Vector Norms

p-norms can be computed in two different ways in numpy:

In [1]:

```
#keep
import numpy as np
import numpy.linalg as la
```

In [2]:

```
#keep
x = np.array([1.,2,3])
```

First, let's compute the 2-norm by hand:

In [3]:

```
np.sum(x**2)**(1/2)
```

Out[3]:

```
1.0
```

Next, let's use `numpy` machinery to compute it:

In [4]:

```
la.norm(x, 2)
```

Out[4]:

```
3.7416573867739413
```

Both of the values above represent the 2-norm: $\|x\|_2$.

Different values of p work similarly:

In [5]:

```
#keep
np.sum(np.abs(x)**5)**(1/5)
```

Out[5]:

```
1.0
```

In [6]:

```
la.norm(x, 5)
```

Out[6]:

```
3.0773848853940629
```

The ∞ norm represents a special case, because it's actually (in some sense) the *limit* of p-norms as $p \to \infty$.

Recall that: $\|x\|_\infty = \max(|x_1|, |x_2|, |x_3|)$.

Where does that come from? Let's try with $p = 100$:

In [7]:

```
x**100
```

Out[7]:

```
array([  1.00000000e+00,   1.26765060e+30,   5.15377521e+47])
```

In [8]:

```
np.sum(x**100)
```

Out[8]:

```
5.1537752073201132e+47
```

Compare to last value in vector: the addition has essentially taken the maximum:

In [9]:

```
np.sum(x**100)**(1/100)
```

Out[9]:

```
1.0
```

Numpy can compute that, too:

In [10]:

```
la.norm(x, np.inf)
```

Out[10]:

```
3.0
```

In [ ]:

```

```