APR (E.T.S. de Ingeniería Informática)
Curso 2016-2017

Práctica 0. Introducción a Octave *y* MATLAB

Enrique Vidal, Vicente Bosch, Francisco Casacuberta
Departamento de Sisatemas Informáticos y Computación
Universidad Politécnica de Valencia

Índice

1. Trabajo previo a la sesión de prácticas

Para la realización de esta práctica se supone que se ha adquirido previamente experiencia en el uso de *shell scripts*, *awk* y *gnuplot*, tanto en Sistemas Inteligentes como en otras asignaturas cursadas. El alumno deberá refrescar sus conocimientos y habilidades sobre dichas herramientas. Además, deberá haber leído de forma detallada la totalidad del boletín práctico, para poder centrarse en profundidad en la parte que hay que desarrollar en el laboratorio, la cual durará entre una y dos sesiones.

2. Introducción

Varias de las técnicas empleadas en el Aprendizaje dentro del área de Reconocimiento de Formas (RF) emplean cálculos vectoriales y matriciales. Es el caso de los algoritmos Perceptron o Widrow-Hoff por ejemplo. Por tanto, la aplicación de una herramienta que implemente de forma sencilla estos cálculos matriciales puede ayudar a obtener más rápidamente los sistemas de RF que hacen uso de estas funcionalidades.

Una de las herramientas comerciales más potentes en cálculo matricial es MATLAB, pero existe una herramienta de código libre que presenta capacidades semejantes: *GNU Octave*.

GNU Octave es un lenguaje de alto nivel interpretado definido inicialmente para computación numérica. Posee capacidades de cálculo numérico para solucionar problemas lineales y no lineales, así como otros experimentos numéricos. Posee capacidades gráficas para visualizar y manipular los datos. Se puede usar de forma interactiva o no (empleando ficheros que guarden programas a interpretar). Su sintaxis y semántica es muy semejante a MATLAB, lo que hace que los programas de éste sean fácilmente portables a Octave.

Octave está en continuo crecimiento y puede descargarse y consultarse su documentación y estado en su web `http://www.gnu.org/software/octave/`. Aunque está definido

para funcionar en GNU/Linux, también es portable a otras plataformas como OS X y MS-Windows (los detalles pueden consultarse en su web).

3. Órdenes básicas de *Octave*

Octave puede ejecutarse desde la línea de órdenes o desde el menú de aplicaciones del entorno gráfico. Para ejecutar desde la línea de órdenes se abre un terminal y se escribe:

```
octave
```

Generalmente, obtenemos una salida similar a:

```
GNU Octave, version 3.0.5
Copyright (C) 2008 John W. Eaton and others.
This is free software; see the source code for copying conditions.
There is ABSOLUTELY NO WARRANTY; not even for MERCHANTABILITY or
FITNESS FOR A PARTICULAR PURPOSE.  For details, type ‘warranty’.

Octave was configured for "i486-pc-linux-gnu".

Additional information about Octave is available at http://www.octave.org.

Please contribute if you find this software useful.
For more information, visit http://www.octave.org/help-wanted.html

Report bugs to <bug@octave.org> (but first, please read
http://www.octave.org/bugs.html to learn how to write a helpful report).

For information about changes from previous versions, type ‘news’.

octave:1>
```

La última línea tendrá un cursor indicando que se esperan órdenes de Octave. Estamos pues en el modo **interactivo**.

3.1. Aritmética básica

Octave acepta a partir de este momento expresiones aritméticas sencillas (operadores `+`, `-`, `*`, `/` y `^`, este último para exponenciación), funciones trigonométricas (`sin`, `cos`, `tan`, `arcsin`, `arccos`, `arctan`), logaritmos (`log`, `log10`), exponencial neperiana (`e^n` ó `exp(n)`) y valor absoluto (`abs`). Como respuesta a estas expresiones da valor a la variable `ans`, mostrándola, pero también puede asignarse a otras variables. Por ejemplo:

```
octave:1> sin(1.71)
ans =  0.99033
```

```
octave:2> b=sin(2.16)
b =  0.83138
```

Para consultar el valor de una variable, basta con escribir su nombre:

```
octave:3> b
b =  0.83138
octave:4> ans
ans =  0.99033
```

Aunque también puede emplearse la función `disp`, que muestra el contenido de la variable omitiendo su nombre:

```
octave:5> disp(b)
 0.83138
```

Las variables pueden usarse en otras expresiones:

```
octave:6> c=b*ans
c =  0.82334
```

Puede evitarse que se muestre el resultado en cada operación añadiendo `;` al final de la operación:

```
octave:7> d=ans*b*5;
octave:8> d
d =  4.1167
```

3.2. Operadores básicos en vectores y matrices

La notación matricial en Octave es entre corchetes (`[ ]`); en su interior, las filas se separan por `;` y las columnas por espacios en blanco o `,`. Por ejemplo, para crear un vector fila de dimensión 3, un vector columna de dimensión 2 y una matriz de 3×2, se puede hacer:

```
octave:9> v1=[1 3 -5]
v1 =

   1   3  -5

octave:10> v2=[4;2]
v2 =

   4
   2

octave:11> m=[3,-4;2 1;-5 0]
```

```
m =

   3  -4
   2   1
  -5   0
```

Siempre y cuando las dimensiones de los elementos vectoriales y matriciales implicados sean apropiados, sobre ellos se pueden aplicar operadores de suma (+), diferencia (-) o producto (*). El operador potencia ^ puede aplicarse sobre matrices cuadradas. Los operadores producto, división y potencia tienen la versión "elemento a elemento" (.*, ./, .^). Por ejemplo:

```
octave:12> mv=v1*m
mv =

   34   -1

octave:13> mx=m.*5
mx =

   15  -20
   10    5
  -25    0

octave:14> v3=v1*v2
error: operator *: nonconformant arguments (op1 is 1x3, op2 is 2x1)
octave:15> v3=v1*[3;5;6]
v3 = -12
octave:16> mvv=[3;5;6]*v1
mvv =

    3    9  -15
    5   15  -25
    6   18  -30
```

Se pueden aplicar operadores de comparación como >, <, >=, <=, == y !=, que retornan una matriz binaria con 1 en las posiciones en las que se cumple la condición y 0 en caso contrario:

```
octave:17> m>=0
ans =

   1   0
   1   1
```

```
   0   1

octave:18> m!=0
ans =

   1   1
   1   1
   1   0
```

Se pueden emplear también en la comparación de vectores y matrices de dimensiones congruentes, dando la matriz binaria resultado de comparar los elementos en la misma posición en ambas estructuras.

También existe el operador de transposición (’):

```
octave:19> m2=m’
m2 =

   3   2  -5
  -4   1   0
```

El indexado de los elementos se hace entre paréntesis. Para vectores puede indicarse una posición o lista de posiciones, mientras que para una matriz se espera una fila o secuencia de filas seguida de una columna o secuencia de columnas:

```
octave:20> v1(2)
ans =  3
octave:21> v1([2 3])
ans =

   3  -5
octave:22> v2(2)
ans =  2
octave:23> m3=[1 2 3 4;5 6 7 8;9 10 11 12]
m3 =

    1    2    3    4
    5    6    7    8
    9   10   11   12

octave:24> m3([1 3], [1 4])
ans =

    1    4
    9   12
```

Para indicar una fila o columna completa, se puede emplear ::

```
octave:25> m3(:,[1 3])
ans =

    1    3
    5    7
    9   11
```

Para indicar rangos, se emplea `i:f`, donde `i` es el índice inicial y `f` el final. Se puede emplear la notación `i:inc:f`, donde `inc` indica el incremento (por omisión es unitario).

```
octave:26> m3([1 3],1:3)
ans =

    1    2    3
    9   10   11

octave:27> m3([1 3],1:2:4)
ans =

    1    3
    9   11
```

Para indicar el último índice de una dimensión se puede emplear `end`:

```
octave:28> m3([1 3],end)
ans =

    4
   12
```

3.3. Funciones básicas en vectores y matrices

Octave aporta múltiples funciones para operar con vectores y matrices. Las más importantes son:

- `size(m)`: devuelve número de filas y columnas de la matriz (en el caso de un vector, una de las dimensiones tendrá tamaño 1)
- `eye(f,c)`, `ones(f,c)`, `zeros(f,c)`: dan la matriz identidad, todo unos y nula, respectivamente, de tamaño `f`×`c`; si se pone un solo número, da la matriz cuadrada correspondiente
- `sum(v)`, `sum(m)`: da la suma de los elementos del vector o matriz; en el caso de la matriz, devuelve el vector resultante de las sumas por columnas; si se le pasa un segundo argumento (`sum(m,n)`), éste indica la dimensión a sumar (1 para columnas, 2 para filas).

- `max(v)`, `max(m)`: indica el valor máximo del vector o el vector con los máximos por columna de la matriz; si se pide que devuelva dos resultados (`[r1,r2]=max(v)`, `[r1,r2]=max(m)`), el primer resultado almacena los valores y el segundo su posición
- `det(m)`: determinante de `m`
- `eig(m)`: vector de valores propios de `m` o su versión matricial diagonal
- `diag(v)`: crear matriz diagonal con los valores de `v`
- `inv(m)`: inversa de la matriz `m` si esta es no singular
- `trace(m)`: traza de la matriz `m`
- `sort(v)`: vector ordenado con los valores del vector `v`
- `find(v)`, `find(m)`: se le pasa un vector o matriz e indica aquellos elementos que no son cero (índices absolutos empezando en 1 y haciendo el recorrido por cada columna y por filas ascendentes); si se le piden dos resultados (`[r1,r2]=find(v)`, `[r1,r2]=find(m)`) el primer resultado almacena fila y el segundo columna; se puede aplicar sobre resultados de operaciones lógicas a fin de verificar elementos de la matriz o vector que cumplen una condición; por ejemplo, para obtener las posiciones con números pares:

```
octave:29> [r,c]=find(rem(m3,2)==0)
r =

   1
   2
   3
   1
   2
   3

c =

   2
   2
   2
   4
   4
   4
```

  Donde `rem` es la función que obtiene el resto del primer operador dividido por el segundo
- `repmat(m,f,c)`: crea una matriz de `f`×`c` bloques de `m`; si `c` se omite, será de `f`×`f`

3.4. Carga y salvado de datos

La introducción de datos de forma manual no es apropiada para grandes cantidades de datos. Por tanto, Octave aporta funciones que permiten cargar de y salvar en ficheros. El salvado se hace mediante la orden `save`:

```
octave:30> save "m3.dat" m3
```

El fichero tiene el siguiente formato:

```
# Created by Octave 3.0.5, DATE <user@machine>
# name: m3
# type: matrix
# rows: 3
# columns: 4
 1 2 3 4
 5 6 7 8
 9 10 11 12
```

Los ficheros de datos a cargar deben seguir este formato, indicando en la línea `# name:` el nombre de la variable en la que se cargarán los datos. Por ejemplo, ante un fichero `maux.dat` con contenido:

```
# Created by Octave 3.0.5, DATE <user@machine>
# name: A
# type: matrix
# rows: 4
# columns: 3
 1 2 -3
 5 -6 7
 -9 10 11
 -5 2 -1
```

Se cargaría con:

```
octave:31> load "maux.dat"
octave:32> A
A =

    1    2   -3
    5   -6    7
   -9   10   11
   -5    2   -1
```

La orden `save` puede usarse con opciones como `-text` (grabar en formato texto con cabecera, por omisión), `-ascii` (graba en formato texto sin cabecera), `-z` (graba en formato comprimido), o `-binary` (graba en binario). Por ejemplo:

```
octave:33> save -ascii "m3woh.dat" m3
```

En ocasiones, el salvado de datos puede provocar pérdida de precisión, pues por omisión se salva hasta el cuarto decimal. Para modificar esta precisión de salvado, se puede emplear `save_precision(n)`, donde `n` es el número de posiciones decimales (incluyendo el `.`) que se grabarán.

3.5. Funciones Octave

En Octave se pueden definir funciones que hagan tareas más específicas y complejas. Las funciones Octave pueden recibir varios parámetros y pueden devolver varios valores de retorno (que pueden incluir vectores y matrices).

La sintaxis básica de una función en Octave es:

```
function [ lista_valores_retorno ] = nombre ( [ lista_parametros ] )
  cuerpo
end
```

Un posible ejemplo sería:

```
octave:34> function [m1,m2] = addsub(ma,mb)
> m1=ma+mb
> m2=ma-mb
> end
octave:35> mat1=[1,2;3,4]
mat1 =

   1   2
   3   4

octave:36> mat2=[-1,2;3,-4]
mat2 =

  -1   2
   3  -4

octave:37> addsub(mat1,mat2)
m1 =

   0   4
   6   0

m2 =

   2   0
   0   8
```

```
ans =

   0   4
   6   0

octave:38> [mr1,mr2]=addsub(mat1,mat2)
m1 =

   0   4
   6   0

m2 =

   2   0
   0   8

mr1 =

   0   4
   6   0

mr2 =

   2   0
   0   8
```

Estas funciones es habitual definirlas en ficheros `.m` (de código Octave) cuyo nombre debe ser el mismo que la función escrita en él (en nuestro ejemplo, `addsub.m`), y que deben situarse en el mismo directorio en el que se ejecuta Octave. De esa forma, se puede acceder a las funciones sin deber definirlas cada vez.

3.6. Programas Octave

Octave se puede usar de forma **no interactiva** escribiendo *scripts* que son interpretados por Octave y donde se pueden emplear las mismas instrucciones que en el modo interactivo. Por ejemplo, suponiendo que tenemos en el directorio actual el fichero `addsub.m` con la función previamente definida, desde la terminal editemos el siguiente *script* en el fichero `test.m`:

```
#!/usr/bin/octave -qf
a=[1,2,3;4,5,6;7,8,9]
b=[9,8,7;6,5,4;3,2,1]
c=a+b
```

```
[d,e]=addsub(a,b)
disp(c)
```

Desde la línea de órdenes hay que darle permisos de ejecución (`chmod +x test.m`) y se puede ejecutar como cualquier programa o *script*: `./test.m`. La salida será:

```
a =

   1   2   3
   4   5   6
   7   8   9

b =

   9   8   7
   6   5   4
   3   2   1

c =

   10   10   10
   10   10   10
   10   10   10

m1 =

   10   10   10
   10   10   10
   10   10   10

m2 =

  -8  -6  -4
  -2   0   2
   4   6   8

d =

   10   10   10
   10   10   10
   10   10   10

e =
```

```
-8  -6  -4
-2   0   2
 4   6   8

 10   10   10
 10   10   10
 10   10   10
```

Estos programas también pueden ejecutarse desde la línea interactiva de Octave escribiendo su nombre (sin `.m`). En este caso, se pueden poner argumentos en la línea de órdenes y pueden usarse la variable `nargin` (número de argumentos) y las funciones `argv()` (da la lista de los argumentos pasados). Estos argumentos están en formato de cadena, y pueden convertirse a formato numérico en caso necesario empleando la función `str2num(c)`.

4. Ejercicios propuestos

1. Realiza el producto escalar de los vectores $v_1 = (1, 3, 8, 9)$ y $v_2 = (-1, 8, 2, -3)$.
2. *a*) Obtén la matriz de dimensiones 4×4 a partir del producto de los vectores v_1 y v_2
   b) Calcula su determinante
   c) Calcula sus valores propios.
3. Sobre la matriz del ejercicio **??**:
   a) Calcula su submatriz 2×2 formadas por las filas 1 y 3 y las columnas 2 y 3.
   b) Súmale la matriz todo unos a la submatriz resultante.
   c) Calcula el determinante de la matriz resultante.
   d) Calcula la inversa de la matriz resultante.
4. Sobre la matriz inversa resultado del ejercicio **??**:
   a) Calcula su máximo valor y su posición.
   b) Calcula las posiciones (fila y columna) de los elementos mayores que 0.
   c) Calcula la suma de cada columna.
   d) Calcula la suma de cada fila.
5. Salva la matriz inversa resultante del ejercicio **??** con hasta 7 dígitos decimales; comprueba que se ha salvado correctamente.
6. Define una función Octave que reciba una matriz y devuelva la primera fila y la primera columna de esa matriz.
7. Implementa un *script* Octave que lea una matriz de un fichero `data` y grabe su traspuesta en el fichero `data_trans`.

5. Ejercicios adicionales

En el directorio `$HOME/asigDSIC/ETSINF/apr/p0/dat` (directorio de datos de la práctica 0) se encuentra el fichero `data`. Este fichero contiene dos columnas, la primera es la única dimensión en la que se representan los datos, $\mathbf{x} \in R$, la segunda es el valor objetivo o target t. El objetivo es aprender un vector de pesos mediante el algoritmo Widrow-Hoff que minimice el error cuadrático:

$$J_w = \sum_{i=1}^{n} (t_i - \mathbf{w} \cdot \mathbf{x}_i)^2 \tag{1}$$

Implementa una función en octave que devuelva dicho vector de pesos:

```
function [w]=wh(data)

# número de muestras de aprendizaje y dimensionalidad
[n,d]=size(data)
...
```