18.C06

Ace Chun

December 13, 2024

Contents

1 Basics of Linear Equations

The study of linear algebra is, essentially, the examination of the equation

$$A\mathbf{x} = \mathbf{b}$$

where A is a linear map, often represented as a matrix of numbers, and $\mathbf{x}$, $\mathbf{b}$ are vectors.

An $m \times n$ (m rows, n columns) matrix A takes inputs of vectors in $\mathbb{R}^n$ and outputs vectors in $\mathbb{R}^m$. Dimensions of "neighboring" maps should align, i.e.:

$$\underset{m\times n}{A}\ \underset{n\times 1}{\mathbf{x}} = \underset{m\times 1}{\mathbf{b}}$$

1.1 Row Picture

The "row picture" of linear equations understands them, fittingly, in terms of the rows of the matrix A; more specifically, the rows of A serve as *constraint equations* on the vector $\mathbf{x}$ to produce $\mathbf{b}$. Take, for example,

$$\begin{bmatrix} 1 & 2 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

The row picture says that the solution(s) to this linear equation are all points (x, y) that satisfy

$$\begin{aligned} x + 2y &= 1 \\ -x + y &= 2 \end{aligned}$$

1.2 Column Picture

The "column picture" of linear equations instead understands linear equations as linear combinations of column vectors (more specifically, the columns of A). Taking the example above,

$$\begin{bmatrix} 1 & 2 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

With the row picture, this would instead be interpreted as the values x and y that make the following combination possible:

$$x \begin{bmatrix} 1 \\ -1 \end{bmatrix} + y \begin{bmatrix} 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

1.3 Linearity

The matrix A is a linear operator, meaning that it has two qualities:

1. Additivity: $A(\mathbf{x} + \mathbf{y}) = A\mathbf{x} + A\mathbf{y}$
2. Homogeneity: $A(\alpha\mathbf{x}) = \alpha A\mathbf{x}$

2 Vector Spaces

2.1 Definitions

A vector space V is any set of "vectors" that support the addition operation and multiplication by a scalar. In addition, for any $x, y \in V$, their linear combinations must also be in V:

$$x, y \in V \Rightarrow \alpha x + \beta y \in V; \alpha, \beta \in \mathbb{R}$$

We put "vectors" in quotations because vector spaces *do not* actually have to be composed of vectors — we can create a vector space for any object, as long as it obeys the closure under addition and scalar multiplication.

A subspace S is a subset $S \subseteq V$ that is, itself, a vector space (i.e., for $x, y \in S$, $\alpha x + \beta y \in S$). A consequence of this is that *every* subspace must contain $\mathbf{0}$, because we must accomodate for linear combinations where the scalar multipliers $\alpha, \beta, \cdots$ are all 0.

In terms of linear equations, there are four particular subspaces that we're interested in, which are covered in the following sections.

2.2 Column Space

The column space of some matrix A is defined as the set of all vectors $\mathbf{x}$ that can be made from some linear combination of the columns of A. For

$$A = \begin{bmatrix} | & | & \cdots & | \\ | & | & & | \\ \mathbf{a}_1 & \mathbf{a}_2 & \cdots & \mathbf{a}_n \\ | & | & & | \\ | & | & \cdots & | \end{bmatrix}$$

the column space is then

$$C(A) = \{\mathbf{x} | \mathbf{x} = c_1\mathbf{a}_1 + c_2\mathbf{a}_2 + \cdot + c_n\mathbf{a}_n\}$$

In line with our "column picture," the equation

$$A\mathbf{x} = \mathbf{b}$$

is only solvable when $\mathbf{b} \in C(A)$, because that then means that there *does* exist some linear combination of the columns of $\mathbf{A}$ that yield $\mathbf{b}$.

From this, it is obvious that the basis for the column space of A is some set of linearly independent columns in A.

2.3 Null Space

The null space of A is defined as the set of vectors $\mathbf{x}$ that are mapped to $\mathbf{0}$ via A. By default, $\mathbf{0}$ itself is in the null space (which is pretty boring), so most of the time, we are interested in vectors that aren't $\mathbf{0}$ that are also in the null space (if there are any). More formally,

$$N(A) = \{\mathbf{x} | A\mathbf{x} = \mathbf{0}\}$$

Finding the basis for the null space is less obvious, and will be covered in section 4.2.3.

2.4 Row Space

The row space (also denoted $C(A^T)$) is the space obtained by taking linear combinations of the *rows* of A. Taking a linear combination of the rows is equivalent to "leftiplying" A by some row vector:

$$\begin{bmatrix} c_1 & c_2 & \cdots & c_n \end{bmatrix} A$$

2.5 Left Null Space

The left null space (also denoted $N(A^T)$) is the null space of the transpose of A, or all row vectors mapped to the zero vector by A.

$$\begin{bmatrix} c_1 & c_2 & \cdots & c_n \end{bmatrix} A = \mathbf{0}$$

3 Linear Independence

3.1 Definitions

A set of vectors $\{\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_n\}$ is said to be *linearly independent* iff

$$c_1\mathbf{v}_1 + c_2\mathbf{v}_2 + \cdots + c_n\mathbf{v}_n = 0$$

only when $c_1 = c_2 = \cdots = c_n = 0$. Stated differently, the vectors are linearly independent if

$$\begin{bmatrix} | & | & \cdots & | \\ \mathbf{v}_1 & \mathbf{v}_2 & \cdots & \mathbf{v}_n \\ | & | & \cdots & | \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

$$A\mathbf{c} = \mathbf{0}$$

only if $\mathbf{c} = \mathbf{0}$. A set of linearly independent vectors is called a *basis* for the space that they cover.

Conversely, a set of vectors is *linearly dependent* if any one of them can be expressed as linear combinations of the others. In other words, there exists a $\mathbf{c} \neq \mathbf{0}$ that, when multiplied by the matrix containing the set of vectors, yields the 0 vector.

We can discuss this in terms of the null space of the matrix A — its columns are *independent* if $N(A)$ only contains $\mathbf{0}$. Conversely, the columns of A are *dependent* if $N(A)$ contains vectors other than $\mathbf{0}$.

3.1.1 Vector Span

The span of a set of vectors $\{\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_n\}$ is defined to be all other vectors "reachable" by taking linear combinations of the set. Put mathematically,

$$\text{span } \mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_n = \left\{ c_1\mathbf{v}_1 + c_2\mathbf{v}_2 + \cdots + c_n\mathbf{v}_n \middle| c_1, c_2, \cdots, c_n \in \mathbb{R} \right\}$$

Or, in a cleaner matrix representation, all $\mathbf{b}$ where

$$\begin{aligned}\mathbf{b} &= \begin{bmatrix} | & | & & | \\ | & | & \cdots & | \\ \mathbf{v}_1 & \mathbf{v}_2 & \cdots & \mathbf{v}_n \\ | & | & \cdots & | \\ | & | & & | \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix} \\ &= A\mathbf{c}\end{aligned}$$

Note that the set of all $\mathbf{b}$s is exactly the *column space* of A.

3.2 Rank

There are many equivalent ways of describing rank, but they all boil down to expressing the number of "independent things" we have.

Formally, the rank of a matrix is the number of independent columns (or rows). The idea that the number of independent rows and columns is the same is not obvious, but it is important. The rank can also be defined as the *smallest number* r for which the following factorization of A exists:

$$\underset{m\times n}{A} = \underset{m\times r}{C}\ \underset{r\times n}{R}$$

The rank is also the *dimension* of both the column space and the row space, where dimension is defined as the number of basis vectors needed to define it.

For rectangular matrices, the maximum value the rank could have is the minimum of m, n. This is because you cannot have more independent rows than columns, or vice versa.

3.2.1 Rank-Nullity Theorem

The rank nullity theorem asserts that the dimension of the null space and the dimension of the column space of A add up to the total number of columns of A. That is,

$$\dim\ N(A) = n - r$$

The dimension of the left null space, similarly, is

$$\dim\ N(A^T) = m - r$$

3.2.2 Cases

If A is an $m \times n$ matrix, we have four following cases:

1. $m = n = r$ — A is square and full-rank.
2. $m < n, r = m$ — A is a wide matrix and is full row rank.
3. $m > n, r = n$ — A is a tall matrix and is full column rank.
4. $r < m, r < n$ — A is rank-deficient.

4 Solving Linear Equations

4.1 Existence and Uniqueness of Solutions

$$A\mathbf{x} = \mathbf{b}$$

will only have solutions for $\mathbf{x}$ when $\mathbf{b}$ is in the *column space* of A. Note that both $\mathbf{b}$ and $C(A)$ exist in the *ambient* space $\mathbb{R}^m$ — that is, the constituent parts of $C(A)$ and $\mathbf{b}$ are m-entry vectors. If $C(A)$ only covers part of that space ($r < m$), then there is a possibility that $\mathbf{b} \in \mathbb{R}^m$ is *not* in $C(A)$. This especially happens when matrices are "tall" (have more rows than columns), as the maximum rank of the matrix will definitively be less than m.

On the other hand, if $C(A)$ covers the entire ambient space $\mathbb{R}^m$, then $A\mathbf{x} = \mathbf{b}$ will always have a solution. However, the uniqueness of that solution depends on $N(A)$. To see why, consider a case where $N(A)$ contains vectors other than $\mathbf{0}$. We'll take one of these vectors and call it $\mathbf{x}_c$. In addition, we'll take a vector $\mathbf{x}_p$ that fulfills $A\mathbf{x}_p = \mathbf{b}$. $\alpha \in \mathbb{R}$. By linearity,

$$A(\mathbf{x}_p + \alpha\mathbf{x}_c) = A\mathbf{x}_p + \alpha A\mathbf{x}_c$$

However, $\mathbf{x}_c$ is in the null space of A by definition. So

$$A\mathbf{x}_p + \alpha A\mathbf{x}_c = \mathbf{b} + \mathbf{0} = \mathbf{b}$$

Therefore, if $\mathbf{x}_p$ is a solution to $A\mathbf{x} = \mathbf{b}$, then $\mathbf{x}_p + \alpha\mathbf{x}_c$ for an arbitrary constant α is *also* a solution to $A\mathbf{x} = \mathbf{b}$. In fact, if $N(A)$ has any non-zero vectors, then there will be an infinite number of solutions to $A\mathbf{x} = \mathbf{b}$ (due to multiplication by an arbitrary constant), given that there exists some $\mathbf{x}_p$ that solves the equation.

4.2 Gaussian Elimination

Gaussian elimination is the formalized version of the process that subtracts multiples of rows from other rows to produce something known as an *upper triangular* matrix, which a matrix that has non-zero entries on and above its diagonal, and zeroes everywhere else. For example,

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ 0 & a_{22} & a_{23} \\ 0 & 0 & a_{33} \end{bmatrix}$$

would be considered upper triangular. This is also called row-echelon form.

To solve a system $A\mathbf{x} = \mathbf{b}$, we would *augment* $\mathbf{b}$ to A and perform Gaussian elimination. For example, take some system

$$\begin{bmatrix} 1 & 3 & 1 \\ 1 & 1 & -1 \\ 3 & 11 & 6 \end{bmatrix} \mathbf{x} = \begin{bmatrix} 9 \\ 1 \\ 35 \end{bmatrix}$$

The augmented representation of this system would look like

$$\left[\begin{array}{ccc|c} 1 & 3 & 1 & 9 \\ 1 & 1 & -1 & 1 \\ 3 & 11 & 6 & 35 \end{array}\right]$$

We can now perform Gaussian elimination:

$$\left[\begin{array}{ccc|c} \boxed{1} & 3 & 1 & 9 \\ 1 & 1 & -1 & 1 \\ 3 & 11 & 6 & 35 \end{array}\right] \xrightarrow{r_2 - r_1} \left[\begin{array}{ccc|c} \boxed{1} & 3 & 1 & 9 \\ 0 & \boxed{-2} & -2 & -8 \\ 3 & 11 & 6 & 35 \end{array}\right]$$

$$\xrightarrow{r_3 - 3r_1} \left[\begin{array}{ccc|c} \boxed{1} & 3 & 1 & 9 \\ 0 & \boxed{-2} & -2 & -8 \\ 0 & 2 & 3 & 8 \end{array}\right] \xrightarrow{r_3 + r_2} \left[\begin{array}{ccc|c} \boxed{1} & 3 & 1 & 9 \\ 0 & \boxed{-2} & -2 & -8 \\ 0 & 0 & \boxed{1} & 0 \end{array}\right]$$

The boxed numbers represent *pivots*, which are numbers that are the leftmost non-zero quantity for their respective rows in the reduced matrix. A *pivot column* is a column that contains a pivot. It turns out that the rank of A can also be expressed as the number of pivot columns that it has.

Our resultant system (on the bottom right) can be translated back into:

$$\begin{bmatrix} 1 & 3 & 1 \\ 0 & -2 & -2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 9 \\ -8 \\ 0 \end{bmatrix}$$

Or, equivalently, in equation form:

$$\begin{aligned} x_1 + 3x_2 + x_3 &= 9 \\ -2x_2 - 2x_3 &= -8 \\ x_3 &= 0 \end{aligned}$$

Doing so makes it really clear what the solution to this system is; we known x_3 outright, and we can *backsubstitute* for each equation above to fully solve the system.

$$\begin{aligned} x_3 &= 0 \\ -2x_2 - 2x_3 = -2x_2 &= -8 \\ x_2 &= 4 \\ x_1 + 3x_2 + x_3 = x_1 + 12 &= 9 \\ x_1 &= -3 \\ \mathbf{x} &= \begin{bmatrix} -3 \\ 4 \\ 0 \end{bmatrix} \end{aligned}$$

The ability to use backsubstitution makes solving the linear system much easier, which is why we want to use Gaussian elimination to solve problems.

In general, the procedure goes:

1. Take the focused row's pivot (we'll call this number p). If everything underneath the pivot is 0, move to the next row and restart this process.

2. If there are nonzero elements in the column beneath the row (we'll call this number a), multiply the current row by $\frac{a}{p}$ and subtract it from a's row. This number $\frac{a}{p}$ is called a *multiplier*.

3. Continue until all of the rows beneath the focused row contain a zero in the column corresponding to the focused row's pivot. Once done, restart the process with the next row.

Reduced row echelon form would be normalizing each row to make the *pivot* equal to 1. In the case of the example above,

$$\left[\begin{array}{ccc|c} \boxed{1} & 3 & 1 & 9 \\ 0 & \boxed{1} & 1 & 4 \\ 0 & 0 & \boxed{1} & 0 \end{array}\right]$$

Gauss-Jordan elimination takes this one step further, and "backsubstitutes" ahead of time:

$$\left[\begin{array}{ccc|c} \boxed{1} & 3 & 1 & 9 \\ 0 & \boxed{1} & 1 & 4 \\ 0 & 0 & \boxed{1} & 0 \end{array}\right] \xrightarrow{r_2 - r_3} \left[\begin{array}{ccc|c} \boxed{1} & 3 & 1 & 9 \\ 0 & \boxed{1} & 0 & 4 \\ 0 & 0 & \boxed{1} & 0 \end{array}\right]$$

$$\xrightarrow{r_1 - r_3} \left[\begin{array}{ccc|c} \boxed{1} & 3 & 0 & 9 \\ 0 & \boxed{1} & 0 & 4 \\ 0 & 0 & \boxed{1} & 0 \end{array}\right] \xrightarrow{r_1 - r_3} \left[\begin{array}{ccc|c} \boxed{1} & 0 & 0 & -3 \\ 0 & \boxed{1} & 0 & 4 \\ 0 & 0 & \boxed{1} & 0 \end{array}\right]$$

Note that Gaussian elimination changes the *column space* of A, but not its *null space*. We can therefore find the basis of $N(A)$ through Gaussian elimination.

4.2.1 Where it fails

What happens if a column doesn't have a pivot, or the number that is supposed to be the pivot (if we follow the diagonal pattern) is zero? Notice that if this is the case, then the multiplier $\frac{a}{p}$ will be undefined, and the procedure breaks.

There are one of two cases.

Case 1: The pivot exists, but the matrix doesn't follow echelon form

We may end up in a case like

$$\left[\begin{array}{ccc|c} \boxed{1} & -1 & 2 & 1 \\ 0 & 0 & \boxed{1} & 1 \\ 0 & \boxed{-4} & 8 & 0 \end{array}\right]$$

All of the columns technically have pivots, but the rows are not in the nice row-echelon form that we'd like to be in to perform backsubstitution. In this case, we can simply permute the matrix's rows (covered in section 4.2.2).

Case 2: The pivot doesn't exist

If we end up with a row of all zeroes, and the column in question has all zeroes below it, then we've reached a block. We can't simply permute any of the rows, because we won't get anywhere that way. In this case, we simply don't have a pivot for that column, and A is rank-deficient.

4.2.2 Permutation

Permutation matrices have exactly 1 entry (of magnitude 1) in each row and column. For example,

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

The effect of multiplying P by some matrix A is that we end up with a permutation of the rows or the columns (depending on if we left-multiply or right-multiply).

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{31} & a_{32} & a_{33} \\ a_{21} & a_{22} & a_{23} \end{bmatrix}$$

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{13} & a_{12} \\ a_{21} & a_{23} & a_{22} \\ a_{31} & a_{33} & a_{32} \end{bmatrix}$$

The inverse of a permutation matrix is its transpose P^T (we will discover why in section 5).

4.2.3 Null Space Basis

Suppose we have a rectangular system

$$A = \begin{bmatrix} 1 & 2 & 3 & 1 \\ 1 & 4 & 5 & -3 \\ 1 & 6 & 7 & -7 \end{bmatrix}$$

This matrix is "wide," which means that all of its columns cannot be pivot columns. We can perform Gaussian elimination, which yields

$$\begin{bmatrix} \boxed{1} & 2 & 3 & 1 \\ 0 & \boxed{2} & 2 & -4 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

We have a rank of 2, which means that $N(A)$ has dimensions of $4 - 2 = 2$.

We call the columns without pivots in them "free columns," and the variables in the input the columns correspond to are the degrees of freedom of the system. We can choose any value for these free variables and backsolve for the pivot variables, which will provide a basis for our null space. To demonstrate this, let us call the portion of the matrix containing only the pivot variables U_r:

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

Meanwhile, we will place the non-zero rows of the free columns in another matrix:

$$F = \begin{bmatrix} 3 & 1 \\ 2 & -4 \end{bmatrix}$$

Given U_r and F, the generic form of a post-elimination matrix is:

$$\begin{bmatrix} \underset{r\times r}{U_r} & \underset{r\times(n-r)}{F} \\ \underset{(m-r)\times r}{0} & \underset{(m-r)\times(n-r)}{0} \end{bmatrix}$$

Creating a system:

$$\begin{bmatrix} U_r & F \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{p} \\ \mathbf{f} \end{bmatrix} = \mathbf{0}$$

which then turns into

$$\begin{bmatrix} U_r\mathbf{p} + F\mathbf{f} \\ 0 \end{bmatrix} = 0$$

We must therefore solve the system

$$U_r\mathbf{p} = -F\mathbf{f}$$

Recall that $\mathbf{f}$ represents our *free variables*; we can choose anything for $\mathbf{f}$ and determine a unique $\mathbf{p}$, which leads us to a nullspace vector

$$\begin{bmatrix} \mathbf{p} \\ \mathbf{f} \end{bmatrix}$$

In addition, we can take linear combinations of any vectors in the nullspace, the result of which will also be in the nullspace. It is therefore *easy* for us to choose basis vectors for $\mathbf{f}$, and then reconstruct the rest of the nullspace from linear combinations of them — commonly, we tend to choose the unit basis vectors $\mathbf{i}, \mathbf{j}$. Applying this to our example system, we end up with:

$$\begin{bmatrix} 1 & 2 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \end{bmatrix} = - \begin{bmatrix} 3 & 1 \\ 2 & -4 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

After solving for $p_{11}, p_{12}, p_{21}, p_{22}$, we can construct the basis of our nullspace:

$$c_1 \begin{bmatrix} p_{11} \\ p_{21} \\ 1 \\ 0 \end{bmatrix} + c_2 \begin{bmatrix} p_{12} \\ p_{22} \\ 0 \\ 1 \end{bmatrix}$$

which, in the context of the example, happens to be

$$c_1 \begin{bmatrix} -1 \\ -1 \\ 1 \\ 0 \end{bmatrix} + c_2 \begin{bmatrix} -5 \\ 2 \\ 0 \\ 1 \end{bmatrix}$$

If we had performed Gauss-Jordan elimination on the original matrix instead, and ended up with a matrix of the form

$$\begin{bmatrix} I & F \\ 0 & 0 \end{bmatrix}$$

then the basis of our null space would be the columns of the matrix

$$\begin{bmatrix} -F \\ I \end{bmatrix}$$

with appropriate dimensions for I and F.

4.3 Inverses

If we think about A as applying some operation on a vector $\mathbf{x}$, its inverse A^{-1} is the operation that "undoes" A.

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

It follows that

$$AA^{-1} = I$$

Inverses only exist for full-rank and square matrices.

Given a system

$$A\mathbf{x} = \mathbf{b}$$

it is conceivable that we can multiply both sides on the left ("leftiply") by A^{-1}:

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

The inverse of a matrix product is

$$(AB)^{-1} = B^{-1}A^{-1}$$

We can envision this in terms of the steps needed to "undo" the operation AB; because we applied A last, we must undo it *first*, and then undo B to return to our original vector. This result also implies that a matrix's product with its inverse is commutative:

$$(A^{-1}A)^{-1} = I^{-1} = I = AA^{-1}$$

Matrix inverse are *unique* for each A, provided that it does have one. This also implies that the solution $\mathbf{x} = A^{-1}\mathbf{b}$ itself is unique — therein lies another condition for the existence and uniqueness of solutions to $A\mathbf{x} = \mathbf{b}$:

$$\exists!\ \mathbf{x} \mid A\mathbf{x} = \mathbf{b} \longleftrightarrow \exists\ A^{-1}$$

Only square matrices can have matrix inverses. However, for "tall" and "wide" matrices with full column rank and full row rank respectively, we can define what are known as left and right inverses.

$$A_{left}^{-1} = (A^T A)^{-1} A^T$$
$$A_{right}^{-1} = A^T (A A^T)^{-1}$$

Note that these only apply when A is either full row rank or full column rank. In addition, we cannot multiply a rank deficient A with either a left or a right inverse, as this result depends on the idea that either $A^T A$ or AA^T will be square and invertible (depending on the nature of the independence of the matrix).

In general, computing matrix inverses is computationally difficult, which is why we don't want to explicitly compute the inverse itself. We instead interpret "find the result of $A^{-1}\mathbf{b}$" instead as "solve the system $A\mathbf{x} = \mathbf{b}$ for $\mathbf{x}$." How we do this easily without solving for the inverse will be covered in the following section (4.4).

4.4 LU Factorization

LU factorization stands for "lower-upper triangular factorization." Just like the name suggests, we take a matrix A and factorize it into L and U, which are lower and upper triangular matrices, respectively. It turns out that performing Gaussian elimination and "keeping a record" of the steps we've done is exactly the same as factorizing $A = LU$.

What do we mean by "keeping a record"? If we take our example matrix from an earlier section,

$$\begin{bmatrix} 1 & 3 & 1 \\ 1 & 1 & -1 \\ 3 & 11 & 6 \end{bmatrix}$$

We will perform Gaussian elimination and bookkeep our steps.

$$\begin{bmatrix} \boxed{1} & 3 & 1 \\ 1 & 1 & -1 \\ 3 & 11 & 6 \end{bmatrix} \xrightarrow{r_2 - r_1, r_3 - 3r_1} \begin{bmatrix} \boxed{1} & 3 & 1 \\ 0 & \boxed{-2} & -2 \\ 0 & 2 & 3 \end{bmatrix} \xrightarrow{r_3 + r_2} \begin{bmatrix} \boxed{1} & 3 & 1 \\ 0 & \boxed{-2} & -2 \\ 0 & 0 & \boxed{1} \end{bmatrix} = U$$

If we were to express the rows of A by the rows of U, we would have to work

"backwards":

$$A_1 = U_1$$
$$A_2 - U_1 = U_2 \to A_2 = U_2 + U_1$$
$$A_3 - 3U_1 + 2U_2 = U_3 \to A_2 = U_3 - 2U_2 + 3U_1$$

We end up with a vaguely echelon-like system of equations

$$A_1 = U_1$$
$$A_2 = U_1 + U_2$$
$$A_3 = 3U_1 - 2U_2 + U_3$$

which is equivalent to saying

$$\begin{bmatrix} A_1 \\ A_2 \\ A_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 3 & -2 & 1 \end{bmatrix} \begin{bmatrix} U_1 \\ U_2 \\ U_3 \end{bmatrix}$$

However, remember that each A_i and U_i are essentially *row vectors* — they represent the rows of their parent matrix. So if we expand this representation, we see

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 3 & -2 & 1 \end{bmatrix} \begin{bmatrix} 1 & 3 & 1 \\ 0 & -2 & -2 \\ 0 & 0 & 1 \end{bmatrix}$$

which is an LU factorization!

$$L = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 3 & -2 & 1 \end{bmatrix}$$
$$U = \begin{bmatrix} 1 & 3 & 1 \\ 0 & -2 & -2 \\ 0 & 0 & 1 \end{bmatrix}$$

The process works the same in general for *all* matrices, regardless of shape.

If we end up with a matrix in which the pivots are not in the "correct" order (i.e., not strictly down the diagonals), we attach a permutation matrix P:

$$PA = LU$$

and then we can perform elimination like usual. Permutation matrices are easy to invert, as it turns out, so it doesn't change the way in which we can apply LU decomposition.

4.4.1 Solving with LU factorization

Suppose we have a system that we want to solve

$$A\mathbf{x} = \mathbf{b}$$

As per our earlier discussion about matrix inverses, we could find the solution to this system by taking A^{-1} — however, this is computationally expensive, so we want to avoid taking inverses if at all possible.

Instead, to solve this system, we can first decompose A (or really, PA) into LU.

$$LU\mathbf{x} = \mathbf{b}$$

The solution now becomes

$$\mathbf{x} = (LU)^{-1}\mathbf{b} = U^{-1}L^{-1}\mathbf{b}$$

How are U^{-1} and L^{-1} any better to find?

The trick lies in that we aren't actually going to find the inverses themselves — we will instead take the problem piece by piece.

Let's define a vector

$$\mathbf{y} = L^{-1}\mathbf{b}$$

so that

$$\mathbf{x} = U^{-1}\mathbf{y}$$

We've now decomposed our system into two separate steps. Recall what we said earlier about the significance of matrix inverses:

$$\mathbf{y} = L^{-1}\mathbf{b} \iff \text{Solve } L\mathbf{y} = \mathbf{b} \text{ for } \mathbf{y}$$

The structure of L makes this much easier to do — we can "forward substitute" for the values of $\mathbf{y}$, because of its lower-triangular echelon structure.

Once we have solved for $\mathbf{y}$, we move on to step 2:

$$\mathbf{x} = U^{-1}\mathbf{y} \iff \text{Solve } U\mathbf{x} = \mathbf{y} \text{ for } \mathbf{x}$$

Once again, the echelon structure of U enables us to "back-substitute" for $\mathbf{x}$ and therefore find the solution to the initial system, $A\mathbf{x} = \mathbf{b}$.

Note that if we did have some P prepended to A in our LU factorization, we need to keep track of the rows that we exchanged and permute $\mathbf{x}$ accordingly.

5 Orthogonality

5.1 Transposes

Transposing a column vector will turn it into a row vector, and vice versa:

$$\mathbf{a}^T = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_m \end{bmatrix}^T = \begin{bmatrix} a_1 & a_2 & \cdots & a_m \end{bmatrix}$$

Transposing a matrix consists of swapping its rows and columns.

$$\underset{m\times n}{\begin{bmatrix} | & | & \cdots & | \\ \mathbf{a}_1 & \mathbf{a}_2 & \cdots & \mathbf{a}_n \\ | & | & \cdots & | \end{bmatrix}^T} = \underset{n\times m}{\begin{bmatrix} — & \mathbf{a}_1^T & — \\ — & \mathbf{a}_2^T & — \\ \vdots & \vdots & \vdots \\ — & \mathbf{a}_m^T & — \end{bmatrix}}$$

If $A = A^T$, then we say that A is *symmetric*. In addition,

$$(AB)^T = B^T A^T$$

The intuition behind a transpose is essentially moving an operator from right to left inside a dot product (expanded upon in the next section).

$$\mathbf{x}^T[A\mathbf{y}] = [\mathbf{x}^T A]\mathbf{y} = [A\mathbf{x}]^T\mathbf{y} = A\mathbf{x} \cdot \mathbf{y}$$

5.2 Vector Properties and Dot Products

The *dot product* (also called the *inner product*) of two vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ is defined as the following:

$$\mathbf{x} \cdot \mathbf{y} = \mathbf{x}^T\mathbf{y}$$

Note that $\mathbf{x}$ and $\mathbf{y}$ must be of the same dimension for the statement above to make any sense. $\mathbf{x}^T\mathbf{y}$ yields a 1×1 matrix, also known as a scalar, by definition, and it can be computed as the sum of pairwise products:

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \begin{bmatrix} y_1 & y_2 & \cdots & y_n \end{bmatrix} = x_1y_1 + x_2y_2 + \cdots + x_ny_n$$

The *length* of a dot product is computed via some norm (most commonly, this is the Euclidean norm, stated below):

$$||\mathbf{x}|| = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2}$$

The dot product of two vectors $\mathbf{x}$ and $\mathbf{y}$ can also give us information about the vectors themselves:

$$\mathbf{x}^T\mathbf{y} = ||\mathbf{x}||\ ||\mathbf{y}|| \cos\theta$$

where θ is the angle between the two vectors. It then follows that if $\mathbf{x}$ and $\mathbf{y}$ are *orthogonal*, then their dot product will be 0, and if they are *parallel*, then their dot product will just be the product of their lengths.

5.3 Orthonormal Bases

An orthonormal basis is a set of basis vectors $Q = \{\mathbf{q}_1, \mathbf{q}_2, \cdots, \mathbf{q}_n\}$ such that:

1. Each $||\mathbf{q}_i|| = 1$
2. $\mathbf{q}_i \cdot \mathbf{q}_j = 1$ if $i = j$, else $\mathbf{q}_i \cdot \mathbf{q}_j = 0$

The second statement is essentially a formal statement that each vector must be orthogonal to every other vector in the basis set.

Orthogonal bases are nicer than any arbirary set of basis vectors because they uphold these two properties — it makes it easier to compute certain properties. For example, define a vector $\mathbf{b} \in$ span Q. Therefore,

$$\mathbf{b} = c_1\mathbf{q}_1 + c_2\mathbf{q}_2 + \cdots + c_n\mathbf{q}_n$$

It becomes really easy to find each c_i — if we just take the dot product of $\mathbf{b}$ with any given basis vector $\mathbf{q}_i$, the components and coefficients of all of the other basis vectors simply go away:

$$\mathbf{q}_i^T\mathbf{b} = c_1\mathbf{q}_i^T\mathbf{q}_1 + \cdots + c_i\mathbf{q}_i^T\mathbf{q}_i + \cdots + c_n\mathbf{q}_i^T\mathbf{q}_n = c_i\mathbf{q}_i^T\mathbf{q}_i = c_i$$

Furthermore, given that we've defined $\mathbf{b}$ in an orthogonal basis, we can easily find its length:

$$||\mathbf{b}||^2 = c_1^2 + c_2^2 + \cdots + c_n^2$$

If we put the orthonormal basis vectors into a matrix Q,

$$Q = \begin{bmatrix} | & | & \cdots & | \\ | & | & & | \\ \mathbf{q}_1 & \mathbf{q}_2 & \cdots & \mathbf{q}_n \\ | & | & & | \\ | & | & \cdots & | \end{bmatrix}$$

then we end up with a matrix with some convenient properties. Namely,

$$Q^T Q = I$$

In addition, the linear map Q preserves the length of any vector it is applied to. When Q is square, it is called an *orthogonal matrix* (or *unitary matrix*), and we can say that $Q^{-1} = Q^T$ (we can't say this if Q isn't square — non-square matrices don't have inverses!). In particular, the permutation matrices P are special kinds of orthonormal matrices.

5.4 Orthogonal Subspaces and Complements

Two subspaces $S_1, S_2 \subset V$ are called *orthogonal* to each other if all vectors in S_1 are orthogonal to every vector in S_2.

$$S_1 \perp S_2 \longleftrightarrow \forall \mathbf{x} \in S_1,\ \forall \mathbf{y} \in S_2,\ \mathbf{x}^T \mathbf{y} = 0$$

For some subspace $S \subset V$, its orthogonal complement $S^\perp$ is the set of every vector in V that is orthogonal to every vector in S.

$$S^\perp = \{\mathbf{x} \mid \forall \mathbf{y} \in S,\ \mathbf{x}^T \mathbf{y} = 0\}$$

In particular, the dimension of S and the dimension of $S^\perp$ must add up to the dimension of V.

Since vector spaces have bases, to confirm that two spaces are orthogonal, we only need to take care of its basis vectors (as linear combinations of orthogonal vectors will also be orthogonal). The union of the set of bases of S and $S^\perp$ should be the basis for V.

5.4.1 Four Fundamental Subspaces

Consider a matrix A. We covered the four fundamental subspaces $C(A)$, $C(A^T)$, $N(A)$, and $N(A^T)$ in section 2.

First, we'll consider $C(A)$. $C(A)^T$ must be the set of all vectors $\mathbf{u}$ such that $\mathbf{u}^T(A\mathbf{x}) = 0$ for all vectors $\mathbf{x} \in \mathbb{R}^n$.

$$C(A)^\perp = \{\mathbf{u} \mid \mathbf{u}^T A\mathbf{x} = 0\} = \{\mathbf{u} \mid A^T\mathbf{u} = 0\}$$

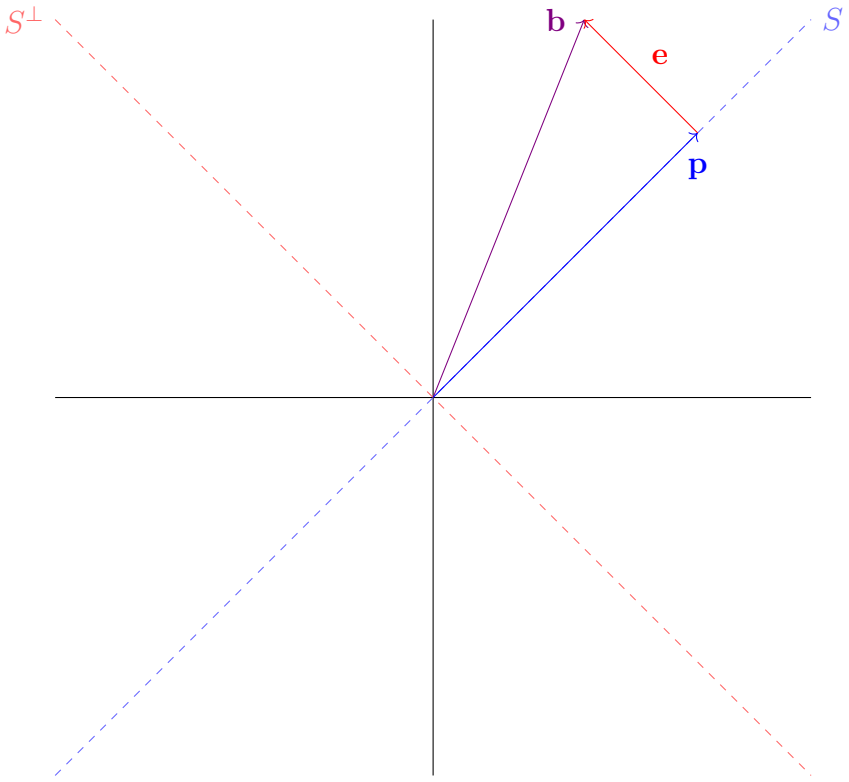
The latter half of the above expression indicates that the left nullspace, $N(A^T)$, is the orthogonal complement of $C(A)$. We can repeat this construction with $C(A^T)$ — we will see that its orthogonal complement is $N(A)$.

5.5 Orthogonal Projection

Given a vector $\mathbf{b} \in \mathbb{R}^m$, we can "split" it into its components along orthogonal complement subspaces S and $S^\perp$, so that

$$\mathbf{b} = \mathbf{p} + \mathbf{e},\ \mathbf{p} \in S,\ \mathbf{e} \in S^\perp$$

For example, in a 2×2 case,



We will call the transformation from $\mathbf{b}$ to $\mathbf{p}$ the projection matrix P (not to be confused with the permutation matrix), so that

$$\mathbf{p} = P\mathbf{b}$$
$$\mathbf{e} = \mathbf{b} - P\mathbf{b} = (I - P)\mathbf{b}$$

As it turns out, given some projection operator P onto a subspace S, the projection operator onto the orthogonal complement $S^{\perp}$ is just $I-P$. In general, the orthogonal projection $\mathbf{p}$ onto the subspace S is the *closest* member of S to $\mathbf{b}$.

In addition, projection matrices have two important properties:

1. $P^T = P$
2. $P^2 = P$ (or any $P^n = P$)

If A is full column rank, we are interested in the subspace $C(A)$ (and its orthogonal complement, $N(A^T)$), especially if we want to solve something like $A\mathbf{x} = \mathbf{b}$ when $\mathbf{b} \notin C(A)$.

Suppose we project $\mathbf{b}$ into $C(A)$, giving us a resulting projection $\mathbf{p}$. $\mathbf{p}$ is, by definition, in $C(A)$, which means we can denote it as some

$$\mathbf{p} = A\mathbf{x}_*$$

for $\mathbf{x}_* \in \mathbb{R}^n$. Therefore,

$$\mathbf{b} = A\mathbf{x}_* + \mathbf{e} \longrightarrow \mathbf{e} = \mathbf{b} - A\mathbf{x}_*$$

We also know that $\mathbf{e} \in C(A)^{\perp} = N(A^T)$. Therefore,

$$A^T\mathbf{e} = A^T(\mathbf{b} - A\mathbf{x}_*) = \mathbf{0}$$

Through distribution, we see that

$$A^TA\mathbf{x}_* = A^T\mathbf{b}$$

The above system is known as the *normal equations*. Furthermore,

$$\mathbf{x}_* = (A^TA)^{-1}A^T\mathbf{b}$$

and

$$A\mathbf{x}_* = P\mathbf{b} = A(A^TA)^{-1}A^T\mathbf{b}$$

Thus, the projection matrix P onto the column space $C(A)$ is

$$P_{C(A)} = A(A^T A)^{-1} A^T$$

Note that if A is invertible (square, full rank), then $P = I$, and the projection matrix onto the orthogonal complement is 0 (if $C(A)$ covers the whole space, there's no more space for an orthogonal complement).

For a "wide" matrix A that is full row rank, we can undergo the same process and come to the conclusion

$$P_{C(A^T)} = A^T(AA^T)^{-1} A$$

5.5.1 Gram-Schmidt Process

The Gram-Schmidt Process generates an orthogonal basis $\mathbf{q_1}, \mathbf{q_2}, \cdots, \mathbf{q_n}$ out of a set of linearly independent vectors $\mathbf{a}_1, \mathbf{a}_2, \cdots, \mathbf{a_n}$ by utilizing projection.

We start with one of the vectors in our original set, say, $\mathbf{a}_1$. We divide it by its magnitude to normalize it:

$$\mathbf{q}_1 = \frac{\mathbf{a}_1}{||\mathbf{a}_1||}$$

From the next vector, we *subtract* the component of the vector in the direction $\mathbf{a}_1$ (or $\mathbf{q}_1$) and normalize it once again to produce $\mathbf{q}_2$.

$$\mathbf{q}_2 = \frac{\mathbf{a}_2 - \mathbf{q}_1(\mathbf{q}_1 \cdot \mathbf{a}_2)}{||\mathbf{a}_2 - \mathbf{q}_1(\mathbf{q}_1 \cdot \mathbf{a}_2)||}$$

Note that the numerator of the above is the same as taking the projection onto the orthogonal complement of q_1:

$$\mathbf{q}_2 = \frac{(I - \mathbf{q}_1\mathbf{q}_1^T)\mathbf{a}_2}{||(I - \mathbf{q}_1\mathbf{q}_1^T)\mathbf{a}_2||} = \frac{\mathbf{a}_2 - \mathbf{q}_1\mathbf{q}_1^T\mathbf{a}_2}{||\mathbf{a}_2 - \mathbf{q}_1\mathbf{q}_1^T\mathbf{a}_2||}$$

The process repeats, but for each successive vector $\mathbf{a}_i$, we subtract the $i-1$ orthogonal components that we've already established from it.

$$\mathbf{q}_3 = \frac{(I - \mathbf{q}_1\mathbf{q}_1^T - \mathbf{q}_2\mathbf{q}_2^T)\mathbf{a}_3}{||''||} = \frac{\mathbf{a}_3 - \mathbf{q}_1\mathbf{q}_1^T\mathbf{a}_3 - \mathbf{q}_2\mathbf{q}_2^T\mathbf{a}_3}{||''||}$$

and so on and so forth.

5.5.2 QR Factorization

The Gram-Schmidt process can be thought of as a matrix factorization. Like LU decomposition, we also work through the process "backwards" to find the QR decomposition.

First, let's name the denominator of each successive basis vector $\mathbf{q}_i$ as some r_{ii}. For example,

$$\begin{aligned} r_{11} &= ||\mathbf{a}_1|| \\ r_{22} &= ||\mathbf{a}_2 - \mathbf{q}_1\mathbf{q}_1^T\mathbf{a}_2|| \\ r_{33} &= ||\mathbf{a}_3 - \mathbf{q}_1\mathbf{q}_1^T\mathbf{a}_3 - \mathbf{q}_2\mathbf{q}_2^T\mathbf{a}_3|| \end{aligned}$$

and so on and so forth, so we can express

$$\begin{aligned} \mathbf{q}_1 &= \frac{\mathbf{a}_1}{r_{11}} \\ \mathbf{q}_2 &= \frac{\mathbf{a}_2 - \mathbf{q}_1\mathbf{q}_1^T\mathbf{a}_2}{r_{22}} \\ \mathbf{q}_3 &= \frac{\mathbf{a}_3 - \mathbf{q}_1\mathbf{q}_1^T\mathbf{a}_3 - \mathbf{q}_2\mathbf{q}_2^T\mathbf{a}_3}{r_{33}} \\ &\vdots \end{aligned}$$

Furthermore, $\mathbf{q}_1^T\mathbf{a}_2$, $\mathbf{q}_1^T\mathbf{a}_3$, $\mathbf{q}_2^T\mathbf{a}_3$, etc., are all numbers (results of dot products). We'll define

$$r_{ij} = \mathbf{q}_i\mathbf{a}_j$$

so that we are left with

$$\begin{aligned} \mathbf{q}_1 &= \frac{\mathbf{a}_1}{r_{11}} \\ \mathbf{q}_2 &= \frac{\mathbf{a}_2 - r_{12}\mathbf{q}_1}{r_{22}} \\ \mathbf{q}_3 &= \frac{\mathbf{a}_3 - r_{13}\mathbf{q}_1 - r_{23}\mathbf{q}_2}{r_{33}} \\ &\vdots \end{aligned}$$

We can do some rearranging, and find that

$$\begin{aligned}
\mathbf{a}_1 &= r_{11}\mathbf{q}_1 \\
\mathbf{a}_2 &= r_{12}\mathbf{q}_1 + r_{22}\mathbf{q}_2 \\
\mathbf{a}_3 &= r_{13}\mathbf{q}_1 + r_{23}\mathbf{q}_2 + r_{33}\mathbf{q}_3 \\
&\vdots
\end{aligned}$$

We see an echelon-like structure emerging once again! In matrix notation:

$$\begin{bmatrix} | & | & \cdots & | \\ \mathbf{a}_1 & \mathbf{a}_2 & \cdots & \mathbf{a}_n \\ | & | & \cdots & | \end{bmatrix} = \begin{bmatrix} | & | & \cdots & | \\ \mathbf{q}_1 & \mathbf{q}_2 & \cdots & \mathbf{q}_n \\ | & | & \cdots & | \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & \cdots \\ & r_{22} & r_{23} & \cdots \\ & & r_{33} & \cdots \\ & & & \ddots \end{bmatrix}$$

$$\underset{m\times n}{A} = \underset{m\times n}{Q} \; \underset{n\times n}{R}$$

6 Singular Value Decomposition

The (full) singular value decomposition of a matrix A is its factorization into

$$\underset{m\times n}{A} = \underset{m\times m}{U} \; \underset{m\times n}{\Sigma} \; \underset{n\times n}{V^T}$$

where U and V are orthogonal matrices and Σ is a diagonal matrix, containing what are called *singular values*. In expanded form,

$$\underset{m\times n}{A} = \begin{bmatrix} | & | & \cdots & | \\ \mathbf{u}_1 & \mathbf{u}_2 & \cdots & \mathbf{u}_m \\ | & | & \cdots & | \end{bmatrix} \begin{bmatrix} \sigma_1 & & & & \\ & \sigma_2 & & & \\ & & \ddots & & \\ & & & \sigma_r & \\ & & & & 0 \\ & & & & & \ddots \end{bmatrix} \begin{bmatrix} — & \mathbf{v}_1^T & — \\ — & \mathbf{v}_2^T & — \\ & \vdots & \\ — & \mathbf{v}_n^T & — \end{bmatrix}$$

where r is the rank of A. By convention, the σ_is are arranged in descending order:

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r \geq \sigma_{r+1} = 0$$

The u_is are referred to as the *left singular vectors*, while the v_is are referred to as the *right singular vectors*. Note that A can also be written as

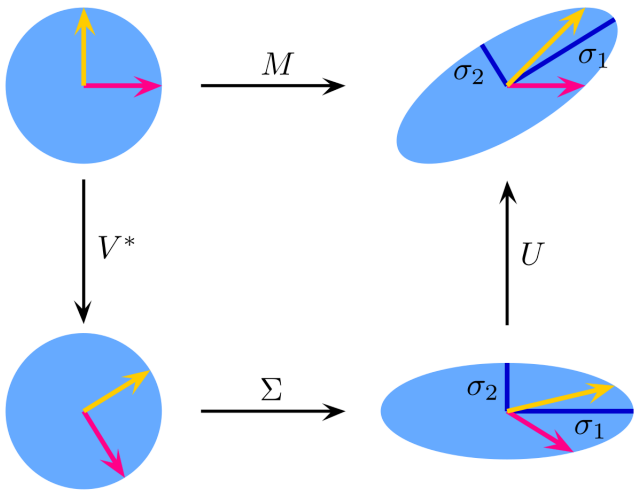
$$A = \sum_{i=0}^{r} \sigma_i \mathbf{u}_i \mathbf{v}_i^T$$

We are effectively summing r independent, rank-1 matrices and *scaling* them by σ_i. σ_i can thus be intuited as the relative contribution of each rank-1 component of A. Conversely, the left and right singular vectors corresponding to a σ_i of 0 does not contribute to A. We can therefore choose to cut them out — this is known as the *compact* SVD.

$$\underset{m\times n}{A} = \underset{m\times r}{\hat{U}}\ \underset{r\times r}{\hat{\Sigma}}\ \underset{r\times n}{\hat{V}^T}$$

$$= \underset{m\times n}{A} = \begin{bmatrix} | & | & \cdots & | \\ \mathbf{u}_1 & \mathbf{u}_2 & \cdots & \mathbf{u}_r \\ | & | & \cdots & | \end{bmatrix} \begin{bmatrix} \sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_r \end{bmatrix} \begin{bmatrix} — & \mathbf{v}_1^T & — \\ — & \mathbf{v}_2^T & — \\ & \vdots & \\ — & \mathbf{v}_r^T & — \end{bmatrix}$$

The compact SVD and the full SVD are equivalent — each have their notational advantages. Importantly, every matrix has an SVD. Geometrically, the SVD has the effect of stretching some unit ball into an ellipsoid, as displayed in the diagram below:



$$M = U \cdot \Sigma \cdot V^*$$

6.1 Properties

From the SVD, we have another definition for rank — the rank of A is the number of non-zero singular values it has. Furthermore, given that the σ_is are arranged in descending order (with the $\sigma_i = 0$s at the end),

1. $\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_r$ are a basis for $C(A)$.
2. $\mathbf{u}_{r+1}, \cdots, \mathbf{u}_m$ are a basis for $N(A^T)$.
3. $\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_r$ are a basis for $C(A^T)$.
4. $\mathbf{v}_{r+1}, \cdots, \mathbf{v}_n$ are a basis for $N(A)$.

We can write

$$A\mathbf{v}_i = \sigma_i \mathbf{u}_i$$

for all i. For a nonzero σ_i, it is clear that A takes the orthonormal basis vector of its input space v_i and outputs a scalar multiple of an orthonormal basis vector u_i for its output space. If σ_i is 0, this means that v_i is in the null space of A.

6.2 Low-rank Approximation

By virtue of the singular values being ordered in descending order of magnitude, this gives us information about the most "relevant" $\mathbf{u}$ and $\mathbf{v}$ vector pairs that contribute to the composition of A. We introduce something called the *Truncated SVD* (TSVD), which cuts off some $r - k$ singular values to approximate A with a rank k matrix.

Why would we want to reduce the rank of a matrix? Matrices with higher rank generally take up much more storage, and tends to be accompanied with noise. Cutting out the singular values less than some ϵ will effectively remove this noise, giving us a nicer matrix to work with.

As it turns out, the TSVD is the *best* rank-k approximation of a given matrix A, as stated by the Eckart-Young theorem.

6.3 Moore-Penrose Pseudoinverse

Any arbitrary matrix A is *not* guaranteed to have an inverse. However, every matrix may have a pseudoinverse A^+ that approximates the behavior of an

inverse because it can be derived from the SVD. In particular, if

$$A = U\Sigma V^T$$

then

$$\underset{n\times m}{A^+} = \underset{n\times n}{V}\ \underset{n\times m}{\Sigma^+}\ \underset{m\times m}{U^T}$$

where

$$\Sigma^+ = \begin{bmatrix} \frac{1}{\sigma_1} & & & & & \\ & \frac{1}{\sigma_2} & & & & \\ & & \ddots & & & \\ & & & \frac{1}{\sigma_r} & & \\ & & & & 0 & \\ & & & & & \ddots \end{bmatrix}$$

with dimensions transposed as necessary.

6.4 Operator Norm

The operator norm (also called the induced norm) of a matrix (and more generally, any linear map) is defined as the following:

$$||A||_2 = \max_{\mathbf{x}\in\mathbb{R}^n} \frac{||A\mathbf{x}||_2}{||\mathbf{x}||_2}$$

Intuitively, $||A||_2$ refers to the maximum "stretchiness" of A (i.e., given an input vector, the maximum factor by which its length can be stretched). As it turns out, given the SVD of A,

$$||A||_2 = \sigma_{max} = \sigma_1$$

6.5 Condition Number

The condition number for a square, invertible A is defined as

$$\kappa = ||A|| \cdot ||A^{-1}|| = \frac{\sigma_{max}}{\sigma_{min}}$$

κ describes the sensitivity of the linear system $Ax = b$ to small perturbations. When κ is close to 1 (i.e., its smallest and largest singular values are not *too* different in magnitude), then the matrix is called *well-conditioned* — it is not extremely sensitive to changes in input. When κ is extremely large, the matrix is *ill-conditioned*, and it is sensitive to changes in inputs (it is extremely close to singular).

7 Determinants

Numerically, the determinant of a matrix A is the alternating sum of the product of its diagonal entries under row exchanges (permutations).

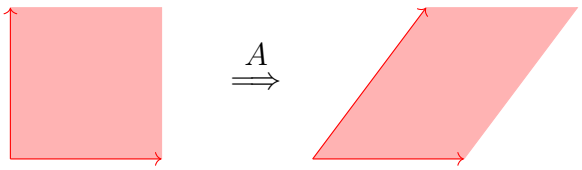
$$\det A = \sum (\text{product of diagonal entries})(-1)^n$$

where n is the number of row swaps performed. An odd number of row swaps will make the term negative, and an even number of row swaps will make it positive. Furthermore, a singular matrix will end up having a determinant of 0.

However, this view of the determinant (and subsequent computation) is not that satisfying — why is the determinant defined this way? Furthermore, computing a determinant in this fashion will take on the order of $O(n!)$ time, where n is the number of rows in A, which is extremely inefficient.

7.1 Geometric Interpretation

The determinant of a matrix (and more generally, a given map) is a number that quantifies how a volume in the input space $\mathbb{R}^m$ is changed/scaled after a map has been applied to it. In the case of matrices, because they are linear maps, the determinant is the volume of the hyper-parallelepiped defined by the *basis* of the map's column space. For example, if we start with the canonical basis vectors:



When we applied the linear map A, we can see that the same area got shifted or skewed in some kind of way. The factor by which the area has changed is $\det A$.

7.2 Properties

Determinants can be built up from ground properties in a way that is more useful than its definition as the sum of permutations.

Property 1

$$\det \underset{m\times m}{I} = 1$$

Property 2

The determinant switches signs whenever we swap any 2 columns.

Property 3

The determinant is *linear* in any single column. This has two consequences:

1. Multiplying one column by some constant α scales the entire determinant by α.
2. Adding some vector $\mathbf{b}$ to exactly one column of A generates another term:

$$\det \begin{bmatrix} | & | & & | \\ \mathbf{a}_1 & \mathbf{a}_2 + \mathbf{b} & \cdots & \mathbf{a}_n \\ | & | & & | \end{bmatrix} = \det A + \det \begin{bmatrix} | & | & & | \\ \mathbf{a}_1 & \mathbf{b} & \cdots & \mathbf{a}_n \\ | & | & & | \end{bmatrix}$$

Property 4

The determinant of A is 0 if 2 columns are equal.

We can prove this from property (2): if we swap the 2 equal columns, the determinant must be negated — however, the matrix has not materially changed at all. Therefore,

$$\det A = -\det A$$

The only number that fulfills the above criteria is 0, so $\det A = 0$.

Property 5

Column (elimination) operations do not change the determinant.

$$\det \begin{bmatrix} | & | & & | \\ \mathbf{a}_1 & \mathbf{a}_2 & \cdots & \mathbf{a}_n \\ | & | & & | \end{bmatrix} = \det \begin{bmatrix} | & | & & | \\ \mathbf{a}_1 & \mathbf{a}_2 - 4\mathbf{a}_1 & \cdots & \mathbf{a}_n \\ | & | & & | \end{bmatrix}$$

for example. This can be proven from properties (3) and (4) — we can separate the post-column operation matrix into two determinants, by linearity.

$$\det \begin{bmatrix} | & | & & | \\ \mathbf{a}_1 & \mathbf{a}_2 - 4\mathbf{a}_1 & \cdots & \mathbf{a}_n \\ | & | & & | \end{bmatrix} = \det \begin{bmatrix} | & | & \cdots & | \\ \mathbf{a}_1 & \mathbf{a}_2 & \cdots & \mathbf{a}_n \\ | & | & \cdots & | \end{bmatrix} - 4 \det \begin{bmatrix} | & | & \cdots & | \\ \mathbf{a}_1 & \mathbf{a}_1 & \cdots & \mathbf{a}_n \\ | & | & \cdots & | \end{bmatrix}$$

However, the second determinant must be equal to 0, because it has 2 identical columns.

Property 6

$$\det(\alpha A) = \alpha^m \det A$$

This can also be proven by (3), linearity.

Property 7

$$\det(AB) = \det A \cdot \det B$$

Property 8

The determinant of an upper-triangular matrix U (or a lower-triangular matrix L) is the product of its diagonal entries, or its pivots.

Property 9

For

$$A^T = P^{-1}LU$$

given n row swaps as a result of P,

$$\det A = (-1)^n \det U^T$$

Property 10

$$\det A = \det A^T$$

This property is especially consequential, as it indicates that all of the properties above that adhered to columns and column operations also apply to row operations. Doing row and column operations to A do the same things to its determinant.

Property 11

$$\det A^{-1} = (\det A)^{-1} = \frac{1}{\det A}$$

assuming that A is invertible in the first place. This gives us yet another indication that a singular matrix (i.e., a matrix that does not have an inverse) has a determinant of 0.

Property 12

$$\det Q = \pm 1$$

Recall that Q is an orthogonal matrix, which means that it merely represents a *rotation*, and not any sort of scaling. Therefore, from a geometric point of view, it makes sense that the magnitude of its determinant is just 1. The $\pm$ accounts for "flips" in rotations, as Q may represent either a proper or improper rotation.

Property 13

The determinant is invariant under a coordinate transform. Put differently,

$$\det(B^{-1}AB) = \det A$$

Even further, given the SVD of A:

$$A = U\Sigma V^T$$

the determinant of A is $\pm$ the product of its singular values (and the product of its eigenvalues, which will be covered in the next section).

8 Eigenvalues and Eigenvectors

An eigenvector-eigenvalue pair is one such pair $\lambda \in \mathbb{R}, \mathbf{x} \in \mathbb{R}^n$ with respect to a square matrix A such that

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

(in addition, $\mathbf{x} \neq \mathbf{0}$). A acts like a scalar when applied to $\mathbf{x}$ — this is useful because multiplication via scalar is a much easier property to deal with than multiplication by a matrix. In particular, given the following expression

$$A^n\mathbf{x}$$

if $\mathbf{x}$ is an eigenvalue of A, then we can just turn this into the simpler vector

$$\lambda^n \mathbf{x}$$

The determinant of a matrix is equal to the product of its eigenvectors, and its trace is the sum of its eigenvectors.

$$\begin{aligned} \det A &= \lambda_1 \cdot \lambda_2 \cdots \lambda_n \\ \mathrm{tr} A &= \lambda_1 + \lambda_2 + \cdots + \lambda_n \end{aligned}$$

8.1 Characteristic Polynomial

We can do some rearranging of the above definition.

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

$\mathbf{x}$ cannot be the $\mathbf{0}$ vector, as stated in the definition. Therefore, the matrix $A - \lambda I$ must have a null space of dimension ≥ 1, so it is singular.

$$\det(A - \lambda I) = 0$$

The above equation is known as the *characteristic polynomial* of A. The roots of this polynomial with respect to λ are the *eigenvalues* of A.

Given the eigenvalues we find from our characteristic polynomial, we can plug them back into $A - \lambda I$. The basis of the null space of this matrix is our corresponding eigenvector(s).

As a sidenote, our characteristic polynomial can have repeated roots — in practice, this rarely happens, but when it does, we can get some interesting behavior. In some cases, the nullity of $A - \lambda I$ can account for more than 1 basis vector, in which case, we have more than one independent eigenvector for the same eigenvalue. In other cases, we have a lack of independent eigenvectors for its dimension n, and we call A *degenerate*.

8.2 Eigenvectors as a basis

If we have n independent eigenvectors for a $n \times n$ matrix A, we can use the set of eigenvectors as a basis for our space $\mathbb{R}^n$ — that is, any arbitrary

vector $\mathbf{y}$ can be expressed as a linear combination of $\{\mathbf{x}_1,\ \mathbf{x}_2, \cdots,\ \mathbf{x}_n\}$. This is particularly useful because of the nice properties A exhibits with its eigenvectors.

$$\begin{aligned}\mathbf{y} &= c_1\mathbf{x}_1 + c_2\mathbf{x}_2 + \cdots + c_n\mathbf{x}_n \\ A\mathbf{y} &= A(c_1\mathbf{x}_1 + c_2\mathbf{x}_2 + \cdots + c_n\mathbf{x}_n) \\ &= c_1\lambda_1\mathbf{x}_1 + c_2\lambda_2\mathbf{x}_2 + \cdots + c_n\lambda_n\mathbf{x}_n\end{aligned}$$

By expressing $\mathbf{y}$ as a linear combination of eigenvectors, we've turned the linear map A into an independent scaling of each of its components, rather than something more complex.

8.3 Diagonalization

Given the eigenvectors and eigenvalues of our matrix A, we can express the equation $A\mathbf{x} = \lambda\mathbf{x}$ into one nice "package":

$$A\begin{bmatrix} | & | & & | \\ \mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_n \\ | & | & & | \end{bmatrix} = \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_n \end{bmatrix}\begin{bmatrix} | & | & & | \\ \mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_n \\ | & | & & | \end{bmatrix}$$

$$\begin{aligned} AX &= \Lambda X \\ &= X\Lambda\end{aligned}$$

X is our matrix of eigenvectors, and Λ is the diagonal matrix with eigenvalues as its entries (note that we can commute X with Λ because it is a diagonal matrix). If X is invertible (which is true when A is not degenerate), then we can further express A as

$$A = X\Lambda X^{-1}$$

This is the *diagonalization* of A. This form gives us another way of looking at the application of a linear map to a vector.

When we apply A to a vector $\mathbf{y}$, we can write this as the following:

$$A\mathbf{y} = X\Lambda(X^{-1}\mathbf{y})$$

Let's take a look at the $X^{-1}\mathbf{y}$ piece. This is equivalent to saying, "find a solution $\mathbf{c}$ such that $X\mathbf{c} = \mathbf{y}$." This vector $\mathbf{c}$ is the vector of coefficients

that we place in front of each column of X in order to find $\mathbf{y}$ — that is, the coefficients we need to write $\mathbf{y}$ in the basis of the eigenvectors of A.

$$\mathbf{y} = X\mathbf{c} = c_1\mathbf{x}_1 + c_2\mathbf{x}_2 + \cdots c_n\mathbf{x}_n$$

Applying Λ to this $\mathbf{c}$, then, has the effect of scaling each of these coefficients c_i by the corresponding λ_i.

$$\Lambda X^{-1}\mathbf{y} = \Lambda\mathbf{c} = \begin{bmatrix} \lambda_1 c_1 \\ \lambda_2 c_2 \\ \vdots \\ \lambda_n c_n \end{bmatrix}$$

Multiplying the matrix X by this resultant vector then gives us a linear combination of eigenvectors.

$$X \begin{bmatrix} \lambda_1 c_1 \\ \lambda_2 c_2 \\ \vdots \\ \lambda_n c_n \end{bmatrix} = \lambda_1 c_1 \mathbf{x}_1 + \lambda_2 c_2 \mathbf{x}_2 + \cdots + \lambda_n c_n \mathbf{x}_n$$

The diagonalization of a matrix is just a nice and concrete mathematical expression for the idea of placing $\mathbf{y}$ into the basis of eigenvectors and scaling each appropriately when A is applied to it.

8.4 Matrix Powers and the Exponential

The diagonalization lends itself nicely to the idea of taking matrix powers and exponentiating by a matrix.

$$\begin{aligned} A^n &= (X\Lambda X^{-1})(X\Lambda X^{-1})\cdots(X\Lambda X^{-1}) \\ &= X\Lambda(X^{-1}X)\Lambda(X^{-1}X)\cdots(X^{-1}X)\Lambda X^{-1} \\ &= X\Lambda^n X^{-1} \end{aligned}$$

Because Λ is a diagonal matrix, it is easy to exponentiate — we just exponentiate each of the entries along its diagonals. Therefore,

$$A^n = X \begin{bmatrix} \lambda_1^n & & & \\ & \lambda_2^n & & \\ & & \ddots & \\ & & & \lambda_n^n \end{bmatrix} X^{-1}$$

Note that, as n gets large, A^n has the tendency to be dominated by the eigenvector-eigenvalue with the largest magnitude λ_i.

What does it mean to exponentiate by a matrix? What does e^A do? Recall the Taylor (Maclaurin) series expansion of e^x:

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots$$

If we replace x with A (forgiving the notational abuse), we see that

$$e^A = I + A + \frac{A^2}{2!} + \frac{A^3}{3!} + \cdots$$

The diagonalization once again allows us to find a "better" computational form of this expression.

$$e^A = XX^{-1} + X\Lambda X^{-1} + \frac{1}{2!}X\Lambda^2 X^{-1} + \frac{1}{3!}X\Lambda^3 X^{-1} + \cdots$$

If we factor X and X^{-1} out,

$$e^A = X\left(I + \Lambda + \frac{1}{2!}\Lambda^2 + \frac{1}{3!}\Lambda^3 + \cdots\right)X^{-1}$$

However, the parenthesized series in the middle is reducible to just placing the exponentials of each of the eigenvectors in the diagonal of a matrix (seen when we expand and add).

$$e^A = X\begin{bmatrix} e^{\lambda_1} & & & \\ & e^{\lambda_2} & & \\ & & \ddots & \\ & & & e^{\lambda_n} \end{bmatrix}X^{-1}$$

This process is not just limited to the exponential e^A — any function with a Taylor series can be applied to the matrix following similar reasoning.

8.5 Matrix Similarity

Two matrices A and B are called "similar" ($A \sim B$) if they share the same eigenvalues, as they will therefore have the same determinant, trace, and characteristic polynomial. Equivalently, A is similar to B if

$$B = PAP^{-1}$$

for an arbitrary invertible matrix P. To see why, we write A as its diagonalization:

$$B = PX\Lambda X^{-1}P^{-1} = (PX)\Lambda(PX)^{-1}$$

B and A share the same Λ matrix, so they share the same eigenvalues. P is essentially a "change-of-basis" matrix between A and B.

8.6 Companion Matrices

Given a monic polynomial

$$p(x) = c_0 + c_1 x + \cdots + c_{n-1}x^{n-1} + x^n$$

there exists a square matrix for which $p(x)$ is its characteristic polynomial. By construction, one such matrix is

$$C(p) = \begin{bmatrix} 0 & 0 & \cdots & 0 & -c_0 \\ 1 & 0 & \cdots & 0 & -c_1 \\ 0 & 1 & \cdots & 0 & -c_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & -c_{n-1} \end{bmatrix}$$

$C(p)$ is called the companion matrix of the polynomial $p(x)$ — however, as eigenvalues are invariant under elementary row (and column) operations, any matrix reachable by performing such operations on $C(p)$ has the same eigenvalues. Furthermore, any matrix *similar* to $C(p)$ yields the same roots to $p(x)$

8.7 Degenerate Matrices

When A does not have n distinct eigenvectors that can form a basis of $\mathbb{R}^n$, it is called a degenerate matrix (or otherwise non-diagonalizable). In practice, a given random matrix A is almost always diagonalizable — degenerate matrices have to be designed specifically by tuning some parameter. It is more common for a matrix to be "almost" defective, which tends to happen for ill-conditioned matrices.

Matrices are defective when, for for some eigenvalue λ, its *algebraic multiplicity* μ_a in the characteristic polynomial $p(\lambda)$ is greater than its *geometric multiplicity* μ_g, the nullity of $A - \lambda I$. In other words, A is defective when

some repeated eigenvalue is not associated with enough eigenvectors to produce a full basis for $\mathbb{R}^n$. However, we *can* introduce another independent vector to fill this gap — we'd wish for it to have some kind of connection to our initial matrix A. This is the motivation behind Jordan vectors, otherwise known as generalized eigenvectors.

8.7.1 Jordan Vectors

For a given repeated eigenvalue λ_k with algebraic multiplicity $\mu_a > \mu_g$, the matrix

$$(A - \lambda_k I)$$

will not have a big enough nullspace to fill a proper basis for its ambient space. Given that it has μ_g independent vectors, we could just choose vectors independent from the rest of its eigen-basis to form a full basis, but this is unsatisfying — we want our chosen additional vectors to be specific and special to A in some way. Thus is the motivation for the concept of *generalized eigenvectors*, otherwise known as *Jordan vectors*.

A vector $\mathbf{x}_i^{(m)}$ is a generalized eigenvector of rank m corresponding to some λ_i iff

$$(A - \lambda I)^m \mathbf{x}_i^{(m)} = \mathbf{0}$$

and

$$(A - \lambda I)^{m-1} \mathbf{x}_i^{(m)} = \mathbf{x}_i^{(1)}$$

Of course, $\mathbf{x}_i^{(1)}$ is just $\mathbf{x}_i$, the regular eigenvector corresponding to λ_i.

Now, we can generate generalized eigenvectors through something known as a Jordan chain. If we need m generalized eigenvectors to fill the space, we can find a set $\{\mathbf{x}_i^{(1)},\ \mathbf{x}_i^{(2)}, \cdots,\ \mathbf{x}_i^{(m)}\}$ where

$$\begin{aligned}
(A - \lambda I)\mathbf{x}_i^{(2)} &= \mathbf{x}_i^{(1)} \\
(A - \lambda I)\mathbf{x}_i^{(3)} &= \mathbf{x}_i^{(2)} \\
&\vdots \\
(A - \lambda I)\mathbf{x}_i^{(m)} &= \mathbf{x}_i^{(m-1)}
\end{aligned}$$

In general, we see the pattern

$$\mathbf{x}_i^{(j)} = (A - \lambda I)\mathbf{x}_i^{(j+1)} = (A - \lambda I)^{m-j}\mathbf{x}_i^{(m)}$$

A set of n linearly independent generalized eigenvectors forms a canonical basis if it is constructed from a Jordan chain.

Generalized eigenvectors should behave almost the same as regular eigenvectors, with some modifications. For example, applying some arbitrary function (with a taylor series) $f(A)$ is not as simple as just looking at its eigenvalues. However, a nice form exists:

$$f(A)\mathbf{x}_i^{(j)} = f(\lambda_i)\mathbf{x}_i^{(j)} + f'(\lambda_i)\mathbf{x}_i^{(1)}$$

For matrix powers A^k,

$$A^k\mathbf{x}_i^{(j)} = \lambda_i^k\mathbf{x}_i^{(j)} + k\lambda_i^{k-1}\mathbf{x}_i^{(1)}$$

This is the basis for multiplying some $e^{\lambda t}$ by powers of t to get solutions to work in linear systems of differential equations with defective matrices.

8.7.2 Jordan Blocks

If A is defective, it cannot be diagonalized into the form $X\Lambda X^{-1}$, where Λ is a diagonal matrix. However, it *can* be factorized into some XJX^{-1}, where J is an *almost* diagonal matrix, in the Jordan Canonical form (in which it has mostly diagonal entries with 1s directly above the diagonals). For example, if I have a set of four independent (generalized or regular) eigenvectors, I can place them in a matrix X:

$$X = \begin{bmatrix} | & | & | & | \\ \mathbf{x}_1^{(1)} & \mathbf{x}_1^{(2)} & \mathbf{x}_2 & \mathbf{x}_3 \\ | & | & | & | \end{bmatrix}$$

When we take the product AX, we get the result

$$AX = \begin{bmatrix} \lambda_1\mathbf{x}_1^{(1)} \\ \lambda_1\mathbf{x}_1^{(2)} + \mathbf{x}_1^{(1)} \\ \lambda_2\mathbf{x}_2 \\ \lambda_3\mathbf{x}_3 \end{bmatrix} = \begin{bmatrix} \lambda_1 & 1 & 0 & 0 \\ 0 & \lambda_1 & 0 & 0 \\ 0 & 0 & \lambda_2 & 0 \\ 0 & 0 & 0 & \lambda_3 \end{bmatrix} X$$

Multiplying X by A has the effect of multiplying X by an *almost diagonal* matrix, contrasting with A in the non-defective case (in which case the matrix

is definitively diagonal). Blocks like

$$\begin{bmatrix} \lambda_1 & 1 \\ 0 & \lambda_1 \end{bmatrix}$$

are called *Jordan blocks*.

9 Special Matrix Structures

9.1 Markov Matrices

A (left) Markov matrix M is a non-negative square matrix whose columns sum to 1. All such matrices have eigenvalues of magnitude less than or equal to 1:

$$|\lambda_k| \leq 1$$

Markov matrices are used to represent the transitions between states of a Markov chain, given a vector of probabilities. When we apply M to a probability vector some n times, we begin to converge to a *steady state* when n is sufficiently large, corresponding to the eigenvectors with $\lambda_i = 1$.

$$M\mathbf{m}_i = 1 \cdot \mathbf{m}_i \iff \lim_{n\to\infty} M^n\mathbf{x} = \mathbf{m}_i$$

Markov matrices are used extensively when analyzing systems probabilistic processes, as well as algorithms such as Google PageRank.

9.2 Real-Symmetric and Hermitian Matrices

A real-symmetric matrix is some $A \in \mathbb{R}^{n\times n}$ such that

$$A = A^T$$

Such a matrix has some nice properties. In particular,

1. The eigenvalues of A are real.
2. A will always be diagonalizable.
3. The eigenvectors of A are orthogonal (and therefore, we can choose an orthonormal basis that functions as the set of eigenvectors of A)

The diagonalization of A can therefore be expressed as some

$$A = Q\Lambda Q^T$$

When we start discussing complex matrices $A \in \mathbb{C}^{n\times n}$, we must reformulate our definitions of taking dot products.

For some $\mathbf{x} \in \mathbb{C}^n,\ \mathbf{y} \in \mathbb{C}^n$, the dot product as it's defined ($\mathbf{x}^T\mathbf{y}$) does not give us the properties we want it to. For one, because of the complex entries and subsequent issues with signs, $\mathbf{x}^T\mathbf{y}$ may not equal 0, even if $\mathbf{x} \perp \mathbf{y}$. A better definition is the Hermitian adjoint $\mathbf{x}^\dagger$ (also notated $\mathbf{x}^H$), the conjugate transpose

$$\mathbf{x}^\dagger = \overline{\mathbf{x}^T}$$

in which we take the *complex conjugate* of every entry in $\mathbf{x}$ and then transpose it (the same goes for any matrix A). The reason works has to do with the definition of the modulus of some $z \in \mathbb{C}$ being defined as

$$|z|^2 = z\overline{z}$$

In complex space, the dot product between two vectors is $\mathbf{x}^\dagger\mathbf{y}$. Note that the real transpose is simply a special case of the Hermitian adjoint (put differently, the Hermitian adjoint is a generalization of the transpose. Anywhere we've used the transpose for real matrices, we can just replace it with the Hermitian adjoint in the general complex case).

A matrix is Hermitian if it is equal to its conjugate transpose.

$$A = A^\dagger$$

The generalization of the orthogonal matrix Q, similarly, is called a unitary matrix that follows the criteria

$$Q^{-1} = Q^\dagger$$

9.3 Positive (Semi-) Definite Matrices

A symmetric positive definite (SPD) matrix A is one that satisfies:

1. $A = B^TB$, where B is full-column rank ($N(B) = \{\mathbf{0}\}$)
2. The quadratic form $\mathbf{x}^TA\mathbf{x} > 0\ \forall \mathbf{x} \in \mathbb{R}^n$

3. All of its eigenvalues $\lambda_k > 0$

As it turns out, all three of these properties are equivalent (that is, if one of them is true, then all of them are true). Note that all inequalities above are "*strictly greater than*" statements. If we instead extend the definitions to include 0, we get a positive semi-definite (PSD) matrix.

A PSD matrix satisfies

1. $A = B^T B$
2. The quadratic form $\mathbf{x}^T A\mathbf{x} \geq 0 \ \forall \mathbf{x} \in \mathbb{R}^n$
3. All of its eigenvalues $\lambda_k \geq 0$

9.3.1 Connection to the SVD

Given a matrix B, we can find its SVD:

$$B = U\Sigma V^T$$

If we take $B^T B$, we see that

$$A = B^T B = V\Sigma^T U^T U\Sigma V^T = V\Sigma^2 V^T = Q\Lambda Q^{-1}$$

The eigenvalues of A (which, remember, is positive semi-definite) are therefore the squares of the singular values of B.

$$\Lambda_A = \Sigma^2 = \begin{bmatrix} \sigma_1^2 & & & & & \\ & \sigma_2^2 & & & & \\ & & \ddots & & & \\ & & & \sigma_r^2 & & \\ & & & & 0 & \\ & & & & & \ddots \end{bmatrix}$$

For any arbitrary matrix A, it is possible to find its singular value σ_k by finding the square root of the corresponding eigenvalue of $A^T A$ ($\sqrt{\lambda_k}$).

9.4 The Jacobian Matrix

Given some multivariable vector-valued function $\mathbf{F}(\mathbf{x}) : \mathbb{R}^n \mapsto \mathbb{R}^m$, we define the Jacobian matrix J as the $m \times n$ matrix of all first-order partial derivatives of $\mathbf{F}$.

$$J = \begin{bmatrix} \frac{\partial \mathbf{F}}{\partial x_1} & \frac{\partial \mathbf{F}}{\partial x_2} & \cdots & \frac{\partial \mathbf{F}}{\partial x_n} \end{bmatrix} = \begin{bmatrix} \frac{\partial F_1}{\partial x_1} & \frac{\partial F_1}{\partial x_2} & \cdots & \frac{\partial F_1}{\partial x_n} \\ \frac{\partial F_2}{\partial x_1} & \frac{\partial F_2}{\partial x_2} & \cdots & \frac{\partial F_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial F_m}{\partial x_m} & \frac{\partial F_1}{\partial x_2} & \cdots & \frac{\partial F_m}{\partial x_n} \end{bmatrix}$$

This is the generalization of the function that the derivative serves in single variable calculus — we can linearize functions by using the Jacobian.

$$\mathbf{F}(\mathbf{x} + \delta\mathbf{x}) \approx \mathbf{F}(\mathbf{x}) + J\Big|_{\mathbf{x}} \cdot (\delta\mathbf{x})$$

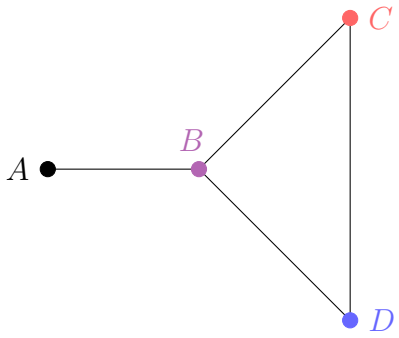
The *gradient* $\nabla\mathbf{F}$ is the transpose of the Jacobian.

$$J = (\nabla\mathbf{F})^T$$

10 Applications

10.1 Graphs

Suppose we have an unweighted, undirected graph, like the diagram below:

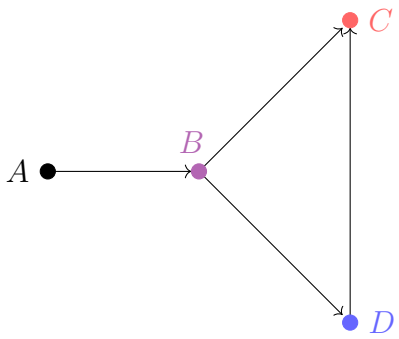


An *adjacency matrix* records connections between different vertices. We can index a matrix with the names of the vertices, like so:

$$A = \begin{bmatrix} AA & AB & AC & AD \\ BA & BB & BC & BD \\ CA & CB & CC & CD \\ DA & DB & DC & DD \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

Note that a vertex isn't considered connected to itself unless there is an explicit singular loop-edge that does so. The adjacency matrix is symmetric, which makes perfect sense — because edges aren't directed, any edge from some vertex i to a vertex j will also lead j back to i.

If we made the graph *directed*, we'd need to take into account this ordering:



$$\begin{bmatrix} AA & AB & AC & AD \\ BA & BB & BC & BD \\ CA & CB & CC & CD \\ DA & DB & DC & DD \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Here, there exists a path from A to B, but not vice versa, and so on.

When we exponentiate an adjacency matrix n times (that is, multiply it with itself n times), the resulting entries contain the number of walks of length n there are from each vertex to another. For example,

$$A^2 = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 3 & 1 & 1 \\ 1 & 1 & 2 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

There is thus 1 path of length 2 from A back to itself, 0 paths of length 2 from A to B, 1 path of length 1 from A to C, and 1 path of length 1 from A to D. This same procedure applies for directed graphs as well.

10.2 Regression and Fitting

If we have an overdetermined system $A\mathbf{x} = \mathbf{b}$, with $n < m$, $r = n$ and $C(A)$ does not cover the full ambient space $\mathbb{R}^m$, we cannot find a solution $\mathbf{x}$

if $\mathbf{b} \notin C(A)$. However, it is useful to find the "next best thing" — that is, the vector $\mathbf{x}_*$ that brings $A\mathbf{x}_*$ the closest to $\mathbf{b}$. This is exactly a projection problem, in that we are finding the projection of $\mathbf{b}$ onto $C(A)$. As per section 5.5,

$$A\mathbf{x}_* = A(A^TA)^{-1}A^T\mathbf{b}$$

It turns out that the above $\mathbf{x}_*$ minimizes the quantity

$$||A\mathbf{x} - \mathbf{b}||^2$$

which is why it is called the least-squares approach.

One application of projecting $\mathbf{b}$ is least-squares linear regression. Given a set of datapoints $(x_0, y_0), (x_1, y_1), \cdots, (x_m, y_m)$, we want to find a polynomial (or, most commonly, a line) that goes fits these datapoints best. That is, we want to find a polynomial

$$c_0 + c_1x + c_2x^2 + c_3x^3 + \cdots + c_nx^n$$

(for an arbitrary degree d) that best fits our data. Given some degree n, we can put our datapoints into the following matrix system

$$\begin{bmatrix} 1 & x_0 & \cdots & x_0^n \\ 1 & x_1 & \cdots & x_1^n \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_m & \cdots & x_m^n \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_m \end{bmatrix}$$

The matrix on the left is known as the *Vandermonde* matrix. Once we project $\mathbf{y}$ onto the column space of the matrix containing the entries of the powers of x, we can find our column vector of coefficients $\mathbf{c}$ that minimizes the least-squares residuals of our fitted polynomial to the actual data.

If, instead, we had an underdetermined system with $n > m$, $r = m$, then solutions for $A\mathbf{x} = \mathbf{b}$ *always* exist — we just have an infinite number of solutions for every $\mathbf{b}$. Thus, our goal is to "choose" one of these solutions — often, we want to find the minimum-norm solution, which turns out to be equal to

$$\mathbf{x}_* = A^T(AA^T)^{-1}\mathbf{b}$$

Note that

$$A\mathbf{x}_* = AA^T(AA^T)^{-1}\mathbf{b} = \mathbf{b}$$

which makes sense — $A\mathbf{x} = \mathbf{b}$ will always be solvable, because A is full row rank.

10.2.1 Regularization

Regularization (more specifically, ridge regularization or Tikhonov regularization) combines the two approaches outlined above, especially for rank-deficient matrices, in which a solution either may not exist or, if it does, has infinite solutions. In particular, it asks for a solution that minimizes the function

$$T(\mathbf{x}) := ||A\mathbf{x} - \mathbf{b}||^2 + \lambda||\mathbf{x}||^2$$

for $\lambda > 0$.

10.3 Statistical Interpretations

Linear algebra provides a nice interpretation of some statistical quantities and processes. Given m samples $x_1, x_2, \cdots, x_m$, we can place them into a vector in $\mathbb{R}^m$:

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix}$$

We define the vector $\mathbf{o}$ as the m dimensional vector of ones.

$$\mathbf{o} \in \mathbb{R}^m, \ \mathbf{o} = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}$$

10.3.1 Mean

The mean μ can be defined as

$$\mu = \frac{1}{m}\mathbf{o}^T\mathbf{x}$$

Furthermore,

$$m = \mathbf{o}^T\mathbf{o}$$
$$\mu = \frac{\mathbf{o}^T\mathbf{x}}{\mathbf{o}^T\mathbf{o}}$$

10.3.2 Variance

Now, let some vector $\mathbf{a}$ be the pointwise subtraction of μ from each element in $\mathbf{x}$.

$$\begin{aligned}\mathbf{a} &= \mathbf{x} - \mathbf{o}\mu \\ &= \mathbf{x} - \frac{\mathbf{o}\mathbf{o}^T\mathbf{x}}{\mathbf{o}^T\mathbf{o}} \\ &= \left(I - \frac{\mathbf{o}\mathbf{o}^T}{\mathbf{o}^T\mathbf{o}}\right)\mathbf{x}\end{aligned}$$

We now define the projection $P = I - \frac{\mathbf{o}\mathbf{o}^T}{\mathbf{o}^T\mathbf{o}}$, which is the projection operator onto the orthogonal complement of $\mathbf{o}$. Therefore, the variance σ^2 of $\mathbf{x}$ can be defined as

$$\sigma^2 = \frac{1}{m-1}||P\mathbf{x}||^2 = \frac{1}{m-1}\mathbf{x}^T P\mathbf{x}$$

We divide by $m-1$, rather than m, due to a concept known as Bessel's correction. One way to go about intuiting this is that we have already "used up" one dimension of our full space in defining $\mathbf{o}$ — P, as the projection operator onto the orthogonal complement of $\mathbf{o}$, is an operator into $m-1$ dimensions.

10.3.3 Covariance

Why are we defining things in this way? For one, it allows us to generalize to higher-dimensional data. I can use the same projection operator P to define something like the variance (now called the *covariance matrix* S^2) to a matrix of data $X \in \mathbb{R}^{m\times n}$ (in which we have m samples, each with n features).

$$S^2 = \frac{1}{m-1}(PX)^T(PX)$$

where each entry S^2_{ij} is the *covariance* between two features, defined by the ith and jth columns. Intuitively, the covariance is a measure of how two features "move" together. If the covariance between two features is positive, they "move" together, and if it is negative, they move opposite to each other. If it is 0, that means that the two features are uncorrelated with each other.

Pearson's correlation coefficient ρ between two variables can be formulated in terms of the covariance:

$$\rho(x,y) = \frac{\mathrm{Cov}(x,y)}{\sqrt{\mathrm{Var}(x)\mathrm{Var}(y)}} = \frac{(P\mathbf{x})^T(P\mathbf{y})}{||P\mathbf{x}||\ ||P\mathbf{y}||}$$

10.3.4 Principal Component Analysis

Given the data

$$A = PX$$

we can find its (compact) SVD

$$A = \hat{U}\hat{\Sigma}\hat{V}^T$$

We can write the rows of A in the basis of $\hat{V}$ (effectively rotating the data, but not modifying its lengths) by rightiplying it by $\hat{V}$. We call this new matrix B.

$$B = AV = \hat{U}\hat{\Sigma}\hat{V}\hat{V}^T = \hat{U}\hat{\Sigma}$$

This B contains the data points in the $\hat{V}$ basis. We can then find the covariance of B:

$$\text{Cov}B = \frac{1}{m-1}B^TB = \frac{1}{m-1}\hat{\Sigma}^2$$

which is a diagonal matrix:

$$\text{Cov}B = \begin{bmatrix} \frac{\sigma_1^2}{m-1} & & & \\ & \frac{\sigma_2^2}{m-1} & & \\ & & \ddots & \\ & & & \frac{\sigma_r^2}{m-1} \end{bmatrix}$$

The measurements in the $\hat{V}$ bases are uncorrelated, which makes it a nice basis to use to quantify the relative "importance" of each feature in the dataset. In addition, the direction of the *biggest variation* of the data is the vector in $\hat{V}$ that corresponds to the largest value on the diagonal of the $\hat{\Sigma}^2$, and furthermore, the value that it corresponds with – $\frac{\sigma_1^2}{m-1}$ – is the variance in that direction. This is the *principal component* of A.

10.4 Linear Recurrences

Given some linear recurrence that depends on more than one previous term (for example, the Fibonacci sequence, or more generally, the Lucas sequences):

$$f_n = f_{n-1} + f_{n-2}$$

we may analyze the growth of this sequence by utilizing matrix powers. We can place the above expression into matrix form:

$$\begin{bmatrix} f_{n+1} \\ f_n \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} f_n \\ f_{n-1} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \left(\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} f_{n-1} \\ f_{n-2} \end{bmatrix} \right) = \cdots$$

Given some initial conditions, we see that

$$\begin{bmatrix} f_{n+1} \\ f_n \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^n \begin{bmatrix} f_1 \\ f_0 \end{bmatrix}$$

For the Fibonacci sequence,

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

Therefore, analyzing the growth of this sequence is akin to analyzing the exponentiation of

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

The characteristic equation of this matrix is

$$\lambda^2 - \lambda - 1 = 0$$

and eigenvalues of this matrix are

$$\lambda_1 = \frac{1+\sqrt{5}}{2},\ \lambda_2 = \frac{1-\sqrt{5}}{2}$$

Does λ_1 look familiar? This is the golden ratio!

As per the properties of matrix exponentiation, as n grows large, the behavior of F^n is dominated by the largest magnitude eigenvalue, which, in this case, is $\lambda_1 = \varphi$.

$$\begin{bmatrix} f_{n+1} \\ f_n \end{bmatrix} = F^n \begin{bmatrix} f_1 \\ f_0 \end{bmatrix} \approx \lambda_1 \begin{bmatrix} f_n \\ f_{n-1} \end{bmatrix} = \lambda_1^n \begin{bmatrix} f_1 \\ f_0 \end{bmatrix}$$

We can check this result numerically — as we compute more and more Fibonacci numbers, the ratio between successive Fibonacci numbers converges to φ.

11 Optimization

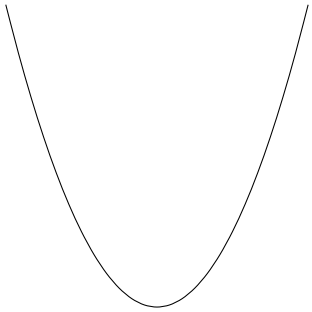
11.1 Quadratic Programming

A quadratic function is a function of the form

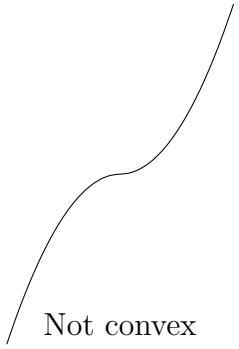
$$f(\mathbf{x}) = \mathbf{x}^T A\mathbf{x} - 2\mathbf{b}^T\mathbf{x} + c$$

where $\mathbf{b}, \mathbf{x} \in \mathbb{R}^n$ and $c \in \mathbb{R}$. If A is positive (semi-) definite, then $f(x)$ is a *convex* function (a bowl opening upwards). It therefore has some minimum point. If A is strictly positive-definite, this minimum is *unique*.

Formally, $f(x)$ is convex if the function is always less than or equal to a line connecting any two points on the function.



Convex



Not convex

Convex functions lend themselves nicely to minimization. For example, suppose we have some problem where we wish to minimize

$$\min_{\mathbf{x}} ||B\mathbf{x} - \mathbf{c}||^2$$

We can examine $||B\mathbf{x} - \mathbf{c}||^2$ further.

$$\begin{aligned} ||B\mathbf{x} - \mathbf{c}||^2 &= (B\mathbf{x} - \mathbf{c})^T(B\mathbf{x} - \mathbf{c}) \\ &= \mathbf{x}B^TB\mathbf{x} - 2\mathbf{c}^TB\mathbf{x} + \mathbf{c}^T\mathbf{c} \end{aligned}$$

This is a convex function! In particular,

$$\begin{aligned} A &= B^T B \\ \mathbf{b} &= B^T \mathbf{c} \\ c &= \mathbf{c}^T \mathbf{c} \\ ||B\mathbf{x} - \mathbf{c}||^2 &= \mathbf{x}^T A \mathbf{x} - 2\mathbf{b}^T \mathbf{x} + c \\ &= f(x) \end{aligned}$$

Minimizing $||B\mathbf{x} - \mathbf{c}||^2$ is equivalent to minimizing $f(x)$. Furthermore, from the normal equations with least-squares regression, we know that the minimum $\mathbf{x}$ solves

$$\begin{aligned} B^T B\mathbf{x} &= B^T \mathbf{c} \\ \implies A\mathbf{x} &= \mathbf{b} \end{aligned}$$

So solving $A\mathbf{x} = \mathbf{b}$ for some positive definite matrix A is also equivalent to minimizing the convex quadratic.

11.2 Gradient Descent

Another way to look at the QP problem is to consider the gradient of $f(\mathbf{x})$ and find where it is equal to 0 (as per multivariable calculus). If we express the function

$$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T A \mathbf{x} - \mathbf{b}^T \mathbf{x} + c$$

component-wise, with expanded dot products and find the derivative with respect to each x_k, we find that

$$\nabla f(\mathbf{x}) = A\mathbf{x} - \mathbf{b}$$

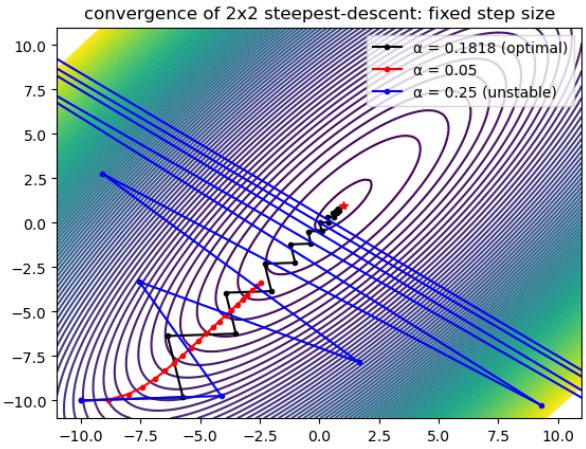
If we set this to 0, we are solving the equation

$$A\mathbf{x} - \mathbf{b} = \mathbf{0} \implies A\mathbf{x} = \mathbf{b}$$

which is the same conclusion we arrived at when we exploited the fact that A was positive-definite in the previous section.

Importantly, for *any* function $f(\mathbf{x})$, $\nabla f(\mathbf{x}_0)$ indicates the direction of greatest *ascent* at some point $\mathbf{x}_0$. $\mathbf{d} = -\nabla f(\mathbf{x}_0)$ indicates the direction of greatest descent. In minimizing some function, then, if we always take a step

in the direction of greatest descent, we are sure to reach a minimum at *some* point — this is the idea behind gradient descent. Most of the work goes behind deciding how large of a step to take. If we take steps that are too large, we run in these step sizesthe risk of "stepping over" the minimum and being unable to find it. If we take steps that are too small, it will take our algorithm forever to converge on a minimum. The differences that step size can make is summarized in the image below (courtesy of Prof. Johnson):



11.2.1 Exact Line Minimization

One way to determine α is to have it depend dynamically upon the position and gradient at a certain point, rather than having it be a fixed number. One such way of going about this is *exact line minimization*, in which, given the direction of the gradient $\mathbf{d}$ and a starting point $\mathbf{x}$, searches along the direction of the gradient until a point where we would no longer be going "downhill" is found.

Formally, we would be finding a new position

$$\mathbf{x} + \alpha\mathbf{d}$$

and our goal is to minimize the function f over α to find the locally optimal step size $\alpha_\star$. In addition,

$$\nabla f\big|_{\mathbf{x}+\alpha_\star\mathbf{d}} \perp \mathbf{d}$$

as, if the gradient at the new position were not perpendicular to $\mathbf{d}$, that would mean that we could still step downhill in some sort of way in the $\mathbf{d}$

direction. Therefore,

$$\begin{aligned}
\mathbf{d}^T \nabla f\big|_{\mathbf{x}+\alpha_\star \mathbf{d}} &= 0 \\
\mathbf{d}^T (A(\mathbf{x} + \alpha_\star \mathbf{d}) - \mathbf{b}) &= 0 \\
\mathbf{d}^T (A\mathbf{x} - \mathbf{b}) + \alpha_\star \mathbf{d}^T A\mathbf{d} &= 0 \\
\alpha_\star &= \boxed{\frac{-\mathbf{d}^T \nabla f\big|_{\mathbf{x}}}{\mathbf{d}^T A\mathbf{d}}}
\end{aligned}$$

Because A is positive-definite, the denominator is always strictly greater than 0, so we will never be dividing by 0. At every step, this $\alpha_\star$ is the "optimal step size."

11.2.2 Fixed learning rate

If, instead of computing some $\alpha_\star$ at every step, we decide on a fixed step size, we decrease the amount of computation needed but we run a greater risk of either converging too slow or diverging completely, as per the diagram. We must therefore find a good learning rate α that is neither too large, nor too small — it turns out that, for quadratic optimization, we can derive such an α given the properties of A.

For a constant α and a direction of steepest descent

$$-\nabla f = \mathbf{b} - A\mathbf{x}$$

on each step, we are setting our new position to be

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha(\mathbf{b} - A\mathbf{x}_k)$$

Now, let's consider the residual error $\mathbf{r} = \mathbf{b} - A\mathbf{x}$.

$$\mathbf{r}_{k+1} = \mathbf{b} - A(\mathbf{x}_k - \alpha \mathbf{r}_k) = \mathbf{r}_k - \alpha A\mathbf{r}_k = (I - \alpha A)\mathbf{r}_k$$

Every step, we are multiplying our residual by the matrix $I - \alpha A$ — this is a matrix powers problem! If we want our residuals to converge to 0, then, the eigenvalues of $I - \alpha A$ must be less than 1. Given that the eigenvalues of A are some λ_k (positive, since we know that A is positive-definite), then the eigenvalues of $I - \alpha A$ are

$$1 - \alpha\lambda_k$$

Therefore,

$$\begin{aligned} |1-\alpha\lambda_k| &< 1 \\ -1 < 1-\alpha\lambda_k &< 1 \\ -2 < -\alpha\lambda_k &< 0 \\ 0 < \alpha &< \frac{2}{\lambda_{max}} \end{aligned}$$

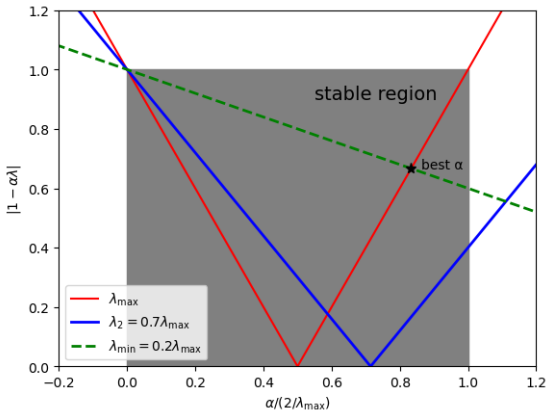
In addition, we want the largest magnitude of $|1-\alpha\lambda_k|$ to be as small as possible, since that determines our rate of convergence. This is given at the point

$$\alpha = \frac{2}{\lambda_{max}+\lambda_{min}}$$

and the biggest magnitude eigenvalue is

$$\alpha\lambda_{max} - 1 = \frac{\lambda_{max}-\lambda_{min}}{\lambda_{max}+\lambda_{min}} = \frac{\kappa-1}{\kappa+1}$$

The plot of stable eigenvalues is seen in the figure below (courtesy of Prof. Johnson)



In the limit case of α being extremely close to 0, we are actually taking what seem like infinitesimally small steps — akin to solving a differential equation.

11.2.3 Accelerated Gradient Descent

A perennial problem with gradient descent is the "zig-zagging" effect, where we move up and down, sometimes in the direction that we just came from,

due to the local gradient. Furthermore, we may get stuck in a local minimum that is not the true global minimum we want to find.

We may wish to introduce some sort of "memory" in choosing our step, so that we are inclined to move *away* from our previous steps, rather than bouncing back. This is the intuition behind accelerated gradient descent (sometimes called "heavy ball algorithms"), which introduces a momentum term to each step to weight our new point in a further-downhill direction. This is summarized in the equation

$$\mathbf{x}_{k+1} = \underbrace{\mathbf{x}_k - \alpha \nabla f\big|_{\mathbf{x}_k}}_{\text{steepest descent}} + \underbrace{\beta(\mathbf{x}_k - \mathbf{x}_{k-1})}_{\text{momentum}}$$

where β is some parameter ≥ 0 that determines the "importance" of our momentum term.

To analyze this, we can look at the residuals:

$$\begin{aligned}
\mathbf{r}_{k+1} &= \mathbf{b} - A(\mathbf{x}_k + \alpha \mathbf{r}_k + \beta(\mathbf{x}_k - \mathbf{x}_{k-1})) \\
&= (I - \alpha A)\mathbf{r}_k + \beta(\mathbf{r}_k - \mathbf{r}_{k-1}) \\
\begin{bmatrix} \mathbf{r}_{k+1} \\ \mathbf{r}_k \end{bmatrix} &= \begin{bmatrix} (1+\beta)I - \alpha A & -\beta I \\ I & 0 \end{bmatrix} \begin{bmatrix} \mathbf{r}_k \\ \mathbf{r}_{k-1} \end{bmatrix}
\end{aligned}$$

By analyzing the eigenvalues of the above $2m \times 2m$ block matrix (which is equivalent to analyzing each 2×2 matrix produced by plugging in the eigenvalues of A), we find that the optimal parameters α and β are

$$\begin{aligned}
\alpha &= \left(\frac{2}{\sqrt{\lambda_{max}} + \sqrt{\lambda_{min}}} \right)^2 \\
\beta &= \left(\frac{\sqrt{\lambda_{max}} - \sqrt{\lambda_{min}}}{\sqrt{\lambda_{max}} + \sqrt{\lambda_{min}}} \right)^2
\end{aligned}$$

with a largest magnitude eigenvalue of

$$\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1}$$

This is roughly a quadratic speedup over gradient descent without a momentum term.

11.3 Constraints

Recall the problem of unconstrained optimization: given some objective function $f(\mathbf{x})$, we find some $\mathbf{x}_\star$ that minimizes the value of f over its domain.

$$\mathbf{x}_\star = \min_{\mathbf{x} \in \mathbb{R}^n} f(\mathbf{x})$$

The necessary condition for optimality (though not sufficient) is that the gradient of f at $\mathbf{x}_\star$ must be zero:

$$\nabla f\big|_{\mathbf{x}_\star} = \mathbf{0}$$

Now, if there exist some boundary conditions or constraints that $\mathbf{x}_\star$ must fulfill (e.g. it must lie inside some particular subset of the domain), the problem becomes more complicated. There are two types of constraints: equality constraints (in which some vector-valued constraint function $\mathbf{h}(\mathbf{x}) = \mathbf{0}$) and inequality constraints (in which some vector-valued constraint function $\mathbf{g}(\mathbf{x}) \preceq \mathbf{0}$). The set of points that fulfill the stated constraints is called the *feasible set*.

11.3.1 Equality Constraints

Given a series of equality constraints $h_i(\mathbf{x}) = 0$ for $i = 1, \cdots, p$, the necessary condition for optimality is that the gradient of the objective function ∇f lies in the span of the gradients of each of the constraints ∇h_i.

$$\nabla f(\mathbf{x}_\star) + \sum_{i=1}^{p} \nu_i \nabla h_i(\mathbf{x}_\star) = \mathbf{0}$$

Recall that the gradient of the vector-valued formulation of our equality constraints $\mathbf{h}(\mathbf{x})$ is the transpose of its Jacobian; therefore, an equivalent way of phrasing these conditions is

$$\nabla f(\mathbf{x}_\star) + J^T \vec{\nu} = \mathbf{0}$$

$\nu_1, \cdots, \nu_p$ are called the *Lagrange Multipliers*. Furthermore, we can define the Lagrangian function

$$\mathcal{L}(\mathbf{x}, \vec{\nu}) = f(\mathbf{x}) + \mathbf{h}(\mathbf{x})^T \vec{\nu}$$

Using the Lagrangian, we can package the necessary conditions nicely:

$$\begin{aligned} \nabla_{\mathbf{x}} \mathcal{L}(\mathbf{x}_\star, \vec{\nu}_\star) &= \mathbf{0} \\ \mathbf{h}(\mathbf{x}) &= \mathbf{0} \end{aligned}$$

A possible pitfall arises when the gradients of distinct equality conditions are linearly dependent at $\mathbf{x}_\star$. To address this, we just add a regularity condition that the gradients of each of the active constraints are linearly independent, known as the Linear Independence Constraint Qualification (LICQ).

11.3.2 Inequality Constraints

Given an objective function $f(\mathbf{x})$ subject to inequality constraints $\mathbf{g}(\mathbf{x}) \preceq \mathbf{0}$, we conceive of inequality constraints to be some "wall" (the boundary of the region fulfilling the inequality, or the feasible region). There are two cases: active and inactive constraints.

Inactive constraints

If there exists some point strictly inside of the feasible region (i.e. not on the "wall") where $\nabla f = \mathbf{0}$, we do not even need the inequality constraint to be in place — we've found the minimum regardless of whether or not the constraint exists. In this case, the constraint is considered *inactive*. Hence, in this case, the corresponding multiplier $\lambda = 0$.

Active Constraints

If instead the constraint is active (and the optimal $\mathbf{x}_\star$ lies on the boundary of the constraint, and $g_i(\mathbf{x}_\star) = 0$), we know that the gradient of the constraints must be pushing us *outwards* directly towards the infeasible region. In this case, the inequality constraints behave like equality constraints, since we are constrained to lie on the boundary. Therefore, just like the equality case, the gradient of the objective function f must lie in the span of the gradients of the constraint functions ∇g_i.

$$\nabla f + \nabla g_i \lambda_i = 0$$

11.3.3 KKT Conditions

The Karush-Kuhn-Tucker (KKT) conditions are a way of packaging the necessary conditions for optimality using the Lagrangian function. They are considered the generalization of the method of Lagrange Multipliers to multidimensional equality constraints and inequality constraints.

More specifically, given an objective function $f(\mathbf{x})$ subject to

$$\mathbf{h}(\mathbf{x}) = \mathbf{0},\ \mathbf{g}(\mathbf{x}) \preceq \mathbf{0}$$

we define the Lagrangian $\mathcal{L}$ such that

$$\mathcal{L}(\mathbf{x}, \vec{\nu}, \vec{\lambda}) = f(\mathbf{x}) + \mathbf{h}^T(\mathbf{x})\vec{\nu} + \mathbf{g}(\mathbf{x})^T\vec{\lambda}$$

The KKT conditions state the following:

$$\begin{aligned} \nabla_{\mathbf{x}}\mathcal{L}\big|_{\mathbf{x}_\star} &= \mathbf{0} \\ \text{Primal Feasibility: } \mathbf{g}(\mathbf{x}_\star) &\preceq \mathbf{0} \\ \mathbf{h}(\mathbf{x}_\star) &= \mathbf{0} \\ \text{Dual Feasibility: } \vec{\lambda}_\star &\succeq \mathbf{0} \\ \text{Complementary Slackness: } \mathbf{g}(\mathbf{x}_\star)^T\vec{\lambda}_\star &= \mathbf{0} \end{aligned}$$

The complementary slackness condition ensures that the inequality constraints are either active ($g_i(\mathbf{x}_\star) = 0$) with $\lambda_i \geq 0$, or inactive with $\lambda_i = 0$.