Linear Algebra Tutorial

Jorge A. Menéndez
July 4, 2019

Contents

1. Matrices and Vectors

An $m \times n$ matrix is simply an array of numbers:

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & & & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix}$$

where we define the indexing $\mathbf{A}_{ij} = a_{ij}$ to designate the component in the ith row and jth column of $\mathbf{A}$. The transpose of a matrix is obtained by flipping the rows with the columns:

$$\mathbf{A}^T = \begin{bmatrix} a_{11} & a_{21} & \dots & a_{n1} \\ a_{12} & a_{22} & \dots & a_{n2} \\ \vdots & & & \vdots \\ a_{1m} & a_{2m} & \dots & a_{nm} \end{bmatrix}$$

which evidently is now an $n \times m$ matrix, with components $(\mathbf{A}^T)_{ij} = \mathbf{A}_{ji} = a_{ji}$. In other words, the transpose is obtained by simply flipping the row and column indeces.

For our purposes, a vector can simply be thought of as a matrix with one column[1]:

$$\mathbf{a} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix}$$

We say that such a vector lives in $n-$dimensional space, since it has n components that determine it. This is written as:

$$\mathbf{a} \in \mathbb{R}^n$$

[1] I will always treat a vector as a column, although others are more loose with this, treating a vector as a more general one-dimensional array of numbers. Because here vectors are always columns, inner and outer products (see below) are respectively expressed as $\mathbf{a}^T\mathbf{b}$ and $\mathbf{a}\mathbf{b}^T$, whereas others (namely physicists) might use $\mathbf{a} \cdot \mathbf{b}$ and $\mathbf{a}\mathbf{b}$ instead.

since its components are all real, i.e. $a_i \in \mathbb{R}$ for $i = 1, \dots, n$. The transpose of a vector yields a row vector, i.e. a matrix with one row and n columns:

$$\mathbf{a}^T = \begin{bmatrix} a_1 & a_2 & \dots & a_n \end{bmatrix}$$

A pair of vectors $\mathbf{a}_1, \mathbf{a}_2$ are said to be linearly independent if and only if there is no linear combination of them that yields zero, i.e. there exists no pair of non-zero scalars c_1, c_2 such that

$$c_1\mathbf{a}_1 + c_2\mathbf{a}_2 = \mathbf{0}$$

where $\mathbf{0}$ is a vector of 0's. If there were, then one would simply be a scaled version of the other:

$$\Leftrightarrow \mathbf{a}_1 = -\frac{c_2}{c_1}\mathbf{a}_2$$

meaning that $\mathbf{a}_1$ and $\mathbf{a}_2$ are parallel. Geometrically, two vectors are linearly independent if they point in different directions (i.e. they are not parallel).

This notion extends to larger sets of vectors: we say that m vectors $\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_m$ are linearly independent if and only if each of them cannot be expressed as a linear combination of the others. This translates to the following simple condition: there does not exist a set of scalars $c_1, \dots, c_m$ that satisfies

$$c_1\mathbf{a}_1 + c_2\mathbf{a}_2 + \dots + c_m\mathbf{a}_m = \mathbf{0}$$

If such a set of scalars did exist, then any one of these vectors could be expressed as a linear combination of the others, e.g.

$$\Leftrightarrow c_1\mathbf{a}_1 = -c_2\mathbf{a}_2 - \dots - c_m\mathbf{a}_m$$

Unlike in the case of just two vectors, however, all m vectors might point in different directions while still not being linearly independent, so the geometric intuition behind linear independence in this case is a little more nuanced.

To see this, consider the following simple example: let $\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3$ be three two-dimensional vectors pointing in different directions. Now consider the set of all linear combinations of $\mathbf{a}_1$ and $\mathbf{a}_2$: is there any 2D vector that cannot be constructed by a linear combination of these two vectors?[2] If $\mathbf{a}_1$ and $\mathbf{a}_2$ point in different directions, then the answer is no. We thus say that the set of vectors $\{\mathbf{a}_1, \mathbf{a}_2\}$ spans 2D space: the (infinite) set of all possible linear combinations of them includes all possible two-dimensional vectors. Consequently, since $\mathbf{a}_3$ is a two-dimensional vector it also can be constructed as a linear combination of $\mathbf{a}_1$ and $\mathbf{a}_2$, even if it points in a different direction. The vectors $\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3$ therefore are *not* linearly independent. The deeper reason for this is that *two* linearly independent vectors are enough to span all of 2D space. Thus, any set of more than *two* *two*-dimensional vectors cannot be linearly independent, since any one of them can be constructed by a linear combination of any other two.

More generally, any set of n linearly independent vectors will span n-dimensional space. By the same argument as above, any set of m n-dimensional vectors therefore cannot be linearly independent if $m > n$. If $m < n$, on the other hand, we say that the set of m vectors spans a subspace of n-dimensional space: a subset of all the vectors in n-dimensional space.[3] This is because m vectors can never be enough to span n-dimensional space if $n > m$. A simple example is obtained by considering two linearly independent three-dimensional vectors and all their possible linear combinations: these will occupy a two-dimensional plane within the full three-dimensional space, see figure 1. In sum: a set of m linearly independent n-dimensional vectors will span an m-dimensional subspace of the full n-dimensional space $\mathbb{R}^n$. If $m = n$, this subspace is actually the full space; if $m < n$ this subspace occupies only a part of the space (figure 1); if $m > n$ then the m vectors cannot possibly be linearly independent. As we'll see below, this notion of linear independence and subspace dimensionality plays a central role in a lot of linear algebra.

[2] Try drawing this!

[3] More precisely, a subspace $\mathcal{S}$ must obey the following three properties:

- $\mathbf{0} \in \mathcal{S}$
- if $\mathbf{u}, \mathbf{v} \in \mathcal{S}$ then $\mathbf{u} + \mathbf{v} \in \mathcal{S}$
- if $\mathbf{u} \in \mathcal{S}$ then $c\mathbf{u} \in \mathcal{S}$ for any scalar c

The latter two requirements are also expressed as: the set of vectors $\mathcal{S}$ is closed under vector addition and scalar multiplication.

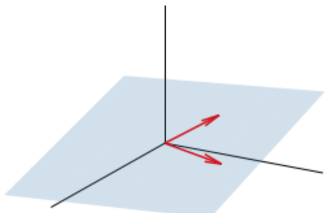

Figure 1 The red vectors are two linearly independent three-dimensional vectors. Note that they live on a plane, depicted by the bluish rectangle. The set of all linear combinations of these two red vectors will also live on this plane, which is thus a two-dimensional subspace of $\mathbb{R}^3$ spanned by the two red vectors.

2. Matrix Multiplication

The product of an $m \times n$ matrix $\mathbf{A}$ and a $n \times p$ matrix $\mathbf{B}$ is given by:

$$\mathbf{AB} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & & & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} & \dots & b_{1p} \\ b_{21} & b_{22} & \dots & b_{2p} \\ \vdots & & & \vdots \\ b_{n1} & b_{n2} & \dots & b_{np} \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^n a_{1i}b_{i1} & \sum_{i=1}^n a_{1i}b_{i2} & \dots & \sum_{i=1}^n a_{1i}b_{ip} \\ \sum_{i=1}^n a_{2i}b_{i1} & \sum_{i=1}^n a_{2i}b_{i2} & \dots & \sum_{i=1}^n a_{2i}b_{ip} \\ \vdots & & & \vdots \\ \sum_{i=1}^n a_{mi}b_{i1} & \sum_{i=1}^n a_{mi}b_{i2} & \dots & \sum_{i=1}^n a_{mi}b_{ip} \end{bmatrix}$$

In other words, we have that

$$(\mathbf{AB})_{ij} = \sum_{k=1}^{n} \mathbf{A}_{ik}\mathbf{B}_{kj} = \sum_{k=1}^{n} a_{ik}b_{kj}$$

It is thus evident that two matrices can only be multiplied if their inner dimensions agree. In this case, $\mathbf{A}$ has n columns and $\mathbf{B}$ has n rows, so they can be multiplied. $\mathbf{BA}$, on the other hand, is not a valid product if $m \neq q$, since $\mathbf{B}$ has p columns and $\mathbf{A}$ has m rows so their inner dimensions don't agree.

This illustrates an important fact about matrix multiplication: in general, $\mathbf{AB} \neq \mathbf{BA}$. Unlike scalar multiplication, matrix multiplication is not commutative. That said, matrix multiplication is associative ($\mathbf{A}(\mathbf{BC}) = (\mathbf{AB})\mathbf{C}$) and distributive ($\mathbf{A}(\mathbf{B}+\mathbf{C}) = \mathbf{AB} + \mathbf{AC}$). Note as well that the dimensionality of a matrix product is given by the outer dimensions: if $\mathbf{A}$ is $m \times n$ and $\mathbf{B}$ is $n \times p$, then $\mathbf{AB}$ is $m \times p$. These are important facts to remember when doing matrix algebra.

One particularly important matrix revealed by the matrix product operation is the identity matrix, which is composed of 1's on the diagonal and 0's everywhere else:

$$\mathbf{I} = \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \end{bmatrix}$$

It is called the identity matrix because the product of any matrix with the identity matrix is identical to itself:

$$\mathbf{AI} = \mathbf{IA} = \mathbf{A}$$

In other words, $\mathbf{I}$ is the equivalent of the number 1 for matrices.

Above we defined the matrix product in terms of operations on the components of the two matrices being multiplied. While this component-wise definition might be useful for calculating matrix products by hand, it is maybe the worst way to think about what a matrix product actually does. A much more useful way of thinking about matrix multiplication is illustrated by first considering the product of a matrix with a vector:

$$\begin{aligned}
\mathbf{Ax} &= \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & & & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} x_1 a_{11} + x_2 a_{12} + \ldots + x_n a_{1n} \\ x_1 a_{21} + x_2 a_{22} + \ldots + x_n a_{2n} \\ \vdots \\ x_1 a_{m1} + x_2 a_{m2} + \ldots + x_n a_{mn} \end{bmatrix} \\
&= \begin{bmatrix} x_1 \begin{bmatrix} a_{11} \\ a_{21} \\ \vdots \\ a_{m1} \end{bmatrix} + x_2 \begin{bmatrix} a_{12} \\ a_{22} \\ \vdots \\ a_{m2} \end{bmatrix} + \ldots + x_n \begin{bmatrix} a_{1n} \\ a_{2n} \\ \vdots \\ a_{mn} \end{bmatrix} \end{bmatrix} \\
&= \begin{bmatrix} x_1 \mathbf{a}_1 + x_2 \mathbf{a}_2 + \ldots + x_n \mathbf{a}_n \end{bmatrix}
\end{aligned}$$

where $\mathbf{a}_1, \ldots, \mathbf{a}_n$ are the n columns of $\mathbf{A}$. What this example illustrates is that the m-dimensional vector given by the matrix-vector product $\mathbf{Ax}$ is in fact a linear combination of the columns of $\mathbf{A}$. It is thus often useful to think of an $m \times n$ matrix as a collection of vectors $\mathbf{a}_1, \ldots, \mathbf{a}_n \in \mathbb{R}^m$ placed side by side. For any n-dimensional vector $\mathbf{x}$, $\mathbf{Ax}$ is then a linear combination of these vectors. Note that this implies that $\mathbf{Ax}$ lives in the (sub)space spanned by $\mathbf{a}_1, \ldots, \mathbf{a}_n$. We call this (sub)space the <u>column space</u> of $\mathbf{A}$. As we noted above, if k of these vectors are linearly independent, then the (sub)space spanned by them - i.e. the column space of $\mathbf{A}$ - is k-dimensional. The dimensionality of the column space of $\mathbf{A}$ (= the number of linearly independent columns of $\mathbf{A}$) is called the <u>matrix rank</u> of $\mathbf{A}$. Matrix-vector products come up all the time in linear algebra, so the notion of a column space plays a big role in a lot of what we do since it tells us the space in which the product $\mathbf{Ax}$ lives.

So what about matrix-matrix products? As just suggested, we'll treat each matrix as collections of column vectors:

$$\begin{aligned}
\mathbf{AB} &= \begin{bmatrix} \begin{bmatrix} a_{11} \\ a_{21} \\ \vdots \\ a_{m1} \end{bmatrix} & \begin{bmatrix} a_{12} \\ a_{22} \\ \vdots \\ a_{m2} \end{bmatrix} & \dots & \begin{bmatrix} a_{1n} \\ a_{2n} \\ \vdots \\ a_{mn} \end{bmatrix} \end{bmatrix} \begin{bmatrix} \begin{bmatrix} b_{11} \\ b_{21} \\ \vdots \\ b_{n1} \end{bmatrix} & \begin{bmatrix} b_{12} \\ b_{22} \\ \vdots \\ b_{n2} \end{bmatrix} & \dots & \begin{bmatrix} b_{1p} \\ b_{2p} \\ \vdots \\ b_{np} \end{bmatrix} \end{bmatrix} \\
&= \begin{bmatrix} \mathbf{a}_1 & \mathbf{a}_2 & \dots & \mathbf{a}_n \end{bmatrix} \begin{bmatrix} \mathbf{b}_1 & \mathbf{b}_2 & \dots & \mathbf{b}_p \end{bmatrix}
\end{aligned}$$

where $\mathbf{a}_i \in \mathbb{R}^m, \mathbf{b}_i \in \mathbb{R}^n$. We then perform the full matrix multiplication by simply performing a series of matrix-vector products: the jth column of $\mathbf{AB}$ is given by $\mathbf{Ab}_j = \sum_{k=1}^n \mathbf{a}_k b_{kj}$:

$$\begin{aligned}
\mathbf{AB} &= \begin{bmatrix} \mathbf{Ab}_1 & \mathbf{Ab}_2 & \cdots & \mathbf{Ab}_p \end{bmatrix} \\
&= \begin{bmatrix} \sum_{k=1}^n \mathbf{a}_k b_{k1} & \sum_{k=1}^n \mathbf{a}_k b_{k2} & \cdots & \sum_{k=1}^n \mathbf{a}_k b_{kp} \end{bmatrix}
\end{aligned}$$

In other words, the columns of $\mathbf{AB}$ are different linear combinations of the columns of $\mathbf{A}$. It is easy to verify that this is equivalent to our above equation $(\mathbf{AB})_{ij} = \sum_k a_{ik} b_{kj}$. Note that this implies that each of the columns of $\mathbf{AB}$ live in the column space of $\mathbf{A}$, so the column space of $\mathbf{AB}$ has to be either equal to the column space of $\mathbf{A}$ or be a subspace of it.

Rather than treating matrices as collections of *column* vectors, we could instead treat them as collections of *row* vectors. This yields yet another interpretation of matrix multiplication as summing rows rather than columns:

$$\mathbf{AB} = \begin{bmatrix} \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \end{bmatrix} \\ \begin{bmatrix} a_{21} & a_{22} & \dots & a_{2n} \end{bmatrix} \\ \vdots \\ \begin{bmatrix} a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix} \end{bmatrix} \begin{bmatrix} \begin{bmatrix} b_{11} & b_{12} & \dots & b_{1p} \end{bmatrix} \\ \begin{bmatrix} b_{21} & b_{22} & \dots & b_{2p} \end{bmatrix} \\ \vdots \\ \begin{bmatrix} b_{n1} & b_{n2} & \dots & b_{np} \end{bmatrix} \end{bmatrix} = \begin{bmatrix} \mathbf{a}_1^T \\ \mathbf{a}_2^T \\ \vdots \\ \mathbf{a}_m^T \end{bmatrix} \begin{bmatrix} \mathbf{b}_1^T \\ \mathbf{b}_2^T \\ \vdots \\ \mathbf{b}_n^T \end{bmatrix}$$

where $\mathbf{a}_i \in \mathbb{R}^n, \mathbf{b}_i \in \mathbb{R}^p$. The ith row of $\mathbf{AB}$ is now given by $\mathbf{a}_i^T \mathbf{B} = \sum_{k=1}^n a_{ik} \mathbf{b}_k^T$:

$$\mathbf{AB} = \begin{bmatrix} \mathbf{a}_1^T \mathbf{B} \\ \mathbf{a}_2^T \mathbf{B} \\ \vdots \\ \mathbf{a}_m^T \mathbf{B} \end{bmatrix} = \begin{bmatrix} \sum_{k=1}^n a_{1k} \mathbf{b}_k^T \\ \sum_{k=1}^n a_{2k} \mathbf{b}_k^T \\ \vdots \\ \sum_{k=1}^n a_{mk} \mathbf{b}_k^T \end{bmatrix}$$

Again, this is simply another equivalent way of looking at matrix multiplication. Analagously, we define the row space of $\mathbf{A}$ as the (sub)space spanned by its rows. We could then define the matrix rank of $\mathbf{A}$ as the number of linearly independent rows rather than the number of linearly independent columns. Interestingly, it turns out that these two definitions of matrix rank are exactly the same: the dimensionality of the row space of $\mathbf{A}$ is always equal to the dimensionality of the column space of $\mathbf{A}$.[4] This is far from obvious (to me), but the proof is relatively straightforward.[5]

In this view of matrices as collections of vectors, we can think of the matrix transpose as a way for switching between a collection of row vectors or a collection of column vectors:

$$\mathbf{A} = \begin{bmatrix} \mathbf{a}_1 & \mathbf{a}_2 & \dots & \mathbf{a}_n \end{bmatrix} \Leftrightarrow \mathbf{A}^T = \begin{bmatrix} \mathbf{a}_1^T \\ \mathbf{a}_2^T \\ \vdots \\ \mathbf{a}_n^T \end{bmatrix}$$

It is thus easy to see that the matrix rank of $\mathbf{A}$ is equal to that of its transpose. Moreover it is easy to see now that the following is true:

$$\text{rank}(\mathbf{A}) = \text{rank}(\mathbf{A}^T) = \text{rank}(\mathbf{A}\mathbf{A}^T) = \text{rank}(\mathbf{A}^T\mathbf{A})$$

We finally note that the rank of any matrix is upper bounded by the smallest of two numbers: the number of columns and the number of rows. If the rank of a matrix is maximal, we say that it is full rank. A square $n \times n$ matrix is full rank only if its rank is n. A rectangular $m \times n$ matrix is said to be full rank if its rank is the smaller of m and n, which is the highest it could be since the row rank = column rank for rectangular matrices too.

3. Outer products

An outer product between two vectors can be treated just like a matrix product, but between an $m \times 1$ matrix (i.e. a column vector) and a $1 \times n$ matrix (i.e. a row vector, or transposed column vector):

$$\mathbf{u}\mathbf{v}^T = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_m \end{bmatrix} \begin{bmatrix} v_1 & v_2 & \dots & v_n \end{bmatrix} = \begin{bmatrix} u_1v_1 & u_1v_2 & \dots & u_1v_n \\ u_2v_1 & u_2v_2 & \dots & u_2v_n \\ \vdots & \vdots & & \vdots \\ u_mv_1 & u_mv_2 & \dots & u_mv_n \end{bmatrix} = \begin{bmatrix} v_1\mathbf{u} & v_2\mathbf{u} & \dots & v_n\mathbf{u} \end{bmatrix} = \begin{bmatrix} u_1\mathbf{v}^T \\ u_2\mathbf{v}^T \\ \vdots \\ u_m\mathbf{v}^T \end{bmatrix}$$

where I have written down the component-wise view, the column-wise view, and the row-wise view as I did above with matrix multiplication.

Outer products are particularly useful because they give us yet another way of expressing matrix multiplication. It turns out that, for an $m \times n$ matrix $\mathbf{A}$ and an $n \times p$ matrix $\mathbf{B}$,

$$\mathbf{AB} = \sum_{k=1}^{n} \mathbf{a}_k \mathbf{b}_k^T$$

where $\mathbf{a}_k \in \mathbb{R}^m$ is the kth column of $\mathbf{A}$ and $\mathbf{b}_k^T \in \mathbb{R}^p$ is the kth row of $\mathbf{B}$. This is exactly equivalent to the component-wise definition above.

Note that an outer product is a rank 1 matrix, since each column (row) is simply a scaled version of the first (second) vector in the product, so none of its columns is linearly independent of any other. And vice versa - any rank 1 matrix can be written down as an outer product. More generally, a cool fact is that any rank k matrix can be expressed as a sum of k rank 1 matrices or outer products.

[4] Even though they might live in different spaces! e.g. if the column space of an $m \times n$ matrix is a k-dimensional subspace of $\mathbb{R}^m$ then its row space will be a k-dimensional subspace of $\mathbb{R}^n$.

[5] https://en.wikipedia.org/wiki/Rank_(linear_algebra)#Proofs_that_column_rank_=_row_rank

4. Inner Products

The Euclidean inner product (also called the dot product) is again another matrix product, but between a $1 \times n$ matrix (i.e. a row vector, or transposed column vector) and an $n \times 1$ matrix (i.e. a column vector):

$$\mathbf{u}^T\mathbf{v} = \begin{bmatrix} u_1 & u_2 & \dots & u_n \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix} = \sum_{k=1}^{n} u_k v_k$$

More generally, an inner product is a mapping from two vectors to a scalar[6]. Other non-Euclidean abstract spaces can be defined by defining alternative inner products that don't directly correspond to a row vector multiplied with a column vector as I have discussed it here. The reason this particular inner product is called "Euclidean" is because it gives you the Euclidean length, or norm, of a vector:

$$\|\mathbf{u}\| := \sqrt{\mathbf{u}^T\mathbf{u}} = \sqrt{\sum_{k=1}^{n} u_k^2}$$

which, by Pythagoras' theorem, is the length of the vector $\mathbf{u}$ (the $:=$ symbol is read as "is defined as"). This norm $\|\cdot\|$ is naturally called the Euclidean norm (or L2 norm), but other non-Euclidean norms exist (e.g. the "Manhattan", or L1, norm $\|\mathbf{u}\|_1 = \sum_k u_k$).

The best part of the Euclidean inner product is that it can actually be interpreted in terms of the angle between two vectors. It turns out that the following is true:[7]

$$\mathbf{u}^T\mathbf{v} = \sum_k u_k v_k = \|\mathbf{u}\|\|\mathbf{v}\| \cos\theta$$

where θ is the angle between $\mathbf{u}$ and $\mathbf{v}$. This means that when $\|\mathbf{u}\| = \|\mathbf{v}\| = 1$, we can directly interpret the inner product as a measure of similarity: if $\mathbf{u}$ and $\mathbf{v}$ line up and are identical, $\theta = 0^o$ and $\mathbf{u}^T\mathbf{v} = \cos 0 = 1$. On the other hand, if $\mathbf{u}$ and $\mathbf{v}$ are orthogonal (i.e. perpendicular), $\theta = 90^o$ and $\mathbf{u}^T\mathbf{v} = \cos\frac{\pi}{2} = 0$. It follows that two vectors $\mathbf{u}, \mathbf{v}$ are orthogonal if and only if $\mathbf{u}^T\mathbf{v} = 0$.

The Euclidean inner product also gives us yet another equivalent way of viewing matrix multiplication:

$$\mathbf{AB} = \begin{bmatrix} \sum_{i=1}^n a_{1i}b_{i1} & \sum_{i=1}^n a_{1i}b_{i2} & \dots & \sum_{i=1}^n a_{1i}b_{ip} \\ \sum_{i=1}^n a_{2i}b_{i1} & \sum_{i=1}^n a_{2i}b_{i2} & \dots & \sum_{i=1}^n a_{2i}b_{ip} \\ \vdots & & & \vdots \\ \sum_{i=1}^n a_{mi}b_{i1} & \sum_{i=1}^n a_{mi}b_{i2} & \dots & \sum_{i=1}^n a_{mi}b_{ip} \end{bmatrix} = \begin{bmatrix} \mathbf{a}_1^T\mathbf{b}_1 & \mathbf{a}_1^T\mathbf{b}_2 & \dots & \mathbf{a}_1^T\mathbf{b}_p \\ \mathbf{a}_2^T\mathbf{b}_1 & \mathbf{a}_2^T\mathbf{b}_2 & \dots & \mathbf{a}_2^T\mathbf{b}_p \\ \vdots & & & \vdots \\ \mathbf{a}_m^T\mathbf{b}_1 & \mathbf{a}_m^T\mathbf{b}_2 & \dots & \mathbf{a}_m^T\mathbf{b}_p \end{bmatrix}$$

where $\mathbf{a}_k^T \in \mathbb{R}^n$ is the kth row of $\mathbf{A}$ and $\mathbf{b}_k \in \mathbb{R}^n$ is the kth column of $\mathbf{B}$. Another way of writing this is: $(\mathbf{AB})_{ij} = \mathbf{a}_i^T\mathbf{b}_j$.

For example, consider the following equation:

$$\mathbf{Ax} = \begin{bmatrix} \mathbf{a}_1^T\mathbf{x} \\ \mathbf{a}_2^T\mathbf{x} \\ \vdots \\ \mathbf{a}_m^T\mathbf{x} \end{bmatrix} = \mathbf{0}$$

where $\mathbf{A}$ is $m \times n$, $\mathbf{x} \in \mathbb{R}^n$, and $\mathbf{a}_i^T \in \mathbb{R}^n$ is the ith row of $\mathbf{A}$. As we just saw, $\mathbf{a}_k^T\mathbf{x} = 0$ implies that $\mathbf{x}$ is orthogonal to $\mathbf{a}_k$. This equation thus tells us that $\mathbf{x}$ is orthogonal to all the rows of $\mathbf{A}$, implying that it is orthogonal to any linear combination of them and therefore orthogonal to the

[6] Strictly, an inner product $\langle \cdot, \cdot \rangle$ must satisfy the following three porperties:

- Conjugate symmetry: $\langle \mathbf{u}, \mathbf{v} \rangle = \overline{\langle \mathbf{v}, \mathbf{u} \rangle}$
- Linearity: $\langle a\mathbf{u}, \mathbf{v} \rangle = a\langle \mathbf{u}, \mathbf{v} \rangle$, $\langle \mathbf{u} + \mathbf{w}, \mathbf{v} \rangle = \langle \mathbf{u}, \mathbf{v} \rangle + \langle \mathbf{w}, \mathbf{v} \rangle$
- Positive-definiteness: $\langle \mathbf{u}, \mathbf{u} \rangle \geq 0$, $\langle \mathbf{u}, \mathbf{u} \rangle = 0 \Leftrightarrow \mathbf{u} = \mathbf{0}$

It is easy to show that $\langle \mathbf{u}, \mathbf{v} \rangle = \mathbf{u}^T\mathbf{v}$ satsifies these.

[7] Easy to prove: https://en.wikipedia.org/wiki/Dot_product#Equivalence_of_the_definitions

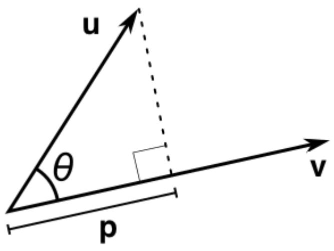

Figure 2 $\mathbf{p}$ is the projection of $\mathbf{u}$ onto $\mathbf{v}$, the scalar projection is $\|\mathbf{p}\|$

entire row space of $\mathbf{A}$. In fact, the set of all vectors $\mathbf{x}$ satisfying the above equation (i.e. the set of all vectors that are orthogonal to the row space of $\mathbf{A}$) forms a subspace of $\mathbb{R}^n$: if $\mathbf{Ax} = \mathbf{0}$ and $\mathbf{Ay} = \mathbf{0}$, then $\mathbf{A}(c_1\mathbf{x}+c_2\mathbf{y}) = 0$ for any $c_1, c_2 \in \mathbb{R}$. This subspace is called the nullspace of $\mathbf{A}$. Note that the above equation also implies that the columns of $\mathbf{A}$ are not linearly independent, since it explicitly says there exists a linear combination of them that equals $\mathbf{0}$. Thus, for a solution to this equation to exist it must be the case that the rank k of $\mathbf{A}$ is less than n. Indeed, it turns out that the dimensionality of the nullspace is given by $n - k$: since the row space of $\mathbf{A}$ is k-dimensional, exactly $n - k$ linearly independent vectors in $\mathbb{R}^n$ can be found that are orthogonal to it. The row- and null- spaces of $\mathbf{A}$ are orthogonal compliments in $\mathbb{R}^n$: the union of all vectors in each of these subspaces is in fact the set of all vectors in the full space $\mathbb{R}^n$.

Inner products are also useful for computing the projection of a vector onto another vector, illustrated in figure 2. In this case, $\mathbf{p}$ is the projection of $\mathbf{u}$ onto $\mathbf{v}$ - the "shadow" that $\mathbf{u}$ casts on $\mathbf{v}$. By definition, $\mathbf{p}$ and $\mathbf{u}$ form a right triangle, so we can use our standard trigonometric rules to show that

$$\|\mathbf{p}\| = \|\mathbf{u}\| \cos\theta = \frac{\mathbf{u}^T\mathbf{v}}{\|\mathbf{v}\|}$$

This quantity (i.e. the length of the projection of $\mathbf{u}$ onto $\mathbf{v}$) is called the scalar projection of $\mathbf{u}$ onto $\mathbf{v}$. To find the actual projection vector $\mathbf{p}$ we first note that, since $\mathbf{p}$ lies in the subspace spanned by $\mathbf{v}$ (i.e. the one-dimensional subspace consisting of all scaled versions of $\mathbf{v}$), $\mathbf{p} = a\mathbf{v}$ for some real scalar a. We then solve for a by equating the norm of $a\mathbf{v}$ to the norm of $\mathbf{p}$ computed from the scalar projection:

$$\|a\mathbf{v}\| = a\|\mathbf{v}\| = \frac{\mathbf{u}^T\mathbf{v}}{\|\mathbf{v}\|} \Leftrightarrow a = \frac{\mathbf{u}^T\mathbf{v}}{\|\mathbf{v}\|^2} = \frac{\mathbf{u}^T\mathbf{v}}{\mathbf{v}^T\mathbf{v}} \Rightarrow \mathbf{p} = \frac{\mathbf{u}^T\mathbf{v}}{\mathbf{v}^T\mathbf{v}}\mathbf{v}$$

Importantly, note that $\mathbf{p}$ is the vector in the subspace spanned by $\mathbf{v}$ that is closest to $\mathbf{u}$. It is easy to prove this formally, but the intuition is evident in figure 2: if I make $\mathbf{p}$ longer or shorter along $\mathbf{v}$ (i.e. move it along the subspace spanned by $\mathbf{v}$), the distance between $\mathbf{u}$ and $\mathbf{p}$ (the length of the dotted line) will only get longer. This property makes projections an extremely useful tool for solving least-squares problems, in which one often wants to minimize the distance between a vector and a subspace. We'll cover one such problem below, called linear regression.

5. Matrix Inverse, Pseudo-inverse

Consider the equation

$$\mathbf{Ax} = \mathbf{b}$$

where $\mathbf{A}$ is $m \times n$ and, accordingly, $\mathbf{b} \in \mathbb{R}^m, \mathbf{x} \in \mathbb{R}^n$. Suppose we know $\mathbf{A}$ and $\mathbf{b}$ and want to solve this equation for the vector $\mathbf{x}$. If we express the matrix-vector product in component form, we note that this is equivalent to solving a system of m linear equations with n unknown variables:

$$\begin{aligned}
a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n &= b_1 \\
a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n &= b_2 \\
&\vdots \\
a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n &= b_m
\end{aligned}$$

Treating this system as a matrix-vector equation will allow us to use the tools of linear algebra to solve it using geometric intuitions. In particular, recall that this equation implies that, for a given $\mathbf{x}$, $\mathbf{b} \in \mathbb{R}^m$ must be in the column space of $\mathbf{A}$. As we'll see, this fact already provides some important clues as to the nature of the solution(s) to this equation.

We consider four cases:

$\mathbf{A}$ is square ($m = n$) and full rank: in this case, the n columns of $\mathbf{A}$ are linearly independent and span $\mathbb{R}^m = \mathbb{R}^n$. Thus, since $\mathbf{b} \in \mathbb{R}^m$, we can be sure there exists a linear combination of the columns of $\mathbf{A}$ that equals $\mathbf{b}$. In fact, it turns out that there is only one such linear combination (this can be easily proved by contradiction). We thus conclude that there exists a unique solution for $\mathbf{x}$, consisting of the coefficients $x_1, \ldots, x_n$ producing the linear combination

$$x_1\mathbf{a}_1 + x_2\mathbf{a}_2 + \ldots + x_n\mathbf{a}_n = \mathbf{b}$$

We can express this solution in terms of the matrix inverse of $\mathbf{A}$ as

$$\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$$

where $\mathbf{A}^{-1}$ is a matrix defined by the identity

$$\mathbf{A}\mathbf{A}^{-1} = \mathbf{A}^{-1}\mathbf{A} = \mathbf{I}$$

so that $\mathbf{A}\mathbf{x} = \mathbf{A}\mathbf{A}^{-1}\mathbf{b} = \mathbf{b}$. Constructing $\mathbf{A}^{-1}$ is in general highly non-trivial, but many algorithms exist for computing it numerically.

$\mathbf{A}$ is square ($m = n$) but not full rank: in this case, the columns of $\mathbf{A}$ span a subspace of $\mathbb{R}^m$. We therefore can't be sure that there exists a linear combination of them equal to $\mathbf{b}$, since $\mathbf{b} \in \mathbb{R}^m$ might live outside that subspace. Moreover, if $\mathbf{b}$ does live in that subspace, it turns out there are infinitely many solutions for $\mathbf{x}$. To see this, consider the following example: let $k = n - 1$ and $\mathbf{a}_2, \ldots, \mathbf{a}_n$ be linearly independent but $\mathbf{a}_1 = r\mathbf{a}_2$. Then for any $c_2, \ldots, c_n$ satisfying

$$c_2\mathbf{a}_2 + \ldots + c_n\mathbf{a}_n = \mathbf{b}$$

we can find a solution for $\mathbf{x}$ of the form

$$x_2 = c_2 - rx_1, x_3 = c_3, \ldots, x_n = c_n$$

such that

$$x_1\mathbf{a}_1 + x_2\mathbf{a}_2 + \ldots + x_n\mathbf{a}_n = (rx_1 + x_2)\mathbf{a}_2 + \ldots + x_n\mathbf{a}_n = c_2\mathbf{a}_2 + \ldots + c_n\mathbf{a}_n = \mathbf{b}$$

for any $x_1 \in \mathbb{R}$. In other words, we have a solution for $\mathbf{x}$ for each value that x_1 can take on - infinitely many of them. Indeed, for any square $\mathbf{A}$ with less than n linearly independent columns there will always be (infinitely) many linear combinations of them that can produce any given vector in the column space of $\mathbf{A}$. We thus conclude that in this case the system of equations can't be solved: either there is no solution or there are infinitely many. By the above logic, this in turn implies that $\mathbf{A}^{-1}$ doesn't exist. $\mathbf{A}$ is therefore not invertible and is said to be a singular matrix. Any square matrix that is not full rank is singular, by virtue of its linearly dependent columns.

$\mathbf{A}$ is skinny ($m > n$): if $m > n$, then our system of equations consists of more equations than it has unkowns: it is an overdetermined system and may or may not have a solution. Geometrically, this translates to the fact that the column space of $\mathbf{A}$ is necessarily a lower-dimensional subspace of $\mathbb{R}^m$ since $n < m$ vectors are not enough to span all of $\mathbb{R}^m$. A solution thus exists only if $\mathbf{b}$ lives in this subspace. We can check if this is the case by projecting $\mathbf{b}$ onto the column space of $\mathbf{A}$. Just like in figure 2, where we projected a vector onto a one-dimensional subspace of $\mathbb{R}^2$ (i.e. another vector), we can also project a vector onto an n-dimensional subspace of $\mathbb{R}^m$ by finding the vector in this subsapce that is closest to it. For finding the vector $\mathbf{b}_{proj} = \mathbf{A}\mathbf{x}_{proj}$ in the column space of $\mathbf{A}$ that is closest to $\mathbf{b}$, we simply look for the linear combination of the columns of $\mathbf{A}$ that has the smallest Euclidean distance from $\mathbf{b}$:

$$\begin{aligned}\mathbf{x}_{proj} &= \arg\min_{\mathbf{x}} \|\mathbf{b} - \mathbf{A}\mathbf{x}\| \\ &= \arg\min_{\mathbf{x}} \|\mathbf{b} - \mathbf{A}\mathbf{x}\|^2 \\ \Leftrightarrow 0 &= \mathbf{A}^T(\mathbf{b} - \mathbf{A}\mathbf{x}_{proj}) \\ \Leftrightarrow \mathbf{x}_{proj} &= (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b}\end{aligned}$$

where in the third line we simply took the derivative[8] of the right-hand side with respect to $\mathbf{x}_{proj}$ and set it to 0. Since $\mathbf{b}_{proj}$ is the vector in the column space of $\mathbf{A}$ that is closest to $\mathbf{b}$, if $\mathbf{b}$ is actually in the column space of $\mathbf{A}$ then it must be the case that $\mathbf{b} = \mathbf{b}_{proj}$, and therefore $\mathbf{x} = \mathbf{x}_{proj}$. Otherwise, there is no solution for $\mathbf{x}$.

The projection matrix $(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T$ is also called the left pseudo-inverse of $\mathbf{A}$, since multiplying $\mathbf{A}$ from the left with this matrix results in the identity. However, one should be careful not to think about this as being the equivalent of an inverse matrix. To see this, note that $\mathbf{x} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b}$ is not necessarily a solution to the equation, since $\mathbf{A}\mathbf{x} = \mathbf{A}(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b}$ does not necessarily equal $\mathbf{b}$! It only does if $\mathbf{b}$ lives in the column space of $\mathbf{A}$:

$$\mathbf{b} = \mathbf{A}\mathbf{x} \Rightarrow \mathbf{A}(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b} = \mathbf{A}(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{A}\mathbf{x} = \mathbf{A}\mathbf{x} = \mathbf{b}$$

It is worth noting that the mechanics of projecting a vector into a subspace using the left pseudo-inverse are essentially the same as that of projecting one vector onto another using the scalar projection as discussed in section 4.. In particular, consider the case where the columns of $\mathbf{A}$ are orthogonal to each other, i.e. $\mathbf{a}_i^T\mathbf{a}_j = 0$ for $i \neq j$, where $\mathbf{a}_i$ is the ith column of $\mathbf{A}$. In this case, we have

$$\begin{aligned}\mathbf{b}_{proj} &= \mathbf{A}(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b} \\ &= \begin{bmatrix}\mathbf{a}_1 & \mathbf{a}_2 & \dots & \mathbf{a}_n\end{bmatrix} \begin{bmatrix}\|\mathbf{a}_1\|^2 & 0 & \dots & 0 \\ 0 & \|\mathbf{a}_2\|^2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \|\mathbf{a}_n\|^2\end{bmatrix}^{-1} \begin{bmatrix}\mathbf{a}_1^T\mathbf{b} \\ \mathbf{a}_2^T\mathbf{b} \\ \vdots \\ \mathbf{a}_n^T\mathbf{b}\end{bmatrix} \\ &= \begin{bmatrix}\mathbf{a}_1 & \mathbf{a}_2 & \dots & \mathbf{a}_n\end{bmatrix} \begin{bmatrix}\frac{\mathbf{a}_1^T\mathbf{b}}{\|\mathbf{a}_1\|^2} \\ \frac{\mathbf{a}_2^T\mathbf{b}}{\|\mathbf{a}_2\|^2} \\ \vdots \\ \frac{\mathbf{a}_n^T\mathbf{b}}{\|\mathbf{a}_n\|^2}\end{bmatrix} \\ &= \sum_{i=1}^{n} \frac{\mathbf{a}_i^T\mathbf{b}}{\|\mathbf{a}_i\|^2}\mathbf{a}_i\end{aligned}$$

which is exactly the sum of the projections of $\mathbf{b}$ onto each of the columns of $\mathbf{A}$ (cf. section 4.)! When the columns of $\mathbf{A}$ are not orthogonal, we can't do exactly this, but we can think of the left-pseudo inverse as effectively "orthogonalizing" them and then computing the projection of $\mathbf{b}$ onto each of them to obtain $\mathbf{b}_{proj}$.

Also note that the left pseudo-inverse only exists when the $n \times n$ matrix $\mathbf{A}^T\mathbf{A}$ is invertible, which only holds if $\mathbf{A}$ is rank n. If the n columns of $\mathbf{A}$ are not linearly independent, then, by the same argument as above, there is an infinite number of linear combinations of its columns that can produce any vector in its column space. In this case, when $\mathbf{b}$ does happen to live in the column space of $\mathbf{A}$ there are an infinite number of solutions for $\mathbf{x}$. This lack of a unique solution is then reflected by the non-existence of the left pseudo-inverse of $\mathbf{A}$.

[8] We used the facts

$$\frac{\mathrm{d}}{\mathrm{d}\mathbf{x}}\|f(\mathbf{x})\|^2 = 2\left(\frac{\mathrm{d}f}{\mathrm{d}\mathbf{x}}\right)^T f(\mathbf{x})$$

$$\frac{\mathrm{d}}{\mathrm{d}\mathbf{x}}\mathbf{A}\mathbf{x} = \mathbf{A}$$

A is fat ($m < n$): if $m < n$, then our system of equations consists of fewer equations than it has unkowns: it is an underdetermined system and therefore has no unique solution. Geometrically, we have more columns of $\mathbf{A}$ than we have dimensions, so they cannot all be linearly independent (cf. end of section 1.). There are therefore infinitely many linear combinations of the columns of $\mathbf{A}$ that could yield $\mathbf{b}$, meaning there is no unique solution for $\mathbf{x}$.

6. Example: Linear Regression

In linear regression, we want to estimate the linear relationship between a dependent variable y and some set of independent variables or predictors $x^{(1)}, \ldots, x^{(k)}$. For example, $x^{(1)}, \ldots, x^{(k)}$ might be features of an image presented to a mouse, and y the firing rate of a neuron responding to the presentation of that image. To better understand how this neuron's response depends on a presented image's features, we'll attempt to estimate the linear dependence between these variables by fitting the following model:

$$y = w_1 x^{(1)} + w_2 x^{(2)} + \ldots + w_k x^{(k)}$$

If, for example, we find that w_2 is large and positive, then we may conclude that the neuron is excited by the presence of feature x_2. Conversely, if we find that w_2 is very negative, then we might infer that the neuron's activity is suppressed by this feature. This model can only capture such simple linear relationships - a strong limitation on its expressiveness but also an advantage in its interpretability.

The problem of linear regression is that of obtaining an estimate for each of the weights $w_1, \ldots, w_k$. We do this by first collecting a large sample of n measurements of each of these variables:

$$\{y_1, \ldots, y_n\},\ \{x_1^{(1)}, \ldots, x_n^{(1)}\},\ \{x_1^{(2)}, \ldots, x_n^{(2)}\},\ \ldots,\ \{x_1^{(k)}, \ldots, x_n^{(k)}\}$$

For example, we might present the mouse with n different visual stimuli and measure the neuron's response to each of these images. We denote the k features of the ith presented image as $x_i^{(1)}, \ldots, x_i^{(k)}$ and the neuron's response to that image as y_i. Given these data, we want to find a set of weights $w_1, \ldots, w_k$ that satisfy

$$y_i = w_1 x_i^{(1)} + w_2 x_i^{(2)} + \ldots + w_k x_i^{(k)}$$

for all i. Note that this is nothing more than a linear system of n equations, with k unknown variables! We can thus naturally formulate it as a matrix equation:

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} w_1 x_1^{(1)} + w_2 x_1^{(2)} + \ldots + w_k x_1^{(k)} \\ w_1 x_2^{(1)} + w_2 x_2^{(2)} + \ldots + w_k x_2^{(k)} \\ \vdots \\ w_1 x_n^{(1)} + w_2 x_n^{(2)} + \ldots + w_k x_n^{(k)} \end{bmatrix} = \begin{bmatrix} x_1^{(1)} & x_1^{(2)} & \ldots & x_1^{(k)} \\ x_2^{(1)} & x_2^{(2)} & \ldots & x_2^{(k)} \\ \vdots & \vdots & & \vdots \\ x_n^{(1)} & x_n^{(2)} & \ldots & x_n^{(k)} \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_k \end{bmatrix} = \mathbf{X}\mathbf{w}$$

Seen now as a matrix equation, the goal of linear regression is to find the linear combination of columns of $\mathbf{X}$ that produces $\mathbf{y}$. For this problem to have a solution, we need $n \geq k$, and in fact we'll generally try to use $n >> k$ to get a good estimate of the weights. Since data collection is always noisy, using $n = k$ puts us at risk of fitting the weights to this noise rather than to the underlying linear relationships we are after.

Because of this noise - plus the fact that the real world is never exactly linear - the n-dimensional vector $\mathbf{y}$ will never exactly live in the k-dimensional subspace that is the column space of $\mathbf{X}$. Thus, rather than finding a weight vector $\mathbf{w}$ that exactly solves this equation, we'll find the one that minimizes the squared error between the measured y_i's and the model's predictions $\hat{y}_i = \sum_j w_j x_i^{(j)}$:

$$\begin{aligned} \mathbf{w}^* &= \arg\min_{\mathbf{w}} \sum_i (y_i - \hat{y}_i)^2 \\ &= \arg\min_{\mathbf{w}} \|\mathbf{y} - \hat{\mathbf{y}}\|^2 \\ &= \arg\min_{\mathbf{w}} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|^2 \\ \Leftrightarrow 0 &= \mathbf{X}^T(\mathbf{y} - \mathbf{X}\mathbf{w}^*) \\ \Leftrightarrow \mathbf{w}^* &= (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} \end{aligned}$$

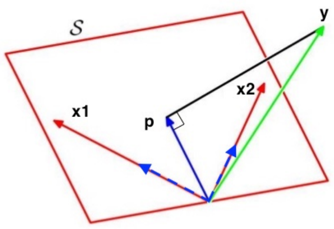

Figure 3 The case of $k = 2$ and the columns of $\mathbf{X}$ are orthogonal, as depicted by the red arrows. These two vectors span the column space $\mathcal{S}$ of $\mathbf{X}$: a 2-dimensional plane depicted by the red square. The blue arrow $\mathbf{p}$ is the projection of $\mathbf{y}$ onto $\mathcal{S}$, which can be computed by summing the projections of $\mathbf{y}$ onto each of the columns of $\mathbf{X}$, depicted as dotted blue arrows.

(see footnote 8 for how the fourth line was derived). As you might have intuited from the discussion in section 5., the solution to this problem is the projection of $\mathbf{y}$ onto the subspace of $\mathbf{X}$, depicted in figure 3.

Moreover, in this case the components of the left pseudo-inverse $(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T$ have statistical meaning:

· $(\mathbf{X}^T\mathbf{X})_{ij} = \sum_{k=1}^n x_k^{(i)} x_k^{(j)}$ is the correlation between features i and j. For example, it could be the correlation between the presence of green and brown in the sample of natural images presented to the mouse.

· $(\mathbf{X}^T\mathbf{y})_i = \sum_{k=1}^n x_k^{(i)} y_k$ is the correlation between feature k and the dependent variable. For example, it could be the correlation between the presence of green in the presented images and the number of spikes emmitted by the neuron responding to them.

Intuitively, it is this latter statistic that is important for measuring the relationship between each feature $x^{(i)}$ and y. But if two features are correlated, then they will both make the same predictions about y and thus overcompensate their contributions to the full prediction when summed together. The inverse correlation matrix $(\mathbf{X}^T\mathbf{X})^{-1}$ accounts for this by decorrelating the predictor variables $x^{(1)}, \ldots, x^{(k)}$.

7. Eigenstuff

I start by first defining eigenvectors and eigenvalues, and then show why they are useful and important.

Consider a square $n \times n$ matrix $\mathbf{A}$. We can think about this matrix as a linear *mapping* $\mathcal{T}$ defined by

$$\mathcal{T} : \mathbf{x} \mapsto \mathbf{A}\mathbf{x}$$

In English: $\mathbf{A}$ maps the vector $\mathbf{x}$ to a new vector $\mathbf{A}\mathbf{x}$. An eigenvector of $\mathbf{A}$ is a vector that maintains its direction through this mapping:

$$\mathbf{A}\mathbf{v} = \lambda\mathbf{v}$$

where λ is just a scalar. In other words, when passed through the mapping $\mathbf{A}$, an eigenvector of $\mathbf{A}$ is simply rescaled, keeping its direction fixed. The rescaling factor λ is called the eigenvalue associated with that particular eigenvector $\mathbf{v}$. Note that for this to be possible, $\mathbf{A}\mathbf{v}$ has to have the same dimensions as $\mathbf{v}$ - in other words, $\mathbf{A}$ must be square. Only square matrices have eigenvectors and eigenvalues. In fact, an $n \times n$ matrix will generally have n linearly independent eigenvectors[9].

This fact allows us to decompose such matrices into a special form that is often very useful to work with. Let $\mathbf{V} = \begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \ldots & \mathbf{v}_n \end{bmatrix}$ be a matrix containing all n eigenvectors $\mathbf{v}_1, \ldots, \mathbf{v}_n \in \mathbb{R}^n$

[9]Some special matrices have less than n, but we won't consider those here. These are called defective matrices, and they are super cool. An $n \times n$ matrix cannot have more than n linearly independent eigenvectors.

of $\mathbf{A}$ as columns. We first note that

$$\begin{aligned}\mathbf{AV} &= \begin{bmatrix}\mathbf{Av}_1 & \mathbf{Av}_2 & \dots & \mathbf{Av}_n\end{bmatrix} \\ &= \begin{bmatrix}\lambda_1\mathbf{v}_1 & \lambda_2\mathbf{v}_2 & \dots & \lambda_n\mathbf{v}_n\end{bmatrix} \\ &= \begin{bmatrix}\mathbf{v}_1 & \mathbf{v}_2 & \dots & \mathbf{v}_n\end{bmatrix}\begin{bmatrix}\lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & \dots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \dots & \lambda_n\end{bmatrix} \\ &= \mathbf{V\Lambda}\end{aligned}$$

where $\mathbf{\Lambda}$ is the $n \times n$ diagonal matrix containing the eigenvalues $\lambda_1, \dots, \lambda_n$ on the diagonal and 0's everywhere else. Because the eigenvectors of $\mathbf{A}$ are linearly independent, $\mathbf{V}$ is full rank and invertible, allowing us to write the following:

$$\mathbf{AV} = \mathbf{V\Lambda} \Leftrightarrow \mathbf{A} = \mathbf{V\Lambda V}^{-1}$$

The right-hand side of the latter equation is called the eigendecomposition of $\mathbf{A}$. Using outer products (section 3.), we can equivalently express this as

$$\mathbf{A} = \sum_{i=1}^{n} \lambda_i \mathbf{v}_i \mathbf{v}_i^\dagger$$

where $\mathbf{v}_i^\dagger$ - often called an adjoint eigenvector - is the ith row of $\mathbf{V}^{-1}$. Note that the eigendecomposition of $\mathbf{A}$ implies that it can be transformed into a diagonal matrix, i.e. *diagonalized*, by the following transformation

$$\mathbf{V}^{-1}\mathbf{AV} = \mathbf{\Lambda}$$

Because this holds for any matrix with n linearly independent eigenvectors, such matrices are said to be diagonalizable.

Why is any of this useful? Consider computing the powers of a square matrix:

$$\mathbf{A}^k = \mathbf{AAA}\dots\mathbf{A} = \mathbf{V\Lambda V}^{-1}\mathbf{V\Lambda V}^{-1}\mathbf{V\Lambda V}^{-1}\mathbf{V}\dots\mathbf{V}^{-1}\mathbf{V\Lambda V}^{-1} = \mathbf{V\Lambda I\Lambda I\Lambda}\dots\mathbf{I\Lambda V}^{-1} = \mathbf{V\Lambda}^k\mathbf{V}^{-1}$$

since $\mathbf{\Lambda V}^{-1}\mathbf{V} = \mathbf{\Lambda I} = \mathbf{\Lambda}$, by definition of the inverse and the identity matrix. Note that

$$\mathbf{\Lambda}^k = \begin{bmatrix}\lambda_1^k & 0 & \dots & 0 \\ 0 & \lambda_2^k & \dots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \dots & \lambda_n^k\end{bmatrix}$$

is a really easy matrix to compute. This can be really useful for computing functions of diagonalizable matrices, since many functions can be defined as a sums of powers[10]. We'll see an example of such a function below - the matrix exponential. We can also use the eigenvalues and eigenvectors to construct the inverse of a diagonalizable matrix:

$$A^{-1} = \mathbf{V\Lambda}^{-1}\mathbf{V}^{-1}$$

which one can easily verify satisfies the definition of an inverse matrix. This property is extremely useful when dealing with inverse matrices analytically.

[10] Namely, a large class of functions - called *analytic* functions - can be expressed as a *power series*:

$$f(x) = \sum_{i=0}^{n} a_i (x - x_0)^i$$

Replacing x and x_0 with matrices gives you the equivalent function for matrices, and requires computing powers of a matrix.

8. Example: Principal Components Analysis (PCA)

Consider a multivariate data set consisting of n observations $\mathbf{x}^{(i)}$ of d variables: $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n)} \in \mathbb{R}^d$. For example, this might be a data set of images of faces, and $x_1^{(i)}, \ldots, x_d^{(i)}$ are the grayscale values of the d pixels in the ith image. The sample covariance between variables i and j is defined as

$$\frac{1}{n}\sum_{i=1}^{n}(x_i^{(i)} - \bar{x}_i)(x_j^{(i)} - \bar{x}_j)$$

where $\bar{x}_i = \frac{1}{n}\sum_{i=1}^{n} x_i$ is the sample mean of that particular variable. We thus define the sample covariance matrix $\mathbf{\Sigma}$ of these data as the matrix containing the covariances between every pair of variables:

$$\begin{aligned}\mathbf{\Sigma} &= \begin{bmatrix} \frac{1}{n}\sum_{i=1}^{n}(x_1^{(i)} - \bar{x}_1)(x_1^{(i)} - \bar{x}_1) & \frac{1}{n}\sum_{i=1}^{n}(x_1^{(i)} - \bar{x}_1)(x_2^{(i)} - \bar{x}_2) & \ldots & \frac{1}{n}\sum_{i=1}^{n}(x_1^{(i)} - \bar{x}_1)(x_d^{(i)} - \bar{x}_d) \\ \frac{1}{n}\sum_{i=1}^{n}(x_2^{(i)} - \bar{x}_2)(x_1^{(i)} - \bar{x}_1) & \frac{1}{n}\sum_{i=1}^{n}(x_2^{(i)} - \bar{x}_2)(x_2^{(i)} - \bar{x}_2) & \ldots & \frac{1}{n}\sum_{i=1}^{n}(x_2^{(i)} - \bar{x}_2)(x_d^{(i)} - \bar{x}_d) \\ \vdots & \vdots & & \vdots \\ \frac{1}{n}\sum_{i=1}^{n}(x_d^{(i)} - \bar{x}_d)(x_1^{(i)} - \bar{x}_1) & \frac{1}{n}\sum_{i=1}^{n}(x_d^{(i)} - \bar{x}_d)(x_2^{(i)} - \bar{x}_2) & \ldots & \frac{1}{n}\sum_{i=1}^{n}(x_d^{(i)} - \bar{x}_d)(x_d^{(i)} - \bar{x}_d) \end{bmatrix} \\ &= \frac{1}{n}\sum_{i=1}^{n}(\mathbf{x}^{(i)} - \bar{\mathbf{x}})(\mathbf{x}^{(i)} - \bar{\mathbf{x}})^T\end{aligned}$$

which can be simply computed as an outer product, where $\bar{\mathbf{x}} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}^{(i)}$ is the vector containing each variable's mean $\bar{x}_i$.

Note that $\mathbf{\Sigma}_{ij} = \mathbf{\Sigma}_{ji}$ and therefore $\mathbf{\Sigma} = \mathbf{\Sigma}^T$. This has the following important implications:

· $\mathbf{\Sigma}$ is full rank

· it is therefore diagonalizable, $\mathbf{\Sigma} = \mathbf{V\Lambda V}^{-1}$, which implies that $\mathbf{V\Lambda V}^{-1} = \mathbf{V}^{-T}\mathbf{\Lambda V}^T \Leftrightarrow \mathbf{V}^{-1} = \mathbf{V}^T$. In other words, the eigenvectors of $\mathbf{\Sigma}$ are orthogonal to each other! See this by noting that $\mathbf{V}^{-1}\mathbf{V} = \mathbf{V}^T\mathbf{V} = \mathbf{I}$ so $\mathbf{v}_i^T\mathbf{v}_j = 0$ for $i \neq j$

· its eigenvalues are all real

Matrices that satisfy the equation $\mathbf{\Sigma} = \mathbf{\Sigma}^T$ are called symmetric matrices. These unique properties make them very easy to work with analytically, since their eigendecomposition $\mathbf{\Sigma} = \mathbf{V\Lambda V}^T$ doesn't require computing a matrix inverse.

Covariance matrices in particular have the additional property that they are positive semi-definite, meaning that their eigenvalues are all greater than or equal to 0. Together with the orthogonality of their eigenvectors, this implies that, given an $n \times n$ covariance matrix $\mathbf{\Sigma}$ with eigenvectors $\mathbf{v}_1, \ldots, \mathbf{v}_n$ and associated eigenvalues $\lambda_1, \ldots, \lambda_n \geq 0$, the following holds for any $\mathbf{u}$:

$$\mathbf{u}^T\mathbf{\Sigma u} = \sum_k \mathbf{u}^T\lambda_k\mathbf{v}\mathbf{v}^T\mathbf{u} = \sum_k \lambda_k(\mathbf{u}^T\mathbf{v})^2 \geq 0$$

This is in fact the defining statement of a positive semi-definite symmetric matrix, and it holds if and only if the matrix has eigenvalues greater than or equal to 0.

In many cases, we are interested in knowing how much such multivariate data varies along a particular direction $\mathbf{v}$. For example, the directions over which images of faces vary most might be particular features of faces that distinguish individuals from each other, e.g. nose size, eye height, mouth width, ... These features might be ones that primate visual systems might care about most, and thus might be of interest to a visual neuroscientist or psychologist.

Given a vector $\mathbf{v}$, we can estimate the variance along that direction by computing the variance of the scalar projections of the data onto $\mathbf{v}$. To make the math simple, we set $\mathbf{v}$ to be a unit vector with length $\|\mathbf{v}\| = 1$, so that the scalar projection of any vector $\mathbf{u}$ onto $\mathbf{v}$ is given by their inner product $\mathbf{u}^T\mathbf{v}$ (since $\mathbf{v}^T\mathbf{v} = 1$). Taking the average over all data points then gives us the following expression for the variance along direction $\mathbf{v}$:

$$\begin{aligned}
\frac{1}{n}\sum_{i=1}^{n}\left(\mathbf{x}^{(i)^T}\mathbf{v}-\left(\frac{1}{n}\sum_{j=1}^{n}\mathbf{x}^{(j)^T}\mathbf{v}\right)\right)^2 &= \frac{1}{n}\sum_{i=1}^{n}\left(\mathbf{x}^{(i)^T}\mathbf{v}-\bar{\mathbf{x}}^T\mathbf{v}\right)^2 \\
&= \frac{1}{n}\sum_{i=1}^{n}\left(\left(\mathbf{x}^{(i)}-\bar{\mathbf{x}}\right)^T\mathbf{v}\right)^2 \\
&= \frac{1}{n}\sum_{i=1}^{n}\left(\mathbf{x}^{(i)}-\bar{\mathbf{x}}\right)^T\mathbf{v}\left(\mathbf{x}^{(i)}-\bar{\mathbf{x}}\right)^T\mathbf{v} \\
&= \frac{1}{n}\sum_{i=1}^{n}\mathbf{v}^T\left(\mathbf{x}^{(i)}-\bar{\mathbf{x}}\right)\left(\mathbf{x}^{(i)}-\bar{\mathbf{x}}\right)^T\mathbf{v} \\
&= \mathbf{v}^T\left(\frac{1}{n}\sum_{i=1}^{n}\left(\mathbf{x}^{(i)}-\bar{\mathbf{x}}\right)\left(\mathbf{x}^{(i)}-\bar{\mathbf{x}}\right)^T\right)\mathbf{v} \\
&= \mathbf{v}^T\mathbf{\Sigma}\mathbf{v}
\end{aligned}$$

where in the fourth line we used the fact that for any two vectors $\mathbf{u},\mathbf{v}$, $\mathbf{u}^T\mathbf{v}=\mathbf{v}^T\mathbf{u}$, and in the first and fifth lines we used the fact that matrix multiplication is distributive ($\mathbf{AB}+\mathbf{AC}=\mathbf{A}(\mathbf{B}+\mathbf{C})$).

In Principal Components Analysis (PCA), we'd like to find the directions in data space along which the data varies the most, i.e. the direction $\mathbf{v}$ along which the data has most variance:

$$\begin{aligned}
\max_{\mathbf{v}} \quad & \mathbf{v}^T\mathbf{\Sigma}\mathbf{v} \\
\text{subject to} \quad & \|\mathbf{v}\|=1
\end{aligned}$$

We proceed to solve this optimization problem by using a Lagrange multiplier λ to implement the constraint $\mathbf{v}^T\mathbf{v}-1=0$, which is equivalent to our constraint $\|\mathbf{v}\|=1$ (doing it this way just makes the algebra easier). The solution is then given by the equation

$$\frac{\mathrm{d}}{\mathrm{d}\mathbf{v}}\left[\mathbf{v}^T\mathbf{\Sigma}\mathbf{v}-\lambda(\mathbf{v}^T\mathbf{v}-1)\right]=\mathbf{0}$$

Using the fact that $\frac{\mathrm{d}}{\mathrm{d}\mathbf{v}}\mathbf{v}^T\mathbf{\Sigma}\mathbf{v}=2\mathbf{\Sigma}\mathbf{v}$ and $\frac{\mathrm{d}}{\mathrm{d}\mathbf{v}}\mathbf{v}^T\mathbf{v}=2\mathbf{v}$, we have:

$$\begin{aligned}
&2\mathbf{\Sigma}\mathbf{v}-2\lambda\mathbf{v}=0 \\
&\Leftrightarrow \mathbf{\Sigma}\mathbf{v}=\lambda\mathbf{v}
\end{aligned}$$

In other words, the optimal direction $\mathbf{v}$ is an eigenvector of the covariance matrix $\mathbf{\Sigma}$! Plugging this back into our objective function we want to maximize, we have that the variance along $\mathbf{v}$ is given by

$$\mathbf{v}^T\mathbf{\Sigma}\mathbf{v}=\lambda\mathbf{v}^T\mathbf{v}=\lambda$$

So the direction along which the data has most variance is given by the eigenvector of $\mathbf{\Sigma}$ with largest eigenvalue. And the k-dimensional subspace that contains the most variance of the data is given by the subspace spanned by the k eigenvectors of $\mathbf{\Sigma}$ with the k largest eigenvalues.

9. Linear Dynamical Systems

Powers of matrices often pop up in dynamical systems. Consider, for example, the following discrete-time linear dynamical system:

$$\mathbf{x}_t=\mathbf{A}\mathbf{x}_{t-1}$$

whose n-dimensional state at time t is a linear transformation of the previous state at time $t-1$. If $\mathbf{A}$ is diagonalizable, we can write:

$$\begin{aligned}
\mathbf{x}_t &= \mathbf{A}\mathbf{x}_{t-1}=\mathbf{A}\mathbf{A}\mathbf{x}_{t-2}=\ldots=\mathbf{A}^t\mathbf{x}_0 \\
&= \mathbf{V}\mathbf{\Lambda}^t\mathbf{V}^{-1}\mathbf{x}_0 \\
&= \sum_{i=1}^{n}c_i\lambda_i^t\mathbf{v}_i, \quad c_i=\mathbf{v}_i^{\dagger}\mathbf{x}_0
\end{aligned}$$

Thus, the state at time t is just a linear combination of the eigenvectors of $\mathbf{A}$, each weighted by the product of (1) its associated eigenvalue raised to the power of t, and (2) the dot product between the initial condition $\mathbf{x}_0$ and its associated adjoint eigenvector. It turns out this simple expression can give us deep insight into the dynamics of this system. Consider the largest and smallest eigenvalues of $\mathbf{A}$, denoted by $\lambda_{max}, \lambda_{min}$, respectively. These two numbers on their own can already tell us a lot about the system:

- if $\lambda_{max} > 1$, then the system is unstable: as t grows, λ_{max}^t goes to infinity!
- if $\lambda_{min} < 1$, then the system is unstable: as t grows, λ_{min} goes to infinity and negative infinity, flipping back and forth between them
- if $-1 < \lambda_{min} < \lambda_{max} < 1$, then the system is stable and will decay to 0: as t grows λ_{min}^t and λ_{max}^t go to 0, as do all the eigenvalues in between
- if $\lambda_{max} = 1$, then the system is stable but doesn't decay to 0: as t grows, $\lambda_{max}^t = 1$ remains fixed, so x_t approaches $c_{max}\mathbf{v}_{max}$. In other words, the steady state of the system is given by the eigenvector associated with λ_{max} (assuming all the other eigenvalues are between -1 and 1 so that they decay to 0).
- if $\lambda_{min} = -1$, then the system is stable but doesn't decay to 0: as t grows, $\lambda_{max}^t = \pm 1$ flips between 1 and -1, so $\mathbf{x}_t$ oscillates between $\pm c_{min}\mathbf{v}_{min}$. The steady state of the system is then given by oscillations along the direction defined by the eigenvector associated with λ_{min} (again, assuming all the other eigenvalues are between -1 and 1 so that they decay to 0).

This illustrates how important and meaningful the eigenvalues and eigenvectors of a matrix are for understanding linear dynamical systems: if you iterate a linear transformation $\mathbf{A}$ over and over again, in the limit of applying it infinitely many times, the behavior of this system is determined by the eigenvectors and eigenvalues of $\mathbf{A}$.

Indeed, these notions generalize to continuous time dynamical systems as well. Consider first the matrix exponential, defined exactly as the classical exponential by the power series[11]

$$\begin{aligned} e^{\mathbf{A}} &= \mathbf{I} + \mathbf{A} + \frac{1}{2!}\mathbf{A}^2 + \frac{1}{3!}\mathbf{A}^3 + \dots \\ &= \mathbf{V}\mathbf{V}^{-1} + \mathbf{V}\mathbf{\Lambda}\mathbf{V}^{-1} + \frac{1}{2!}\mathbf{V}\mathbf{\Lambda}^2\mathbf{V}^{-1} + \frac{1}{3!}\mathbf{V}\mathbf{\Lambda}^3\mathbf{V}^{-1} + \dots \\ &= \mathbf{V}(\mathbf{I} + \mathbf{\Lambda} + \frac{1}{2!}\mathbf{\Lambda}^2 + \frac{1}{3!}\mathbf{\Lambda}^3 + \dots)\mathbf{V}^{-1} \\ &= \mathbf{V}e^{\mathbf{\Lambda}}\mathbf{V}^{-1} \end{aligned}$$

which is an easy expression to work with since taking the matrix exponential of a diagonal matrix is the same as exponentiating each of its diagonal components, i.e.

$$e^{\mathbf{\Lambda}} = \begin{bmatrix} e^{\lambda_1} & 0 & \dots & 0 \\ 0 & e^{\lambda_2} & \dots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \dots & e^{\lambda_n} \end{bmatrix}$$

(this trivially follows from the obvious fact that powers of diagonal matrices are just diagonal matrices with the elements raised to the same power). Analagously to the scalar case, it also holds that

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t}e^{\mathbf{A}t} &= \frac{\mathrm{d}}{\mathrm{d}t}\left[\mathbf{I} + \mathbf{A}t + \frac{1}{2!}\mathbf{A}^2t^2 + \frac{1}{3!}\mathbf{A}^3t^3 + \dots\right] \\ &= \mathbf{0} + \mathbf{A} + \mathbf{A}^2t + \frac{1}{2!}\mathbf{A}^3t^2 + \dots \\ &= \mathbf{A}\left(\mathbf{I} + \mathbf{A}t + \frac{1}{2!}\mathbf{A}^2t^2 + \dots\right) \\ &= \mathbf{A}e^{\mathbf{A}t} \end{aligned}$$

[11] https://en.wikipedia.org/wiki/Exponential_function#Formal_definition

Thus, given a system of linear differential equations,

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \begin{bmatrix} \frac{\mathrm{d}x_1}{\mathrm{d}t} \\ \frac{\mathrm{d}x_2}{\mathrm{d}t} \\ \vdots \\ \frac{\mathrm{d}x_n}{\mathrm{d}t} \end{bmatrix} = \mathbf{A}\mathbf{x}$$

one can easily verify that - whenever $\mathbf{A}$ is diagonalizable - the solution is given by:

$$\mathbf{x}(t) = e^{\mathbf{A}t}\mathbf{x}(0) = \sum_{i=1}^{n} c_i e^{\lambda_i t}\mathbf{v}_i, \quad c_i = \mathbf{v}_i^{\dagger}\mathbf{x}(0)$$

Again, the long run behavior of $\mathbf{x}(t)$ is determined by its eigenvalues and eigenvectors:

- if $\lambda_{max} > 0$, then the system is unstable: as t grows, $e^{\lambda_{max}t}$ goes to infinity
- if $\lambda_{max} < 0$, then the system is stable and decays to 0: as t grows, $e^{\lambda_{max}t}$ goes to 0
- if $\lambda_{max} = 0$, then the system is stable but decays to somewhere along its associated eigenvector $\mathbf{v}_{max}$: as t grows, $e^{\lambda_{max}t} = 1$, so $x(t) \to c_{max}\mathbf{v}_{max}$
- if any $\lambda_k = \alpha + \beta i$ is imaginary, then the system will exhibit oscillations along the plane defined by the real and imaginary components of its associated eigenvector $\mathbf{v}_k$: $e^{\lambda_k t}\mathbf{v}_k = e^{\alpha t}e^{\beta i t}\mathbf{v}_k = e^{\alpha t}(\cos\beta t + i\sin\beta t)\mathbf{v}_k$[12]. Crucially, imaginary eigenvalues always come in conjugate pairs with conjugate associated eigenvectors, such that the sum of their exponential-weighted eigenvectors is always real.

10. Example: Linear Recurrent Neural Network

Consider a recurrently connected network of two neurons (figure 4), connected to each other via a pair of synapses with weights w_1, w_2. A classic approach to modelling the firing activity r_i of a neuron in such a network is to write down a differential equation of the following form:

$$\tau\frac{\mathrm{d}r_i}{\mathrm{d}t} = -r_i + \sum_j w_{ij}r_j$$

This model is made up of three ingredients:

1. A *synaptic input* term $\sum_j w_{ij}r_j$, describing the input current to a neuron arising from synaptic contacts with all the other neurons in the network. Neuron j propagates its activity onto neuron i via a synapse with weight w_{ij}: if this weight is 0 then no connection exists from j to i; if it is positive then neuron j excites neuron i; if it is negative then neuron j inhibits neuron i. These synaptic weights thus allow us to model the excitatory and inhibitory interactions between neurons in the network.
2. A *leak* term $-r_i$, describing the way in which the neuron behaves in the abscence of any input: if the synaptic input term were set to 0, r_i would just decay to zero no matter where it was (since $\mathrm{d}r_i/\mathrm{d}t < 0$ whenever $r_i > 0$ and $\mathrm{d}r_i/\mathrm{d}t > 0$ whenever $r_i < 0$). Biologically, this is modelling the homeostatic effect of passive membrane ion channels pulling the membrane potential back to resting state. The synaptic inputs thus have to compete with this homeostatic leak to change the activity of the neuron.
3. A *time constant* τ, essentially encapsulating the physiological process by which a neuron changes its activity level (i.e. active/passive ion channels, synaptic transmission, etc.) into a single number that gauges how fast a neuron's activity level can change (since $|\mathrm{d}r_i/\mathrm{d}t| \propto 1/\tau$)

[12] This last step is given by Euler's formula, widely considered one of the most beautiful results in mathematics: https://en.wikipedia.org/wiki/Euler%27s_formula

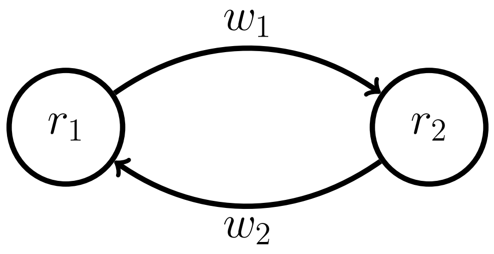

Figure 4 A two-neuron recurrent network.

This last point emphasizes the simplicity of this equation as a model of neural activity.

Another massive simplification in this model is the purely linear interactions between neurons, evident in the synaptic input term consisting of a purely linear operation on the pre-synaptic neural activities r_j. It is simply a fact that neurons interact in non-linear ways, e.g. through dendritic integration, saturation in firing rates (which can't go to infinity, unlike a real number), and positivity of firing rates (right now nothing is constraining r_j to be positive, meaning that an inhibitory connection $w_{ij} < 0$ can actually provide excitatory drive to its post-synaptic partner r_i if $r_j < 0$!). This can be easily fixed by incorporating a non-linearity into the synaptic input[13], but doing so adds a layer of complexity that is very difficult to deal with mathematically. The fundamental reason for this comes back to the intuition provided by the discrete linear dynamical system discussed at the start of section 9.. We saw that in that case, the dynamics in the system arose from repeated application of a *linear* operation, $\mathbf{A}$. Continuous-time dynamical systems can be thought of as taking the limit of the number of repetitions to infinity, by taking the length of the discrete time steps to 0. This is why the solution to a linear differential equation is just a particular infinite sum of powers of $\mathbf{A}$, compactly represented by the exponential function. When we add a non-linearity to the differential equation, however, we are no longer applying a linear operation over and over again. Non-linearities thus prevent us from being able to express the temporal dynamics as powers of matrices, and thus eliminate the utility of eigenvalues and eigenvectors for understanding them.

We'll thus consider here the purely linear model described above. As just mentioned, one reason for this is that we can use linear algebra to fully characterize the behavior of such a system. But another is that even non-linear systems can sometimes be well approximated by linear ones for restricted values of the r_i variables (e.g. near fixed points), meaning that it is often possible to get a qualitative description of a non-linear system's dynamics by stitching together a bunch of linear systems. Thus, understanding the mechanics of linear dynamics is fundamental to understanding network dynamics in general and thus serves as an excellent starting point for understanding recurrent neural networks. It is no coincidence that several seminal papers in the network dynamics literature analyze purely linear recurrent networks. As we'll see below, such networks can in fact exhibit a number of qualitatively different behaviors.

To really gain some intuition for these mechanics, we'll consider the case of just two neurons connected to each other:

$$\begin{aligned}\tau\frac{\mathrm{d}r_1}{\mathrm{d}t} &= -r_1 + w_2 r_2\\ \tau\frac{\mathrm{d}r_2}{\mathrm{d}t} &= -r_2 + w_1 r_1\end{aligned}$$

[13] For example, you might replace the synaptic input term with

$$\phi\left(\sum_j w_{ij} r_j\right)$$

and encapsulate an abstraction of dendritic integration in the function $\phi()$. Another possibility is

$$\sum_j w_{ij}\phi(r_j)$$

which would allow you to incorporate the positivity and saturation of the neural activity variables r_j by using e.g. a logistic function for $\phi()$. Or, some combination thereof.

which can be compactly written in matrix form as

$$\tau\frac{\mathrm{d}\mathbf{r}}{\mathrm{d}t} = -\mathbf{r} + \mathbf{W}\mathbf{r} = \underbrace{(\mathbf{W} - \mathbf{I})}_{\mathbf{A}}\mathbf{r}$$

where

$$\mathbf{W} = \begin{bmatrix} 0 & w_2 \\ w_1 & 0 \end{bmatrix}$$

is the weight matrix containing the synaptic weights of all the connections in the network. As we saw above, the solution to this differential equation given an initial condition $\mathbf{r}(0)$ is given by

$$\mathbf{r}(t) = e^{\mathbf{A}\frac{t}{\tau}}\mathbf{r}(0) = \mathbf{V}e^{\mathbf{\Lambda}\frac{t}{\tau}}\mathbf{V}^{-1}\mathbf{r}(0)$$

where $\mathbf{V}, \mathbf{\Lambda}$, as above, contain the eigenvectors and eigenvalues of $\mathbf{A} = \mathbf{W} - \mathbf{I}$.

As we saw above, knowing the eigenvalues of this system is sufficient to characterize many important aspects of its behavior. So what are they? It turns out we can actually calculate the eigenvectors and eigenvalues of a 2×2 matrix by hand. First, note that the eigenvalue equation implies the following:

$$\begin{aligned} \mathbf{A}\mathbf{v} &= \lambda\mathbf{v} \\ \Leftrightarrow (\mathbf{A} - \lambda\mathbf{I})\mathbf{v} &= 0 \end{aligned}$$

This innocent-looking equation in fact has massive implications. Firstly, it means that $\mathbf{v}$ is in the nullspace of the matrix $\mathbf{A} - \lambda\mathbf{I}$ (cf. section 4.). This in turn implies that this matrix is low-rank, since a full-rank matrix doesn't have a nullspace (namely, it has a 0-dimensional nullspace). If the matrix is low-rank, then it is singular (cf. section 5.). That means that a particular property of this matrix, called its determinant[14] $|\mathbf{A} - \lambda\mathbf{I}|$, must be equal to 0. Thus, all we need to do to obtain the eigenvalues is solve the following equation for λ, called the characteristic equation:

$$|\mathbf{A} - \lambda\mathbf{I}| = 0$$

Importantly, the determinant of a matrix is a polynomial expression of its entries, which entails that the left-hand side of the characteristic equation is just an nth order polynomial, called the characteristic polynomial of $\mathbf{A}$. The roots of the characteristic polynomial are the eigenvalues of $\mathbf{A}$, so solving for the eigenvalues of a matrix requires just solving for the roots of a polynomial. In the case that $\mathbf{A}$ is a 2×2 matrix, plugging in the formula for the determinant of a 2×2 matrix yields a quadratic equation in λ:

$$\begin{aligned} \mathbf{A} &= \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \\ \Rightarrow |\mathbf{A} - \lambda\mathbf{I}| &= (a_{11} - \lambda)(a_{22} - \lambda) - a_{21}a_{12} = 0 \\ \Leftrightarrow \lambda^2 - \underbrace{(a_{11} + a_{22})}_{\mathrm{Tr}[\mathbf{A}]}\lambda &+ \underbrace{a_{11}a_{22} - a_{21}a_{12}}_{|\mathbf{A}|} = 0 \end{aligned}$$

where $\mathrm{Tr}[\mathbf{A}]$, called the trace of $\mathbf{A}$, is the sum of its diagonal terms, and the last term in this quadratic polynomial is just the determinant of $\mathbf{A}$.

[14] The determinant $|\mathbf{A}|$ of a matrix $\mathbf{A}$ is a somewhat difficult concept to define. The most intuitive description is that it is the volume scaling factor of the linear transformation described by the matrix. For example, consider an n-dimensional cube consisting of 2^n vertices given by 2^n vectors. Multiplying each of these vectors by $\mathbf{A}$ then results in a distortion of this cube into some new polygon, which will have a different volume. The determinant of $\mathbf{A}$ determines the ratio between this volume and the volume of the original cube, i.e. the factor by which the volume has changed after transformation by $\mathbf{A}$. The key fact here is that a *singular* (i.e. low-rank) matrix will transform this n-dimensional cube into a lower-dimensional object, and thus it no longer makes sense to compare their volumes. The determinant of a singular matrix is thus always 0. Determinants are hard to compute for large matrices, but for 2×2 matrices it turns out to be pretty easy:

$$\mathbf{A} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \Rightarrow |\mathbf{A}| = ad - bc$$

A pretty cool fact is that the determinant of a matrix turns out to be equal to the product of its eigenvalues. A low-rank matrix has eigenvalues equal to 0, which is consistent with the fact that its determinant is 0.

Coming back to our recurrent network example, we can obtain the eigenvalues of the system by simply plugging in the trace and determinant of **A** into the quadratic equation to find the roots of its characteristic polynomial:

$$\begin{aligned}
\mathrm{Tr}[\mathbf{A}] = -2, \quad |\mathbf{A}| = 1 - w_1 w_2 \\
\Rightarrow \lambda_\pm &= \frac{\mathrm{Tr}[\mathbf{A}] \pm \sqrt{\mathrm{Tr}[\mathbf{A}]^2 - 4|\mathbf{A}|}}{2} \\
&= \frac{-2 \pm \sqrt{4 - 4(1 - w_1 w_2)}}{2} \\
&= -1 \pm \sqrt{w_1 w_2}
\end{aligned}$$

Given these eigenvalues, we could now go on to derive their associated eigenvectors, but I leave that as an exercise for the reader. Armed with this expression for the eigenvalues in terms of w_1, w_2, we're now in a position to deeply understand the behavior of the system under different settings of w_1, w_2.

Consider first the case in which r_1 and r_2 weakly excite each other:

$$\begin{cases} w_1 = 0.5 \\ w_2 = 0.5 \end{cases} \Rightarrow \begin{cases} \lambda_+ = -0.5 \\ \lambda_- = -1.5 \end{cases}$$

The eigenvalues in this case are both real and negative, so the system is stable and simply decays to 0. This is illustrated in figure 5, which depicts the dynamics of this system in two different ways. On the left, $r_1(t), r_2(t)$ are plotted as a function of time, starting from a single random initial condition. On the right, I have plotted the system in so-called state-space, which is just the space of all possible states $\mathbf{r}(t)$ of the system. I have plotted the sequence of states visited starting from six different random initial conditions. This is done by simply plotting $r_1(t)$ vs. $r_2(t)$ for a long sequence of timepoints t. On a mechanistic level, what these figures illustrate is the fact that the incoming excitation to each neuron is insufficient to compensate for the membrane leak, and thus the activity levels of both neurons systematically decay to 0 regardless of where they start.

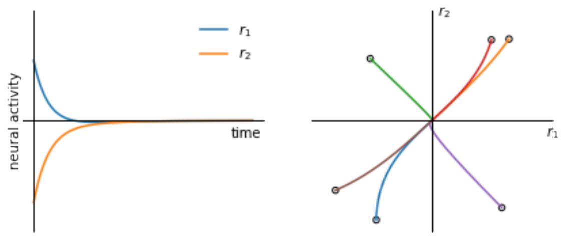

Figure 5 Dynamics of network of two weakly connected excitatory neurons: $w_1 = w_2 = 0.5$. Left panel shows timecourse of $r_1(t), r_2(t)$ for a random initial condition. Right panel shows trajectories of the system in state space, starting from six random initial conditions marked with open black circles. Each of these trajectories moves from its initial condition towards the origin and ends there.

So what happens if we now crank up these excitatory connections? For example, let's triple the weights of the excitatory connections:

$$\begin{cases} w_1 = 1.5 \\ w_2 = 1.5 \end{cases} \Rightarrow \begin{cases} \lambda_+ = 0.5 \\ \lambda_- = -2.5 \end{cases}$$

We now have an eigenvalue that is real and greater than 0, thus making the system unstable. As you can see in figure 6, the activity in the network quickly grows uncontrollably to infinity, even for very small initial conditions. As one neuron's activity grows, the excitatory input to the other grows with it, thus increasing the activity of the second neuron which then drives the activity of the first. This strong positive feedback loop quickly leads to an explosion of the overal activity level in the network. Note as well that the activity only grows along one direction in state space: this is the direction of the eigenvector associated with the unstable eigenvalue λ_+.

So let's try scaling back the strength of this positive feedback loop to the point at which the instability disappears. In other words, lets weaken the positive feedback just enough so that the network is no longer unstable, but not too much so that the leak takes over and brings the network

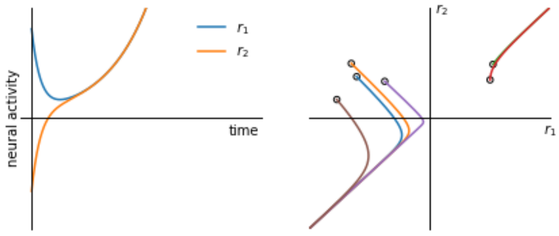

Figure 6 Dynamics of network of two strongly connected excitatory neurons: $w_1 = w_2 = 1.5$. Panels exactly as in figure 5

activity state back to 0. Recall that instability is indicated by a positive eigenvalue and decay to 0 is indicated by a negative eigenvalue. Thus, the point in between is an eigenvalue equal to 0:

$$\begin{cases} w_1 = 1 \\ w_2 = 1 \end{cases} \Rightarrow \begin{cases} \lambda_+ = -1 + 1 = 0 \\ \lambda_- = -1 - 1 = -2 \end{cases}$$

In this case, the math tells us that the presence of the eigenvector associated with λ_+ in the network's time varying state $\mathbf{r}(t)$ won't change over time, since $e^{\lambda_+ t} = e^0 = 1$ does not depend on t. The other eigenvector, however, has a negative eigenvalue associated with it, so activity should decay along that direction. This is exactly what we find in simulation: as figure 7 shows, the network activity in state space always decays to its projection onto the eigenvector associated with λ_+. This leads to a continuum of stable non-zero fixed points lying along this line. Whatever my initial condition may be, I'll always end up at the point on this line closest to where I started. If I start on the line, then I'll just stay there. For this reason, such networks are called line attractors. This flavor of network is typically used to model working memory, since it is capable of keeping a fixed state over an extended period of time (infinite time in the exact linear case here). This state could encode the value of some transiently presented stimulus in the past, thus allowing the recurrent network to store information over time, which is the defining property of (working) memory (cf. *parametric* working memory).

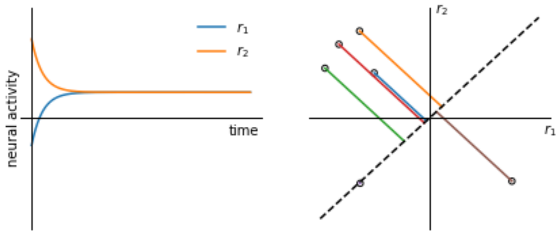

Figure 7 Dynamics of network of two excitatory neurons with finely tuned weights so as to form a line attractor: $w_1 = w_2 = 1.0$. Panels exactly as in figure 5. Dashed black line in second panel indicates direction of eigenvector associated with λ_+.

However, a problem of using such a network in the brain is that, by construction, they are on the border of instability. Perturb the weights just a tiny bit and you might knock λ_+ to a positive value. The eigenvector associated with λ_+ would then become an unstable direction as in figure 6, and the brain would explode into an epileptic fit. Figuring out how to build memory networks that avoid this so-called "fine-tuning" problem is an active area of research in the working memory literature.

Since instability must be prevented at all costs in the brain, it is unsurprising that inhibitory neurons are found all over the brain. To incorporate this into our model, lets take neuron 2 to be inhibitory, so that its synaptic weight onto neuron 1 is negative:

$$\begin{cases} w_1 = 1.5 \\ w_2 = -1.5 \end{cases} \Rightarrow \begin{cases} \lambda_+ = -1 + 1.5i \\ \lambda_- = -1 - 1.5i \end{cases}$$

We now have two imaginary eigenvalues, with negative real parts. What does this mean? As we noted above, Euler's formula (cf. footnote 12) gives us that the exponential of an imaginary number is a sum of trigonometric functions: $e^{\beta t i} = \cos(\beta t) + i\sin(\beta t)$. These two functions[15] oscillate as

[15] An attentive but naive reader might worry about the presence of an imaginary number multiplying the sine

a function of t, leading to oscillations in the network state as shown in figure 8. The negative real part $\mathrm{Re}(\lambda) = -1$ puts a factor of $e^{\mathrm{Re}(\lambda)t/\tau} = e^{-t/\tau}$ in front of these oscillatory functions, making them decay to 0 over time. Mechanistically, what is going on is that r_1 is continually exciting r_2 via its excitatory connection w_1, thus increasing the overall network activity. But this in turn increases the inhibitory input to r_1, subsequently silencing it and decreasing the overall network activity. As r_2 then decreases, the inhibition to r_1 drops with it, thus allowing r_1 to rise and excite r_2 all over again, returning the network's overall activity to a slightly higher level. This flip-flopping continues in a decaying fashion, since the total inhibition in the network still trumps the total excitation due to the inhibitory effect of the membrane leak, thus ultimately bringing the network to a halt. Intuitively, such oscillations will arise whenever the neurons have opposite effects on each other, i.e. one excites and the other inhibits. Mathematically, this turns out to be exactly true, since the eigenvalues will be imaginary whenever w_1 and w_2 are of opposite sign (so that the number inside the square root is negative).

Another interesting insight provided by the Euler formula is that the frequency of the oscillations will depend on the imaginary part of the eigenvalues (the part that multiplies t inside the sine and cosine functions). This is shown in the rightmost panel of figure 8, by plotting the activity of one of the neurons for increasing values of w_1. Naturally, the more strongly the inhibitory neuron gets excited the faster it will silence the excitatory neuron, thus pushing forward the oscillation at a faster pace.

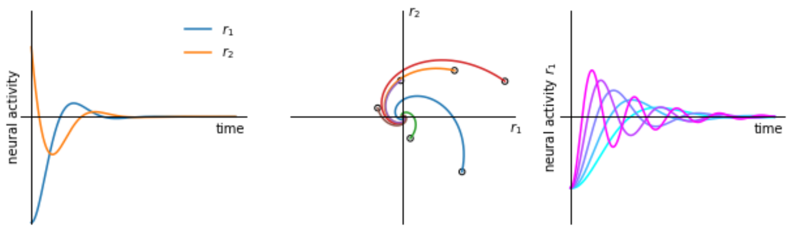

Figure 8 Dynamics of network with one excitatory and one inhibitory neuron: $w_1 = 1.5, w_2 = -1.5$. First two panels exactly as in figure 5. Third panel shows timecourses of excitatory neuron $r_1(t)$ for different settings of w_1, increasing from 1.5 to 24 along the blue to pink color gradient.

11. Useful resources

- MIT OpenCourseWare "Linear Algebra" course by Gilbert Strang (`https://ocw.mit.edu/courses/mathematics/18-06-linear-algebra-spring-2010/video-lectures/`)
- 3Blue1Brown channel on YouTube (`https://www.youtube.com/playlist?list=PLZHQObOWTQDPD3MizzM2xVFit ab`)

in this expression. Note that the two eigenvalues are complex conjugates, so that $\lambda_+ + \lambda_- = -2$ is always real. In fact, if you work out their associated eigenvectors and adjoint eigenvectors you'll find that they also come in conjugate pairs. The final result is that when you work out the actual solution to the system, i.e. $\mathbf{V}e^{\mathbf{\Lambda}\frac{t}{\tau}}\mathbf{V}^{-1}\mathbf{r}(0)$, the conjugacies lead all the imaginary components to cancel out, leaving a real solution in terms of sines and cosines of $|\mathrm{Im}(\lambda)|t$ (where $\mathrm{Im}(\lambda)$ denotes the imaginary part of the eigenvalue λ).