Machine Learning Lab #1: MATLAB

Joe Mama

August 31, 2018

1 Introduction

This lab introduces features of MATLAB that will be useful for machine learning applications.
It is assumed that you already have some familiarity with MATLAB.

2 Vectors

A numerical vector $\vec{v}$ is stored as a MATLAB array.
Examples of creating numerical vectors in MATLAB:

```
q = []
r = rand ( 6, 1 )
s = randn ( 5, 1 )
t = ones ( 4, 1 )
u = zeros ( 3, 1 )
v = randperm ( 8 )
w = 10 : 20
x = linspace ( 0.0, 1.0, 7 )
y = [ 1.1, 2.2, 3.3 ]
z = [ 4.4; 5.5; 6.6 ]
```

Row and column vectors are different objects in MATLAB. A row vector has a shape (1,n), while a column vector has a shape (m,1). You can find the shape of any array with the **size()** command: size (x) and you can save these values using a command like: [m, n] = size (x) The **length()** command returns the number of elements of a vector; for a row vector, it returns n, for a column vector, m;
Each entry in a vector has an index; the first entry in the vector a is a(1). You can print, use, or alter any element of a vector by indexing it. u(2) = 20 A *colon* can be used to refer to a range of vector indices: w(3:8) We can specify a stepsize between successive indexes: w(3:2:8)
Most arithmetic functions can be applied to a vector, giving a vector of results: abs(x), cos(x), exp(x), log(x) sin(x), sqrt(x) The ' operator will transpose a vector from one form to the other. rt = r' Some functions return a single result based on all the vector values: max(x), mean(x), min(x), norm(x), std(x), var(x)
For pairs of vectors, the operator '*' requests a form of the vector dot product. To get the desired scalar result, the vectors must be written so that their dimensions have the form (1xn) * (nx1), that is, row vector times column vector. If x and y are column vectors, as is common in mathematics convention, then the dot product is $x' * y$. If both are row vectors, then the express x*y' is necessary.
When applied to vectors, the operations *, / and ^ can be preceded by a period, indicating that the operation is to be applied element by element, returning a vector of results.

```
x .* y <-- returns a vector of elementwise products.
x ./ y <-- returns a vector of elementwise fractions.
x .^ 2 <-- returns a vector of elementwise squares;
x .^ y <-- returns a vector of elementwise powers
```

3 Matrices

An *mxn* numerical matrix is stored in a MATLAB array.

```
O = pascal ( 5 )
Q = []
R = rand ( 3, 4 )
S = randi ( 5, 3 )
T = randn ( 2, 2 )
U = ones ( 5, 3 )
V = zeros ( 4, 3 )
W = eye ( 3, 3 )
X = [ 1, 2, 3; 4, 5, 6 ]
Y = [ 1, 2;
      3, 4 ]
Z = magic ( 4 )
```

(MATLAB programmers typically use a capital letter to represent a matrix.)
The **size()** command returns the array dimensions: [m, n] = size (x)
Matrix values are accessed by specifying the row and column index, typically written as (i,j). You can print, use, or alter any element of a vector by indexing it. U(2,1) = 20 The colon can be used to define a portion of the array: R(2,2) a single entry R(2,1:4) the second row R(1:2,3) part of the third column R(2:3,2:3) a 2x2 submatrix
Most arithmetic functions can be applied to an array, giving an array of results: abs(X), cos(X), exp(X), log(X), norm(X), sin(X), sqrt(X) The ' operator will transpose an array from one form to the other. Xt = X' Some functions return a row vector of results, by applying the operation separately to each column: max(X), mean(X), min(X), std(X), var(X)
To multiply a matrix A times a vector x, we write y=A*x. If A is mxn, then x should be nx1 and the result y will be mx1.
To multiply a matrix A times a matrix B, we write C=A*B. If A is mxn, B must be nxo, and C will have size mxo.
In some cases, it may be necessary to use the transpose operator, writing expressions like u=A'*x or v=B*y' or even w=C'*z'.
When applied to matrices, the operations *, / and ^ indicating that the operation is to be applied element by element, returning a matrix of results.

```
x .* y <-- returns a matrix of elementwise products.
x ./ y <-- returns a matrix of elementwise fractions.
x .^ 2 <-- returns a matrix of elementwise squares;
x .^ y <-- returns a matrix of elementwise powers
```

4 Linear Algebra

Scalar functions of a matrix include the condition number, determinant, and rank: cond(A), det(A), rank(A) but only condition number is really useful and reliable; Vector functions include returning the diagonal elements of a matrix: diag(A) Matrix functions include the inverse, the LU factorization, the pseudoinverse,

and the QR factorization: inv(A), lu(A), pinv(A), qr(A) Computing the inverse matrix inv(A) is numerically unreliable and unnecessary.
The set of linear equations A*x=b is often posed, with the vector x unknown. If A is nxn and nonsingular, while b is nx1, then a unique solution may be expected. MATLAB can provide that solution by the command x = A b
Consider the following commands: A = pascal(4) x = ones(4,1) b = A * x x2 = A b
[p,l,u] = lu (a) a = p' * l * u
[q, r] = qr (A) A = Q * R
Ap = pinv (A) A*x=b x2=Ap*b A*x2
Eigenvalues:
[V, D] = eig (A) A*V=V*D
SVD: [U,S,V] = svd (A) A = U * S * V'

5 Plotting

Line plot
plot (x, y)
Multiple line plots
hold on plot (x1, y1) plot (x2, y2) hold off
Scatter plot
plot (x, y, '.')
Histogram
Bar chart
Contour plot
Surface plot

6 Operations on Data

7 Reading and Writing Data Files

To save all of your data, or to restore it, use the SAVE or LOAD command.
To save an item in a text file: save ('filename', '-ascii', 'variable') and then restore it by variable = load ('filename')
Many sets of data are stored in tabular form. Numeric data AND no header line AND separated by spaces, commas or tabs: use LOAD (or CSVREAD) Numeric data AND header line AND separated by commas: use CSVREAD (filename, 1, 0) Numeric data AND header line AND only first data item is text: use CSVREAD (filename, 1, 1) Numeric and text: use readtable (filename), which creates a MATLAB 'table' If the data is completely numeric, and is separated by spaces, commas or tabs, use LOAD