1 Numpy

Numpy is the library that supports linear algebra operations. The two most important data structures are the n-dimensional array and the matrix. The matrix always has 2 dimensions. It is the matrix you should be using for most of your code in this course, so let’s look at some matrix operations in the Python interpreter.

First, let’s make a row vector, storing it as a matrix:

```python
>>> import numpy as np
>>> a = np.matrix([[1, 3]])
>>> a
matrix([[1, 3]])
```

To make a column vector, we take the transpose.

```python
>>> b = a.T
>>> b
matrix([[1], [3]])
```

But b is not a copy of a, merely another way to view the same data. We see this by changing a value of a and then looking the the contents of b (they have been affected, too). To access an element we use two indices (always use 2 indices for a matrix, even if it has only one row or only one column!). The row index is first and the column index is second.

```python
>>> a[0, 0] = 4
>>> a
matrix([[4, 3]])
>>> b
matrix([[1], [3]])
```

To decouple b from a, we make a copy of a:

```python
>>> b = a.copy().T
>>> b
matrix([[4], [3]])
>>> a[0, 1] = 2
>>> a
matrix([[4, 2]])
>>> b
```

To see the shape of `a`, we use the `shape` field. This is very helpful for debugging. Instead of printing the contents of `a`, you can print the shape. It is a tuple: the first element is the number of rows, the second is the number of columns.

```python
>>> print a.shape
(1, 2)
```

Now let’s look at slicing. First make a $3 \times 3$ matrix.

```python
>>> A = np.matrix([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
```

Now we take the third column and then the second row.

```python
>>> a2 = A[:, 2]
>>> print a2
[[3]
 [6]
[9]]
```

```python
>>> ab = A[1, :]
...]
```

```python
>>> ab
matrix([[4, 5, 6]])
```

```python
And we try to change an entry in the row vector `ab` using only one index. It fails:

```python
>>> ab[1] = 10
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IndexError: index 1 is out of bounds for axis 0 with size 1
```

So, like good little programmers, we use two indices and it works. Also we see that we are changing the data in `A`. So, the slices are just views on the same data.

```python
>>> ab[0, 1] = 10
```

```python
>>> a2
matrix([[3],
 [6],
[9]])
```

```python
>>> a2[1,0] = 3.0
```

```python
>>> a2
matrix([[3],
 [3],
[9]])
```

```python
>>> A
matrix([[1, 2, 3],
 [4, 10, 3],
[7, 8, 9]])
```

The take-home message:

- Matrices are always 2D and you should use two indices to slice them or index into them.
• Slicing and transposing a matrix does not also copy the matrix.