Notes on linear algebra

James Aspnes

October 11, 2012

1 Matrices

We've seen that a **sequence** $a_1, a_2, \ldots, a_n$ is really just a function from some index set ($\{1 \ldots n\}$ in this case) to some codomain, where $a_i = a(i)$ for each i. What if we have two index sets? Then we have a two-dimensional structure:

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \\ A_{31} & A_{32} \end{bmatrix}$$

where $A_{ij} = a(i, j)$ and the domain of the function is just the cross-product of the two index sets. Such a structure is called a **matrix**. The values A_{ij} are called the **elements** or **entries** of the matrix. A sequence of elements with the same first index is called a **row** of the matrix; similarly, a sequence of elements with the same second index is called a **column**. The **dimension** of the matrix specifies the number of rows and the number of columns: the matrix above has dimension $(3, 2)$, or, less formally, it is a 3×2 matrix.[1] A matrix is **square** if it has the same number of rows and columns.

Note: The convention in matrix indices is to count from 1 rather than 0. In Computer Science terms, matrices are written in FORTRAN.

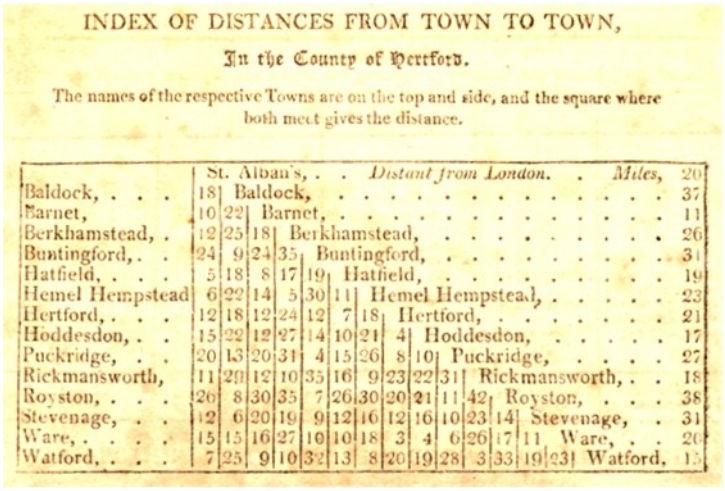
By convention, variables representing matrices are usually written with capital letters. (This is to distinguish them from lower-case **scalars**, which are single numbers.)

1.1 Interpretation

We can use a matrix any time we want to depict a function of two arguments (over small finite sets if we want it to fit on one page). A typical example (that predates the formal notion of a matrix by centuries) is a table of distances between cities or towns, such as this example from 1807:[2]

[1] The convention for both indices and dimension is that rows come before columns.

[2] The original image is taken from `http://www.hertfordshire-genealogy.co.uk/data/books/books-3/book-0370-cooke-1807.htm`. As an exact reproduction of a public domain document, this image is not subject to copyright in the United States.



INDEX OF DISTANCES FROM TOWN TO TOWN,

In the County of Hertford.

The names of the respective Towns are on the top and side, and the square where both meet gives the distance.

	St. Alban's,		*Distant from London.*		*Miles,*
Baldock,	18	Baldock,			37
Barnet,	10 22	Barnet,			11
Berkhamstead,	12 25 18	Berkhamstead,			26
Buntingford,	24 9 24 35	Buntingford,			31
Hatfield,	5 18 8 17 19	Hatfield,			19
Hemel Hempstead	6 22 14 5 30 11	Hemel Hempstead,			23
Hertford,	12 18 12 24 12 7 18	Hertford,			21
Hoddesdon,	15 22 12 27 14 10 21 4	Hoddesdon,			17
Puckridge,	20 13 20 31 4 15 26 8 10	Puckridge,			27
Rickmansworth,	11 28 12 10 35 16 9 23 22 31	Rickmansworth,			18
Royston,	26 8 30 35 7 26 30 20 21 11 42	Royston,			38
Stevenage,	12 6 20 19 9 12 16 12 16 10 23 14	Stevenage,			31
Ware,	15 15 16 27 10 10 18 3 4 6 26 17 11	Ware,			20
Watford,	7 25 9 10 32 13 8 20 19 28 3 33 19 23	Watford,			15

Because distance matrices are *symmetric* (see below), usually only half of the matrix is actually printed.

Another example would be a matrix of counts. Suppose we have a set of destinations D and a set of origins O. For each pair $(i, j) \in D \times O$, let C_{ij} be the number of different ways to travel from j to i. For example, let origin 1 be Bass Library, origin 2 be AKW, and let destinations 1, 2, and 3 be Bass, AKW, and SML. Then there is 1 way to travel between Bass and AKW (walk), 1 way to travel from AKW to SML (walk), and 2 ways to travel from Bass to SML (walk above-ground or below-ground). If we assume that we are not allowed to stay put, there are 0 ways to go from Bass to Bass or AKW to AKW, giving the matrix

$$C = \begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 2 & 1 \end{bmatrix}$$

Wherever we have counts, we can also have probabilities. Suppose we have a particle that moves between positions $1 \ldots n$ by flipping a coin, and moving up with probability $\frac{1}{2}$ and down with probability $\frac{1}{2}$ (staying put if it would otherwise move past the endpoints). We can describe this process by a **transition matrix** P whose entry P_{ij} gives the probability of moving to i starting from j. For example, for $n = 4$, the transition matrix is

$$P = \begin{bmatrix} 1/2 & 1/2 & 0 & 0 \\ 1/2 & 0 & 1/2 & 0 \\ 0 & 1/2 & 0 & 1/2 \\ 0 & 0 & 1/2 & 1/2 \end{bmatrix}.$$

Finally, the most common use of matrices in linear algebra is to represent the coefficients of a **linear transformation**, which we will describe later.

1.2 Operations on matrices

1.2.1 Transpose of a matrix

The **transpose** of a matrix A, written A' or $A^\top$, is obtained by reversing the indices of the original matrix; $(A')_{ij} = A_{ji}$ for each i and j. This has the effect of turning rows into columns and vice versa:

$$A' = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \\ A_{31} & A_{32} \end{bmatrix}' = \begin{bmatrix} A_{11} & A_{21} & A_{31} \\ A_{12} & A_{22} & A_{32} \end{bmatrix}$$

If a matrix is equal to its own transpose (i.e., if $A_{ij} = A_{ji}$ for all i and j), it is said to be **symmetric**. The transpose of an $n \times m$ matrix is an $m \times n$ matrix, so only square matrices can be symmetric.

1.2.2 Sum of two matrices

If we have two matrices A and B with the same dimension, we can compute their sum $A + B$ by the rule $(A + B)_{ij} = A_{ij} + B_{ij}$. Another way to say this is that matrix sums are done term-by-term: there is no interaction between entries with different indices.

For example, suppose we have the matrix of counts C above of ways of getting between two destinations on the Yale campus. Suppose that upperclassmen are allowed to also take the secret Science Hill Monorail from the sub-basement of Bass Library to the sub-basement of AKW. We can get the total number of ways an upperclassman can get from each origin to each destination by adding to C a second matrix M giving the paths involving monorail travel:

$$C + M = \begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 2 & 1 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 2 & 0 \\ 2 & 1 \end{bmatrix}.$$

1.2.3 Product of two matrices

Suppose we are not content to travel once, but have a plan once we reach our destination in D to travel again to a final destination in some set F. Just as we constructed the matrix C (or $C + M$, for monorail-using upperclassmen) counting the number of ways to go from each point in O to each point in D, we can construct a matrix Q counting the number of ways to go from each point in D to each point in F. Can we combine these two matrices to compute the number of ways to travel $O \to D \to F$?

The resulting matrix is known as the **product** QC. We can compute each entry in QC by taking a sum of products of entries in Q and C. Observe that the number of ways to get from k to i via some single intermediate point j is just $Q_{ij}C_{jk}$. To get all possible routes, we have to sum over all possible intermediate points, giving $(QC)_{ik} = \sum_j Q_{ij}C_{jk}$.

This gives the rule for multiplying matrices in general: to get $(AB)_{ik}$, sum $A_{ij}B_{jk}$ over all intermediate values j. This works only when the number of columns in A is the same as the number of rows in B (since j has to vary over the same range in both matrices), i.e., when A is an $n \times m$ matrix and B is an $m \times s$ matrix for some n, m, and s. If the dimensions of the matrices don't match up like this, the matrix product is *undefined*. If the dimensions do match, they are said to be **compatible**.

For example, let $B = (C + M)$ from the sum example and let A be the number of ways of getting from each of destinations 1 = Bass, 2 = AKW, and 3 = SML to final destinations 1 = Heaven and 2 = Hell. After consulting with appropriate representatives of the Divinity School, we determine that one can get to either Heaven or Hell from any intermediate destination in one way by dying (in a state of grace or sin, respectively), but that Bass Library provides the additional option of getting to Hell by digging. This gives a matrix

$$A = \begin{bmatrix} 1 & 1 & 1 \\ 2 & 1 & 1 \end{bmatrix}.$$

We can now compute the product

$$A(C+M) = \begin{bmatrix} 1 & 1 & 1 \\ 2 & 1 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 2 & 0 \\ 2 & 1 \end{bmatrix} = \begin{bmatrix} 1\cdot 0 + 1\cdot 2 + 1\cdot 2 & 1\cdot 1 + 1\cdot 0 + 1\cdot 1 \\ 2\cdot 0 + 1\cdot 2 + 1\cdot 2 & 2\cdot 1 + 1\cdot 0 + 1\cdot 1 \end{bmatrix} = \begin{bmatrix} 4 & 2 \\ 4 & 3 \end{bmatrix}.$$

One special matrix I (for each dimension $n \times n$) has the property that $IA = A$ and $BI = B$ for all matrices $AandB$ with compatible dimension. This matrix is known as the **identity matrix**, and is defined by the rule $I_{ii} = 1$ and $I_{ij} = 0$ for $i \neq j$. It is not hard to see that in this case $(IA)_{ij} = \sum_k I_{ik}A_{kj} = I_{ii}A_{ij} = A_{ij}$, giving $IA = A$; a similar computation shows that $BI = B$. With a little more effort (omitted here) we can show that I is the *unique* matrix with this identity property.

1.2.4 The inverse of a matrix

A matrix A is **invertible** if there exists a matrix A^{-1} such that $AA^{-1} = A^{-1}A = 1$. This is only possible if A is square (because otherwise the dimensions don't work) and may not be possible even then. Note that it is enough to find a matrix such that $A^{-1}A = I$ to show that A is invertible.

To try to invert a matrix, we start with the pair of matrices A, I (where I is the identity matrix defined above) and multiply both sides of the pair from the left by a sequence of transformation matrices $B_1, B_2, \ldots B_k$ until $B_k B_{k-1} \cdots B_1 A = I$. At this point the right-hand matrix will be $B_k B_{k-1} \cdots B_1 = A^{-1}$. (We could just keep track of all the B_i, but it's easier to keep track of their product.)

How do we pick the B_i? These will be matrices that (a) multiply some row by a scalar, (b) add a multiple of one row to another row, or (c) swap two rows. We'll use the first kind to make all the diagonal entries equal one, and

the second kind to get zeroes in all the off-diagonal entries. The third kind will be saved for emergencies, like getting a zero on the diagonal.

That the operations (a), (b), and (c) correspond to multiplying by a matrix is provable but tedious.[3] Given these operations, we can turn any invertible matrix A into I by working from the top down, rescaling each row i using a type (a) operation to make $A_{ii} = 1$, then using a type (b) operation to subtract A_{ji} times row i from each row $j > i$ to zero out A_{ji}, then finally repeating the same process starting at the bottom to zero out all the entries above the diagonal. The only way this can fail is if we hit some $A_{ii} = 0$, which we can swap with a nonzero A_{ji} if one exists (using a type (c) operation). If all the rows from i on down have a zero in the i column, then the original matrix A is not invertible. This entire process is known as Gauss-Jordan elimination.

This procedure can be used to solve matrix equations: if $AX = B$, and we know A and B, we can compute X by first computing A^{-1} and then multiplying $X = A^{-1}AX = A^{-1}B$. If we are not interested in A^{-1} for its own sake, we can simplify things by substituting B for I during the Gauss-Jordan elimination procedure; at the end, it will be transformed to X.

Example Original A is on the left, I on the right.

Initial matrices:

$$\begin{bmatrix} 2 & 0 & 1 \\ 1 & 0 & 1 \\ 3 & 1 & 2 \end{bmatrix} \quad \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Divide top row by 2:

$$\begin{bmatrix} 1 & 0 & 1/2 \\ 1 & 0 & 1 \\ 3 & 1 & 2 \end{bmatrix} \quad \begin{bmatrix} 1/2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Subtract top row from middle row and 3*top row from bottom row:

$$\begin{bmatrix} 1 & 0 & 1/2 \\ 0 & 0 & 1/2 \\ 0 & 1 & 1/2 \end{bmatrix} \quad \begin{bmatrix} 1/2 & 0 & 0 \\ -1/2 & 1 & 0 \\ -3/2 & 0 & 1 \end{bmatrix}$$

Swap middle and bottom rows:

$$\begin{bmatrix} 1 & 0 & 1/2 \\ 0 & 1 & 1/2 \\ 0 & 0 & 1/2 \end{bmatrix} \quad \begin{bmatrix} 1/2 & 0 & 0 \\ -3/2 & 0 & 1 \\ -1/2 & 1 & 0 \end{bmatrix}$$

Multiply bottom row by 2:

[3] The tedious details: To multiple row r by a, use a matrix B with $B_{ii} = 1$ when $i \neq r$, $B_{rr} = a$, and $B_{ij} = 0$ for $i \neq j$; to add a times row r to row s, use a matrix B with $B_{ii} = 1$ when $i \neq r$, $B_{rs} = a$, and $B_{ij} = 0$ for all other pairs ij; to swap rows r and s, use a matrix B with $B_{ii} = 1$ for $i \notin \{r, s\}$, $B_{rs} = B_{sr} = 1$, and $B_{ij} = 0$ for all other pairs ij.

$$\begin{bmatrix} 1 & 0 & 1/2 \\ 0 & 1 & 1/2 \\ 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} 1/2 & 0 & 0 \\ -3/2 & 0 & 1 \\ -1 & 2 & 0 \end{bmatrix}$$

Subtract (1/2)*bottom row from top and middle rows:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} 1 & -1 & 0 \\ -1 & -1 & 1 \\ -1 & 2 & 0 \end{bmatrix}$$

and we're done. (It's probably worth multiplying the original A by the alleged A^{-1} to make sure that we didn't make a mistake.)

1.2.5 Scalar multiplication

Suppose we have a matrix A and some constant c. The **scalar product** cA is given by the rule $(cA)_{ij} = cA_{ij}$; in other words, we multiply (or *scale*) each entry in A by c. The quantity c in this context is called a **scalar**; the term *scalar* is also used to refer to any other single number that might happen to be floating around.

Note that if we only have scalars, we can pretend that they are 1×1 matrices; $a + b = a_{11} + b_1$ and $ab = a_{11}b_{11}$. But this doesn't work if we multiply a scalar by a matrix, since cA (where c is considered to be a matrix) is only defined if A has only one row. Hence the distinction between matrices and scalars.

1.3 Matrix identities

For the most part, matrix operations behave like scalar operations, with a few important exceptions:

1. Matrix multiplication is only defined for matrices with compatible dimensions.

2. Matrix multiplication is *not commutative*: in general, we do not expect that $AB = BA$. This is obvious when one or both of A and B is not square (one of the products is undefined because the dimensions aren't compatible), but may also be true even if A and B are both square.

For a simple example of a non-commutative pair of matrices, consider

$$\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 0 \\ 1 & 1 \end{bmatrix} \neq \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1 & 2 \end{bmatrix}.$$

On the other hand, matrix multiplication *is* associative: $A(BC) = (AB)C$. The proof is by expansion of the definition. First compute $(A(BC))_{ij} = \sum_k A_{ik}(BC)_{kj} = \sum_k \sum_m A_{ik}B_{km}C_{mj}$. Then compute $((AB)C)_{ij} = \sum_m (AB)_{im}C_{mj} = \sum_m \sum_k A_{ik}B_{km}C_{mj} = \sum_k \sum_m A_{ik}B_{km}C_{mj} = (A(BC))_{ij}$.

So despite the limitations due to non-compatibility and non-commutativity, we still have:

Associative laws $A + (B + C) = (A + B) + C$ (easy), $(AB)C = A(BC)$ (see above). Also works for scalars: $c(AB) = (cA)B = A(cB) and (cd)A = c(dA) = d(cA)$.

Distributive laws $A(B+C) = AB+BC$, $A(B+C) = AB+AC$. Also works for scalars: $c(A + B) = cA + cB, (c + d)A = cA + dA$.

Additive identity $A + 0 = 0 + A = A$, where 0 is the all-zero matrix of the same dimension as A.

Multiplicative identity $AI = A, IA = A, 1A = A, A1 = A$, where I is the identity matrix of appropriate dimension in each case and 1 is the scalar value 1.

Inverse of a product $(AB)^{-1} = B^{-1}A^{-1}$. Proof: $(B^{-1}A^{-1})(AB) = B^{-1}(A^{-1}A)B = B^{-1}(IB) = B^{-1}B = I$, and similarly for $(AB)(B^{-1}A^{-1})$.

Transposes $(A + B)' = A' + B'$ (easy), $(AB)' = B'A'$ (a little trickier). $(A^{-1})' = (A')^{-1}$, provided A^{-1} exists (proof: $A'(A^{-1})' = (A^{-1}A)' = I' = I$).

Using these identities, we can do arithmetic on matrices without knowing what their actual entries are, so long as we are careful about non-commutativity. So for example we can compute

$$(A + B)^2 = (A + B)(A + B) = A^2 + AB + BA + B^2.$$

Similarly, if for square A we have

$$S = \sum_{n \in \mathbb{N}} A^n,$$

(where $A^0 = I$) we can solve the equation

$$S = I + AS$$

by first subtracting AS from both sides to get

$$IS - AS = I$$

then applying the distributive law:

$$(I - A)S = I$$

and finally multiplying both sides from the left by $(I - A)^{-1}$ to get

$$S = (I - A)^{-1},$$

assuming $I - A$ is invertible.

2 Vectors

A $1 \times n$ or $n \times 1$ matrix is called a **vector**. A $1 \times n$ matrix is called a **row vector** for obvious reasons; similarly, an $n \times 1$ matrix is called a **column vector**.

Vectors behave exactly like matrices in every respect; however, they are often written with lowercase letters to distinguish them from their taller and wider cousins. If this will cause confusion with scalars, we can disambiguate by writing vectors with a little arrow on top: $\vec{x}$ or in boldface: $\mathbf{x}$. Often we will just hope it will be obvious from context which variables represent vectors and which represent scalars, since writing all the little arrows can take a lot of time.

When extracting individual coordinates from a vector, we omit the boring index and just write x_1, x_2, etc. This is done for both row and column vectors, so rather than write x'_i we can just write x_i.

2.1 Geometric interpretation

We can use a vector to represent the coordinates of a point in space. In general, given some point in the n-**dimensional Euclidean space** $\mathbb{R}^n$, we consider it as an $n \times 1$ column vector (row vectors work too, but the convention is to use column vectors because it makes the matrix-vector product Ax look like function application). The set of all such vectors for a given fixed dimension form a **vector space**.

For example, we could represent the latitude and longitude of the major world landmarks Greenwich Observatory, Mecca, and Arthur K. Watson Hall by the column vectors $\begin{bmatrix} 0.0 \\ 0.0 \end{bmatrix}$, $\begin{bmatrix} 21.45 \\ 39.82 \end{bmatrix}$, and $\begin{bmatrix} 41.31337 \\ -72.92508 \end{bmatrix}$.

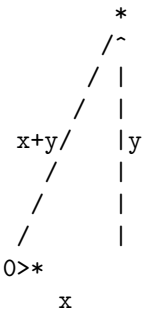
Pedantic note: The surface of the Earth is not really a Euclidean space, despite the claims of the Flat Earth Society.

In running text, we can represent these column vectors as transposed row vectors, e.g. $[21.4539.82]'$ for the coordinates of Mecca. Quite often we will forget whether we are dealing with a column vector or a row vector specifically and just write the sequence of entries, e.g. $(21.5439.82)$ or $(21.54, 39.82)$.

We can use vectors to represent any quantities that can be expressed as a sequence of coordinates in $\mathbb{R}$ (or more generally, over any *field*: see AlgebraicStructures and AbstractLinearAlgebra). A common use is to represent *offsets*—the difference between the coordinates of two locations—and *velocities*—the rate at which the coordinates of a moving object change over time. So, for example, we might write the velocity of a car that is moving due northeast at 100 km/h as the vector $(100/\sqrt{2}100/\sqrt{2})'$, where each entry corresponds to the speed if we look only at the change in the north-south or west-east position. We can also think of coordinates of points as themselves offsets from some **origin** at position 0—Greenwich Observatory in the case of latitude and longitude.

2.2 Sums of vectors

We can add vectors term-by-term just as we can add any other matrices. For vectors representing offsets (or velocities) the geometric interpretation of $x + y$ is that we attach y to the end of x (or vice versa, since addition *is* commutative) and take the combined offset, as in this picture:



This can be used to reduce the complexity of pirate-treasure instructions:

1. Yargh! Start at the olde hollow tree on Dead Man's Isle, *if ye dare*.
2. Walk 10 paces north.
3. Walk 5 paces east.
4. Walk 20 paces south.
5. Walk $6\sqrt{2}$ paces northwest.
6. Dig 8 paces down.
7. Climb back up 6 paces. There be the treasure, argh!

In vector notation, this becomes:

1. $(000)'$
2. $+ (1000)'$
3. $+ (050)'$
4. $+ (-2000)'$
5. $+ (6 - 60)'$
6. $+ (00 - 8)'$
7. $+ (006)'$

which sums to $(-4 - 1 - 2)'$. So we can make our life easier by walking 4 paces south, 1 pace west, and digging only 2 paces down.

2.3 Length

The **length** of a vector x, usually written as $\|x\|$ or sometimes just $|x|$, is defined as $\sqrt{(\sum_i x_i)}$; the definition follows from the Pythagorean theorem: $\|x\|^2 = \sum x_i^2$. Because the coordinates are squared, all vectors have non-negative length, and only the zero vector has length 0.

Length interacts with scalar multiplication exactly as you would expect: $\|cx\| = c\|x\|$. The length of the sum of two vectors depends on how the are aligned with each other, but the **triangle inequality** $\|x + y\| \leq \|x\| + \|y\|$ always holds.

A special class of vectors are the *unit vectors*, those vectors x for which $\|x\| = 1$. In geometric terms, these correspond to all the points on the surface of a radius-1 sphere centered at the origin. Any vector x can be turned into a unit vector $x/\|x\|$ by dividing by its length. In two dimensions, the unit vectors are all of the form $(xy)' = (\cos\Theta, \sin\Theta)'$, where by convention Θ is the angle from due east measured counterclockwise; this is why traveling 9 units northwest corresponds to the vector $9(\cos 135°, \sin 135°)' = (-9/\sqrt{2}, 9/\sqrt{2})'$. In one dimension, the unit vectors are (± 1). (There are no unit vectors in zero dimensions: the unique zero-dimensional vector has length 0.)

2.4 Dot products and orthogonality

Suppose we have some column vector x, and we want to know how far x sends us in a particular direction, where the direction is represented by a unit column vector e. We can compute this distance (a scalar) by taking the **dot product**

$$e \cdot x = e'x = \sum e_i x_i.$$

For example, if $x = (34)'$ and $e = (10)'$, then the dot product is

$$e \cdot x = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} 3 \\ 4 \end{bmatrix} = 1 \cdot 3 + 0 \cdot 4 = 3.$$

In this case we see that the $(10)'$ vector conveniently extracts the first coordinate, which is about what we'd expect. But we can also find out how far x takes us in the $(1/\sqrt{2}1/\sqrt{2})'$ direction: this is $(1/\sqrt{2}1/\sqrt{2})x = 7/\sqrt{2}$.

By convention, we are allowed to take the dot product of two row vectors or of a row vector times a column vector or vice versa, provided of course that the non-boring dimensions match. In each case we transpose as appropriate to end up with a scalar when we take the matrix product.

Nothing in the definition of the dot product restricts either vector to be a unit vector. If we compute $x \cdot y$ where $x = ce$ and $\|e\| = 1$, then we are effectively multiplying $e \cdot y$ by c. It follows that the dot product is proportional to the length of both of its arguments. This often is expressed in terms of the geometric formulation, memorized by vector calculus students since time immemorial:

The dot product of x and y is equal to the product of their lengths times the cosine of the angle between them.

This formulation is a little misleading, since modern geometers will often *define* the angle between two vectors x and y as $\cos^{-1}(x \cdot y/(\|x\| \cdot \|y\|))$, but it gives a good picture of what is going on. One can also define the dot-product as the area of the parallelogram with sides x and y, with the complication that if the parallelogram is flipped upside-down we treat the area as negative. The simple version in terms of coordinates is harder to get confused about, so we'll generally stick with that.

Two vectors are **orthogonal** if their dot product is zero. In geometric terms, this occurs when either one or both vectors is the zero vector or when the angle between them is $\pm 90°$ (since $\cos(\pm 90°) = 0$). In other words, two non-zero vectors are orthogonal if and only if they are perpendicular to each other.

Orthogonal vectors satisfy the **Pythagorean theorem**: If $x \cdot y = 0$, then $\|x+y\|^2 = (x+y)\cdot(x+y) = x\cdot x + x\cdot y + y\cdot x + y\cdot y = x\cdot x + y\cdot y = \|x\|^2 + \|y\|^2$. It is not hard to see that the converse is also true: any pair of vectors for which $\|x+y\|^2 = \|x\|^2 + \|y\|^2$ must be orthogonal (at least in $\mathbb{R}^n$).

Orthogonality is also an important property of vectors used to define coordinate systems, as we will see below.

3 Linear combinations and subspaces

A **linear combination** of a set of vectors $x_1 \ldots x_n$ is any vector that can be expressed as $\sum c_i x_i$ for some coefficients c_i. The **span** of the vectors, written $\langle x_1 \ldots x_n \rangle$, is the set of all linear combinations of the x_i.[4]

The span of a set of vectors forms a **subspace** of the vector space, where a subspace is a set of vectors that is closed under linear combinations. This is a succinct way of saying that if x and y are in the subspace, so is $ax + by$ for any scalars a and b. We can prove this fact easily: if $x = \sum c_i x_i$ and $y = \sum d_i x_i$, then $ax + by = \sum (ac_i + bd_i)x_i$.

A set of vectors $x_1, x_2, \ldots, x_n$ is **linearly independent** if there is no way to write one of the vectors as a linear combination of the others, i.e., if there is no choice of coefficients that makes some $x_i = \sum_{j \neq i} c_j x_j$. An equivalent definition is that there is no choice of coefficients c_i such that $\sum c_i x_i = 0$ and at least one c_i is nonzero (to see the equivalence, subtract x_i from both sides of the $x_i = \sum c_j x_j$ equation).

3.1 Bases

If a set of vectors is both (a) linearly independent, and (b) spans the entire vector space, then we call that set of vectors a **basis** of the vector space. An example of a basis is the **standard basis** consisting of the vectors $(10\ldots00)', (01\ldots00)', \ldots, (00\ldots10)', (00\ldots01)'$.

[4] Technical note: If the set of vectors $\{x_i\}$ is infinite, then we will only permit linear combinations with a finite number of nonzero coefficients. We will generally not consider vector spaces big enough for this to be an issue.

This has the additional nice property of being made of of vectors that are all orthogonal to each other (making it an **orthogonal basis**) and of unit length (making it a **normal** basis).

A basis that is both orthogonal and normal is called **orthonormal**. We like orthonormal bases because we can recover the coefficients of some arbitrary vector v by taking dot-products. If $v = \sum a_i x_i$, then $v \cdot x_j = \sum a_i (x_i \cdot x_j) = a_i$, since orthogonality means that $x_i \cdot x_j = 0$ when $i \neq j$, and normality means $x_i \cdot x_i = \|x_i\|^2 = 1$.

However, even for non-orthonormal bases it is still the case that any vector can be written as a unique linear combination of basis elements. This fact is so useful we will state it as a theorem:

Theorem 1. *If $\{x_i\}$ is a basis for some vector space V, then every vector y has a unique representation $y = a_1 x_1 + a_2 x_2 + \ldots + a_n x_n$.*

Proof. Suppose there is some y with more than one representation, i.e., there are sequences of coefficients a_i and b_i such that $y = a_1 x_1 + a_2 x_2 + \ldots + a_n x_n = b_1 x_1 + b_2 x_2 + \ldots + b_n x_n$. Then $0 = y - y = a_1 x_1 + a_2 x_2 + \ldots + a_n x_n - b_1 x_1 + b_2 x_2 + \ldots + b_n x_n = (a_1 - b_1) x_1 + (a_2 - b_2) x_2 + \ldots + (a_n - b_n) x_n$. But since the x_i are independent, the only way a linear combination of the x_i can equal 0 is if all coefficients are 0, i.e., if $a_i = b_i$ for all i. □

Even better, we can do all of our usual vector space arithmetic in terms of the coefficients a_i. For example, if $a = \sum a_i x_i$ and $b = \sum b_i x_i$, then it can easily be verified that $a + b = \sum (a_i + b_i) x_i$ and $ca = \sum (c a_i) x_i$.

However, it may be the case that the same vector will have different representations in *different* bases. For example, in $\mathbb{R}^2$, we could have a basis $B_1 = \{(1,0), (0,1)\}$ and a basis $B_2 = \{(1,0), (1,-2)\}$. The vector $(2,3)$ would be represented as $(2,3)$ using basis B_1 but would be represented as $(5/2, -3/2)$ in basis B_2. In the standard basis $\{(1,0), (0,1)\}$, the representation of $(2,3)$ is just $(2,3)$.

Both bases above have the same size. This is not an accident; if a vector space has a finite basis, then all bases have the same size. We'll state this as a theorem, too:

Theorem 2. *Let $x_1 \ldots x_n$ and $y_1 \ldots y_m$ be two finite bases of the same vector space V. Then $n = m$.*

Proof. Assume without loss of generality that $n \leq m$. We will show how to replace elements of the x_i basis with elements of the y_i basis to produce a new basis consisting only of $y_1 \ldots y_n$. Start by considering the sequence $y_1, x_1 \ldots x_n$. This sequence is not independent since y_1 can be expressed as a linear combination of the x_i (they're a basis). So from Theorem 1 there is some x_i that can be expressed as a linear combination of $y_1, x_1 \ldots x_{i-1}$. Swap this x_i out to get a new sequence $y_1, x_1 \ldots x_{i-1}, x_{i+1}, \ldots x_n$. This new sequence is also a basis, because (a) any z can be expressed as a linear combination of these vectors by substituting the expansion of x_i into the expansion of z in the original basis, and (b) it's independent, because if there is some nonzero linear combination

that produces 0 we can substitute the expansion of x_i to get a nonzero linear combination of the original basis that produces 0 as well. Now continue by constructing the sequence $y_2, y_1, x_1 \dots x_{i-1}, x_{i+1}, \dots x_n$, and arguing that some $x_{i'}$ in this sequence must be expressible as a combination of earlier terms by Theorem 1 (it can't be y_1 because then y_2, y_1 is not independent), and drop this $x_{i'}$. By repeating this process we can eventually eliminate all the x_i, leaving the basis $y_n, \dots, y_1$. But then any y_k for $k > n$ would be a linear combination of this basis, so we must have $m = n$. □

The size of any basis of a vector space is called the **dimension** of the space.

4 Linear transformations

When we multiply a column vector by a matrix, we transform the vector into a new vector. This transformation is **linear** in the sense that $A(x+y) = Ax + Ay$ and $A(cx) = cAx$; thus we call it a **linear transformation**. Conversely, any linear function f from column vectors to column vectors can be written as a matrix M such that $f(x) = Mx$. We can prove this by decomposing each x using the standard basis.

Theorem 3. *Let $f : \mathbb{R}^n \to \mathbb{R}^m$ be a linear transformation. Then there is a unique $n \times m$ matrix M such that $f(x) = Mx$ for all column vectors x.*

Proof. We'll use the following trick for extracting entries of a matrix by multiplication. Let M be an $n \times m$ matrix, and let e^i be a column vector with $e^i_j = 1$ if $i = j$ and 0 otherwise.[5] Now observe that $(e^i)' M e^j = \sum_k e^i_k (Me^j)_k = (Me^j)_i = \sum_k M_{ik} e^j_k = M_{ij}$. So given a particular linear f, we will now *define* M by the rule $M_{ij} = (e^i)' f(e^j)$. It is not hard to see that this gives $f(e^j) = Me^j$ for each basis vector j, since multiplying by $(e^i)'$ grabs the i-th coordinate in each case. To show that $Mx = f(x)$ for all x, decompose each x as $\sum_k c_k e^k$. Now compute $f(x) = f(\sum_k c_k e^k) = \sum_k c_k f(e^k) = \sum_k c_k M(e^k) = M(\sum_k c_k e^k) = Mx$. □

4.1 Composition

What happens if we compose two linear transformations? We multiply the corresponding matrices:

$$(g \circ f)(x) = g(f(x)) = g(M_f x) = M_g(M_f x) = (M_g M_f)x.$$

This gives us another reason why the dimensions have to be compatible to take a matrix product: If multiplying by an $n \times m$ matrix A gives a map $g : \mathbb{R}^m \to \mathbb{R}^n$, and multiplying by a $k \times l$ matrix B gives a map $f : \mathbb{R}^l \to \mathbb{R}^k$, then the composition $g \circ f$ corresponding to AB only works if $m = k$.

[5] We are abusing notation by not being specific about how long e^i is; we will use the same expression to refer to any column vector with a 1 in the i-th row and zeros everywhere else. We are also moving what would normally be a subscript up into the superscript position to leave room for the row index—this is a pretty common trick with vectors and should not be confused with exponentiation.

4.2 Role of rows and columns of M in the product Mx

When we multiply a matrix and a column vector, we can think of the matrix as a sequence of row or column vectors and look at how the column vector operates on these sequences.

Let $M_{i\cdot}$ be the i-th row of the matrix (the "$\cdot$" is a stand-in for the missing column index). Then we have

$$(Mx)_i = \sum_k M_{ik}x_k = M_{i\cdot} \cdot x.$$

So we can think of Mx as a vector of dot-products between the rows of M and x:

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 2 \end{bmatrix} = \begin{bmatrix} (1,2,3)\cdot(1,1,2) \\ (4,5,6)\cdot(1,1,2) \end{bmatrix} = \begin{bmatrix} 9 \\ 21 \end{bmatrix}.$$

Alternatively, we can work with the columns $M_{\cdot j}$ of M. Now we have

$$(Mx)_i = \sum_k M_{ik}x_k = \sum_k (M_{\cdot k})_i x_k.$$

From this we can conclude that Mx is a linear combination of columns of M: $Mx = \sum_k x_k M_{\cdot k}$. Example:

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 2 \end{bmatrix} = 1\begin{bmatrix} 1 \\ 4 \end{bmatrix} + 1\begin{bmatrix} 2 \\ 5 \end{bmatrix} + 2\begin{bmatrix} 3 \\ 6 \end{bmatrix} = \begin{bmatrix} 1 \\ 4 \end{bmatrix} + \begin{bmatrix} 2 \\ 5 \end{bmatrix} + \begin{bmatrix} 7 \\ 12 \end{bmatrix} = \begin{bmatrix} 9 \\ 21 \end{bmatrix}.$$

The set $\{Mx\}$ for all x is thus equal to the span of the columns of M; it is called the **column space** of M.

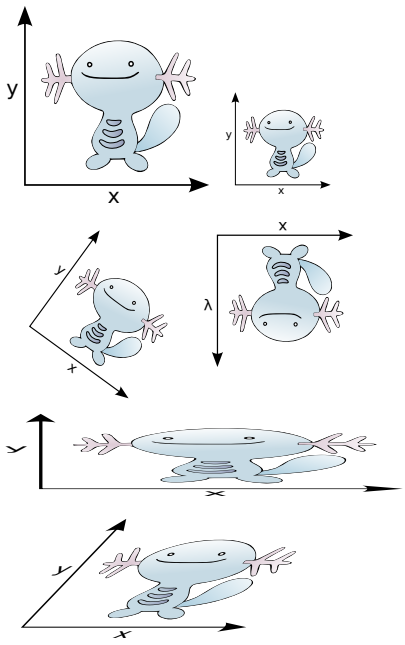
For yM, where y is a row vector, similar properties hold: we can think of yM either as a row vector of dot-products of y with columns of M or as a weighted sum of rows of M; the proof follows immediately from the above facts about a product of a matrix and a column vector and the fact that $yM = (M'y')'$. The span of the rows of M is called the **row space** of M, and equals the set $\{yM\}$ of all results of multiplying a row vector by M.

4.3 Geometric interpretation

Geometrically, linear transformations can be thought of as changing the basis vectors for a space: they keep the origin in the same place, move the basis vectors, and rearrange all the other vectors so that they have the same coordinates in terms of the new basis vectors. These new basis vectors are easily read off of the matrix representing the linear transformation, since they are just the columns of the matrix. So in this sense all linear transformations are transformations from some vector space to the column space of some matrix.[6]

[6]The situation is slightly more complicated for infinite-dimensional vector spaces, but we will try to avoid them.

This property makes linear transformations popular in graphics, where they can be used to represent a wide variety of transformations of images. Below is a picture of an untransformed image (top left) together with two standard basis vectors labeled x and y. In each of the other images, we have shifted the basis vectors using a linear transformation, and carried the image along with it.[7]



Note that in all of these transformations, the origin stays in the same place. If you want to move an image, you need to add a vector to everything. This gives an **affine transformation**, which is any transformation that can be written as $f(x) = Ax + b$ for some matrix A and column vector b.

Many two-dimensional linear transformations have standard names. The simplest transformation is **scaling**, where each axis is scaled by a constant, but the overall orientation of the image is preserved. In the picture above, the top right image is scaled by the same constant in both directions and the second-from-the-bottom image is scaled differently in each direction.

Recall that the product Mx corresponds to taking a weighted sum of the

[7]The thing in the picture is a Pokémon known as a *Wooper*, which evolves into a *Quagsire* at level 20. This evolution is not a linear transformation.

columns of M, with the weights supplied by the coordinates of x. So in terms of our basis vectors x and y, we can think of a linear transformation as specified by a matrix whose columns tell us what vectors for replace x and y with. In particular, a scaling transformation is represented by a matrix of the form

$$\begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix},$$

where s_x is the scale factor for the x (first) coordinate and s_y is the scale factor for the y (second) coordinate. Flips (as in the second image from the top on the right) are a special case of scaling where one or both of the scale factors is -1.

A more complicated transformation, as shown in the bottom image, is a **shear**. Here the image is shifted by some constant amount in one coordinate as the other coordinate increases. Its matrix looks like this:

$$\begin{bmatrix} 1 & c \\ 0 & 1 \end{bmatrix}.$$

Here the x vector is preserved: $(1, 0)$ maps to the first column $(1, 0)$, but the y vector is given a new component in the x direction of c, corresponding to the shear. If we also flipped or scaled the image at the same time that we sheared it, we could represent this by putting values other than 1 on the diagonal.

For a rotation, we will need some trigonometric functions to compute the new coordinates of the axes as a function of the angle we rotate the image by. The convention is that we rotate counterclockwise: so in the figure above, the rotated image is rotated counterclockwise approximately 315° or −45°. If Θ is the angle of rotation, the rotation matrix is given by

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

For example, when $\Theta = 0°$, then we have $\cos\Theta = 1$ and $\sin\Theta = 0$, giving the identity matrix. When $\Theta = 90°$, then $\cos\Theta = 0$ and $\sin\Theta = 1$, so we rotate the x axis to the vector $(\cos\Theta, \sin\Theta) = (0, 1)$ and the y axis to $(-\sin\Theta, \cos\Theta) = (-1, 0)$. This puts the x axis pointing north where the y axis used to be, and puts the y axis pointing due west.

4.4 Rank and inverses

The dimension of the column space of a matrix—or, equivalently, the dimension of the range of the corresponding linear transformation—is called the **rank**. The rank of a linear transformation determines, among other things, whether it has an inverse.

Theorem 4. *If $f : \mathbb{R}^n \to \mathbb{R}^m$ is a linear transformation with an inverse f^{-1}, then we can show all of the following:*

1. *f^{-1} is also a linear transformation.*

2. *$n = m$, and f has **full rank**, i.e.,* $\text{rank}(f) = \text{rank}(f^{-1}) = m$.

Proof. 1. Let x and y be elements of codomain(f) and let a be a scalar. Then $f(af^{-1}(x)) = a(f(f^{-1}(x))) = ax$, implying that $f^{-1}(ax) = af^{-1}(x)$. Similarly, $f(f^{-1}(x) + f^{-1}(y)) = f(f^{-1}(x)) + f(f^{-1}(y)) = x + y$, giving $f^{-1}(x + y) = f^{-1}(x) + f^{-1}(y)$. So f^{-1} is linear.

2. Suppose $n < m$. Pick any basis e^i for $\mathbb{R}^n$, and observe that $\{f(e^i)\}$ spans range(f) (since we can always decompose x as $\sum a_i e^i$ to get $f(x) = \sum a_i f(e^i)$). So the dimension of range(f) is at most n. If $n < m$, then range(f) is a proper subset of $\mathbb{R}^m$ (otherwise it would be m-dimensional). This implies f is not surjective and thus has no inverse. Alternatively, if $m < n$, use the same argument to show that any claimed f^{-1} isn't. By the same argument, if either f or f^{-1} does not have full rank, it's not surjective.

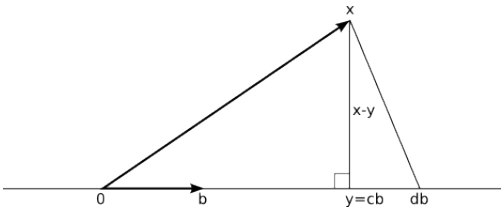
□

The converse is also true: If $f : \mathbb{R}^n \to \mathbb{R}^n$ has full rank, it has an inverse. The proof of this is to observe that if dim(range(f)) $= n$, then range$(f) = \mathbb{R}^n$ (since $\mathbb{R}^n$ has no full-dimensional subspaces). So in particular we can take any basis $\{e^i\}$ for $\mathbb{R}^n$ and find corresponding $\{x^i\}$ such that $f(x^i) = e^i$. Now the linear transformation that maps $\sum a_i e^i$ to $\sum a_i x^i$ is an inverse for f, since $f(\sum a_i x^i) = \sum a_i f(x_i) = \sum a_i e^i$.

4.5 Projections

Suppose we are given a low-dimensional subspace of some high-dimensional space (e.g. a line (dimension 1) passing through a plane (dimension 2)), and we want to find the closest point in the subspace to a given point in the full space. The process of doing this is called **projection**, and essentially consists of finding some point z such that $(x - z)$ is orthogonal to any vector in the subspace.

Let's look at the case of projecting onto a line first, then consider the more general case.



A line consists of all points that are scalar multiples of some fixed vector b. Given any other vector x, we want to extract all of the parts of x that lie in the direction of b and throw everything else away. In particular, we want to find a vector $y = cb$ for some scalar c, such that $(x - y) \cdot b = 0$. This is is enough information to solve for c.

We have $(x - cb) \cdot b = 0$, so $x \cdot b = c(b \cdot b)$ or $c = (x \cdot b)/(b \cdot b)$. So the projection of x onto the subspace $\{cb | c \in \mathbb{R}\}$ is given by $y = b(x \cdot b)/(b \cdot b)$ or

$y = b(x \cdot b)/\|b\|^2$. If b is normal (i.e. if $\|b\| = 1$), then we can leave out the denominator; this is one reason we like orthonormal bases so much.

Why is this the right choice to minimize distance? Suppose we pick some other vector db instead. Then the points x, cb, and db form a right triangle with the right angle at cb, and the distance from x to db is $\|x - db\| = \sqrt{(\|x - cb\|^2 + \|cb - db\|^2)} \geq \|x - cb\|$.

But now what happens if we want to project onto a larger subspace? For example, suppose we have a point x in three dimensions and we want to project it onto some plane of the form $\{c_1 b_1 + c_2 b_2\}$, where b_1 and b_2 span the plane. Here the natural thing to try is to send x to $y = b_1(x \cdot b_1)/\|b_1\|^2 + b_2(x \cdot b_2)/\|b_2\|^2$. We then want to argue that the vector $(x - y)$ is orthogonal to any vector of the form $c_1 b_1 + c_2 b_2$. As before, $(x - y)$ is orthogonal to any vector in the plane, it's orthogonal to the difference between the y we picked and some other z we didn't pick, so the right-triangle argument again shows it gives the shortest distance.

Does this work? Let's calculate: $(x-y)\cdot(c_1 b_1 + c_2 b_2) = x\cdot(c_1 b_1 + c_2 b_2) - (b_1(x \cdot b_1)/\|b_1\|^2 + b_2(x \cdot b_2)/\|b_2\|^2) \cdot (c_1 b_1 + c_2 b_2) = c_1(x \cdot b_1 - (b_1 \cdot b_1)(x \cdot b_1)/(b_1 \cdot b_1)) + c_2(x \cdot b_2 - (b_2 \cdot b_2)(x \cdot b_2)/(b_2 \cdot b_2)) - c_1(b_1 \cdot b_2)(x \cdot b_1)/(b_1 \cdot b_1) - c_2(b_1 \cdot b_2)(x \cdot b_2)/(b_2 \cdot b_2)$.

The first two terms cancel out very nicely, just as in the one-dimensional case, but then we are left with a nasty $(b_1 \cdot b_2)$(much horrible junk) term at the end. *It didn't work!*

So what do we do? We could repeat our method for the one-dimensional case and solve for c_1 and c_2 directly. This is probably a pain in the neck. Or we can observe that the horrible extra term includes a $(b_1 \cdot b_2)$ factor, and if b_1 and b_2 are orthogonal, it disappears. The moral: We can project onto a 2-dimensional subspace by projecting independently onto the 1-dimensional subspace spanned by each basis vector, *provided the basis vectors are orthogonal*. And now we have another reason to like orthonormal bases.

This generalizes to subspaces of arbitrary high dimension: as long as the b_i are all orthogonal to each other, the projection of x onto the subspace $\langle b_i \rangle$ is given by $\sum b_i(x \cdot b_i)/\|b_i\|^2$. Note that we can always express this as matrix multiplication by making each row of a matrix B equal to one of the vectors $b_i/\|b_i\|^2$; the product Bx then gives the coefficients for the basis elements in the projection of x, since we have already seen that multiplying a matrix by a column vector corresponds to taking a dot product with each row. If we want to recover the projected vector $\sum c_i b_i$ we can do so by taking advantage of the fact that multiplying a matrix by a column vector also corresponds to taking a linear combination of columns: this gives a combined operation of $B'Bx$ which we can express as a single **projection matrix** $P = B'B$. So projection corresponds to yet another special case of a linear transformation.

One last detail: suppose we aren't given orthonormal b_i but are instead given some arbitrary non-orthogonal non-normal basis for the subspace. Then what do we do?

The trick here is to use a technique called Gram-Schmidt orthogonalization. This constructs an orthogonal basis from an arbitrary basis by induction. At each step, we have a collection of orthogonalized vectors $b_1 \ldots b_k$ and some that

we haven't processed yet $a_{k+1} \ldots a_m$; the induction hypothesis says that the $b_1 \ldots b_k$ vectors are (a) orthogonal and (b) span the same subspace as $a_1 \ldots a_k$. The base case is the empty set of basis vectors, which is trivially orthogonal and also trivially spans the subspace consisting only of the 0 vector. We add one new vector to the orthogonalized set by projecting a_{k+1} to some point c on the subspace spanned by $b_1 \ldots, b_k$; we then let $b_{k+1} = a_{k+1} - c$. This new vector is orthogonal to all of $b_1 \ldots b_k$ by the definition of orthogonal projection, giving a new, larger orthogonal set $b_1 \ldots b_{k+1}$. These vectors span the same subspace as $a_1 \ldots a_{k+1}$ because we can take any vector x expressed as $\sum_{i=1}^{k+1} c_i a_i$, and rewrite it as $\sum_{i=1}^{k} c_i b_i + c_{k+1}(c + b_{k+1})$, and in the second term $c_{k+1}c$ reduces to a linear combination of $b_1 \ldots b_k$; the converse essentially repeats this argument in reverse. It follows that when the process completes we have an orthogonal set of vectors $b_1 \ldots b_m$ that span precisely the same subspace as $a_1 \ldots a_m$, and we have our orthogonal basis. (But not orthonormal: if we want it to be orthonormal, we divide each b_i by $\|b_i\|$ as well.)

5 Further reading

Linear algebra is vitally important in Computer Science: it is a key tool in graphics, scientific computing, robotics, neural networks, and many other areas. If you do further work in these areas, you will quickly find that we have not covered anywhere near enough linear algebra in this course. Your best strategy for remedying this deficiency may be to take an actual linear algebra course; failing that, a very approachable introductory text is *Linear Algebra and Its Applications*, by Gilbert Strang. You can also watch an entire course of linear algebra lectures through YouTube: `http://www.youtube.com/view_play_list?p=E7DDD91010BC51F8`.

Some other useful books on linear algebra:

- Golub and van Loan, *Matrix Computations*. Picks up where Strang leaves off with practical issues in doing computation.
- Halmos, *Finite-Dimensional Vector Spaces*. Good introduction to abstract linear algebra, i.e. properties of vector spaces without jumping directly to matrices.

Matlab (which is available on the Zoo machines: type 'matlab' at a shell prompt) is useful for playing around with operations on matrices. There are also various non-commercial knockoffs like Scilab or Octave that are not as comprehensive as Matlab but are adequate for most purposes. Note that with any of these tools, if you find yourselves doing lots of numerical computation, it is a good idea to talk to a numerical analyst about round-off error: the floating-point numbers inside computers are not the same as real numbers, and if you aren't careful about how you use them you can get very strange answers.