Multi-Dimensional Arrays

MCLA

Computer Science Department

Mark A. Cohen, Ph.D

Array Arithmetic Useful in ML

4	8
2	1

x 2 =

8	16
4	2

4	8
2	1

/ 2 =

2	4
1	0.5

4	8
2	1

== 2 =

False	False
True	False

4	8
2	9

>

7	1
3	2

=

False	True
False	True

4	8
2	1

*

2	1
3	2

=

8	8
6	2

NOTE: Not matrix multiplication!

Arrays (lists) in Python

Dimensions	Syntax
1 x 4 array	myAry = [7, 3, 8, 2]
2 x 2 array	myAry = [[3, 7], [2, 1]]
3 x 2 array	myAry = [[1, 3], [2, 5], [8, 2]]
3 x 3 array	myAry = [[1, 3, 7], [2, 5, 9], [8, 2, 1]]

1	3	7
2	5	9
8	2	1

3	7
2	1

The Trouble With Python Lists of Lists

- When printed they look like lists of lists instead of multi-dimensional arrays.
- They don’t support array arithmetic.
- Slicing lists of lists is troublesome.

```
a = [[1, 2, 3], [4, 5, 6], [7, 8, 9] ]
print(a)
```

```
[[1, 2, 3], [4, 5, 6], [7, 8, 9]]
```

```
a = [[1, 2, 3], [4, 5, 6], [7, 8, 9] ]
b = [[2, 2, 2], [2, 2, 2], [2, 2, 2] ]
c = a * b
```

```
Traceback (most recent call last):
  File "arrays.py", line 3, in <module>
    c = a * b

TypeError: can't multiply sequence by non-int of type 'list'
```

NumPy Arrays

They Print Nice!

```
a =
[[1 2 3]
 [4 5 6]
 [7 8 9]]
```

Slicing Works Great!

```
a =
[[1 2 3]
 [4 5 6]
 [7 8 9]]

a[0:3, 1] = [2 5 8]
```

They Support Array Arithmetic!

```
a =
[[1 2 3]
 [4 5 6]
 [7 8 9]]

b =
[[1 1 1]
 [1 1 1]
 [2 2 2]]

a - b =
[[0 1 2]
 [3 4 5]
 [5 6 7]]
```

More On Slicing

a[row-start : up-to-row-end, col-start : up-to-col-end]

a[0, 0]

4	8
2	1

a[0, 0:2] or a[0, :]

4	8
2	1

a[1, 0:2] or a[1, :]

4	8
2	1

a[0:2, 0] or a[:, 0]

4	8
2	1

a[0:2, 1] or a[:, 1]

4	8
2	1

a[0:2, 0:2]

4	8	7
2	1	3
9	5	6

a[1:3, 0:2]

4	8	7
2	1	3
9	5	6

a[0:2, 1:3]

4	8	7
2	1	3
9	5	6

a[:, 0:2]

4	8	7
2	1	3
9	5	6

a[1:3, :]

4	8	7
2	1	3
9	5	6

More On Slicing

a

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

a[1:3, 0:3] * 2

4	2	6
18	10	12

a[:, 1:3] > 4

True	True
False	False
True	True
False	False

a[1, 1:3] * a[3, 0:2]

2	9