ARRAYS AND VECTORS WITH NUMPY

José M. Garrido
Department of Computer Science

January 2016

College of Computing and Software Engineering
Kennesaw State University

© 2015 J. M. Garrido
1 Arrays

In general, an array is a term used in programming and defined as a data structure that is a collection of values and these values are organized in several ways. In programming, a one-dimensional array is often known as a vector. The following arrays: $X$, $Y$, and $Z$ have their data arranged in different manners. Array $X$ is a one-dimensional array with $n$ elements and it is considered a row vector because its elements $x_1, x_2, \ldots, x_n$ are arranged in a single row.

$$X = [x_1 \ x_2 \ x_3 \ \cdots \ x_n] \quad Z = \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_m \end{bmatrix}$$

Array $Z$ is also a one-dimensional array; it has $m$ elements organized as a column vector because its elements: $z_1, z_2, \ldots, z_m$ are arranged in a single column.

The following array, $Y$, is a two-dimensional array organized as an $m \times n$ matrix; its elements are arranged in $m$ rows and $n$ columns. The first row of $Y$ consists of elements: $y_{11}, y_{12}, \ldots, y_{1n}$. Its second row consists of elements: $y_{21}, y_{22}, \ldots, y_{2n}$. The last row of $Y$ consists of elements: $y_{m1}, y_{m2}, \ldots, y_{mn}$.

$$Y = \begin{bmatrix} y_{11} & y_{12} & \cdots & y_{1n} \\ y_{21} & y_{22} & \cdots & y_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ y_{m1} & y_{m2} & \cdots & y_{mn} \end{bmatrix}$$

2 Vectors and Operations

A vector is a mathematical entity that has magnitude and direction. In physics, it is used to represent characteristics such as the velocity, acceleration, or momentum of a physical object. A vector $v$ can be represented by an n-tuple of real numbers:

$$v = (v_1, v_2, \ldots, v_n)$$

Several operations with vectors are performed with a vector and a scalar or with two vectors.
2.1 Addition of a Scalar and a Vector

To add a scalar to a vector involves adding the scalar value to every element of the vector. In the following example, the scalar $\alpha$ is added to the elements of vector $Z$, element by element.

$$Z = \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_m \end{bmatrix}$$

$$Z + \alpha = \begin{bmatrix} z_1 + \alpha \\ z_2 + \alpha \\ \vdots \\ z_m + \alpha \end{bmatrix}$$

2.2 Vector Addition

Vector addition of two vectors that are n-tuple involves adding the corresponding elements of each vector. The following example illustrates the addition of two vectors, $Y$ and $Z$.

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}$$

$$Z = \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_m \end{bmatrix}$$

$$Y + Z = \begin{bmatrix} y_1 + z_1 \\ y_2 + z_2 \\ \vdots \\ y_m + z_m \end{bmatrix}$$

2.3 Multiplication of a Vector and a Scalar

Scalar multiplication is performed by multiplying the scalar with every element of the specified vector. In the following example, scalar $\alpha$ is multiplied by every element $z_i$ of vector $Z$.

$$Z = \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_m \end{bmatrix}$$

$$Z \times \alpha = \begin{bmatrix} z_1 \times \alpha \\ z_2 \times \alpha \\ \vdots \\ z_m \times \alpha \end{bmatrix}$$

2.4 Dot Product of Two Vectors

Given vectors $v = (v_1, v_2, \ldots, v_n)$ and $w = (w_1, w_2, \ldots, w_n)$, the dot product $v \cdot w$ is a scalar defined by:

$$v \cdot w = \sum_{i=1}^{n} v_i w_i = v_1 w_1 + v_2 w_2 + \ldots + v_n w_n$$
Therefore, the dot product of two vectors in an n-dimensional real space is the sum of the product of the vectors’ components.

When the elements of the vectors are complex, then the dot product of two vectors is defined by the following relation. Note that \( \overline{v}_i \) is the complex conjugate of \( v_i \).

\[
v \cdot w = \sum_{i=1}^{n} \overline{v}_i w_i = \overline{v}_1 w_1 + \overline{v}_2 w_2 + \ldots + \overline{v}_n w_n
\]

### 2.5 Length (Norm) of a Vector

Given a vector \( v = (v_1, v_2, \ldots, v_n) \) of dimension \( n \), the Euclidean norm of the vector denoted by \( \|v\|_2 \), is the length of \( v \) and is defined by the square root of the dot product of the vector:

\[
\|v\|_2 = \sqrt{v \cdot v} = \sqrt{v_1^2 + v_2^2 + \ldots + v_n^2}
\]

In the case that vector \( v \) is a 2-dimensional vector, the Euclidean norm of the vector is the value of the hypotenuse of a right angled triangle. When vector \( v \) is a 1-dimensional vector, then \( \|v\|_2 = |v_1| \), the absolute value of the only component \( v_1 \).

### 3 Vector Properties and Characteristics

A vector \( v = (v_1, v_2, \ldots, v_n) \) in \( \mathbb{R}^n \) (an n-dimensional real space) can be specified as a column or row vector. When \( v \) is an \( n \) column vector, its transpose \( v^T \) is an \( n \) row vector.

#### 3.1 Orthogonal Vectors

Vectors \( v \) and \( w \) are said to be orthogonal if their dot product is zero. The angle \( \theta \) between vectors \( v \) and \( w \) is defined by:

\[
\cos(\theta) = \frac{v \cdot w}{\|v\|_2 \|w\|_2}
\]
where θ is the angle from v to w, non-zero vectors are orthogonal if and only if they are perpendicular to each other, i.e. when \( \cos(\theta) = 0 \) and θ is equal to \( \pi/2 \) or 90 degrees. Orthogonal vectors v and w are called orthonormal if they are of length one, i.e. \( v \cdot v = 1 \), and \( w \cdot w = 1 \).

### 3.2 Linear Dependence

A set \( k \) of vectors \( \{x_1, x_2, \ldots, x_k\} \) is linearly dependent if at least one of the vectors can be expressed as a linear combination of the others. Assuming there exists a set of scalars \( \{\alpha_1, \alpha_2, \ldots, \alpha_k\} \), vector \( x_k \) is defined as follows:

\[
x_k = \alpha_1 x_1 + \alpha_2 x_2 + \ldots + \alpha_{k-1} x_{k-1}
\]

If a vector \( w \) depends linearly on vectors \( \{x_1, x_2, \ldots, x_k\} \), this is expressed as follows:

\[
w = \alpha_1 x_1 + \alpha_2 x_2 + \ldots + \alpha_k x_k
\]

### 4 Using Arrays in Python with Numpy

Arrays are created and manipulated in Python and Numpy by calling the various library functions. Before using an array, it needs to be created. Numpy function `array` creates an array given the values of the elements. When an array is no longer needed in the program, it can be destroyed by using the `del` Python command.

Numpy function `zeros` creates an array with the specified number of elements, all initialized to zero. Similarly, function `ones` creates an array with its elements initialized to value 1.0. Note that the default type of these arrays is `float`. Function `arange` creates an array of integers starting at value 0 and increasing up to \( n - 1 \).

The following short Python program illustrates the various Numpy functions used to create arrays. The program is stored in file `test_arrays.py`.

```python
import numpy as np
print "Creating arrays"
x = np.array([4.5, 2.55, 12.0 -9.785])
print "Array x: ", x
y = np.zeros(12)
print "Array y: ", y
z = np.ones((3, 4)) # 3 rows, 4 cols
print "Array z: 
print z
```

© 2015 J. M. Garrido
Polynomial Models with Python

$n = \text{np.arange}(12)$

print "Array n: ", n
del x  # delete array x

Executing the Python interpreter and running the program yields the following output. Note that array $z$ is a two-dimensional array with three rows and four columns with all its elements initialized to 1.0

$ python test_arrays.py
Creating arrays
Array x:  [ 4.5  2.55  2.215]
Array y:  [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
Array z:
[[ 1.  1.  1.  1.]
 [ 1.  1.  1.  1.]
 [ 1.  1.  1.  1.]]
Array n:  [ 0  1  2  3  4  5  6  7  8  9 10 11]

A vector is manipulated by accessing its individual elements and changing and/or retrieving the value of the elements using indexing.

Listing 1 shows the source code of a Python program stored in file test2_arrays.py that uses Numpy functions to create and manipulate a vector. Line 4 calls function zeros to create vector $p$ with $N$ elements. In lines 7–8, elements with index $j$ (from 0 to $k - 1$) of vector $p$ are set to value 5.25. In line 11 vector $p$ is destroyed.

Listing 1: Program that creates and manipulates a vector.

1 import numpy as np
2 print "Creating and manipulate an array"
3 N = 8  # number of elements
4 p = np.zeros(N)
5 print "Array p: ", p
6 k = 5
7 for j in range(k):
8     p[j] = 5.25
9 print "Array p: 
10 print p
11 del p  # delete array p

Executing the Python interpreter and running the program yields the following output. Note that only the first $k$ elements of array $p$ are set to value 5.25.
Creating and manipulate an array
Array p: [ 0. 0. 0. 0. 0. 0. 0. 0.]
Array p:
[ 5.25 5.25 5.25 5.25 5.25 0. 0. 0.]

Slicing can be used to indicate more selective indexing with the : operator. For example, x[0:4] references the elements of array x but only the elements with indices from 0 up to 3. Most of the time it is more efficient (time-wise) to use slicing than using a loop. The following program is a variation of the previous one, instead of the loop, now slicing is used in line 7 to assign values to select elements of array p. The results are the same as the previous program.

1 import numpy as np
2 print "Creating and manipulate an array"
3 N = 8  # number of elements
4 p = np.zeros(N)
5 print "Array p: ", p
6 k = 5
7 p[0:k] = 5.25
8 print "Array p: 
9 print p

Arrays can be stacked into a single array by calling Numpy function hstack. Arrays can also be split into separate arrays by calling function hsplit. The following program creates two arrays p and q in lines 3 and 6, then it stacks them into array newa in line 7. Array newa is split into three arrays with equal shape in line 10. Arrays x, y, and z are used to reference the three arrays created in lines 12–14. Array newa is split after column 4 and up to column 6, basically creating three arrays in line 17.

1 import numpy as np
2 print "Stacking and splitting array"
3 p = np.array([1.5, 2.5, 3.5, 4.5, 5.5, 6.5, 7.5, 8.5])
4 print "Array p: 
5 print p
6 q = np.array([2.35, 5.75, 7.75, 3.15])
7 newa = np.hstack((p, q))
8 print "newa: 
9 print newa
10 r = np.hsplit(newa,3) # three equally shaped arrays
11 print "Array r:"

© 2015 J. M. Garrido
Polynomial Models with Python

12 x = r[0]
13 y = r[1]
14 z = r[2]
15 print "Array x: ", x
16 # after col 4 up to col 6
17 newb = np.hsplit (newa,(4,6))
18 print "Array newb:"
19 print newb[0]
20 print newb[1]
21 print newb[2]

$ python test2c_arrays.py
Stacking and splitting array
Array p:
[1.5, 2.5, 3.5, 4.5, 5.5, 6.5, 7.5, 8.5]
newa:
[ 1.5  2.5  3.5  4.5  5.5  6.5  7.5  8.5  2.35  5.75
  7.75  3.15]
Array r:
Array x:  [ 1.5  2.5  3.5  4.5]
Array newb:
[ 1.5  2.5  3.5  4.5]
[ 5.5  6.5]
[ 7.5  8.5  2.35  5.75  7.75  3.15]

5 Simple Vector Operations

Operations on vectors are performed on an individual vector, with a vector and a scalar, or with two vectors.

5.1 Arithmetic Operations

To add a scalar to a vector involves adding the scalar value to every element of the vector. This operation adds the specified constant value to the elements of the vector specified.

The following Python program illustrates the arithmetic operations on vectors and scalars. The program is stored in file test3_arrays.py. In line 4, vector $p$ is created and its elements initialized with zero values. In line 8, a scalar value $xv$ is added to vector $p$. In line 11 the constant 2 (another scalar) is subtracted from...
vector $p$ and assigned to array $q$. In line 13, vector $q$ is multiplied by the scalar 3 and assigned to vector $q2$. In line 15, vector $q2$ is divided by the scalar 2.5.

```python
1 import numpy as np
2 print "Arithmetic operations with a scalar"
3 N = 8 # number of elements
4 p = np.zeros(N)
5 print "Array p: ", p
6 print p
7 xv = 3.75
8 p = p + xv
9 print "Array p: ", p
10 print p
11 q = p - 2.0
12 print "Array q: ", q
13 q2 = q * 3
14 print "Array q2: ", q2
15 q3 = q2 / 2.5
16 print "Array q3: ", q3
```

Executing the Python interpreter with program `test3_arrays.py` yields the following output.

```
$ python test3_arrays.py
Arithmetic operations with a scalar
Array p: 
[ 0.  0.  0.  0.  0.  0.  0.  0.]
Array p: 
[ 3.75 3.75 3.75 3.75 3.75 3.75 3.75 3.75]
Array q: 
[ 1.75 1.75 1.75 1.75 1.75 1.75 1.75 1.75]
Array q2: 
[ 5.25 5.25 5.25 5.25 5.25 5.25 5.25 5.25]
Array q3: 
[ 2.1 2.1 2.1 2.1 2.1 2.1 2.1 2.1]
```

To add two vectors involves adding the corresponding elements of each vector, and a new vector is created. This addition operation on vectors is only possible if the row vectors (or column vectors) are of the same size. In a similar manner, subtracting two vectors of the same size can be performed.

In the following program, vectors $p$ and $q$ are created with size 8. The operation in line 8 adds the elements of vector $q$ and the elements of vector $p$ and the elements are assigned to vector $q3$, the new vector created. The operation in line 10 subtracts the elements of vector $p$ from the elements of vector $q$ and the new vector created is $q4$. This program is stored in file `test4_arrays.py`. 

© 2015 J. M. Garrido
Polynomial Models with Python

1 import numpy as np
2 print "Vector arithmetic operations"
3 p = np.zeros(8)
4 p = p + 3.5
5 print "Array p: ", p
6 q = p * 3
7 print "Array q: ", q
8 q3 = q + p
9 print "Array q3: ", q3
10 q4 = q - p
11 print "Array q4: ", q4

Executing the Python interpreter with program test4_arrays.py yields the following output.

$python test4_arrays.py
Vector arithmetic operations
Array p: [ 3.5  3.5  3.5  3.5  3.5  3.5  3.5  3.5]
Array q: [10.5 10.5 10.5 10.5 10.5 10.5 10.5 10.5]
Array q4: [ 7.  7.  7.  7.  7.  7.  7.  7.]

5.2 Element Multiplication and Division Operations

Applying element by element multiplication, the corresponding elements of two vectors are multiplied. This operation is applied to two vectors of equal size. This operation multiplies the elements of the specified vector by the elements of the second specified vector. Using element by element division, the corresponding elements of two vectors are divided. This operation is applied to two vectors of equal size.

The following program shows element-wise multiplication and division with vectors. In line 8, vectors p and q are multiplied and the results are stored in vector p3. In line 10, vector q is divided by vector p and the results are stored in vector q4. The program is stored in file test5_arrays.py.

1 import numpy as np
2 print "Element multiplication and division operations"
3 p = np.zeros(8)
4 p = p + 3.5
5 print "Array p: ", p
6 q = p * 3
7 print "Array q: ", q

© 2015 J. M. Garrido
Polynomial Models with Python

8 q3 = q * p
9 print "Array q3: ", q3
10 q4 = q / p
11 print "Array q4: ", q4

Executing the Python interpreter with program test5_arrays.py yields the following output.

$ python test5_arrays.py
Element multiplication and division operations
Array p: [ 3.5  3.5  3.5  3.5  3.5  3.5  3.5  3.5]
Array q: [10.5 10.5 10.5 10.5 10.5 10.5 10.5 10.5]
Array q3: [36.75 36.75 36.75 36.75 36.75 36.75 36.75 36.75]
Array q4: [ 3.  3.  3.  3.  3.  3.  3.  3.]

5.3 Vector Multiplication

Multiplication of vectors is carried out by using the Numpy function dot or vdot. This operation is known as the dot multiplication of vectors. The following program stored in file test5b_arrays illustrates these operations. Lines 8 and 10 apply dot multiplication of vectors p and q.

1 import numpy as np
2 print "Vector dot multiplication"
3 p = np.zeros(8)
4 p = p + 3.5
5 print "Array p: ", p
6 q = p * 3
7 print "Array q: ", q
8 q4 = np.dot(p,q)
9 print "Dot product of p, q: ", q4
10 q4v = np.vdot(p,q)
11 print "Dot product of p, q", q4v

Executing the Python interpreter with program test5b_arrays.py yields the following output.
Vector dot multiplication
Array p: [ 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5]
Array q: [ 10.5 10.5 10.5 10.5 10.5 10.5 10.5 10.5]
Dot product of p, q: 294.0
Dot product of p, q 294.0

5.4 Additional Vector Operations

In addition to slicing, selective indexing can also be carried out using arrays of indices. The following program stored in file test9_arrays.py creates an array of indices in line 6 and applies it to array p to select the corresponding elements in line 7. These selected elements are assigned the value 2.0 in line 9.

```python
1 import numpy as np
2 print "Array of indices"
3 p = np.array([1.5, 2.5, 3.5, 4.5, 5.5, 6.5, 7.5, 8.5])
4 print "Array p: "
5 print p
6 pindx = np.array([1, 4, 6, 7])
7 q = p[pindx]
8 print "q: ", q
9 p[pindx] = 2.0
10 print "Array p:"
11 print p
```

Executing the Python interpreter with program test9_arrays.py yields the following output.

$ python test9_arrays.py
Array of indices
Array p:
[ 1.5 2.5 3.5 4.5 5.5 6.5 7.5 8.5]
q: [ 2.5 5.5 7.5 8.5]
Array p:
[ 1.5 2.0 3.5 4.5 2.0 6.5 2.0 2.0 ]

Various additional operations can be applied to vectors. For vector assignment, the resulting vector is only a view of the first vector. The two vectors refer to the same list of values. For the copy operation, method copy is called, the two vectors

© 2015 J. M. Garrido
must have the same length and the element values are all copied to the second vector.

The vector comparison operation is applied to the entire vectors as a whole by calling Numpy function `array_equal`. The result is a single boolean value. The element comparison of two vectors for equality is performed element by element by calling Numpy function `equal` and the operation creates a new vector of the same size with values `True` or `False`. Similarly, when comparing for greater than operation, the Numpy function `greater` is called.

The following program illustrates the use of these operations. Line 10 is an assignment of vector `q` to vector `qq`. The two vectors now refer to the same list of values. In line 12, the elements of vector `p` are copied to vector `qqq` by calling method `copy` and `qqq` becomes a new vector. In line 14, Numpy function `array_equal` compares vectors `p` and `qqq`. This produces a single boolean result. In line 16, the Numpy function `equal` is called to compare the elements of vector `p` and `qq`, the result is another vector with the boolean result of the comparison for each corresponding pair of element values. In a similar manner, the Numpy function `greater` is called to compare the elements of vectors `q` and `p` in line 18.

```python
import numpy as np
print "Vector assignment, comparisons operations"
p = np.zeros(8)
p = p + 3.5
p[2] = 1.75
print "Array p: ", p
q = p * 3
print "Array q: ", q
qq = q
print "Array qq: ", qq
qqq = p.copy()
print "Array qqq: ", qqq
yn = np.array_equal(p, qqq)
print "Vector equality: ", yn
yn = np.equal(p, qq) # p == qq
print "Element equality: ", yn
byn = np.greater(q, p) # q > p
print "Greater: ", byn
```

Executing the Python interpreter with program `test6_arrays.py` yields the following output.
$ python test6_arrays.py

Vector assignment, comparisons operations
Array p: [ 3.5  3.5  1.75  3.5  3.5  3.5 12.35  3.5 ]
Array q: [ 10.5 10.5  5.25 10.5 10.5 10.5 37.05 10.5 ]
Array qq: [ 10.5 10.5  5.25 10.5 10.5 10.5 37.05 10.5 ]
Array qqq: [ 3.5  3.5  1.75  3.5  3.5  3.5 12.35  3.5 ]

Vector equality: True
Element equality: [False False False False False False False False]
Greater: [ True True True True True True True True ]

A boolean expression can be used in indexing, and this is very convenient to select the elements for which the boolean expression is true. In the following program, a boolean expression is applied to all elements of array \( p \) in line 6. The array of truth values created, \( pindx \), is used to index array \( p \).

1 import numpy as np
2 print "Indexing with boolean expression"
3 p = np.array([1.5, 2.5, 3.5, 4.5, 5.5, 6.5, 3.25, 1.65])
4 print "Array p: "
5 print p
6 pindx = p <= 2.25
7 print "Array pindx: ", pindx
8 q = p[pindx]
9 print "q: ", q

Executing the Python interpreter with program tmat11.py yields the following output.

$ python tmat11.py

Indexing with boolean expression
Array p:
[ 1.5  2.5  3.5  4.5  5.5  6.5  3.25  1.65]
Array pindx: [ True False False False False False False True]
q: [ 1.5  1.65]

Function \( \text{max} \) gets the maximum value stored in the specified vector. The following function call gets the maximum value in vector \( pv \) and assigns this value to \( x \).

© 2015 J. M. Garrido
In addition to the maximum value in a vector, the index of the element with that value may be desired. Calling function `argmax` returns the index value of the element with the maximum value in a specified vector. In the following function call, the index value of the element with the maximum value in vector \( pv \) is returned and assigned to integer variable \( idx \).

\[
\text{idx} = \text{numpy.argmax (pv)}
\]

In a similar manner, function `min` gets the minimum value stored in a vector. Function `argmin` returns the index value of the minimum value in the specified vector.

Listing 2 shows a Python source listing of a program that includes the application of the operations on vectors that have been discussed and some additional ones. The program is stored in file `vectorops.py`.

Listing 2: Program that shows various operations on vectors.

```python
import numpy as np
print "Vector maximum, minimum, sum, mean, other operations"
p = np.zeros(8)
p = p + 3.5
p[2] = 1.75
print "Array p: ", p
q = p * 3
print "Array q: ", q
x = np.max(p)
print "Max in vector p: ", x
idx = np.argmax(p)
print "Index of max in p: ", idx
y = np.min(q)
print "Min in vector q: ", y
ydx = np.argmin(q)
print "Index of min in q: ", ydx
mym = np.mean(p)
print "Mean of p: ", mym
xm = np.median(p)
print "Median of p: ", xm
mystd = p.std()
print "STD of p: ", mystd
```

© 2015 J. M. Garrido
24 mysum = p.sum()
25 print "Sum of p: ", mysum

Executing the Python interpreter with program vectorops.py yields the following output.

$ python vectorops.py
Vector maximum, minimum, sum, mean, other operations
Array p: [ 3.5  3.5  1.75  3.5  3.5  3.5  12.35
   3.5 ]
Array q: [ 10.5  10.5  5.25  10.5  10.5  10.5  37.05
   10.5 ]
Max in vector p: 12.35
Index of max in p: 6
Min in vector q: 5.25
Index of min in q: 2
Mean of p: 4.3875
Median of p: 3.5
STD of p: 3.06357124122
Sum of p: 35.1