for y. If y is a vector, x is a scalar value, if y is a matrix, each row of y is treated as observations, and the corresponding mean is calculated.

```
x =
  1   2   3
  4   5   6

>> mean(x)
ans =
  2.5000   3.5000   4.5000
```
Introduction to Matlab

• Some matrix functions
  – \( y = \text{cov}(x) \) calculate the covariance matrix of \( x \)
  – \([y, \text{ind}] = \text{sort}(x, \text{option})\) sort vector \( x \) according to the option, can either be ascending order or descending order.
  – \( y = \text{inv}(x) \) calculate the inverse of matrix \( x \)
  – \( y = \text{det}(x) \) calculate the determinant of matrix \( x \)
  – \([\text{vec}, \text{val}] = \text{eig}(x)\) calculate the eigenvalue and eigenvector of matrix \( x \)
Introduction to Matlab

• Plotting figures
  – plot(x, y) plot y as a function of x, x and y must have the same number of elements.
  – plot(x) is equivalent to plot(1:length(x), x)
  – use ‘hold on’ and ‘hold off’ to plot multiple functions in one figure.

• Displaying images
  – imshow(f) f is the image
Introduction to Matlab

```
>> x = randn(1, 1000)*5;
>> y = randn(1, 1000);
>> m=mean([x',y']);
>> plot(x, y, '+');
>> hold on;
>> x1 = -20:0.001:20;
>> x2 = a(2,1)/a(1,1)*(x1-m(1))+m(2);
>> plot(x1, x2, 'r');
>> hold off;
```
Introduction to Matlab

```matlab
>> f = imread('./homework2-data/train0/00000.pgm');
>> imshow(f);
```
Introduction to Matlab

• Matlab programming
  – Expressions
  – Flow Controls
    • Condition
    • Iteration
  – Scripts
  – Functions
Introduction to Matlab

• Relational operators
  – Less than <
  – Less than or equal <=
  – Greater than >
  – Greater than or equal >=
  – Equal to ==
  – Not equal to ~=

• Logical operators
  – Not ~, and &, or |
Introduction to Matlab

• Conditional structures
  if condition
    expressions
  elseif condition (optional)
    expressions
  else
    expressions
  end
Introduction to Matlab

• Iterations

  for variable = expression
   expressions
  end

  while condition
   expressions
  end

  for i=1:2:10
   do whatever
  end

  while i < 10
   do whatever
  end
Introduction to Matlab

• “.m” files
  – Plain text files containing Matlab programs (functions/scripts). Can be called from command line by typing the filename, or from other M-files.
  – Scripts are like main() function in C/C++, except they do not return values and do not take input arguments.
  – Functions take input arguments and return values.
Introduction to Matlab

• Functions
  – File name must be the same as the function name.
  – Can contain many sub-functions in one function file.

    ‘function_name.m’
    function [out1, out2, ..., outN] = function_name(arg1, arg2, ..., argN)
    .......
Introduction to Matlab

• Good programming practice in Matlab
  – Do not use loops unless there is no other way to do it, loops are slow in Matlab. Most functions in Matlab take matrix/vector input and runs very fast (look at the help documents).
  – Always prefer matrix operations.
  – Allocate spaces for matrix/vectors before assign them values.
Hints on HW2-4

• Load image
  – image = imread(‘image path’);
  – imagepath = sprintf(‘path/%d.extension’, number);
  – imagepath = [‘path/’, num2str(number), ‘.extension’];
  – double_image = im2double(image);
  – use function zeros/ones to allocate space for training and testing vectors
Hints on HW2-4

• Compute PCA
  – use function reshape(x, r, c) to convert image into vector form, and convert vector images to original size for displaying.
  – cov(x) function calculates the covariance matrix of x, each row of x should be an observation.
  – [evec, eval] = eig(x) function will calculate the eigenvalues and eigenvectors of x. Each column of evec will be a eigen vector, eval will be a diagonal matrix, we can use diag(eval) to convert it to a vector.
  – [val, idx] = sort(x, option) will sort vector x according to option, ‘descend’ or ‘ascend’.
Hints on HW2-4

```matlab
>> figure;
for i = 1 : 5
   for j = 1 : 4
      subplot(5, 4, i+(j-1)*5); imshow(reshape(pvec(:,i+(j-1)*5), 28, 28), []);
   end
end
```
Hints on HW2-4

• Classification
  – If our data have $M$ dimensions, and we have $N$ samples, and we choose the top $D$ vectors from PCA, then each $M$ dimensional data point will have a $D$ dimensional representation after applying PCA.
  – If we organize the original samples in a $N \times M$ matrix $S$, with each row contains an observation. Assume $V$ is the eigenvector we got from PCA, to transform the data points it is simply an multiplication of two matrix $S \times V$. 
Hints on HW2-4

• Classification
  – To do classification we just need to find the nearest neighbors in the lower dimensional space.