LINEAR TRANSFORMATIONS

MATH 196, SECTION 57 (VIPUL NAIK)

Corresponding material in the book: Section 2.1.

Executive summary

(1) A $n \times m$ matrix defines a function from $\mathbb{R}^m$ to $\mathbb{R}^n$ via matrix-vector multiplication. A function arising in this manner is termed a *linear transformation*. Every linear transformation arises from a unique matrix, i.e., there is a bijection between the set of $n \times m$ matrices and the set of linear transformations from $\mathbb{R}^m$ to $\mathbb{R}^n$.

(2) A function (also called map) $f : A \to B$ of sets is termed *injective* if no two elements of A map to the same element of B. It is termed *surjective* if B equals the range of f. It is termed *bijective* if it is both injective and surjective. A bijective map has a unique inverse map.

(3) The standard basis vector $\vec{e_i}$ is the vector with a 1 in the i^{th} coordinate and 0s elsewhere. The image of $\vec{e_i}$ is precisely the i^{th} column of the matrix describing the linear tranformation.

(4) A linear transformation can alternatively be defined as a map that preserves addition and scalar multiplication. Explicitly, $T : \mathbb{R}^m \to \mathbb{R}^n$ is a linear transformation if and only if $T(\vec{x} + \vec{y}) = T(\vec{x}) + T(\vec{y})$ (additivity) and $T(a\vec{x}) = aT(\vec{x})$ (scalar multiples) for all $\vec{x}, \vec{y} \in \mathbb{R}^m$ and all $a \in \mathbb{R}$.

(5) A linear transformation $T : \mathbb{R}^m \to \mathbb{R}^n$ is *injective* if the matrix of T has full column rank, which in this case means rank m, because the dimensions of the matrix are $n \times m$. Note that this in particular implies that $m \leq n$. The condition $m \leq n$ is a *necessary but not sufficient condition* for the linear transformation to be injective.

(6) A linear transformation $T : \mathbb{R}^m \to \mathbb{R}^n$ is *surjective* if the matrix of T has full row rank, which in this case means rank n, because the dimensions of the matrix are $n \times m$. Note that this in particular implies that $n \leq m$. The condition $n \leq m$ is a *necessary but not sufficient condition* for the linear transformation to be surjective.

(7) A linear transformation $T : \mathbb{R}^m \to \mathbb{R}^n$ is *bijective* if the matrix of T has full row rank and full column rank. Thus forces $m = n$, and forces the (now square) matrix to have full rank. As mentioned before, this is equivalent to invertibility.

(8) The inverse of a diagonal matrix with all diagonal entries nonzero is the matrix obtained by inverting each diagonal entry individually.

(9) A permutation matrix is a matrix where each row has one 1 and all other entries 0, and each column has one 1 and all other entries 0. A permutation matrix acts by permuting the standard basis vectors among themselves. It can be inverted by permuting the standard basis vectors among themselves in the reverse direction. Hence, the inverse is also a permutation matrix. If the permutation matrix just flips two basis vectors, it is self-inverse.

(10) The inverse of a shear operation is the shear operation obtained by using the negative of the shear. In particular:

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

(11) Inverting a matrix requires solving a system of simultaneous linear equations with that as coefficient matrix and with the augmenting column a *generic* output vector, then using the expression for the input in terms of the output to get the matrix of the transformation. It can alternatively be done by augmenting the matrix with the identity matrix, row-reducing the matrix to the identity matrix, and then looking at what the augmented side has become.

(12) We can use linear transformations for the purposes of coding, compression, and extracting relevant information. In many of these practical applications, we are working, not over the real numbers, but over the field of two elements, which is ideally suited for dealing with the world of bits (binary digits).

1. Linear transformations from one space to another

1.1. Matrix-vector multiplication defines a map between vector spaces.

Recall that if A is a $n \times m$ matrix, and $\vec{x}$ is a $m \times 1$ column vector (i.e., $\vec{x}$ is a m-dimensional vector written as a column vector) then $A\vec{x}$ is a $n \times 1$ vector, i.e., $A\vec{x}$ is a n-dimensional vector written as a column vector. We can thus think of the matrix A as describing a function:

m-dimensional column vectors → n-dimensional column vectors

Further, recall that a vector can be thought of as describing the coordinates of a generic point in the space of the same number of dimensions. In particular, m-dimensional column vectors can be considered points in $\mathbb{R}^m$ and n-dimensional column vectors can be considered points in $\mathbb{R}^n$. Thus, A can be viewed as defining, via matrix-vector multiplication, a map:

$$\mathbb{R}^m \to \mathbb{R}^n$$

In other words, every $n \times m$ matrix defines a map from $\mathbb{R}^m$ to $\mathbb{R}^n$ via matrix-vector multiplication.

Please note very carefully that the dimension of the input space is the *number of columns* in the matrix and coincides with the *dimension of the row vectors in the matrix.* The dimension of the output space is the *number of rows* in the matrix and coincides with the *dimension of the column vectors in the matrix.*

Note also that the i^{th} coordinate of the output is obtained as the dot product of the i^{th} row of A with the input vector.

1.2. First definition of linear transformation.

We give the first definition:

Definition (Linear transformation)**.** A function $T : \mathbb{R}^m \to \mathbb{R}^n$ is termed a *linear transformation* if there exists a $n \times m$ matrix A (dependent on T) such that $T(\vec{x}) = A\vec{x}$ for all $\vec{x}$ in $\mathbb{R}^m$ (writing $\vec{x}$ as a column vector).

In other words, a transformation is linear if it can be represented via multiplication by a matrix.

We can think of a map at the *meta* level:

(Set of $n \times m$ matrices over $\mathbb{R}$) → (Linear transformations $\mathbb{R}^m \to \mathbb{R}^n$)

The map takes a matrix as input and outputs the linear transformation induced via multiplication by the matrix.

Note that the set of outputs of the map is itself a set of functions. That's why I said above that this is a map at the *meta* level.

1.3. Injectivity, surjectivity, and bijectivity.

The following terminology, which you may or may not have seen, will be useful in coming discussions.

Suppose $f : A \to B$ is a function (also called a *map* or a *mapping*). We say that f is:

- *injective* or *one-one* or *one-to-one* if different inputs always give different outputs, i.e., if whenever $x, y \in A$ satisfy $f(x) = f(y)$, then $x = y$. Another framing of this is that every element of B is the image of *at most* one element of A.
- *surjective* if the range of f is B, i.e., every element of B arises as the image of *at least* one element of A.
- *bijective* if it is both injective and surjective. Another framing is that every element of B arises as the image of *exactly* one element of A. If f is bijective, it establishes a "one-one correspondence" between the sets A and B.

1.4. **Bijectivity between matrices and linear transformations.** We noted a while back the existence of the map:

(Set of $n \times m$ matrices over $\mathbb{R}$) $\rightarrow$ (Linear transformations $\mathbb{R}^m \rightarrow \mathbb{R}^n$)

The map is *surjective* by the definition of linear transformation. But is it injective? That's a trickier question. At its heart is the question of whether two matrices that are not identical can give rise to the same linear transformation. If that cannot happen, the map is injective.

The map is indeed injective, but to better understand this, we need to think a little bit more about the outputs of specific vectors.

The space $\mathbb{R}^m$ has m very special vectors called the *standard basis vectors*:[1] these are vectors with a 1 in one coordinate and 0s in all the other coordinates. The basis vector e_i is the vector whose i^{th} coordinate is a 1 and remaining coordinates are all 0s. In other words, the standard basis vectors $\vec{e}_1, \vec{e}_2, \ldots, \vec{e}_m$ are the vectors:

$$\vec{e}_1 = \begin{bmatrix} 1 \\ 0 \\ \cdot \\ \cdot \\ \cdot \\ 0 \end{bmatrix}, \vec{e}_2 \begin{bmatrix} 0 \\ 1 \\ 0 \\ \cdot \\ \cdot \\ 0 \end{bmatrix}, \ldots, \vec{e}_m = \begin{bmatrix} 0 \\ 0 \\ \cdot \\ \cdot \\ \cdot \\ 1 \end{bmatrix}$$

The following will turn out to be true:

- If a matrix A describes a linear transformation $T : \mathbb{R}^m \rightarrow \mathbb{R}^n$, then $T(\vec{e}_i)$ is the i^{th} column of A. Note that $T(\vec{e}_i)$ is a n-dimensional column vector.
- In particular, knowledge of the outputs $T(\vec{e}_1), T(\vec{e}_2), \ldots T(\vec{e_m})$ is sufficient to reconstruct the whole matrix A.
- Thus, knowledge of T as a function (which would entail knowledge of the image of every input under T) is sufficient to reconstruct A.
- Thus, the mapping from the set of $n \times m$ matrices over $\mathbb{R}$ to linear transformations $\mathbb{R}^m \rightarrow \mathbb{R}^n$ is injective. Since we already noted that the mapping is surjective, it is in fact bijective. In other words, every matrix gives a linear transformation, and every linear transformation arises from a unique matrix.

1.5. **An alternative definition of linear transformation.** We can also define linear transformation in another, albeit related, way. This definition relies on some universal equalities that the map must satisfy. First, recall that we can add vectors (addition is coordinate-wise) and multiply a real number and a vector (the real number gets multiplied with each coordinate of the vector).

Definition (Linear transformation (alternative definition)). A function $T : \mathbb{R}^m \rightarrow \mathbb{R}^n$ is termed a *linear transformation* if it satisfies the following two conditions:

(1) *Additivity*: $T(\vec{x} + \vec{y}) = T(\vec{x}) + T(\vec{y})$ for all vectors $\vec{x}, \vec{y} \in \mathbb{R}^m$. Note that the addition on the inside in the left side is in $\mathbb{R}^m$ and the addition on the right side is in $\mathbb{R}^n$.
(2) *Scalar multiples*: $T(a\vec{x}) = aT(\vec{x})$ for all real numbers a and vectors $\vec{x} \in \mathbb{R}^m$.

To show that this definition is equivalent to the earlier definition, we need to show two things:

- Every linear transformation per the original definition (i.e., it can be represented using a matrix) is a linear transformation per the new definition: This follows from the fact that matrix-vector multiplication is linear in the vector. We noted this earlier when we defined matrix-vector multiplication.
- Every linear transformation per the new definition is a linear transformation per the original definition: The idea here is to start with the linear transformation T per the new definition, then construct the putative matrix for it using the values $T(\vec{e}_1), T(\vec{e}_2), \ldots, T(\vec{e_m})$ as columns. Having constructed

[1] This is a lie in some ways, but that does not matter right now

the putative matrix, we then need to check that this matrix gives the linear transformation (per the new definition) that we started with.

2. Behavior of linear transformations

2.1. **Rank and its role.** The rank of a linear transformation plays an important role in determining whether it is injective, whether it is surjective, and whether it is bijective. Note that our earlier discussion of injective, surjective and bijective was in the context of a "meta" map from a set of matrices to a set of linear transformations. Now, we are discussing the injectivity, surjectivity, and bijectivity of a specific linear transformation.

Note that finding the pre-images for a given linear transformation is tantamount to solving a linear system where the augmenting column is the desired output. And, we have already studied the theory of what to expect with the solutions to systems of linear equations. In particular, we have that:

- A linear transformation is injective if and only if its matrix has full column rank. In other words, $T : \mathbb{R}^m \to \mathbb{R}^n$ is injective if and only if its matrix, which is a $n \times m$ matrix, has rank m. Note that this is possible only if $m \leq n$.

  This is because, as we saw earlier, the dimension of the solution space for any specific output is the difference (number of variables) - (rank). In order to get injectivity, we wish for the solution space to be zero-dimensional, i.e., we wish that whenever the system has a solution, the dimension of the solution space is zero. This means that the rank must equal the number of variables.

  The condition $m \leq n$ is *necessary but not sufficient* for the linear transformation to be injective. Intuitively, this makes sense. For a map to be injective, the target space of the map must be *at least as big* as the domain. Since the appropriate size measure of vector spaces in the context of linear transformations is the dimension, we should expect $m \leq n$. Note that $m \leq n$ is not sufficient: any linear transformation whose matrix does not have full column rank is not injective, even if $m \leq n$. An extreme example is the zero linear transformation, whose matrix is the zero matrix. This is not injective for $m > 0$.
- A linear transformation is surjective if and only if its matrix has full row rank. In other words, $T : \mathbb{R}^m \to \mathbb{R}^n$ is surjective if and only its matrix, which is a $n \times m$ matrix, has rank n. Note that this is possible only if $n \leq m$.

  This is because, as we saw earlier, if the coefficient matrix has full row rank, the system is always consistent. Hence, every output vector occurs as the image of some input vector, so the map is surjective.

  The condition $n \leq m$ is *necessary but not sufficient* for the linear trasnformation to be surjective. Intuitively, this makes sense. For a map to be surjective, the target space must be *at most as large* as the domain. The appropriate size measure of vector spaces in the context of linear transformations is dimension, so $n \leq m$ follows. Note that $n \leq m$ is not sufficient: any linear transformation whose matrix does not have full row rank is not surjective, even if $n \leq m$. An extreme example is the zero linear transformation, given by the zero matrix. This is not surjective if $n > 0$.
- A linear transformation can be bijective only if its domain and co-domain space have the same dimension, so that its matrix is a square matrix, *and* that square matrix has full rank.

Our interest for now will be in bijective linear transformations of the form $T : \mathbb{R}^n \to \mathbb{R}^n$, for which, as noted above, the rank is n, and thus, the rref is the identity matrix.

For a bijective linear transformation, we can define an *inverse* transformation that sends each element of the target space to the unique domain element mapping to it. This inverse transformation will also turn out to be linear.

Note that the inverse is unique, and the inverse of the inverse is the original linear transformation.

We will later on see the matrix algebra definition and meaning of the inverse.

3. Some important linear transformations and their inverses

3.1. **Linear transformations with diagonal matrices.** Consider a linear transformation $T : \mathbb{R}^n \to \mathbb{R}^n$ whose matrix is a *diagonal* matrix. For instance, consider this linear transformation T given by the matrix:

$$A = \begin{bmatrix} 3 & 0 & 0 \\ 0 & 7 & 0 \\ 0 & 0 & 4 \end{bmatrix}$$

This means that:

$$T(\vec{e_1}) = 3\vec{e_1}, T(\vec{e_2}) = 7\vec{e_2}, T(\vec{e_3}) = 4\vec{e_3}$$

Diagonal linear transformations are special in that they send each standard basis vector to a multiple of it, i.e., they do not "mix up" the standard basis vectors with each other. To invert a diagonal linear transformation, it suffices to invert the operation for each standard basis vector, while keeping it diagonal. In other words, we invert each diagonal entry, keeping it in place.

$$A^{-1} = \begin{bmatrix} 1/3 & 0 & 0 \\ 0 & 1/7 & 0 \\ 0 & 0 & 1/4 \end{bmatrix}$$

Note that for a diagonal matrix where one or more of the diagonal entries is zero, the matrix does not have full rank. In fact, the corresponding standard basis vector gets killed by the linear transformation. Thus, the inverse does not exist.

3.2. **Linear transformations that permute the coordinates.** Consider the linear transformation T with matrix:

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

We can describe T as follows:

$$T(\vec{e_1}) = \vec{e_2}, T(\vec{e_2}) = \vec{e_1}$$

In other words, T interchanges the two standard basis vectors $\vec{e_1}$ and $\vec{e_2}$. To invert T, we need to do T again: interchage them back! Thus, the inverse matrix is:

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

Let's take a 3 × 3 matrix that cycles the vectors around:

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

We can read off the images of $\vec{e_1}$, $\vec{e_2}$, and $\vec{e_3}$ by looking at the respective columns. The first column describes $T(\vec{e_1})$, and it is $\vec{e_3}$. The second column describes $T(\vec{e_2})$, and it is $\vec{e_1}$. The third column describes $T(\vec{e_3})$, and it is $\vec{e_2}$. In other words:

$$T(\vec{e_1}) = \vec{e_3}, T(\vec{e_2}) = \vec{e_1}, T(\vec{e_3}) = \vec{e_2}$$

Thus, this linear transformation cycles the vectors around as follows:

$$\vec{e_1} \to \vec{e_3} \to \vec{e_2} \to \vec{e_1}$$

To invert this, we need to cycle the vectors in the reverse order:

$$\vec{e_1} \to \vec{e_2} \to \vec{e_3} \to \vec{e_1}$$

For $\vec{e_1} \to \vec{e_2}$, we need the first column to be $\vec{e_2}$. For $\vec{e_2} \to \vec{e_3}$, we need the second column to be $\vec{e_3}$. For $\vec{e_3} \to \vec{e_1}$, we need the third column to be $\vec{e_1}$.

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

A square matrix is termed a *permutation matrix* if it has one 1 and the remaining entries 0 in each row, and similarly it has one 1 and the remaining entries 0 in each column. There is a rich theory of permutation matrices that relies only on the theory of what are called permutation groups, which deal with finite sets. Enticing though the theory is, we will not explore it now.

3.3. **Shear operations and their inverses.** Consider the shear operation T with matrix:

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

This matrix does not keep each coordinate within itself, nor does it merely permute the coordinates. Rather, it adds one coordinate to another. Explicitly:

$$T(\vec{e}_1) = \vec{e}_1, T(\vec{e}_2) = \vec{e}_1 + \vec{e}_2$$

The operation T^{-1} should behave as follows:

$$T^{-1}(\vec{e}_1) = \vec{e}_1, T^{-1}(\vec{e}_1 + \vec{e}_2) = \vec{e}_2$$

Thus:

$$T^{-1}(\vec{e}_2) = T^{-1}(\vec{e}_1 + \vec{e}_2) - T^{-1}(\vec{e}_1)$$

We get that:

$$T^{-1}(\vec{e}_2) = \vec{e}_2 - \vec{e}_1$$

Thus, we get that:

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

Note that the choice of the word "shear" is not a coincidence. We had talked earlier of operations on systems of equations. One of these operation types, called a *shear operation*, involved adding a multiple of one row to another row. You can see that this is similar in broad outline to what we are trying to do here. Is there a formalization of the relationship? You bet! It has to do with matrix multiplication, but we'll have to skip it for now.

4. Computing inverses in general

4.1. **Procedure for computing the inverse: a sneak preview.** *Note*: We will return to this topic in more detail later, after describing matrix multiplication. For the moment, think of this as a very quick sneak preview. It's a tricky collection of ideas, so multiple rounds of exposure help. At this stage, you are not expected to acquire computational fluency.

Consider a $n \times n$ matrix A. Suppose that A has full rank. Hence, it is invertible. For any vector $\vec{x}$, $A^{-1}\vec{x}$ is the unique vector $\vec{y}$ such that $A\vec{y} = \vec{x}$. To obtain this vector, we need to solve the linear system with A as the coefficient matrix and $\vec{y}$ as the augmenting column. The fact that A has full row rank guarantees consistency. The fact that A has full column rank guarantees that the solution is unique.

This is great for finding $A^{-1}\vec{x}$ for individual vectors x. But it would be ideal if we can obtain A^{-1} *qua* matrix, rather than solving the linear system separately for each $\vec{x}$.

There are two equivalent ways of thinking about this.

One is to solve the linear system for an arbitrary vector $\vec{x}$, i.e., without putting in actual numerical values, and get $A^{-1}\vec{x}$ functionally in terms of $\vec{x}$, then pattern match that to get a matrix.

A more sophisticated approach is to calculate $A^{-1}\vec{e}_1, A^{-1}\vec{e}_2, \dots A^{-1}\vec{e}_n$ separately, then combine them into a matrix. Note that all of them use the coefficient matrix A, so instead of writing each augmented matrix separately, we could just write A and augment it with multiple columns, one for each standard basis vector. That is equivalent to augmenting A with the identity matrix. Row reduce A, and keep applying the operations to the identity matrix. When A gets row reduced to the identity matrix, the identity matrix is converted to A^{-1}. We will return to this procedure later, after we have an understanding of matrix multiplication.

Yet another way of thinking of this is that each of the elementary row operations on A *undoes* A a bit, and all of them put together undo A to the identity matrix. Doing all these operations one after the other gives us the entire recipe for undoing A, and hence, applying that recipe to the identity matrix gives A^{-1}.

4.2. **Formula for the inverse of a 2×2 matrix.** Suppose we have a 2×2 matrix:

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

It can be verified that:

- The matrix has full rank (or equivalently, is invertible) if and only if $ad - bc \neq 0$.
- If the matrix has full rank, then the inverse is:

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}^{-1} = \frac{1}{ad - bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

The expression $ad - bc$ controls whether or not the matrix has full rank. This expression is termed the *determinant* of the matrix. We hope to discuss determinants at a later stage in the course.

4.3. **Time complexity of computing the inverse.** The row reduction approach described above for computing the inverse has roughly the same complexity as ordinary Gauss-Jordan elimination. In particular, the arithmetic time complexity of the process is $\Theta(n^3)$. There are better approaches to matrix inversion in many situations. However, as a general rule, the time complexity cannot be brought down to a lower order using known and easy-to-implement algorithms.

5. Real-world applications

5.1. **Binary content storage.** Content on computers is mostly stored in the form of bits. A *bit* or *binary digit* can take values 0 or 1. The set $\{0, 1\}$ has the structure of a field. This is called the field of two elements. The general theory of vector spaces and linear transformations applies here. This is part of a general theory of finite fields, exploring which would take us too far afield.

A piece of content looks like a sequence of bits. For instance, a 1 KB file is a sequence of 2^{13} bits (2^{10} bytes, and each byte is 2^3 bits). We can think of the individual bits in the file as the coordinates of a vector. The dimension of the vector space for a 1 KB file is 2^{13}. Note that the number of possible 1 KB files is $2^{2^{13}}$.

Transformations that take in files of a certain size and output files of another size may well arise as linear transformations between the corresponding vector spaces. For instance, a linear transformation:

$$\{0, 1\}^{2^{33}} \to \{0, 1\}^{2^{13}}$$

represents a transformation that takes as input a 1 GB file and outputs a 1 KB file.

Note that not every transformation must be linear, but a lot of the kinds of things we wish to achieve can be achieved via linear transformations.

5.2. **Coding/encryption.** Suppose there are two people, Alice and Bob, who wish to communicate messages (which we can think of as binary files) with each other over an insecure channel, where a third party, called Carl, can eavesdrop. Alice and Bob have a little time before they start sending messages (but before they actually know what communications they would like to exchange). They can use this secure, eavesdropper-free meeting to decide on a protocol to encode and decode messages.

Alice and Bob know the size of the file they expect to need to exchange. Let us say the file is n bits long. Alice and Bob agree upon an invertible linear transformation $T : \{0, 1\}^n \to \{0, 1\}^n$. When Alice wants to send Bob a file that can be described by a vector $\vec{v}$ with n coordinates, she does not send $\vec{v}$. Instead, she sends $T\vec{v}$ over the insecure channel.

Bob, who knows T already, also knows T^{-1}. Therefore, he can apply T^{-1} to $T(\vec{v})$ and recover $\vec{v}$. Another way of thinking about this is that Bob needs to solve the linear system whose coefficient matrix is the matrix of T and whose augmenting column is the vector $T(\vec{v})$.

Carl, who overhears $T(\vec{v})$, does not know T (because this was decided by Alice and Bob earlier through a secure communication channel that Carl could not eavesdrop upon). Since Carl has no idea about what T is, it is very difficult for him to decipher $\vec{v}$.

In the process above, the linear transformation T is termed the *encoding transformation* and its inverse transformation T^{-1} is termed the *decoding transformation.* The matrix for T is the *encoding matrix* and the matrix for T^{-1} is termed the *decoding matrix.*

What kind of linear transformation should ideally be chosen for T? There are many competing considerations, which we cannot cover here. The criteria are similar to the criteria for choosing a good password.

There is a problem with this sort of coding. It is not a secure form of encryption. The main problem is that insofar as the encoding matrix needs to be communicated between the sender and receiver, people may intercept it. Once they know the encoding matrix, all they need is elementary linear algebra and they have the decoding matrix. Now, if they intercept any communication, they will be able to decode it. The key problem with this method is that the decoding matrix is easy to deduce from the encoding matrix, because the algorithm for matrix inversion can run in polynomial time, since it essentially relies on Gauss-Jordan elimination.

There are more secure cryptographic methods, including RSA and the Diffie-Hellman key exchange. The key ideas are described below:

(1) The key principle of RSA (public key cryptography) is to use the fact that factoring a number is much harder than finding large primes to construct a pair of (encoding, decoding) such that, even if the encoding procedure is publicly known, the decoding procedure cannot be deduced quickly from that. With RSA, Bob has an (encoding, decoding) pair. He publishes the encoding information (called the *public key*) to the public. This includes Alice and Carl. Alice then sends Bob a message encoded using the public key. Bob then uses his decoding procedure (his *private key*) to decode the message. Alice herself cannot decode her own message after she has encoded it, even though she knows the original message.

    Carl, who is overhearing, knows the encoding procedure and knows the result of the encoding, but he still cannot deduce what the original message was.

    The reason that linear transformations are not amenable to this approach is that with linear transformations, the inverse of a linear transformation can be computed quickly (in time $\Theta(n^3)$ where n is the dimension of the spaces being transformed). So, publishing the encoding procedure is tantamout to publishing the decoding procedure. Note that when working over the field of two elements, as we are here, we do not need to distinguish between arithmetic complexity and bit complexity, because the numbers are always 0 or 1. They don't get arbitrarily complicated.

(2) Diffie-Hellman key exchange involves two people using a possibly insecure channel to exchange information that allows both of them to agree upon an encoding and decoding procedure whereas other people listening in on the same channel cannot figure out what procedure has been decided upon. The idea is that each person contributes part of the procedure in a way that third parties have no way of figuring out what's going on.

In both cases (RSA and Diffie-Hellman) the security of the protocols rests on the computational difficulty of certain mathematical problems related to prime numbers and modular arithmetic. This computational difficulty has not been mathematically proved, so it is possible that there exist fast algorithms to break these allegedly secure protocols. However, theoretical computer scientists and cryptographers believe it is highly unlikely that such fast algorithms exist.

The upshot of that is that the reason linear encoding is insecure is precisely that linear algebra is *computationally too easy!* Even if you don't think of it that way right now.

5.3. **Redundancy.** Return to the use of linear transformations for encoding. One of the problems with using an invertible (i.e., bijective) linear transformation is that it leaves no room for redundancy. If any bit gets corrupted, the original message can appear very different when decoded from what it was intended to look like. One way of handling with is to build redundancy by using a linear transformation that is injective but not surjective. Explicitly, use an injective linear transformation:

$$T : \{0,1\}^n \to \{0,1\}^{n+p}$$

for the purpose of encoding. Here, $p > 0$ describes the extent of redundancy in the message.

Injectivity still guarantees that the original message can be recovered uniquely from the encoded message. What we need to do is solve a linear system with a $(n + p) \times n$ coefficient matrix and full column rank. Assuming the message is not corrupted, the solution is unique.

What happens if the message gets corrupted? Note that since the system has rank n but has $n + p$ rows (equations), there are p redundant conditions. These p redundant conditions means that with a corrupted message, it is quite likely that the system will become inconsistent and we will not be able to decode the message, as we rightly should not, since it is corrupt. The probability that a corrupted message will still appear like a genuine message (i.e., it will give a consistent system to solve) is $1/2^p$. By choosing p reasonably large, we can make this probability quite small. Note that over the reals, the probability is effectively zero, but we are working over the field of two elements.

If you are interested more in this, look up *error-detecting codes* and *error-correcting codes.*

5.4. **Compression and more on redundancy.** In some cases, linear transformations can be used for purposes such as hashing, checksums, and various forms of compression. The idea is to take the original vector, which may live in a very huge dimension, and use a linear transformation to map it to a smaller dimension. The mapping is not injective, hence it is theoretically conceivable for two different vectors to give the same image. Nonetheless, the probability that two randomly chosen vectors give the same output is very small.

Suppose there are two files, $\vec{v}$ and $\vec{w}$, stored on different nodes of a network, both of size 1 GB. This means that both files can be viewed as vectors with 2^{33} coordinates. We want to check if $\vec{v}$ and $\vec{w}$ are equal. It would be very difficult to send the entire vectors across the network. One approach to checking equality probabilistically is to check if, say, the first 2^{10} coordinates of $\vec{v}$ are the same as the first 2^{15} coordinates of $\vec{w}$ (this is basically the first 4 KB of the files). This method may be misleading because there may have been copy errors made near the end which will not be caught by just looking at a subset of the coordinates.

A better way is to choose a linear transformation $T : \{0,1\}^{2^{33}} \to \{0,1\}^{2^{15}}$ that uses all the coordinates in a nontrivial fashion, apply T to $\vec{v}$ and $\vec{w}$ separately, then look at the outputs $T(\vec{v})$ and $T(\vec{w})$ and check whether they are equal (by communicating them across the network). The files that need to be compared are now just 4 KB long, and this comparison can be made relatively easily.

A couple of other remarks:

- If we wish, we could pick a *random* linear transformation T. A random linear transformation will most likely have rank equal to the output dimension, which is in this case 2^{15}. The probability of full rank is hard to calculate precisely, but it is almost 1.
- The probability of a collision, i.e., the probability that two different vectors $\vec{v}$ and $\vec{w}$ give rise to the same output, is quite low. Explicitly, if T has full rank, the probability is $1/2^{2^{15}}$. The intuitive explanation is that the probability that each bit happens to agree is $1/2$, and we multiply these probabilities. The independence of the probabilities requires full rank.

5.5. **Getting relevant information: the case of nutrition.** Suppose there are m different foodstuffs, and n different types of nutrients, and each food contains a fixed amount of each given nutrient per unit quantity of the good. The "foodstuffs vector" of your diet is a vector whose coordinates describe how much of each foodstuf you consume. The "nutrient vector" of your diet describes how much of each nutrient you obtain. There is a matrix describing the nutritional content of the food that defines a linear transformation from your food vector to your nutrition vector. The matrix has rows corresponding to nutrients and columns corresponding to foodstuffs, with the entry in each cell describing the amount of the row nutrient in the column food. This matrix defines a linear transformation from food vectors to nutrient vectors, i.e., whatever your foodstuff vector is, multiplying it by the matrix gives your nutrient vector.

If you wish for a balanced, healthy, and nutritious diet, this typically involves some constraints on the nutrient vector. Specifically, for some nutrients, there are both minimum and maximum values. For some nutrients, there may be only minimum values. The goal is to find a foodstuff vector whose image under the linear transformation satisfies all the constraints. This requires solving a system of linear *inequalities*. Whether a solution exists or not depends on the nature of the matrix describing the nutritional contents of the foods. For instance, if the only foodstuffs available to you are Coca Cola and Doritos, you are unlikely to be able to find a food vector that gives you a nutrient vector satisfying the constraints.

Note that the linearity assumption is a reasonable approximation for most nutrients, but there may be cases where this linear model fails. This occurs if there are complicated interactions between the nutrients or between various types of foods, and/or diminishing returns in nutrient content as you consume more of a certain food. For instance, it may be the case that consuming food A and food B does not just give you the sum of the nutritional content you would get from consuming foods A and B separately. The linear assumption is probably a reasonable approximation that works for most nutrients and hence one worth using insofar as it keeps the problem tractable.

5.6. **Many different dot products with a fixed vector.** One way of thinking of matrix-vector multiplication is in terms of there being many different dot products we want to compute where one of the vectors in the dot product is fixed, and the other one takes a finite list of values. The varying list is put as rows in a matrix, and the vector that is fixed is put as the column vector to be multiplied.