Mini Lecture on Numpy

CS7646 – Georgia Tech

James Chan

9/15/17

What is Numpy?

- Library for variety of basic and advanced numeric computations
- Optimized for matrix and multi-dimensional array operations
- May help eliminate loops under the right circumstance, which significantly reduce running time. This is known as vectorization.
- The underlying data structure of pandas is in numpy.
- Commonly imported as just “np”
- Make sure you watch Udacity video *01-03 The Power of Numpy*

np.ndarray

- Stands for n-dimensional array.
- Why do we care? Decision tree is represented in table form!

N =

1	5

1-d array

N.shape = (2,)
N.shape[0] = 2

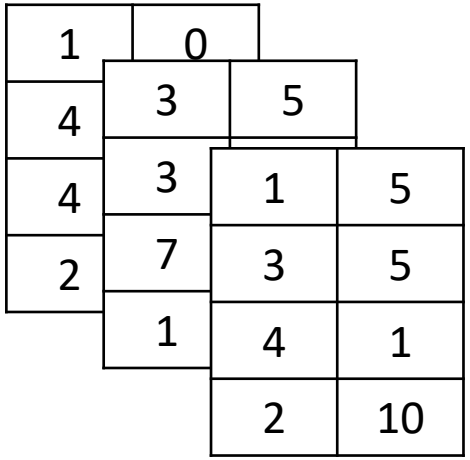
N =

1	5
3	5
4	1
2	10

2-d array

N.shape = (4,2)
N.shape[0] = 4
N.shape[1] = 2

N =



3-d array

N.shape = (3,4,2)
N.shape[0] = 3
N.shape[1] = 4
N.shape[2] = 2

Construct a 2d-array

1	1
1	1
1	1
1	1

N = np.ones((4,2))

1	2
3	4
5	6
7	8

N = np.array([[1,2],[3,4],[5,6],[7,8]])

0.2151651	0.5191198
0.9984162	0.3191896
0.1945381	0.6999632
0.1144863	0.1848499

N = np.random.random((4,2))

Slicing

N =

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

- Zero based indexing
- ":" used to specify range
- 1:3 means elements between 1 inclusive and 3 exclusive.
- N[*row specifier*, *column specifier*]

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

N[:,:]

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

N[2,3]

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

N[1:,:4]

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

N[0,2:4]

Modify Array

N =

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

N[:,:] = 0

1	2	3	4
5	6	7	8
9	10	11	0
13	14	15	16

N[2,3] = 0

1	2	3	4
0	0	0	8
0	0	0	12
0	0	0	16

N[1:,:4] = 0

1	2	0	0
5	6	7	8
9	10	11	12
13	14	15	16

N[0,2:4] = 0

Concatenate

- What is it? Attach tables together to form bigger table.
- There are many short-hand for concatenating. I will show you just one.
- np.concatenate()

Concatenate (continued)

A =

7
3
5
2

B =

1	2
5	6
9	10
13	14

C =

4	2
4	6

- Examples of proper concatenation
- Axis = 1 to concat to column
- Axis = 0 to concat to row

7	1	2
3	5	6
5	9	10
2	13	14

N = np.concatenate([A,B],axis=1)

1	2	7
5	6	3
9	10	5
13	14	2

N = np.concatenate([B,A],axis=1)

1	2
5	6
9	10
13	14
4	2
4	6

N = np.concatenate([B,C],axis=0)

Concatenate (continued)

A =

7
3
5
2

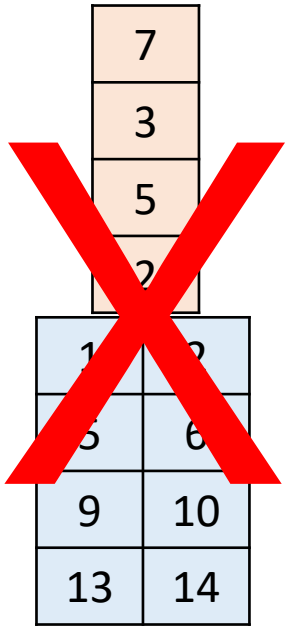
B =

1	2
5	6
9	10
13	14

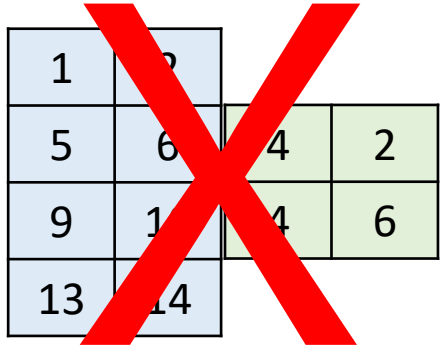
C =

4	2
4	6

- Illegal concatenation!
- Pay attention to your axis



N = np.concatenate([A,B],axis=0)



N = np.concatenate([B,C],axis=1)

Numerical operations

N =

1	2
3	4

- Basic operations are carried out in element-wise manner.

3	6
9	12

N * 3

3	4
5	6

N + 2

2	4
6	8

N + N

Numerical operations (continued)

N =

1	2
3	4

- Basic operations are carried out in element-wise manner.
- May specify axis

Sum of everything

1	2
3	4

=

10

np.sum(N)

Sum this way only ↓

1	2
3	4

=

4	6

np.sum(N, axis = 0)

Sum this way only →

1	2
3	4

=

3
7

np.sum(N, axis = 1)

Numerical operations (continued)

N =

0	2	3
4	6	6
8	4	9

- Supports common statistical operations
- Mean, Median, Max, Min

Max of everything

0	2	3
4	6	6
8	4	9

= 9

np.max(N)

Average this way ↓

0	2	3
4	6	6
8	4	9

=

4	4	6

np.mean(N, axis = 0)

Median this way →

0	2	3
4	6	6
8	4	9

=

2
6
8

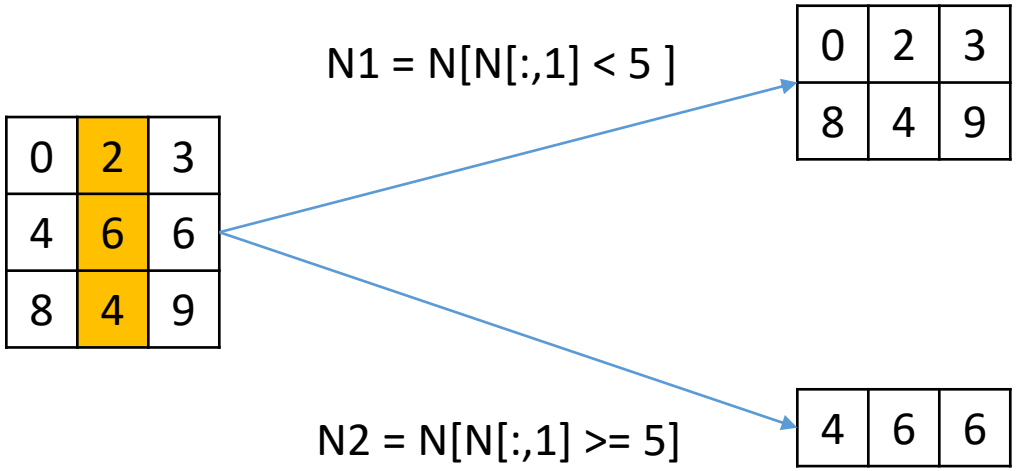
np.median(N, axis = 1)

Masking

N =

0	2	3
4	6	6
8	4	9

- Quick way to separate data by logic.



Divide the table into two parts by whether the elements in N[:,1] is larger or smaller than 5

Not covered

- Vectorization: will cover later in class by Dave