Basic Linear Algebra

For this topic you need to know a little bit of linear algebra. I know that this class has a high variety of mathematical background, so I will assume only that you have seen some matrices and vectors in high school, and know what they are. If you are at all interested in learning more about data analysis or networks, I recommend taking a linear algebra class first and foremost. Here are the basic concepts you will need to know. They are all shown on small vectors and matrices, but it should be clear how this generalizes to larger ones. Let

- **Norm of a vector.** The p-norm of a vector is a generalized way of measuring its magnitude. Suppose that $\vec{x} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$. Then the p-norm of $\vec{x}$ is given by $\|\vec{x}\|_p = \sqrt[p]{x_1^p + x_2^p + x_3^p}$. When $p = 1$, this is simply the sum of the elements of $\vec{x}$. When $p = 2$, this is the Euclidean magnitude of the vector. For example, $\left\| \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} \right\|_2 = \sqrt{14}$.

```
import numpy as np
from numpy import linalg as LA
x=np.array([1,2,3])
LA.norm(x,2)
```

- **Dot product.** The dot product of two vectors $\vec{x}$ and $\vec{y}$ is defined as $\vec{x} \cdot \vec{y} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \cdot \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = x_1y_1 + x_2y_2 + x_3y_3$. For example, $\begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} \cdot \begin{pmatrix} 3 \\ 4 \\ 3 \end{pmatrix} = 20$.

```
y=np.array([3,4,3])
np.dot(x,y)
```

- **Angle between two vectors.** The angle between two vectors can be computed using dot products and norms. The angle θ between two vectors $\vec{x}$ and $\vec{y}$ is computed as $\cos(\theta) = \frac{\vec{x} \cdot \vec{y}}{\|\vec{x}\|_2 \times \|\vec{y}\|_2}$. For example, the cosine of the angle between vectors $\begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$ and $\begin{pmatrix} 3 \\ 4 \\ 3 \end{pmatrix}$ is given by $\cos(\theta) = \frac{\begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} \cdot \begin{pmatrix} 3 \\ 4 \\ 3 \end{pmatrix}}{\left\| \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} \right\|_2 \times \left\| \begin{pmatrix} 3 \\ 4 \\ 3 \end{pmatrix} \right\|_2} = \frac{20}{\sqrt{14 \cdot 34}}$. Using numpy, we can find θ using the arc cosine function (inverse cosine). Note that the angle is given in radians.

```
costheta=np.dot(x,y)/(LA.norm(x,2)*LA.norm(y,2))
np.arccos(costheta)
```

- **Matrix multiplication.** Matrix multiplication of an $n \times p$ matrix A and an $p \times m$ matrix B yields an $n \times m$ matrix C, where each element $C[i, j]$ is the dot product of the ith row of A and the jth column of B. For example: $\begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix} \begin{pmatrix} 0 & 4 \\ 3 & 1 \end{pmatrix} = \begin{pmatrix} 6 & 6 \\ 3 & 9 \end{pmatrix}$. In numpy:

```
A=np.array([[1,2],[2,1]])
B=np.array([[0,4],[3,1]])
np.matmul(A,B)
```

Pearson Correlation Coefficient

The Pearson correlation coefficient, r, is a commonly used measure of linear correlation between two variables $\vec{x}$ and $\vec{y}$, taking the forms of two sample datasets $\vec{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$ and $\vec{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}$. The usual definition does not use linear algebra and looks something like:

$$r = \frac{\sum_{i=1}^{n}(x_i - \mu_x)(y_i - \mu_y)}{\sqrt{\sum_{i=1}^{n}(x_i - \mu_x)^2}\sqrt{\sum_{i=1}^{n}(y_i - \mu_y)^2}}.$$

If we "centralize" the vectors $\vec{x}$ and $\vec{y}$ around the means, or in other words, consider the deviation vectors $\vec{x_c} = \begin{pmatrix} x_1 - \mu_x \\ x_2 - \mu_x \\ \vdots \\ x_n - \mu_x \end{pmatrix}$ and $\vec{y_c} = \begin{pmatrix} y_1 - \mu_y \\ y_2 - \mu_y \\ \vdots \\ y_n - \mu_y \end{pmatrix}$, then we see that

$$r = \cos(\theta) = \frac{\vec{x_c} \cdot \vec{y_c}}{\|\vec{x_c}\|_2 \times \|\vec{y_c}\|_2}.$$

This corresponds to thinking of the centralized datasets as two vectors in n-dimensional space, where each dimension corresponds to an observation. These two vectors are correlated if they point in the same direction, i.e. if a positive deviation from the mean of one vector in some direction results in a positive deviation of the other vector. (And same for negative deviations.) The two variables are anti-correlated if they point in opposite directions. Remember that the cosine of an angle is 1 when the angle is 0, and -1 when the angle is 180°.

A big advantage of using linear algebra when programming is that numpy, for example, has built-in linear algebra functions that are highly optimized for speed. In general, they are much faster than for-loops. For data analysis this is important because interesting datasets are usually large.