Vectors, Matrices, and Linear Algebra

Linear Algebra is strikingly similar to the algebra you learned in high school, except that in the place of ordinary single numbers, it deals with vectors. Many of the same algebraic operations you're used to performing on ordinary numbers (a.k.a. scalars), such as addition, subtraction and multiplication, can be generalized to be performed on vectors. We'll better start by defining what we mean by *scalars* and *vectors*.

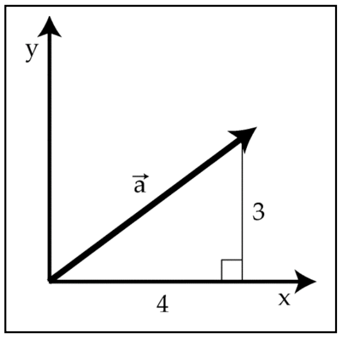
Definition: A scalar is a number. Examples of scalars are temperature, distance, speed, or mass – all quantities that have a magnitude but no "direction", other than perhaps positive or negative.

Okay, so scalars are what you're used to. In fact we could go so far as to describe the algebra you learned in grade school as *scalar* algebra, or the calculus many of us learned in high school as *scalar* calculus, be cause both dealt almost exclusively with scalars. This is to be contrasted with *vector* calculus or *vector* algebra, that most of us either only got in college if at all. So what *is* a vector?

Definition: A vector is *a list of numbers*. There are (at least) two ways to interpret what this list of numbers mean: One way to think of the vector as being *a point in a space*. Then this list of numbers is a way of identifying that point in space, where each number represents the vector's component that dimension. Another way to think of a vector is *a magnitude and a direction*, e.g. a quantity like velocity ("the fighter jet's velocity is 250 mph north-by-northwest"). In this way of think of it, a vector is a directed arrow pointing from the origin to the end point given by the list of numbers.

In this class we'll denote vectors by including a small arrow overtop of the symbol like so: $\vec{a}$. Another common convention you might encounter in other texts and papers is to denote vectors by use of a boldface font ($\boldsymbol{a}$). An example of a vector is $\vec{a} = [4,3]$. Graphically, you can think of this vector as an arrow in the *x*-*y* plane, pointing from the origin to the point at *x*=3, *y*=4 (see illustration.)

In this example, the list of numbers was only two elements long, but in principle it could be any length. The dimensionality of a vector is the length of the list. So, our example $\vec{a}$ is 2-dimensional because it is a list of two numbers. Not surprisingly all 2-dimentional vectors live in a plane. A 3-dimensional vector would be a list of three numbers, and they live in a 3-D volume. A 27-dimensional vector would be a list of twenty-seven numbers, and would live in a space only Ilana's dad could visualize.



Magnitudes and direction

The “magnitude” of a vector is the distance from the endpoint of the vector to the origin – in a word, it’s length. Suppose we want to calculate the magnitude of the vector $\vec{a} = [4,3]$. This vector extends 4 units along the x-axis, and 3 units along the y-axis. To calculate the magnitude $\|\vec{a}\|$ of the vector we can use the Pythagorean theorem ($x^2 + y^2 = z^2$).

$$\|\vec{a}\| = \sqrt{x^2 + y^2} = \sqrt{4^2 + 3^2} = 5$$

The magnitude of a vector is a scalar value – a number representing the length of the vector independent of the direction. There are a lot of examples were the magnitudes of vectors are important to us: velocities are vectors, speeds are their magnitudes; displacements are vectors, distances are their magnitudes.

To complement the magnitude, which represents the length independent of direction, one might wish for a way of representing the direction of a vector independent of its length. For this purpose, we use “unit vectors,” which are quite simply vectors with a magnitude of 1. A unit vector is denoted by a small “carrot” or “hat” above the symbol. For example, $\hat{a}$ represents the unit vector associated with the vector $\vec{a}$. To calculate the unit vector associated with a particular vector, we take the original vector and divide it by its magnitude. In mathematical terms, this process is written as:

$$\hat{a} = \frac{\vec{a}}{\|\vec{a}\|}$$

Definition: A unit vector is a vector of magnitude 1. Unit vectors can be used to express the direction of a vector independent of its magnitude.

Returning to the previous example of $\vec{a} = [4,3]$, recall $\|\vec{a}\| = 5$. When dividing a vector ($\vec{a}$) by a scalar ($\|\vec{a}\|$), we divide each component of the vector individually by the scalar. In the same way, when multiplying a vector by a scalar we will proceed component by component. Note that this will be very different when multiplying a vector by another vector, as discussed below. But for now, in the case of dividing a vector by a scalar we arrive at:

$$\hat{a} = \frac{[4,3]}{5}$$

$$\hat{a} = \left[\frac{4}{5}, \frac{3}{5}\right]$$

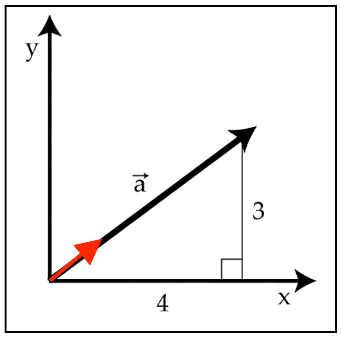
As shown in red in the figure, by dividing each component of the vector by the same number, we leave the direction of the vector unchanged, while we change the magnitude. If we have done this correctly, then the magnitude of the unit vector must be equal to 1 (otherwise it would not be a unit vector). We can verify this by calculating the magnitude of the unit vector $\|\hat{a}\|$.

$$\|\hat{a}\|^2 = \left(\frac{4}{5}\right)^2 + \left(\frac{3}{5}\right)^2$$

$$\|\hat{a}\|^2 = \left(\frac{16}{25}\right) + \left(\frac{9}{25}\right)$$

$$\|\hat{a}\|^2 = \left(\frac{25}{25}\right) = 1$$

$$\|\hat{a}\| = 1$$



So we have demonstrated how to create a unit vector $\hat{a}$ that has a magnitude of 1 but a direction identical to the vector $\vec{a}$. Taking together the magnitude $\|\vec{a}\|$ and the unit vector $\hat{a}$ we have all of the information contained in the vector $\vec{a}$, but neatly separated into its magnitude and direction components. We can use these two components to re-create the vector $\vec{a}$ by multiplying the vector $\hat{a}$ by the scalar $\|\vec{a}\|$ like so:

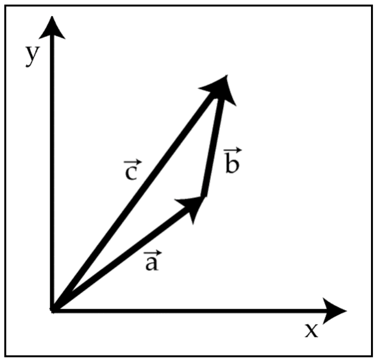
$$\vec{a} = \hat{a} * \|\vec{a}\|$$

Vector addition and subtraction

Vectors can be added and subtracted. Graphically, we can think of adding two vectors together as placing two line segments end-to-end, maintaining distance and direction. An example of this is shown in the illustration, showing the addition of two vectors $\vec{a}$ and $\vec{b}$ to create a third vector $\vec{c}$.

$$\vec{a} + \vec{b} = \vec{c}$$

Numerically, we add vectors component-by-component. That is to say, we add the x components together, and then separately we add the y components together. For example, if $\vec{a} = [4,3]$ and $\vec{b} = [1,2]$, then:



$$\vec{c} = \vec{a} + \vec{b}$$

$$\vec{c} = [4,3] + [1,2]$$

$$\vec{c} = [4 + 1, 3 + 2]$$

$$\vec{c} = [5,5]$$

Similarly, in vector subtraction:

$$\vec{c} = \vec{a} - \vec{b}$$

$$\vec{c} = [4,3] - [1,2]$$

$$\vec{c} = [3,1]$$

Vector addition has a very simple interpretation in the case of things like displacement. If in the morning a ship sailed 4 miles east and 3 miles north, and then in the afternoon it sailed a further 1 mile east and 2 miles north, what was the total displacement for the whole day? 5 miles east and 5 miles north – vector addition at work.

Linear independence

If two vectors point in different directions, even if they are not very different directions, then the two vectors are said to be *linearly independent*. If vectors $\vec{a}$ and $\vec{b}$ point in the same direction, then you can multiply vector $\vec{a}$ by a constant, scalar value and get vector $\vec{b}$, and vice versa to get from $\vec{b}$ to $\vec{a}$. If the two vectors point in different directions, then this is not possible to make one out of the other because multiplying a vector by a scalar will never change the direction of the vector, it will only change the magnitude. This concept generalizes to families of more than two vectors. Three vectors are said to be linearly independent if there is no way to construct one vector by combining scaled versions of the other two. The same definition applies to families of four or more vectors by applying the same rules.

The vectors in the previous figure provide a graphical example of linear independence. Vectors $\vec{a}$ and $\vec{c}$ point in slightly different directions. There is no way to change the length of vector $\vec{a}$ and generate vector $\vec{c}$, nor vice-versa to get from $\vec{c}$ to $\vec{a}$. If, on the other hand, we consider the family of vectors that contains $\vec{a}$, $\vec{b}$ and $\vec{c}$, it is now possible, as shown, to add vectors $\vec{a}$ and $\vec{b}$ to generate vector $\vec{c}$. So the family of vectors $\vec{a}$, $\vec{b}$ and $\vec{c}$ is *not* linearly independent, but is instead said to be linearly dependent. Incidentally, you could change the length of any or all of these three vectors and they would still be linearly dependent.

Definition: A family of vectors is linearly independent if no one of the vectors can be created by any linear combination of the other vectors in the family. For example, $\vec{c}$ is linearly independent of $\vec{a}$ and $\vec{b}$ if and only if it is *impossible* to find scalar values of α and β such that $\vec{c} = \alpha\vec{a} + \beta\vec{b}$

Vector multiplication: dot products

Next we move into the world of vector multiplication. There are two principal ways of multiplying vectors, called *dot products* (a.k.a. *scalar products*) and *cross products*. The dot product:

$$d = \vec{a} \cdot \vec{b}$$

generates a scalar value from the product of two vectors and will be discussed in greater detail below. Do not confuse the dot product with the cross product:

$$\vec{c} = \vec{a} \times \vec{b}$$

which is an entirely different beast. The cross product generates a vector from the product of two vectors. Cross products so up in physics sometimes, such as when describing the interaction between electrical and magnetic fields (ask your local fMRI expert), but we'll set those aside for now and just focus on dot products in this course. The dot product is calculated by multiplying the x components, then separately multiplying the y components (and so on for z, etc… for products in more than 2 dimensions) and then adding these products together. To do an example using the vectors above:

$$\vec{a} \cdot \vec{b} = [4,3] \cdot [1,2]$$

$$\vec{a} \cdot \vec{b} = (4 * 1) + (3 \ * 2)$$

$$\vec{a} \cdot \vec{b} = 11$$

Another way of calculating the dot product of two vectors is to use a geometric means. The dot product can be expressed geometrically as:

$$\vec{a} \cdot \vec{b} = \|\vec{a}\| \|\vec{b}\| \cos\theta$$

where θ represents the angle between the two vectors. Believe it or not, calculation of the dot product by either procedure will yield exactly the same result. Recall, again from high school geometry, that $\cos 0° = 1$, and that $\cos 90° = 0$. If the angle between $\vec{a}$ and $\vec{b}$ is nearly $0°$ (i.e. if the vectors point in nearly the same direction), then the dot product of the two vectors will be nearly $\|\vec{a}\| \|\vec{b}\|$.

Definition: A dot product (or scalar product) is the numerical product of the lengths of two vectors, multiplied by the cosine of the angle between them.

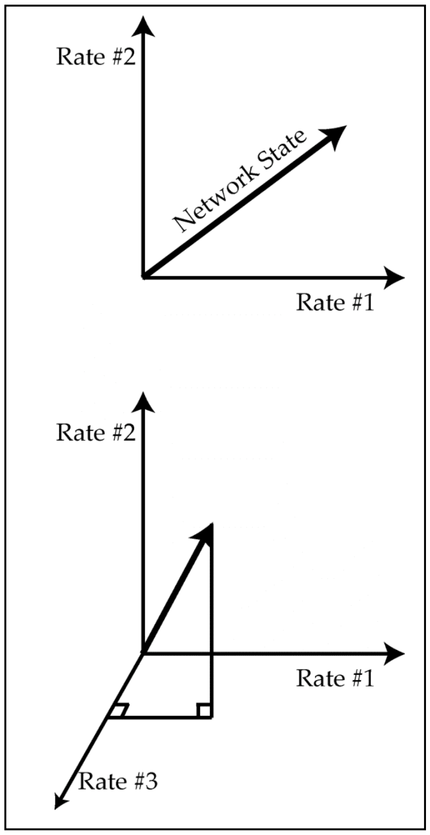
Orthogonality

As the angle between the two vectors opens up to approach $90°$, the dot product of the two vectors will approach 0, regardless of the vector magnitudes $\|\vec{a}\|$ and $\|\vec{b}\|$. In the special case that the angle between the two vectors is exactly $90°$, the dot product of the two vectors will be 0 regardless of the magnitude of the vectors. In this case, the two vectors are said to be *orthogonal*.

Definition: Two vectors are orthogonal to one another if the dot product of those two vectors is equal to zero.

Orthogonality is an important and general concept, and is a more mathematically precise way of saying "perpendicular." In two- or three-dimensional space, orthogonality is identical to perpendicularity and the two ideas can be thought of interchangeably. Whereas perpendicularity is restricted to spatial representations of things, orthogonality is a more general term. In the context of neural networks, neuroscientists will often talk in terms of two patterns of neuronal firing being orthogonal to one another. Orthogonality can also apply to functions as well as to things like vectors and firing rates. As we will discuss later in this class in the context of *fourier transforms*, the sin and cos functions can be said to be orthogonal functions. In any of these contexts, orthogonality will always mean something akin to "totally independent" and is specifically refering to two things having a dot product of zero.

Vector spaces



All vectors live within a *vector space*. A vector space is exactly what it sounds like – the space in which vectors live. When talking about spatial vectors, for instance the direction and speed with which a person is walking through a room, the vector space is intuitively spatial since all available directions of motion can be plotted directly onto a spatial map of the room.

A less spatially intuitive example of a vector space might be all available states of a neural network. Imagine a very simple network, consisting of only five neurons which we will call n_1, n_2, n_3, n_4, and n_5. At each point in time, each neuron might not fire any action potentials at all, in which case we write $n_i = 0$, where i denotes the neuron number. Alternatively, the neuron might be firing action potentials at a rate of up to 100 Hz, in which case we write that $n_i = x$, where $0 \le x \le 100$. The state of this network at any moment in time can be depicted by a vector that describes the firing rates of all five neurons:

$$s = [n_1, n_2, n_3, n_4, n_5]$$

The set of all possible firing rates for all five neurons represents a vector space that is every bit as real as the vector space represented by a person walking through a room. The vector space represented by the neurons, however, is a 5-dimensional vector space. The math is identical to the two dimensional situation, but in this case we must trust the math because our graphical intuition fails us.

If we call state 1 the state in which neuron #1 is firing at a rate of 1 Hz and all others are silent, we can write this as:

$$s_1 = [1, 0, 0, 0, 0]$$

We may further define states 2, 3, 4, and 5 as follows:

$$s_2 = [0, 1, 0, 0, 0]$$

$$s_3 = [0, 0, 1, 0, 0]$$

$$s_4 = [0, 0, 0, 1, 0]$$

$$s_5 = [0, 0, 0, 0, 1]$$

By taking combinations of these five *basis vectors*, and multiplying them by scalar constants, we can describe any state of the network in the entire vector space. For example, to generate the network state [0, 3, 0, 9, 0] we could write:

$$(3 * s_2) + (9 * s_4) = [0, 3, 0, 9, 0]$$

If any one of the basis vectors is removed from the set, however, there will be some states of the network we will be unable to describe. For example, no combination of the vectors s_1, s_2, s_3, and s_4 can describe the network state [1, 0, 5, 3, 2] without also making use of s_5. Every vector space has a set of basis vectors. The definition of a set of basis vectors is twofold: (1) linear combinations (meaning addition, subtraction and multiplication by scalars) of the basis vectors can describe any vector in the vector space, and (2) every one of the basis vectors must be required in order to be able to describe all of the vectors in the vector space. It is also worth noting that the vectors s_1, s_2, s_3, s_4, and s_5 are all orthogonal to one another. You can test this for yourself by calculating the dot product of any two of these five basis vectors and verifying that it is zero. Basis vectors are not always orthogonal to one another, but they must always be linearly independent. The vector space that is defined by the set of all vectors you can possibly generate with different combinations of the basis vectors is called the *span* of the basis vectors.

Definition: A basis set is a linearly independent set of vectors that, when used in linear combination, can represent every vector in a given vector space.

It's worth taking a moment to consider what vector space is actually defined by these basis vectors. It's not all of 5-dimensional space, because we don't allow negative firing rates. So it's sort of the positive "quadrant" of 5-dimensional space. But we also said that we don't allow firing rates over 100Hz, so it's really only the part of that quadrant that fits between 0 and 100. This is a 5-dimensional hypercube. Doesn't that just sound so cool.

Matrices

A matrix, like a vector, is also a collection of numbers. The difference is that a matrix is a table of numbers rather than a list. Many of the same rules we just outlined for vectors above apply equally well to matrices. (In fact, you can think of vectors as matrices that happen to only have one column or one row.) First, let's consider matrix addition and subtraction. This part is uncomplicated. You can add and subtract matrices the same way you add vectors – element by element:

$$A + B = C$$

$$\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} + \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix} = \begin{pmatrix} a_{11} + b_{11} & a_{12} + b_{12} \\ a_{21} + b_{21} & a_{22} + b_{22} \end{pmatrix}$$

Matrix multiplication gets a bit more complicated, since multiple elements in the first matrix interact with multiple elements in the second to generate each element in the product matrix. This means that matrix multiplication can be a tedious task to carry out by hand, and can be time consuming on a computer for very large matrices. But as we shall see, this also means that, depending on the matrices we are multiplying, we will be able to perform a vast array of computations. Shown here is a simple example of multiplying a pair of 2×2 matrices, but this same procedure can generalize to matrices of arbitrarily large size, as will be discussed below:

$$AB = D$$

$$\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \cdot \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix} = \begin{pmatrix} a_{11}b_{11} + a_{12}b_{21} & a_{11}b_{12} + a_{12}b_{22} \\ a_{21}b_{11} + a_{22}b_{21} & a_{21}b_{12} + a_{22}b_{22} \end{pmatrix}$$

If you look at what's going on here, you may notice that what's going on it that we're taking dot products between rows of the ***A*** matrix with columns of the ***B*** matrix. for example to find the entry in our new matrix that's in the first-row-second-column position, we take the first row of ***A*** and compute it's dot product with the second column of ***B***. Matrix multiplication and dot products are intimately linked.

When multiplying two matrices together, the two matrices do not need to be the same size. For example, the vector $\vec{a} = (4,3)$ from earlier in this chapter represents a very simple matrix. If we multiply this matrix by another simple matrix the result is:

$$\begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \begin{pmatrix} 4 \\ 3 \end{pmatrix} = \begin{pmatrix} 3 \\ -4 \end{pmatrix}$$

Linear Transformations

Linear transformations are functions (functions that take vectors as inputs), but their a special subset of functions. Linear transformations are the set of all functions that can be written as a matrix multiplication:

$$\vec{y} = A\vec{x}$$

This sees like a pretty restrictive class of function, and indeed it is. But as it turns out, these simple functions can accomplish some surprisingly interesting things.

Consider the matrix we used for matrix multiplication in our last example.

$$A = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

When multiplied the vector [4,3] by this matrix, we got the vector [-3,4]. If you draw those to vectors you'll notice that the two vectors have the same length, but the second vector is rotated 90º counter clockwise from the first vector. Coincidence? Pick another vector, say [100, 0]; that vector gets transformed to the vector [0, 100], rotating 90º from the 3 o'clock position to the 12 o'clock position. As it turns out this is true for any vector $\vec{x}$ you pick: $\vec{y}$ will always be rotated 90º counter clockwise.

As it turns out our matrix $\boldsymbol{A}$ is a very special matrix. It's a *rotation matrix* – it never changes the length of a vector, but changes every vector's direction by a certain fixed amount – in this case 90º. This is just one special case of a rotation matrix. We can generate a rotation matrix that will allow us to rotate a vector by any angle we want. The general rotation matrix to rotate a vector by an angle θ looks like this:

$$R_\theta = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$$

First you plug in the desired value for θ to calculate the different elements of the matrix, then you multiply the matrix by the vector and you have a rotated vector.

This may sound like an awful lot to go through when a protractor would do just as well. The reason this is worth all the trouble is that it doesn't stop at rotations, it doesn't even stop at two dimensions. Above we defined a rotation matrix $\boldsymbol{R}$ for two dimensions. We can just as well define a rotation matrix in three dimensions, it would just be a 3×3 matrix rather than the 2×2 one above. In the case above, we think of the rotation matrix as "operating on" the vector to perform a rotation. Following this terminology, we call the rotation matrix the *operator*. We can make all sorts of other linear operators (matrixes) to do all sorts of other transformations. A few other useful linear transformations include identity $(\boldsymbol{I})$, Stretch and squash $(\boldsymbol{S})$, Skew $(\boldsymbol{W})$ and Flip $(\boldsymbol{F})$.

You may notice that since a linear operator is anything that can be expressed as a matrix, that means that you can stick any series of linear operators together to make a new function, and that function will also be a linear operator. For example if we want to rotate a vector and then stretch it we could, premultiply those to matrices together, and the resulting matrix operator will still be linear and will still follow all the same rules. So if $\boldsymbol{S} \cdot \boldsymbol{R} = \boldsymbol{X}$, then we can apply $\boldsymbol{X}$ as its own operator. Extending this, any set of linear transformations, no mater how long can be multiplied together and condensed into a single operator. For instance, if we wanted to stretch-skew-rotate-skew-stretch-flip-rotate ($\boldsymbol{S} \cdot \boldsymbol{W} \cdot \boldsymbol{R} \cdot \boldsymbol{W} \cdot \boldsymbol{S} \cdot \boldsymbol{F} \cdot \boldsymbol{R}$), we can define all of that into a single new operator $\boldsymbol{Y}$.