1 Introduction

2 Mathematical modelling and why we need computers

2.1 Why we need to be clever about our computing

Mathematical problems can be classed into problems that are *well-posed* and *ill-posed.* A problem is *well-posed* if

1. A solution exists
2. the solution is unique
3. A small change in the initial condition induces only a small change in the solution

A problem that is not well-posed is ill-posed.

We are interested in the last criterion which can be termed *sensitivity.* Suppose our problem has inputs x and has a solution (or output) y. An *insensitive* or well-conditioned problem is when a change in x causes a change in y that is of similar relative size. A *sensitive* or *ill-conditioned* problem is one in which the change in solution/output can be large relative the the change in input.

Based on this idea, define the *condition number* by

$$cond = \frac{|\text{relative change in solution}|}{|\text{relative change in input data}|} = \left|\frac{\Delta y/y}{\Delta x/x}\right|.$$

Thus a problem is *ill-conditioned* is $cond \gg 1$.

Example: what is the condition number when we evaluate a function $y = f(x)$ at an approximation of x, $\hat{x} = x + \Delta x$, rather than at the true value x?

Solution:

$$cond = \left|\frac{\Delta y/y}{\Delta x/x}\right| = \left|\frac{f(x+\Delta x) - f(x)/f(x)}{\Delta x/x}\right| = \left|\frac{f(x+\Delta x) - f(x)}{\Delta x}\frac{x}{f(x)}\right| \approx \left|\frac{xf'(x)}{f(x)}\right|.$$

So, depending on the function f and the input x, we could get very large condition numbers. □

Example: What is the condition number for the functions $f(x) = x^n$ and $f(x) = e^x$?

Solution: From above, the condition number is

$$cond \approx \left|\frac{xf'(x)}{f(x)}\right|.$$

When $f(x) = x^n$, $f'(x) = nx^{n-1}$, so

$$cond = \left|\frac{xf'(x)}{f(x)}\right| = \left|\frac{x.nx^{n-1}}{x^n}\right| = \left|\frac{nx^n}{x^n}\right| = |n|.$$

So as the degree of the polynomial increases, the problem becomes increasingly ill-conditioned.

Similarly, when $f(x) = e^x$, $f'(x) = e^x$ so

$$cond = \left| \frac{xf'(x)}{f(x)} \right| = \left| \frac{x.e^x}{e^x} \right| = |x|.$$

In this case, the condition number depends on the input argument, x. If x is very large, the problem can be considered ill-conditioned. $\square$

What does this mean for computation? In nearly all our computations, we will replace exact formulations with approximations. For example, we approximate continuous quantities with discrete quantities, we are limited in size by floating point representation of numbers which often necessitates rounding or truncation and introduces (hopefully) small errors. If we are not careful, these error will be amplified within an ill-conditioned problem an produce inaccurate results.

This leads us to consider the *stability* and *accuracy* of our algorithms.

An algorithm is *stable* if the result is relatively unaffected by perturbations during computation. This is similar to the idea of conditioning of problems.

An algorithm is *accurate* is the competed solution is close to the true solution of the problem. Note that a stable algorithm could be inaccurate.

We seek to design and apply stable and accurate algorithms to find accurate solutions to well-posed problems (or to find ways of transforming or approximating ill-posed problems by well-posed ones).

3 Approximating a function by a Taylor series

First, a little notation. A real-valued function $f : \mathbb{R} \to \mathbb{R}$ is a function f that takes a real number argument, $x \in \mathbb{R}$,and returns a real number, $f(x) \in \mathbb{R}$. We write $f'(x)$ to denote the derivative of f, so $f'(x) = \frac{df}{dx}$, and $f''(x)$ is the second derivative (so $f'(x) = \frac{d}{dx}(\frac{df}{dx}) = \frac{d^2 f}{dx^2}$) and $f^{(k)}(x)$ is the kth derivative of f evaluated at x.

As we have just seen, simply evaluating a real valued function can be prone to instability (if the function has a large condition number). For that reason, approximations of functions are often used. The most common method of approximating the real-valued function $f : \mathbb{R} \to \mathbb{R}$ by a simpler function is to use the Taylor series representation for f.

The Taylor series has the form of a polynomial where the coefficients of the polynomial are the derivatives of f evaluated at a point. So long as all derivatives of the function exists at the point $x = a$, $f(x)$ can be expressed in terms of of the value of the function and it's derivatives at a as:

$$f(x) = f(a) + (x-a)f'(a) + \frac{(x-a)^2}{2!}f''(a) + \ldots + \frac{(x-a)^k}{k!}f^{(k)}(a) + \ldots$$

This can be written more compactly as

$$f(x) = \sum_{k=0}^{\infty} \frac{(x-a)^k}{k!} f^{(k)}(a),$$

where $f^{(0)} = f$ and $0! = 1$ by definition.

This is known as the *Taylor series* for f about a. It is valid for x "close" to a (strictly, within the "radius of convergence" of the series). When $a = 0$, the Taylor series is known as a *Maclaurin series.*

This is an infinite series (the sum contains infinitely many terms) so cannot be directly computed. In practice, we truncate the series after n terms to get the *Taylor polynomial of degree* n centred at a, which we denote $\hat{f}_n(x;a)$:

$$f(x) \approx \hat{f}_n(x;a) = \sum_{k=0}^{n} \frac{(x-a)^k}{k!} f^{(k)}(a).$$

This is an approximation of f that can be readily calculated so long as the first n derivatives of f evaluated at a can be calculated. The approximation can be made arbitrarily accurate by increasing n. The quality of the approximation also depends on the distance of x from a — the closer x is to a, the better the approximation.

Example: Find the Taylor approximation of $f(x) = \exp(x) = e^x$ for values of x close to 0.

Solution: The kth derivative of $f(x) = e^x$ is simply e^x for all k. Since we want values of x close to 0, find the Taylor series about $a = 0$ (the Maclaurin series). Then

$$\widehat{f}_n(x;0) = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots = \sum_{k=0}^{n} \frac{x^k}{k!}$$

This series converges to e^x everywhere: $\lim_{\to\infty} \widehat{e}_n(x) = e^x$. The quality of the approximation for various values of n and x are studied in the table below.

n	1	2	3	4	...	true value e^x
$\widehat{f}_n(x=1;0)$	2.0000	2.5000	2.6667	2.7083	...	2.7183
Relative error	0.26	0.08	0.019	0.0037		
$\widehat{f}_n(x=2;0)$	3.0000	5.0000	6.3333	7.0000	...	7.3891
Relative error	0.59	0.32	0.14	0.053		

Notice that the error is smaller for x close to a and decreases as n, the number of terms in the polynomial increases. □

4 Finding the roots of equations

Given a real valued function $f : \mathbb{R} \to \mathbb{R}$, a fundamental problem is to find the values of x for which $f(x) = 0$. This is known as finding the roots of f. The problem crops up again and again and many problems can be reformulated as this problem. For example, if the trajectory of one object is described by $h(x)$, while another object has trajectory $g(x)$, then the two objects intercept one another exactly when $f(x) = g(x) - h(x) = 0$.

Another common application is when we wish to find the minimum or maximum value that a function takes. We know from basic calculus that if f is the derivative of some function F, then $F(x)$ takes its maximum or minimum values when $f(x) = 0$. Thus finding the maximum value of F is a matter of finding the roots of f.

We will consider two simple yet effective root finding algorithms. The bisection method and Newton's method. In both cases, we assume that $f : \mathbb{R} \to \mathbb{R}$ is continuous.

4.1 Bisection method

We want to find x^* such that $f(x^*) = 0$. Let $\text{sign}(f(x)) = -1$ if $f(x) < 0$ and $\text{sign}(f(x)) = 1$ if $f(x) > 0$. The bisection method proceeds as follows:

- **Initialise:** Choose initial guesses a, b such that $\text{sign}f(a) \neq \text{sign}f(b)$
- **Iterate** until the absolute difference $|a - b| \approx 0$
  - Calculate $c = \frac{a+b}{2}$
  - If $\text{sign}f(a) = \text{sign}f(c)$, then $a \leftarrow c$; otherwise $b \leftarrow c$

The method provides slow but sure convergence.

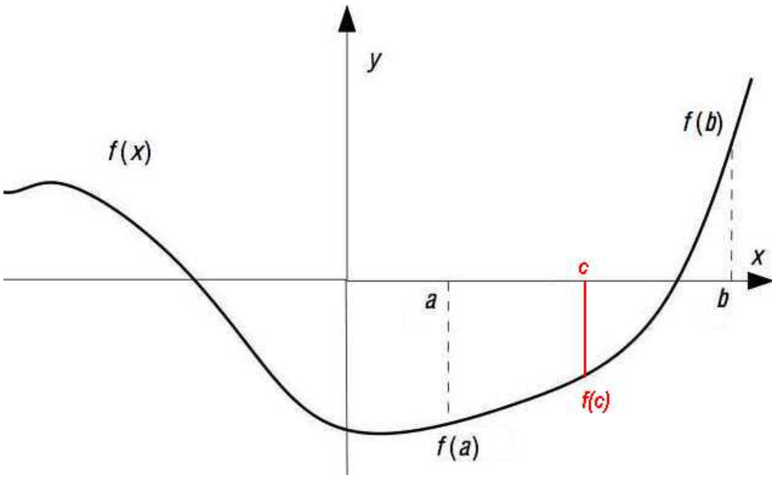


Figure 1: The idea behind the bisection method. Here, $\text{sign}f(a) = \text{sign}f(c)$ so at the next iteration we will set $a \leftarrow c$. The algorithm stops when a and b are sufficiently close to each other.

4.2 Newton's method

This method is also known as the Newton-Raphson method and is based on the approximation first two terms of the Taylor series expansion. Recall that we want to find x such that $f(x) = 0$. From the first two terms of the Taylor series of $f(x)$, we know that $f(x) \approx f(a) + (x - a)f'(a)$. If $f(x)$ is zero, then the expression on the right hand side to 0 and solve for x to get

$$f(a) + (x - a)f'(a) = 0 \implies x = a - \frac{f(a)}{f'(a)}.$$

We can treat this iteratively, starting at x_0, and finding $x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$. This leads to the algorithm:

- **Initialise:** Choose x_0 as an initial guess.
- **Iterate** until the absolute difference $|x_i - x_{i-1}| \approx 0$
  - Set $x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$.

Example: $f(x) = x^2 - 2$.
Solution: $f'(x) = 2x$ so

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} = x_i - \frac{x_i^2 - 2}{2x_i} = \frac{x_i}{2} + \frac{1}{x_i}.$$

See slide for example starting at $x_0 = 0.5$. □

Compare with bisection method: Start at $a = 1/2$ and $b = 2$ to get to $a = 1.34375$ and $b = 1.4375$ at the fifth step. This produces an absolute error of 0.02688 or 1.9%. The absolute error in the Newton case is 0.00020 which is 2 orders of magnitude smaller.

5 Numerical linear algebra

Numerical linear algebra is one of the cornerstones of modern mathematical modelling. Topics as important as solving systems of ordinary differential equations (arising in engineering, economics, physics, biotech, etc), to network analysis (telecoms, sociologic, epidemiology), internet search, data mining and many more rely on linear algebra.

These days, applied linear algebra and numerical linear algebra are virtually interchangeable — problems of all sizes are routinely solved numerically and rely on a wealth of mathematical and computational insight.

We'll start out with a brief review of topics that you should be somewhat familiar with.

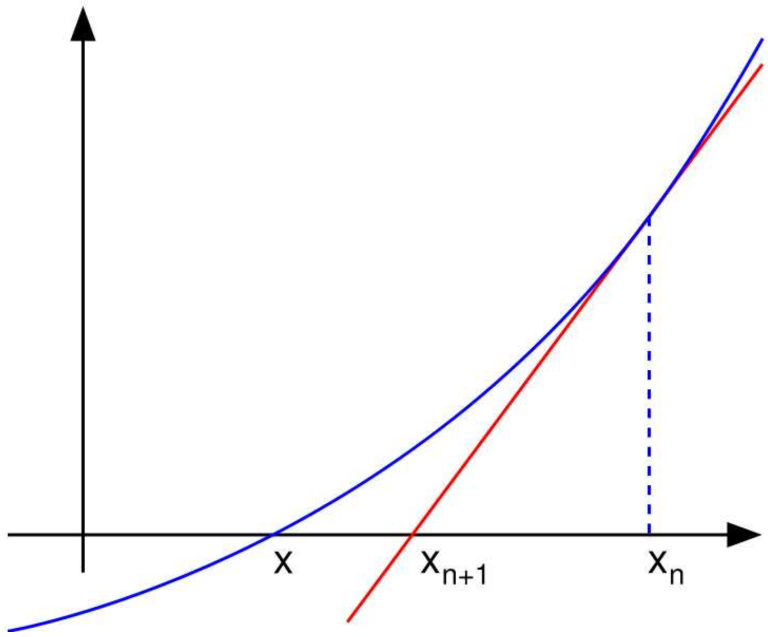


Figure 2: The idea behind the Newton's method. Starting at x_n, we find the tangent line (red) and calculate the point it intercepts the x-axis. This point of intercept is x_{n+1}.

5.1 Review

Let $\mathbf{a} = \begin{bmatrix} a_1 \\ \vdots \\ a_n \end{bmatrix}$ $\mathbf{b} = \begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix}$ be vectors. The *inner* or *dot product* of $\mathbf{a}$ and $\mathbf{b}$ is the scalar $c = \mathbf{a} \bullet \mathbf{b} \equiv \mathbf{a}^\mathsf{T}\mathbf{b} = \sum_{i=1}^{n} a_i b_i$. The dot product is also called *multiplication* of vectors.

The *norm* or *magnitude* of a vector $\mathbf{a}$ is $||\mathbf{a}|| = \sqrt{a \cdot a} = \sqrt{\mathbf{a}^\mathsf{T}\mathbf{a}} = \sqrt{a_1^2 + \ldots a_n^2}$.

The *product* of an $m \times n$ matrix $\mathbf{A} = \begin{bmatrix} A_{11} & \ldots & A_{1n} \\ \vdots & \ddots & \vdots \\ A_{m1} & \ldots & A_{mn} \end{bmatrix}$ and an $n \times 1$ (n-dimensional) vector $\mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$ is the m-dimensional vector $\mathbf{y} = \mathbf{A}\mathbf{x}$ with the elements $y_i = \sum_{j=1}^{m} A_{ij} x_j$.

The *product* of a $k \times m$ matrix $\mathbf{A}$ and an $m \times n$ matrix $\mathbf{B}$ is the $k \times n$ matrix $\mathbf{C} = \mathbf{AB}$ with the elements $C_{ij} = \sum_{\alpha=1}^{m} A_{i,\alpha} B_{\alpha,j}$

The *outer product* of an m-dimensional vector $\mathbf{a}$ with an n-dimensional vector $\mathbf{b}$ is the

$m \times n$ matrix

$$\mathbf{ab}^\mathsf{T} \equiv \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_m \end{bmatrix} \begin{bmatrix} b_1 & b_2 & \dots & b_n \end{bmatrix} = \begin{bmatrix} a_1b_1 & a_1b_2 & \dots & a_1b_n \\ a_2b_1 & a_2b_2 & \dots & a_2b_n \\ \vdots & \vdots & \ddots & \vdots \\ a_mb_1 & a_mb_2 & \dots & a_mb_n \end{bmatrix}$$

The *identity matrix* of size n, $\mathbf{I}_n$, is the $n \times n$ matrix with (i,j)th entry $= 0$ if $i \neq j$ and 1 if $i = j$.

The *inverse* of a square matrix $\mathbf{A}$ of size n is the square matrix $\mathbf{A}^{-1}$ such that $\mathbf{AA}^{-1} = \mathbf{I}_n = \mathbf{A}^{-1}\mathbf{A}$. When such a matrix exists, A is called *invertible* or *non-singular*. $\mathbf{A}$ is *singular* if no inverse exists. Finding the inverse of $\mathbf{A}$ is typically difficult.

The *determinant* of an $n \times n$ matrix $\mathbf{A}$, written $det(\mathbf{A}) = \begin{vmatrix} A_{11} & \dots & A_{1n} \\ \vdots & \ddots & \vdots \\ A_{n1} & \dots & A_{nn} \end{vmatrix}$, is given by a somewhat complex formula that we need not reproduce here (look it up at `http://en.wikipedia.org/wiki/Determinant`). For $n = 2$, $det(A) = A_{11}A_{22} - A_{21}A_{12}$. For $n = 3$, $det(A) = A_{11}A_{22}A_{33} - A_{31}A_{22}A_{13} + A_{12}A_{23}A_{31} - A_{32}A_{23}A_{11} + A_{13}A_{21}A_{32} - A_{33}A_{21}A_{12}$.

Example: Find the determinant of $\mathbf{A} = \begin{pmatrix} 3 & 5 \\ 1 & -1 \end{pmatrix}$.

Solution: From above, $det(\mathbf{A}) = |\mathbf{A}| = 3.-1 - 1.5 = -3 - 5 = -8$. □

It is worth recalling a few properties of the determinant (as listed on the wiki page):

- $det(\mathbf{I}) = 1$
- $det(A^T) = det(A)$ (transposing the matrix does not affect the determinant)
- $det(A^{-1}) = \frac{1}{det(A)}$ (the determinant of the inverse is the inverse of the determinant)
- For A, B square matrices of equal size, $det(AB) = det(A)det(B)$
- $det(cA) = c^n\, det(A)$ for any scalar c
- If A is triangular (so has all zeros in the upper or lower triangle) then $det(A) = \prod_{i=1}^n A_{ii}$.

An *eigenvector* of the square matrix $\mathbf{A}$ is a non-zero vector $\mathbf{e}$ such that $\mathbf{Ae} = \lambda\mathbf{e}$ for some scalar λ. λ is known as the *eigenvalue* of $\mathbf{A}$ corresponding to $\mathbf{e}$. Note that λ may be 0. So the effect of multiplying e by A is simply to scale e by the corresponding scalar λ.

The determinant can be used to find the eigenvalues of $\mathbf{A}$: they are the roots of the *characteristic polynomial* $p(\lambda) = \det(\mathbf{A} - \lambda\mathbf{I}_n)$ where $\mathbf{I}_n$ is the identity matrix.

Example: Find the eigenvalues of $\mathbf{A} = \begin{pmatrix} 3 & 5 \\ 1 & -1 \end{pmatrix}$.

Solution: We need to solve $p(\lambda) = \det(\mathbf{A} - \lambda \mathbf{I}_2) = 0$.

$$\begin{aligned}
|\mathbf{A} - \lambda \mathbf{I}_2| &= \left| \begin{pmatrix} 3 & 5 \\ 1 & -1 \end{pmatrix} - \lambda \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \right| \\
&= \begin{vmatrix} 3-\lambda & 5 \\ 1 & -1-\lambda \end{vmatrix} \\
&= (3-\lambda)(-1-\lambda) - 5 \\
&= -\lambda^2 - 2\lambda - 8 \\
&= (\lambda+2)(\lambda-4)
\end{aligned}$$

which is zero when $\lambda = 4$ or $\lambda = -2$. So the eigenvalues of $\mathbf{A}$ are $\lambda = 4$ and $\lambda = -2$. $\square$

Example: Find the eigenvector of $\mathbf{A} = \begin{pmatrix} 3 & 5 \\ 1 & -1 \end{pmatrix}$ corresponding to the eigenvalue $\lambda = -2$.

Solution: The eigenvector $\mathbf{e}$ corresponding to $\lambda = -2$ satisfies the equation $\mathbf{Ae} = -2\mathbf{e}$. That is,

$$\begin{pmatrix} 3 & 5 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} e_1 \\ e_2 \end{pmatrix} = -2 \begin{pmatrix} e_1 \\ e_2 \end{pmatrix}.$$

This is the system of linear equations

$$\begin{aligned}
3e_1 + 5e_2 &= -2e_1, & (1) \\
e_1 - e_2 &= -2e_2. & (2)
\end{aligned}$$

Rearranging either equation, we get $e_1 = -e_2$, so both equations are the same. We thus fix $e_1 = 1$ and the eigenvector associated with $\lambda = -2$ is $\mathbf{e} = \begin{pmatrix} 1 \\ -1 \end{pmatrix}$. Notice that the choice to fix $e_1 = 1$ was arbitrary. We could choose any value so, strictly, $\mathbf{e} = c \begin{pmatrix} 1 \\ -1 \end{pmatrix}$ for any $c \neq 0$. Often, c is chosen so that $\mathbf{e}$ is normalised (see below). In this case, choose $c = 1/\sqrt{2}$ to normalise $\mathbf{e}$. $\square$

Vectors a and b are *orthogonal* if the dot product $a^T b = 0$. Orthogonal generalises the of the idea of the perpendicular. In particular, a set of vectors $\{\mathbf{e}_1, \ldots, \mathbf{e}_n\}$ is *mutually orthogonal* if each pair of vectors e_i, e_j is orthogonal for $i \neq j$.

A vector $\mathbf{e}_i$ is normalised if $\mathbf{e}_i^T e_i = 1$.

A set of vectors that is mutually orthogonal and has each vector normalise is called *orthonormal.*

Any symmetric, square matrix $\mathbf{A}$ of size n has exactly n eigenvectors that are mutually orthogonal.

Any square matrix A of size n that has n mutually orthogonal eigenvectors can be represented via the *eigenvector representation* as follows:

$$\mathbf{A} = \sum_{i=1}^{n} \lambda_i \underbrace{\mathbf{e}_i \mathbf{e}_i^{\mathsf{T}}}_{\mathbf{U}_i}$$

where $\mathbf{U}_i = \mathbf{e}_i \mathbf{e}_i^{\mathsf{T}}$ is an $n \times n$ matrix.

The *Range*, range($\mathbf{A}$), or span of an $m \times n$ matrix $\mathbf{A}$ is the set of vectors $\mathbf{y} \in \mathbb{R}^m$ such that $\mathbf{y} = \mathbf{A}\mathbf{x}$ for some $\mathbf{x} \in \mathbb{R}^n$. The range is also referred to as the *column space* of $\mathbf{A}$ as it is the space of all linear combinations of the columns of $\mathbf{A}$.

The *Nullspace*, null($\mathbf{A}$), of an $m \times n$ matrix $\mathbf{A}$ is the set of vectors $\mathbf{x} \in \mathbb{R}^n$, such that $\mathbf{A}\mathbf{x} = \mathbf{0} \in \mathbb{R}^m$

The *Rank*, rank($\mathbf{A}$), of an $m \times n$ matrix $\mathbf{A}$ is the dimension of the range of $\mathbf{A}$ or of the column space of $\mathbf{A}$. $\text{rank}(\mathbf{A}) \leq \min\{m, n\}$.

5.2 Review of eigenvectors and eigenvalues

- λ is an eigenvalue of $\mathbf{A}$ if determinant $|\mathbf{A} - \lambda\mathbf{I}| = 0$
- This determinant is a polynomial in λ of degree n: so it has n roots $\lambda_1, \lambda_2, \ldots, \lambda_n$
- Every symmetric matrix $\mathbf{A}$ has a full set (basis) of n orthogonal unit eigenvectors $\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_n$
- No algebraic formula for the polynomial roots for $n > 4$
  - Thus, the eigenvalue problem needs own special algorithms
  - Solving the eigenvalue problem is harder than solving $\mathbf{Ax} = \mathbf{b}$
- Determinant $|\mathbf{A}| = \prod_{i=1}^{n} \lambda_i = \lambda_1\lambda_2 \cdots \lambda_n$ (the product of eigenvalues)
- The *trace* of a matrix is the sum of the diagonal elements. That is, trace$(\mathbf{A}) = \sum_{i=1}^{n} a_{ii} = a_{11} + a_{22} + \ldots + a_{nn}$.
- It turns out that trace$(\mathbf{A}) = \sum_{i=1}^{n} \lambda_i = \lambda_1 + \lambda_2 + \ldots + \lambda_n$ (the sum of eigenvalues)
- $\mathbf{A}^k = \underbrace{\mathbf{A} \cdots \mathbf{A}}_{k \text{ times}}$ has the same eigenvectors as $\mathbf{A}$: e.g. for $\mathbf{A}^2$

$$\mathbf{Ae} = \lambda\mathbf{e} \;\Rightarrow\; \mathbf{AAe} = \lambda\mathbf{Ae} = \lambda^2\mathbf{e}$$

- Eigenvalues of $\mathbf{A}^k$ are $\lambda_1^k, \ldots, \lambda_n^k$
- Eigenvalues of $\mathbf{A}^{-1}$ are $\frac{1}{\lambda_1}, \ldots, \frac{1}{\lambda_n}$

Example: Find the eigenvalues and eigenvectors of $\mathbf{A} = \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix}$

Solution: First, find the eigenvalues of $\mathbf{A}$ by solving

$$|\mathbf{A} - \lambda\mathbf{I}| = \begin{vmatrix} 2-\lambda & -1 \\ -1 & 2-\lambda \end{vmatrix} = \lambda^2 - 4\lambda + 3 = (\lambda - 1)(\lambda - 3) = 0.$$

So the eigenvalues are $\lambda_1 = 1$ and $\lambda_2 = 3$

The eigenvector associated with $\lambda_1 = 1$ is $\mathbf{e}_1$ and satisfies $\mathbf{Ae}_1 = \lambda_1\mathbf{e}_1$. Putting $\mathbf{e}_1 = \begin{bmatrix} x_1 \\ y_1 \end{bmatrix}$ we need to solve

$$\begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \begin{bmatrix} x_1 \\ y_1 \end{bmatrix}.$$

The second row gives $-x_1 + 2y_1 = y_1$, so $y_1 = x_1$. So fix $x_1 = 1$ and $e_1 = c\begin{bmatrix} 1 \\ 1 \end{bmatrix}$ for any $c \neq 0$. If we choose c so that e_1 is normalised, $\mathbf{e}_1 = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 \\ 1 \end{bmatrix}$. A similar argument shows $\mathbf{e}_2 = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 \\ -1 \end{bmatrix}$.

Before leaving this example, it is worth looking at some of the properties of the eigenvalues of $\mathbf{A}$:

- Determinant $\det \mathbf{A} \equiv |\mathbf{A}| = 4 - 1 = 3 \Longleftrightarrow \lambda_1 \cdot \lambda_2 \equiv 1 \cdot 3 = 3$
- $\text{trace}(\mathbf{A}) = 2 + 2 = 4 \Longleftrightarrow \lambda_1 + \lambda_2 \equiv 1 + 3 = 4$
- Inverse matrix $\mathbf{A}^{-1} = \frac{1}{3}\begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$: eigenvalues $\lambda_1 = \frac{1}{3}$ and $\lambda_2 = 1$
- Matrix $\mathbf{A}^2 = \begin{bmatrix} 5 & -4 \\ -4 & 5 \end{bmatrix}$: eigenvalues $\lambda_1 = 1$ and $\lambda_2 = 9$
- Matrix $\mathbf{A}^3 = \begin{bmatrix} 14 & -13 \\ -13 & 14 \end{bmatrix}$: eigenvalues $\lambda_1 = 1$ and $\lambda_2 = 27$

$\square$

5.3 Review of systems of linear equations

A linear equation in n unknowns $x_1, \dots, x_n$ is of the form $a_1 x_1 + \dots a_n x_n = b$. Given m such equations, we can write the ith equation as $a_{i1} x_1 + \dots a_{in} x_n = b_i$. We will seek to solve these systems of linear equations.

Example a system of 3 equations in 3 unknowns and its solution is

$$\left\{ \begin{array}{rcrcrcr} 4x_1 & + & x_2 & + & 2x_3 & = & 24 \\ 2x_1 & - & x_2 & - & 2x_3 & = & -6 \\ -x_1 & + & 2x_2 & - & x_3 & = & -4 \end{array} \right. \Longrightarrow \left\{ \begin{array}{l} x_1 = 3 \\ x_2 = 2 \\ x_3 = 5 \end{array} \right.$$

□

These systems can be represented as a matrix equation $\mathbf{Ax} = \mathbf{b}$ where $\mathbf{A}$ is the $m \times n$ matrix of coefficients, a_{ij}, $\mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$ is the n-dimensional column vector of unknowns and $\mathbf{b}$ is a vector of dimension m.

Example cont. In the example above, $\mathbf{A} = \begin{bmatrix} 4 & 1 & 2 \\ 2 & -1 & -2 \\ -1 & 2 & -1 \end{bmatrix}$ and $\mathbf{b} = \begin{bmatrix} 24 \\ -6 \\ -4 \end{bmatrix}$ □

We'll initially look at systems of n equations and n unknowns. Systems with $m < n$ are known as *under-determined* as there are less equations than unknowns while systems with $m > n$ are *over-determined* with more equations than there are unknowns.

When A is non-singular (so A^{-1} exists), the system has a unique solution given by $\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$.

Recall that $\mathbf{A}$ is nonsingular if and only if:

(*i*) inverse matrix $\mathbf{A}^{-1}$ exists; or
(*ii*) $\det(\mathbf{A}) \neq 0$; or
(*iii*) $\text{rank}(\mathbf{A}) = m$, or
(*iv*) $\mathbf{Ax} \neq \mathbf{0}$ for any vector $\mathbf{x} \neq \mathbf{0}$, or
(*v*) $\text{range}(\mathbf{A}) = \mathbb{R}^m$, or
(*vi*) $\text{null}(\mathbf{A}) = \{\mathbf{0}\}$.

If $\mathbf{A}$ is singular, the system may have infinitely many solutions or no solutions at all, depending on $\mathbf{b}$.

Example If $\mathbf{A} = \begin{bmatrix} 2 & 3 \\ 4 & 6 \end{bmatrix}$, $\mathbf{Ax} = \mathbf{b}$ has no solution if $\mathbf{b} \notin \text{range}(\mathbf{A})$ or infinitely many solutions when $\mathbf{b} \in \text{range}(\mathbf{A})$. Thus, when $\mathbf{b} = \begin{bmatrix} 4 \\ 7 \end{bmatrix}$ there is no solution, while when $\mathbf{b} = \begin{bmatrix} 4 \\ 8 \end{bmatrix}$, $\mathbf{x} = \begin{bmatrix} \gamma \\ \frac{2}{3}(2-\gamma) \end{bmatrix}$ is a solution for any real γ. □

6 Solving linear equations

In principle, all we need to do to solve the system of equations $\mathbf{Ax} = \mathbf{b}$ is find the inverse of $\mathbf{A}$, $\mathbf{A}^{-1}$. Then $\mathbf{Ax} = \mathbf{b} \implies \mathbf{A^{-1}Ax} = \mathbf{A^{-1}b} \implies \mathbf{x} = \mathbf{A^{-1}b}$. In practice, however, things are more complicated. First, $\mathbf{A}$ only has an inverse if it is square (so $m = n$) and $det(A) \neq 0$. In most cases, $m \neq n$ and often even when $m = n$, $det(A) = 0$ is not unusual. Second, supposing that A is indeed square, m and n are often large (10^4 is common, as are much larger values). In these cases, even calculating $det(A)$ is a hugely expensive and complex computational task while finding $\mathbf{A}^{-1}$ is even harder.

We'll initially concentrate on easily solvable systems and look at how we can coerce other systems into a form where they (or some close approximation) too are easily solvable.

6.1 Easily solvable systems 1: Diagonal matrix

All the simple systems we consider here are assumed to be square, so $m = n$. We want to solve $\mathbf{Ax} = \mathbf{b}$.

$\mathbf{A}$ is *diagonal* all entries the off-diagonal are zero. That is $a_{ij} = 0$ when $i \neq j$. So to specify a diagonal matrix, we need only specify the n diagonal elements. We can thus use the simplifying notation, $\mathbf{A} = \text{diag}\{a_1, \ldots, a_n\}$.

When A is diagonal, $x_i = \frac{b_i}{a_i}$ for all $i = 1, \ldots, n$. That is, $\mathbf{A}^{-1} = \text{diag}\{\frac{1}{a_1}, \ldots, \frac{1}{a_n}\}$. Or, to use less compact notation:

$$\mathbf{A} = \begin{bmatrix} a_1 & & \\ & \ddots & \\ & & a_n \end{bmatrix} \Rightarrow \mathbf{A}^{-1} = \begin{bmatrix} \frac{1}{a_1} & & \\ & \ddots & \\ & & \frac{1}{a_n} \end{bmatrix}.$$

6.2 Easily solvable systems 2: Triangular matrix

A matrix is *lower triangular* when all entries above the main diagonal are 0. That is, $\mathbf{A}$ is lower triangular if and only if $a_{ij} = 0$ when $i < j$. Similarly, a matrix is *upper triangular* when all entries above the main diagonal are 0 ($a_{ij} = 0$ for $i > j$). Lower triangular is also called left triangular, and upper called right triangular, for obvious reasons. E.g., a lower triangular matrix:

$$\mathbf{A} = \begin{bmatrix} a_{11} & 0 & \ldots & 0 \\ a_{21} & a_{22} & \ddots & \vdots \\ \vdots & \vdots & \ddots & 0 \\ a_{n1} & a_{n2} & \ldots & a_{nn} \end{bmatrix}.$$

The system $\mathbf{Ax} = \mathbf{b}$ is easy to solve for triangular $\mathbf{A}$ and it does not require that we calculate the inverse of $\mathbf{A}$.

For the lower triangular matrix, the solution is given by

$$x_i = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j \right),$$

so that

$$x_1 = \frac{b_1}{a_{11}};\ x_2 = \frac{b_2 - a_{21}x_1}{a_{22}};\ \ldots;\ x_n = \frac{b_n - a_{n1}x_1 - \ldots - a_{n-1,n}x_{n-1}}{a_{nn}}.$$

A similar simple formula is available for the upper triangular case, this time working backwards from x_n:

$$x_n = \frac{b_n}{a_{nn}}$$

and

$$x_i = \frac{1}{a_{ii}} \left(b_i - a_{i,i+1}x_{i+1} - \ldots - a_{i,n}x_n \right) \text{ for } i = n-1, \ldots, 1.$$

The method of Gaussian elimination, or row reduction, which we assume you have seen before transforms the matrix $\mathbf{A}$ into a triangular one to solve the system. This method is reviewed and discussed in Section 7.1

6.3 Easily solvable systems 3: Orthonormal or orthogonal matrix

Matrix $\mathbf{A}$ is *orthogonal* or *orthonormal* if the columns of $\mathbf{A}$ are mutually orthogonal unit vectors.

That is, $\mathbf{A} = [\mathbf{a}_1\ \mathbf{a}_2\ \ldots\ \mathbf{a}_n]$ where $\mathbf{a}_i = [a_{i1}\ a_{i2}\ \ldots\ a_{in}]^\mathsf{T}$ are unit vectors and the set $\{\mathbf{a}_1\ \mathbf{a}_2\ \ldots\ \mathbf{a}_n\}$ is mutually orthogonal (so $\mathbf{a}_i \cdot \mathbf{a}_j = 0$ for $i \neq j$).

When $\mathbf{A}$ is orthonormal, $\mathbf{A}^{-1} = \mathbf{A}^T$. This result is true since

$$\mathbf{A}^\mathsf{T}\mathbf{A} \equiv \begin{bmatrix} \mathbf{a}_1^\mathsf{T} \\ \vdots \\ \mathbf{a}_n^\mathsf{T} \end{bmatrix} [\mathbf{a}_1\ \mathbf{a}_2\ \ldots\ \mathbf{a}_n] = \mathbf{I}_n \equiv \text{diag}\{1, 1, \ldots, 1\}.$$

Also check that $\mathbf{A}\mathbf{A}^\mathsf{T} = \mathbf{I}_n$: $\mathbf{A}\mathbf{A}^\mathsf{T}\mathbf{A} \equiv \mathbf{A}\underbrace{(\mathbf{A}^\mathsf{T}\mathbf{A})}_{\mathbf{I}_n} = \mathbf{A}$ and $\mathbf{A}\mathbf{A}^\mathsf{T}\mathbf{A} \equiv (\mathbf{A}\mathbf{A}^\mathsf{T})\mathbf{A}$

These properties can be taken as a definition of an orthonomal matrix: $\mathbf{A}^{-1} = \mathbf{A}^T$ if and only if $\mathbf{A}$ is orthonormal.

Thus, if $\mathbf{A}$ is orthonormal, the solution to $\mathbf{A}\mathbf{x} = \mathbf{b}$ is simply $\mathbf{x} = \mathbf{A}^T\mathbf{b}$.

Example: Find the solution to the set of equations

$$\begin{aligned} 0.48x_1 + 0.64x_2 + 0.60x_3 &= 3.56 \\ 0.36x_1 + 0.48x_2 - 0.80x_3 &= -1.08 \\ 0.80x_1 - 0.60x_2 &= -0.40 \end{aligned}$$

or

$$x_1 \overbrace{\begin{bmatrix} 0.48 \\ 0.36 \\ 0.80 \end{bmatrix}}^{\mathbf{a}_1} + x_2 \overbrace{\begin{bmatrix} 0.64 \\ 0.48 \\ -0.60 \end{bmatrix}}^{\mathbf{a}_2} + x_3 \overbrace{\begin{bmatrix} 0.60 \\ -0.80 \\ 0.00 \end{bmatrix}}^{\mathbf{a}_3} = \begin{bmatrix} 3.56 \\ -1.08 \\ -0.40 \end{bmatrix}.$$

So $\mathbf{A} = [\mathbf{a}_1\, \mathbf{a}_2\, \mathbf{a}_3]$.

Solution: By checking that $\mathbf{a}_i \cdot \mathbf{a}_j = 1$ for $i = j = 0$ for $i \neq j$, it is easy to see that $\mathbf{A}$ is orthonormal. So we have the solution

$$\mathbf{x} = \mathbf{A}^\mathsf{T}\mathbf{b} = \begin{bmatrix} \mathbf{a}_1^T \\ \mathbf{a}_2^T \\ \mathbf{a}_3^T \end{bmatrix} \mathbf{b}$$

and

$$\begin{aligned} x_1 = \mathbf{a}_1^T\mathbf{b} &= 0.48 \cdot 3.56 - 0.36 \cdot 1.08 - 0.80 \cdot 0.40 \\ &= 1.7088 - 0.3888 - 0.3200 && = 1.0 \\ x_2 = \mathbf{a}_2^T\mathbf{b} &= 0.64 \cdot 3.56 - 0.48 \cdot 1.08 + 0.60 \cdot 0.40 \\ &= 2.2784 - 0.5184 + 0.2400 && = 2.0 \\ x_3 = \mathbf{a}_3^T\mathbf{b} &= 0.60 \cdot 3.56 + 0.80 \cdot 1.08 \\ &= 2.136 + 0.864 && = 3.0. \end{aligned}$$

□

7 Factorising matrices

As we saw in the previous section, matrices with special forms are often much easier to work with than arbitrary matrices. The remainder of this part of the course is focused on how we can manipulate an arbitrary given matrix into a form that is convenient for a stated problem. This is known as *factorising* or *decomposing* matrices.

There are 3 factorisations we will study in various degrees of depth: LU-factorisation, Singular Value Decomposition (SVD) and QR decomposition. A brief summary is given here:

- **Elimination** (LU decomposition): $\mathbf{A} = \mathbf{LU}$
  - Lower triangular matrix ◣ $\times$ ◥ Upper triangular matrix
- **Singular Value Decomposition** (SVD): $\mathbf{A} = \mathbf{UDV}^\mathsf{T}$

- ▥ $\times$ ⧅ diag(singular values) $\times$ ▤ Orthogonal (rows)
- Orthonormal columns in $\mathbf{U}$ and $\mathbf{V}$:
  the left and right **singular vectors**, respectively
- Left singular vector: an **eigenvector** of the square $m \times m$ matrix $\mathbf{A}\mathbf{A}^\top$
- Right singular vector: an **eigenvector** of the square $n \times n$ matrix $\mathbf{A}^\top\mathbf{A}$
- Singular value: the square root of an eigenvalue of $\mathbf{A}^\top\mathbf{A}$ (or $\mathbf{A}\mathbf{A}^\top$).

• **Orthogonalisation** (QR decomposition): $\mathbf{A} = \mathbf{QR}$

- Orthogonal matrix (columns) ▥ $\times$ ◥

7.1 LU decomposition via Gaussian elimination

7.1.1 Gaussian elimination to solve systems linear equations (review)

You should be familiar with the process of *Gaussian elimination* (or *row reduction*) in which the equation $\mathbf{Ax} = \mathbf{b}$ (where A is arbitrary) in transformed into the equivalent equation $\mathbf{C}x = \mathbf{d}$ where $\mathbf{C}$ is triangular, making the equation easy to solve. We review the process here.

It is easy to show that *multiplying* both sides of $\mathbf{Ax} = \mathbf{b}$ from the left by any nonsingular matrix $\mathbf{M}$ does not affect the solution. That is $\mathbf{MAx} = \mathbf{Mb}$ has the same solution as $\mathbf{Ax} = \mathbf{b}$, since

$$\mathbf{MAx} = \mathbf{Mb} \Rightarrow \mathbf{x} = (\mathbf{MA})^{-1}\,\mathbf{Mb} = \mathbf{A}^{-1}\mathbf{M}^{-1}\mathbf{Mb} = \mathbf{A}^{-1}\mathbf{b}.$$

We know from the above result that we can multiple both sides by a series of *elementary matrices* which perform various *row operations* on $\mathbf{A}$: the three types of operation are row swapping, row multiplication and adding some multiple of one row to another row. Repeated application of these three operations (that is, repeated multiplication by elementary matrices) to both sides of the equation transforms it to $\mathbf{C}x = \mathbf{d}$ where $\mathbf{C} = \mathbf{M}_1 \ldots \mathbf{M}_k\mathbf{A}$ is in upper triangular form (so the only non-zero elements of $\mathbf{C}$ are on or above the diagonal) and $\mathbf{d} = \mathbf{M}_1 \ldots \mathbf{M}_k\mathbf{b}$.

7.1.2 Gaussian elimination as LU decomposition

It turns out that Gaussian elimination can be viewed a LU decomposition in which a matrix $\mathbf{A}$ is written as the product of a lower triangular matrix $\mathbf{L}$ and an upper triangular matrix $\mathbf{U}$ so that $\mathbf{A} = \mathbf{LU}$. Recall that the Gaussian elimination process starts with an arbitrary square matrix $\mathbf{A}$, multiplies it by a series of elementary vectors, $\mathbf{M}_1 \ldots \mathbf{M}_k$ to get $\mathbf{U} = \mathbf{M}_1 \ldots \mathbf{M}_k\mathbf{A}$ where $\mathbf{U}$ is upper triangular (we called it $\mathbf{C}$ in the earlier discussion).

Now (check that you understand the following statements), each of the elementary matrices is lower triangular (so long as there are no row swapping operations) and the inverse of lower triangular matrices are lower triangular too, so each of $\mathbf{M}_i^{-1}$, for $i = 1, \ldots, k$ is lower triangular. Finally, the product of lower triangular matrices is also lower triangular so

$$\mathbf{U} = \mathbf{M}_1 \ldots \mathbf{M}_k\mathbf{A} \implies \mathbf{LU} = \mathbf{A},$$

where $\mathbf{L} = (\mathbf{M}_1 \ldots \mathbf{M}_k)^{-1} = \mathbf{M}_k^{-1} \ldots \mathbf{M}_1^{-1}$. If row permutations (row swaps) are needed in the Gaussian elimination process, we can't find an LU decomposition for $\mathbf{A}$ but can find an LU decomposition for the permuted matrix $\mathbf{PA}$, where $\mathbf{P}$ describes the necessary permutations. Obviously, the permuted system has the same solution as the unpermuted system.

The computational complexity of solving a system of n equations in n unknown using Gaussian elimination is $O(n^3)$. It is typically a stable algorithm, though potential for instability arises when a leading non-zero entry is very small (as we divide through by this entry). Reordering of the rows before the start of the row reduction process so that the largest leading non-zero elements are selected first can avoid this cause of instability. This technique is known as pivoting.

We won't look further at LU decompositions but will spend considerable time looking first at singular value decomposition and its uses and, later, when we consider the Least Squares framework, QR decompositions and its applications.

8 Singular Value Decomposition (SVD)

Singular Value Decomposition is a method of factorising any ordinary rectangular $m \times n$ matrix. It is most frequently applied to problems where $m \geq n$ (more equations than unknowns).

It has applications in signal processing, pattern recognition, and statistics where it is used for least squares data fitting, regularised inverse problems, finding pseudoinverses and performing principal component analysis (PCA). The application areas are many and varied but include computational tomography, seismology, weather forecast, image compression, image denoising, genetic analyses and more.

It is particularly useful when a given set of linear equations is singular or very close to singular in which case conventional solutions (e.g. by LU decomposition) are either not available or produce senseless results (due to the problems being ill-posed). In these cases, SVD can diagnose and, in some cases, solve the problem giving an useful numerical answer (though not necessarily the expected one!).

8.1 Overview of an SVD

SVD represents an ordinary $m \times n$ matrix $\mathbf{A}$ as $\mathbf{A} = \mathbf{U}\mathbf{D}\mathbf{V}^{\mathsf{T}}$ where:

- **U** : an $m \times m$ column-orthogonal matrix; its m columns are the m eigenvectors $\mathbf{u}$ of the $m \times m$ matrix $\mathbf{A}\mathbf{A}^{\mathsf{T}}$. The vectors $\{\mathbf{u}\}$ are known as the *left singular vectors* of $\mathbf{A}$.
- **V** : an $n \times n$ orthogonal matrix; its n columns are the eigenvectors $\mathbf{v}$ of the $n \times n$ matrix $\mathbf{A}^{\mathsf{T}}\mathbf{A}$. The vectors $\{\mathbf{v}\}$ are known as the *right singular vectors* of $\mathbf{A}$.
- **D** : an $m \times n$ matrix whose only non-zero elements are the first r entries on the diagonal where r is the rank of $\mathbf{A}$ and $d_{kk} = \sigma_k = \sqrt{\lambda_k}$ where λ_k is the eigenvalue associated with $\mathbf{v}_k$.

The *singular values*, σ_k are ordered so that $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r > 0$.
Since $\mathbf{A} = \mathbf{U}\mathbf{D}\mathbf{V}^{\mathsf{T}}$, we can write

$$\begin{aligned} \mathbf{A} &= \sum_{k=1}^{r} \sigma_k \mathbf{u}_k \mathbf{v}_k^T \qquad (3) \\ &= \sigma_1 \mathbf{u}_1 \mathbf{v}_1^{\mathsf{T}} + \sigma_2 \mathbf{u}_2 \mathbf{v}_2^{\mathsf{T}} + \ldots + \sigma_r \mathbf{u}_r \mathbf{v}_r^{\mathsf{T}}. \end{aligned}$$

This representation suggests the approximation of $\mathbf{A}$ by the truncated series,

$$\widehat{\mathbf{A}}_\rho = \sum_{k=1}^{\rho} \sigma_k \mathbf{u}_k \mathbf{v}_k^T \text{ for } \rho < r.$$

Notice that when $m > n$ (that is, the problem is over-determined), there are at most n non-zero singular values. In this case, we can truncate the matrix $\mathbf{U}$ to be $m \times n$ and the matrix $\mathbf{D}$ to be a $n \times n$ diagonal matrix. This leaves the sum in Equation 3 unaltered as the rows or columns that are removed contribute nothing to that sum. In the following example, we employ this strategy.

Example: Find the SVD of the matrix $\mathbf{A}$ where

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

Solution: First find the eigenvectors and eigenvalues of $\mathbf{A}\mathbf{A}^\mathsf{T}$ and $\mathbf{A}^\mathsf{T}\mathbf{A}$. Since $\mathbf{A}$ is 3×2, we need only find the top two eigenvalues and eigenvectors of each of these matrices.

$$\mathbf{A}^\mathsf{T}\mathbf{A} = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$$

which has eigenvalues $\lambda_1 = 3$ and $\lambda_2 = 1$. The associated eigenvectors are, respectively,

$$\mathbf{v}_1 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix} \text{ and } \mathbf{v}_2 = \frac{1}{\sqrt{2}} \begin{bmatrix} -1 \\ 1 \end{bmatrix}.$$

Notice that the eigenvectors have been normalised.

Similarly,

$$\mathbf{A}\mathbf{A}^\mathsf{T} = \begin{bmatrix} 0 & 1 \\ 1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 & 1 \\ 1 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

The top two eigenvalues are $\mu_1 = 3$ and $\mu_2 = 1$. Notice that these are the same as the top two eigenvalues of $\mathbf{A}^\mathsf{T}\mathbf{A}$. The associated eigenvectors are

$$\mathbf{u}_1 = \frac{1}{\sqrt{6}} \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \text{ and } \mathbf{u}_2 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}.$$

The singular values are given by $\sigma_i = \sqrt{\lambda_i}$, so $\sigma_1 = \sqrt{3}$ and $\sigma_2 = 1$.

We can thus write $\mathbf{A}$ as

$$\begin{aligned}\mathbf{A} = \begin{bmatrix} 0 & 1 \\ 1 & 1 \\ 1 & 0 \end{bmatrix} &= \underbrace{[\mathbf{u}_1\ \mathbf{u}_2]}_{\mathbf{U}} \underbrace{\operatorname{diag}(\sigma_1, \sigma_2)}_{\mathbf{D}} \underbrace{\begin{bmatrix} \mathbf{v}_1^\mathsf{T} \\ \mathbf{v}_2^\mathsf{T} \end{bmatrix}}_{\mathbf{V}^\mathsf{T}} \\ &= \underbrace{\begin{bmatrix} 1/\sqrt{6} & 1/\sqrt{2} \\ 2/\sqrt{6} & 0 \\ 1/\sqrt{6} & -1/\sqrt{2} \end{bmatrix}}_{\mathbf{U}} \underbrace{\begin{bmatrix} \sqrt{3} & 0 \\ 0 & 1 \end{bmatrix}}_{\mathbf{D}} \underbrace{\begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ -1/\sqrt{2} & 1/\sqrt{2} \end{bmatrix}}_{\mathbf{V}^\mathsf{T}}.\end{aligned}$$

The matrix approximation $\widehat{\mathbf{A}}_1$ is calculated as follows:

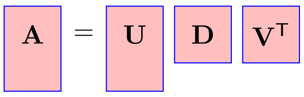
$$\begin{aligned}
\widehat{\mathbf{A}}_1 &= \sigma_1 \mathbf{u}_1 \mathbf{v}_1^{\mathsf{T}} \\
&= \sqrt{3} \cdot \tfrac{1}{\sqrt{6}} \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \cdot \tfrac{1}{\sqrt{2}} [1 \;\; 1] = \tfrac{1}{2} \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} [1 \;\; 1] = \begin{bmatrix} 0.5 & 0.5 \\ 1 & 1 \\ 0.5 & 0.5 \end{bmatrix}
\end{aligned}$$

while the approximation can be extended to $\widehat{\mathbf{A}}_2 (= \mathbf{A})$ by

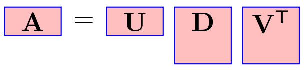
$$\begin{aligned}
\widehat{\mathbf{A}}_2 &= \widehat{\mathbf{A}}_1 + \sigma_2 \mathbf{u}_2 \mathbf{v}_2^{\mathsf{T}} \\
&\equiv \begin{bmatrix} 0.5 & 0.5 \\ 1 & 1 \\ 0.5 & 0.5 \end{bmatrix} + 1 \cdot \tfrac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} \tfrac{1}{\sqrt{2}} [-1 \;\; 1] \\
&\equiv \begin{bmatrix} 0.5 & 0.5 \\ 1 & 1 \\ 0.5 & 0.5 \end{bmatrix} + \begin{bmatrix} -0.5 & 0.5 \\ 0 & 0 \\ 0.5 & -0.5 \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & 1 \\ 1 & 1 \\ 1 & 0 \end{bmatrix}}_{\mathbf{A}}.
\end{aligned}$$

8.2 Structure of SVD

In the overdetermined case, in which $m > n$, so that we have more equations than unknowns, we have the following structure:



In the underdetermined case, in which $m < n$, so that we have fewer equations than unknowns, we have the following structure:



Note that, in the overdetermined case, we truncate $\mathbf{U}$ and $\mathbf{D}$ since there are at most $r \leq n < m$ non-zero singular values of $\mathbf{A}$ we can omit the $\mathbf{u}_i$ that contribute nothing to matrix product.

The matrix $\mathbf{V}$ is orthonormal, so $\mathbf{V}\mathbf{V}^T = \mathbf{V}^T\mathbf{V} = \mathbf{I}_n$. $\mathbf{U}$ is orthonormal when $m \geq n$, but if $m < n$, the singular values $\sigma_j = 0$ for $j = m+1, \ldots, n$ and the corresponding columns of $\mathbf{U}$ are also 0 so that $\mathbf{U}\mathbf{U}^T = \text{diag}(1, \ldots, 1, 0, \ldots, 0)$ where only the first m elements of the diagonal are 1 and the elements from $m+1$ to n are zero.

Note that the SVD of matrix $\mathbf{A}$ is only unique up to permutations of the columns/rows. For this reason, we insist that the singular values and corresponding singular vectors are arranged so that the singular values are in descending order $\sigma_1 \geq \sigma_2 \geq \ldots$. Even then, some of the σ_i's may have the same value so columns of $\mathbf{U}$ and $\mathbf{V}$ could be permuted. Aside from these possible permuations, the representation is unique. Be aware when calculating the SVD with various software that the you may need to enforce this canonical representation.

8.3 Condition number of a matrix

The concept of a condition number was introduced in Section 2. This concept can be applied to a matrices and is useful, for example, when considering solutions to the equation $\mathbf{Ax} = \mathbf{b}$. Solutions to this equation will change greatly with small changes in $\mathbf{b}$ when $\mathbf{A}$ has a large condition number, while the small changes in $\mathbf{b}$ will lead to only small changes in the solution when the matrix has a small condition number. We can define the condition number of a matrix as the maximum of the ratio of the relative error in $\mathbf{x}$ divided by the relative error in $\mathbf{b}$, where the maximum is taken over all possible $\mathbf{x}$ and $\mathbf{b}$.

To give a full description of how to derive the condition number of $\mathbf{A}$, we would have to introduce matrix norms which we do not have time to do here. Instead, we simply present the result here that the condition number of $\mathbf{A}$ can be defined as the ratio of the largest to the smallest non-zero singular values:

$$\mathit{cond}(\mathbf{A}) = \frac{\sigma_{\max}}{\sigma_{\min}}.$$

If the smallest singular value of $\mathbf{A}$ is 0, $\mathbf{A}$ is singular (has no inverse) but the condition number of $\mathbf{A}$ is still defined.

The condition number of $\mathbf{A}$ is considered to be large, and the matrix is *ill-conditioned*, if roughly $\log(\mathit{cond}(\mathbf{A})) \geq k$ where k is the number of digits of precision in the matrix entries.

Example: Find the condition number of the matrix $\mathbf{A} = \begin{bmatrix} 2 & -3 \\ 1 & -1 \end{bmatrix}$

Solution: Using a matrix algebra package, find the singular values of $\mathbf{A}$ to be 3.864 and 0.259, so $\mathit{cond}(\mathbf{A}) = \frac{3.864}{0.259} \approx 14.9$. □

Example: The singular values of the matrix $\mathbf{A} = \begin{bmatrix} 1.2969 & 0.8648 \\ 0.2161 & 0.1441 \end{bmatrix}$ are approximately 1.58 and 6.33×10^{-9} so the condition number is about 2.5×10^{8}. This very large condition number means that $\mathbf{A}$ is an ill-conditioned matrix.

Ill-conditioning means that standard approaches to solving linear systems can be very unstable. For example, consider the linear system $\mathbf{Ax} = \mathbf{b}$ where $\mathbf{b} = \begin{bmatrix} 0.8642 \\ 0.1140 \end{bmatrix}$. This has the exact solution $\mathbf{x} = \begin{bmatrix} 2 \\ -2 \end{bmatrix}$.

But standard matrix software (`linsolve` in Matlab), gives the solution as $\begin{bmatrix} 2.59 \\ -3.89 \end{bmatrix} \times 10^{6}$ which is radically wrong!

It also means that if some number in the system is measured slightly differently, the results we get can change enormously. For example, if the measurement vector $\mathbf{b}$ is just slightly different, say $\mathbf{b} = \begin{bmatrix} 0.86419999 \\ 0.11400001 \end{bmatrix}$, then the exact solution is now close to $\mathbf{x} = \begin{bmatrix} 0.9911 \\ -0.4870 \end{bmatrix}$ which represents an enormous change in the solution relative to the small change in the original system. $\square$

8.4 Principal Components Analysis (PCA)

PCA is a common technique for identifying patterns in high-dimensional data. It transforms the original correlated measurements into uncorrelated measurements. One of the main uses of PCA is as a dimension reduction tool, in which only the directions in which the data varies the most are considered. This can lead to enormous simplifications of the data and provide insights for a wide variety of data. PCA is alternatively known as the Karhunen-Loéve transform (KLT), the Hotelling transform or proper orthogonal decomposition (POD)

These new coordinate axes (along which the data varies the most) are are known as *principal components* and are, by construction, orthogonal.

A useful visualisation tool to aid your understanding of PCA is at http://setosa.io/ev/principal-component-analysis/.

Suppose we have a $m \times n$ matrix of measurement data A. For example, n trials where m properties were measured in each trial. Then, if $\mathbf{a}_i$ are the measurements from the ith trial,

$$\mathbf{A} = \begin{bmatrix} \mathbf{a}_1 & \mathbf{a}_2 & \dots & \mathbf{a}_n \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix}.$$

For example, $\mathbf{A}$ could be $n = 100$ observations of the position of an object measured in $m = 3$ dimensions.

In the following, assume that the rows of $\mathbf{A}$ have been centred, so that the mean of each row is 0 (each rows of $\mathbf{A}$ corresponds to a dimension in the original data). If this is not already the case, it can be achieved by subtracting the mean of each row of $\mathbf{A}$ from each element of that row. That is, set element

$$a_{ij} = a_{ij} - \sum_{j=1}^{n} a_{ij}/n$$

to centre the rows of $\mathbf{A}$. This is a critical assumption and allows us to concentrate on the variance.

Each observation $\mathbf{A}$ is just the m-vector $\mathbf{a}_i$. The idea of PCA is to chose a new basis $\mathbf{u}_1, \dots, \mathbf{u}_k$ to express the data points (the $\mathbf{a}_i$'s) so that the variance of the measurements is greatest in the direction of $\mathbf{u}_1$, the next greatest variance is in the direction of $\mathbf{u}_2$ and so on, down to $\mathbf{u}_k$. Ideally, $k < m$.

Define the *covariance matrix* of $\mathbf{A}$ by

$$\mathbf{\Sigma} = \frac{1}{n-1}\mathbf{A}\mathbf{A}^T.$$

Then $\mathbf{\Sigma}$ is an $m \times m$ matrix where the diagonal terms of $\mathbf{\Sigma}$ are the variance of the ith dimension of the measurement, while the off-diagonal terms of $\mathbf{\Sigma}$ are the covariances between different measurements.

It turns out that the best basis to choose are the k eigenvectors of $\mathbf{\Sigma}$ corresponding its k largest eigenvalues. These are known as the *principal components* of $\mathbf{A}$. Call them $\mathbf{u}_1, \ldots, \mathbf{u}_k$ and form the matrix

$$\mathbf{U}_k = [\mathbf{u}_1, \ldots, \mathbf{u}_k]$$

From results we have seen earlier about symmetric matrices, this an orthogonal matrix. We include the subscript k as we may decide to truncate this matrix by including only the eigenvectors corresponding to the largest eigenvalues. That is, if $\mathbf{\Sigma}$ has K eigenvectors and associated eigenvalues, and the largest k eigenvalues are substantially larger than the remaining $K - k$, it is reasonable to form $\mathbf{U}_k$ containing only the most significant k eigenvectors.

We can now represent the original measurements, $\mathbf{A}$, in this this new co-ordinate system. The amount of measurement vector $\mathbf{a}_i$ in direction $\mathbf{u}_j$ is given by $\mathbf{u}_j^\top \mathbf{a}_i$: this is the jth coordinate of $\mathbf{a}_i$ in this new coordinate system. So if we consider just the two dimension space defined by the top two principal components, $\mathbf{a}_i$ has coordinates

$$\mathbf{U}_2^\top \mathbf{a}_i = \begin{bmatrix} \mathbf{u}_1^\top \\ \mathbf{u}_2^\top \end{bmatrix} \mathbf{a}_i = \begin{bmatrix} \mathbf{u}_1^\top \mathbf{a}_i \\ \mathbf{u}_2^\top \mathbf{a}_i \end{bmatrix}.$$

We usually consider this space independently of how it relates to the original m-dimensional space but we can consider it as embedded in the original space.

Example: Find the principal components of the data matrix $\mathbf{A}$ where

$$\mathbf{A} = \begin{bmatrix} -4 & 3 & -5 & 18 & 6 & -5 \\ 2 & 6 & -2 & 10 & 1 & -1 \\ 7 & 11 & 3 & 6 & 9 & 3 \end{bmatrix}.$$

Find the amount of the first principal component in the first measurement vector of $\mathbf{A}$ (that is, the first column), and calculate the projection matrix for projecting $\mathbf{A}$ onto the first two principal components.

Solution: First, centre the rows of $\mathbf{A}$ so that each row has mean zero. Call this centred matrix $\mathbf{B}$.

$$\mathbf{B} = \begin{bmatrix} -6.1667 & 0.8333 & -7.1667 & 15.8333 & 3.8333 & -7.1667 \\ -0.6667 & 3.3333 & -4.6667 & 7.3333 & -1.6667 & -3.6667 \\ 0.5 & 4.5 & -3.5 & -0.5 & 2.5 & -3.5 \end{bmatrix}.$$

Now form the covariance matrix for the centred data matrix, $\mathbf{\Sigma} = \frac{1}{n}\mathbf{B}\mathbf{B}^{\mathsf{T}}$:

$$\mathbf{\Sigma} = \frac{1}{5}\begin{bmatrix} 406.8333 & 176.3333 & 52.5 \\ 176.3333 & 103.3333 & 36.0 \\ 52.5000 & 36.0000 & 51.5 \end{bmatrix}.$$

This matrix has eigenvalues 99.31, 9.46 and 3.561 corresponding to eigenvectors

$$[\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3] = \begin{bmatrix} 0.8987 & 0.2829 & 0.3352 \\ 0.4158 & -0.3062 & -0.8564 \\ 0.1396 & -0.9090 & 0.3928 \end{bmatrix} = \mathbf{U}.$$

These eigenvectors are the principal components of $\mathbf{A}$ (and of $\mathbf{B}$).

The amount of the first principal component in $\mathbf{a}_1$ is

$$\mathbf{u}_1^{\mathsf{T}}\mathbf{a}_1 = [0.8987, 0.4158, 0.1396]\begin{bmatrix} -4 \\ 2 \\ 7 \end{bmatrix} = -1.756.$$

8.5 What is connection between PCA and SVD?

Given $\mathbf{A}$ such that the rows of $\mathbf{A}$ have zero mean, define $\mathbf{Y} = \frac{1}{\sqrt{n-1}}\mathbf{A}^T$ (which has columns with zero mean). Then $\mathbf{Y}^T\mathbf{Y} = \mathbf{\Sigma}$, the covariance of $\mathbf{A}$. We have seen that the principal components of $\mathbf{A}$ are the eigenvectors of $\mathbf{\Sigma}$.

Now, if we calculate the SVD of $\mathbf{Y}$ to get $\mathbf{Y} = \mathbf{U}\mathbf{D}\mathbf{V}^T$, the columns of $\mathbf{V}$ are the eigenvectors of $\mathbf{Y}^T\mathbf{Y} = \mathbf{\Sigma}$. Therefore, the columns of $\mathbf{V}$ are the principal components of $\mathbf{A}$.

8.6 Problems with SVD and PCA

As we have seen, SVD and PCA are powerful analysis tools and SVD is a very stable procedure. They do not, however, come free of cost.

The time complexity of SVD is $O(m^2n + n^3)$ to calculate all of $\mathbf{U}, \mathbf{V}$ and $\mathbf{D}$ (where, typically, $m \gg n$) while faster algorithms are available when some elements of the SVD are not required.

However, the matrices $\mathbf{U}$ and $\mathbf{V}$ are not at all *sparse*, where we say a matrix is sparse when it mainly consists of zeros. Spareness is a commonly assumed property in large systems as it reflects the observation that most effects are local and do not influence all parameters in the system — a large world with small neighbourhoods. Sparse matrices are typically computationally efficient to work with and store.

A second potential set-back is that SVD and PCA only work with data that can be (coherently) expressed as a two dimensional array (that is, a matrix). When data naturally has 3 or 4 dimensions arrays (*tensors*), as is common in many engineering applications, there is no perfect analogue to SVD or PCA or even eigenvectors.

Finally, when using PCA for data analysis, you should be aware of the strong assumptions being made. In particular, dependencies in the data are assumed to be linear, which may not be the case. PCA and SVD will always give an answer but it is up to the user to interpret whether or not it is a valid answer to any question they are interested in.

9 Least squares

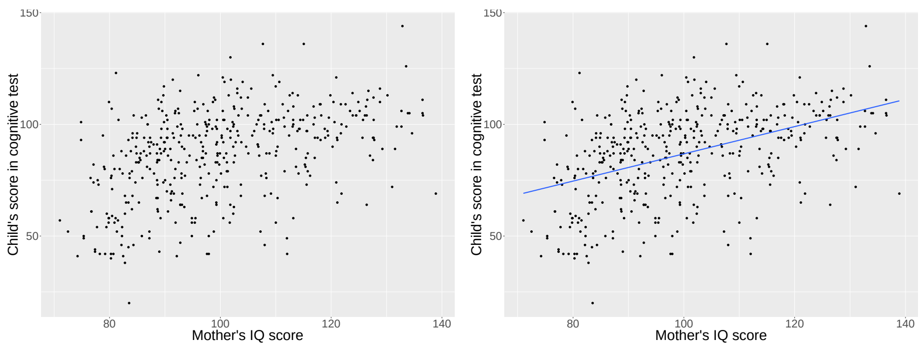


Figure 3: Left: Relationship between cognitive test scores for 3-4 year old children and mother's IQ score. Right: The same data with a least squares best fit line added. Discussed in Gelman and Hill, 2007, Cambridge University Press, data at http://www.stat.columbia.edu/ gelman/arm/examples/child.iq/kidiq.dta

You are probably familiar with the basic idea of least squares: we have a set of measurements and we want to fit a model to them. But no sufficiently simple model exactly fits all of the points at the same time. So how choose the model that is most satisfactory? The answer often given is that we chose the model that satisfies the *least squares* criterion: that is, the model for which the sum of the squares of differences between the predictions from the model and the actual observations is minimised.

For example, in Figure 3, we might want to fit a linear model to the relationship between a mother's IQ score and her young child's score in a cognitive test. This should be familiar to you as the linear regression problem in statistics.

This problem arises when we have an *overdetermined* linear system: recall that $\mathbf{Au} = \mathbf{b}$ is overdetermined when $\mathbf{A}$ is $m \times n$ matrix with $m > n$:

$$\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}.$$

In this case, $\mathbf{A}^{-1}$ does not exist and there is no $\mathbf{u}$ that solves this problem. (We ignore the highly unusual cases where a solution does exist.)

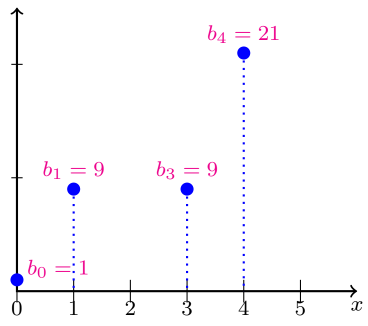
The goal, then, is to find the best solution $\mathbf{u}^*$ to the problem.

Example 1: Fitting $m = 4$ measurements by a small number $n = 2$ of parameters (e.g. linear regression in statistics)

Want to find the straight line $b_x = u_1 + u_2 x$ where we have observed the points b_x at x.

$$\begin{cases} u_1 + u_2 \cdot 0 &= 1 \\ u_1 + u_2 \cdot 1 &= 9 \\ u_1 + u_2 \cdot 2 &= 9 \\ u_1 + u_2 \cdot 4 &= 21 \end{cases} \Leftrightarrow$$

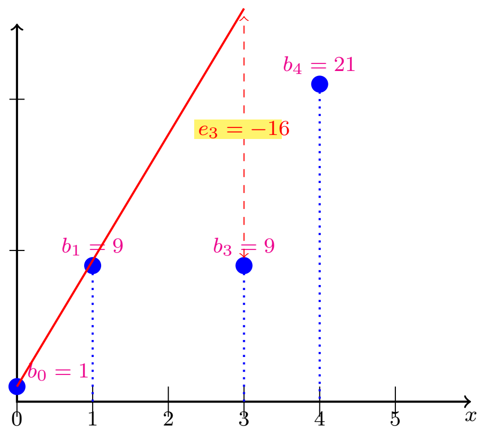
$$\left\{ \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 3 \\ 1 & 4 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 9 \\ 9 \\ 21 \end{bmatrix} \right.$$



The above set of equations clearly has no solution as vector **b** is not a linear combination of the two column vectors from **A**:

$$\begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 3 \\ 1 & 4 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \neq \begin{bmatrix} 1 \\ 9 \\ 9 \\ 21 \end{bmatrix}$$

For example, The line $b = 1 + 8x$ through the first two points is almost certainly not the best line:



But why is this not the best line: look at the *error* or *residual* , $\mathbf{e} = \mathbf{b} - \mathbf{A}\mathbf{u}$. For the two points the line does not pass through the error is $e_x = b_x - (1 + 8x)$ is large: $e_3 = 16$ and $e_4 = 12$. The *Total square error*, $E(\mathbf{u}) = 0 + 0 + 256 + 144 = 400$.

Notice that the *total square error* is given by.

$$E(\mathbf{u}) = \mathbf{e}^{\mathsf{T}}\mathbf{e} \equiv \| \mathbf{e} \|^2 = (\mathbf{b} - \mathbf{A}\mathbf{u})^{\mathsf{T}}(\mathbf{b} - \mathbf{A}\mathbf{u})$$

The Least Squares method to find the chooses a solution $\mathbf{u}^*$ that minimises $E(\mathbf{u})$.

How do we find $\mathbf{u}^*$? To find the minimum of $E(\mathbf{u})$, we can differentiate with respect to $\mathbf{u}$, set to 0 and attempt to solve for $\mathbf{u}$:

$$\begin{aligned} E(\mathbf{u}) &= (\mathbf{b} - \mathbf{A}\mathbf{u})^{\mathsf{T}}(\mathbf{b} - \mathbf{A}\mathbf{u}) \\ &= \mathbf{b}^{\mathsf{T}}\mathbf{b} - 2\mathbf{u}^{\mathsf{T}}\mathbf{A}^{\mathsf{T}}\mathbf{b} + \mathbf{u}^{\mathsf{T}}\mathbf{A}^{\mathsf{T}}\mathbf{A}\mathbf{u} \end{aligned}$$

Differentiating and setting to 0:

$$\begin{aligned} \tfrac{\partial E(\mathbf{u})}{\partial \mathbf{u}} &= 0 \\ \Longrightarrow \quad -2\mathbf{A}^{\mathsf{T}}\mathbf{b} + 2\mathbf{A}^{\mathsf{T}}\mathbf{A}\mathbf{u} &= \mathbf{0} \\ \Longrightarrow \quad \mathbf{A}^{\mathsf{T}}\mathbf{A}\mathbf{u} &= \mathbf{A}^{\mathsf{T}}\mathbf{b} \end{aligned}$$

This equation, $\mathbf{A}^{\mathsf{T}}\mathbf{A}\mathbf{u} = \mathbf{A}^{\mathsf{T}}\mathbf{b}$ is called the *normal equation.*

The least squares estimate, $\mathbf{u}^*$, is the solution to the normal equation.

Notice that $\mathbf{A}^T\mathbf{A}$ is square and symmetric. In some cases it may be possible to directly find the inverse (in particular, when $\mathbf{A}$ has independent columns, then $\mathbf{A}^T\mathbf{A}$ is positive definite and $\mathbf{A}^T\mathbf{A}$ is invertible in which case $\mathbf{u}^* = (\mathbf{A}^{\mathsf{T}}\mathbf{A})^{-1}\mathbf{A}^{\mathsf{T}}\mathbf{b}$). In other cases, this approach may be highly unstable, so stable numerical techniques need to be employed.

9.1 Computing the Least Squares solution, $\mathbf{u}^*$

We consider three methods for computing the least squares solution to a linear system. They are Gaussian elimination, QR Decomposition (aka Orthogonalisation) and computation of the pseudo-inverse via SVD.

9.2 Computing $\mathbf{u}^*$ via Gaussian elimination

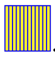
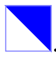
Given the normal equation $\mathbf{A}^T\mathbf{A}\mathbf{u} = \mathbf{A}^T\mathbf{b}$, we may be tempted to find the solution by Gaussian elimination, where we reduce the the matrix $\mathbf{A}^T\mathbf{A}$ to upper triangular form using elementary row operations.

This solution can work but is highly unstable. To see why it is unstable, consider the condition number of the matrix $\mathbf{A}^T\mathbf{A}$. It can be shown that the condition number of $\mathbf{A}^T\mathbf{A}$ is the square of the condition number of $\mathbf{A}$ (if we take $\sigma_{\min}$ to be the smallest non-zero singular value in the definition of condition number). So even if $\mathbf{A}$ has only moderately widely spread singular values, $\mathbf{A}^T\mathbf{A}$ can have a very large condition number and solution by row reduction can be very unstable.

9.3 Computing $\mathbf{u}^*$ via orthogonalisation (QR decomposition)

QR-decompostion presents a much more stable solution to the normal equation $\mathbf{A}^\mathsf{T}\mathbf{A}\mathbf{u} = \mathbf{A}^\mathsf{T}\mathbf{b}$.

The Orthogonalisation of matrix $\mathbf{A}$ is given by $\mathbf{A} = \mathbf{QR}$ where

- $\mathbf{Q}$ is an $m \times n$ matrix with n orthonormal columns: . Construction of $\mathbf{Q}$ is discussed below.
- $\mathbf{R}$ is an $n \times n$ upper triangular matrix: . $\mathbf{R}$ is given by $\mathbf{R} = \mathbf{Q}^\mathsf{T}\mathbf{A}$.

This factorisation reduces the normal equation to a much simpler equation:

$$\begin{aligned}
\mathbf{A}^\mathsf{T}\mathbf{A}\mathbf{u} &= \mathbf{A}^\mathsf{T}\mathbf{b} \\
\implies (\mathbf{QR})^\mathsf{T}\mathbf{QR}\mathbf{u}^* &= (\mathbf{QR})^\mathsf{T}\mathbf{b} \\
\implies \mathbf{R}^\mathsf{T}\mathbf{Q}^\mathsf{T}\mathbf{QR}\mathbf{u}^* &= \mathbf{R}^\mathsf{T}\mathbf{Q}^\mathsf{T}\mathbf{b} \\
\implies \mathbf{R}^\mathsf{T}\mathbf{R}\mathbf{u}^* &= \mathbf{R}^\mathsf{T}\mathbf{Q}^\mathsf{T}\mathbf{b} \text{ since } \mathbf{Q}^\mathsf{T}\mathbf{Q} = \mathbf{I} \\
\implies \mathbf{R}\mathbf{u}^* &= \mathbf{Q}^\mathsf{T}\mathbf{b} \text{ multiplying both sides by } (\mathbf{R}^\mathsf{T})^{-1}.
\end{aligned}$$

This is easy to solve via back-substitution, since $\mathbf{R}$ is upper triangular.

9.3.1 Constructing the orthogonal matrix Q by Gram-Schmidt

The orthonormal columns of $\mathbf{Q}$, call them $\mathbf{q}_1, \ldots, \mathbf{q}_n$, are obtained iteratively from the columns $\mathbf{a}_1, \ldots, \mathbf{a}_n$ of $\mathbf{A}$. The basic idea is that we set $\mathbf{q}_1$ to be $\mathbf{a}_1$. q_2 is then set to be a_2 and any part of it in the direction of $q_1(= a_1)$ is subtracted out, so ensure that it is orthogonal to q_1. Similarly, q_3 is set to be a_3 with any parts in the direction of q_1 or q_2 are subtracted. All these vectors are normalised to have magnitude 1 at each step.

This is called the *Gram-Schmidt* process and is more formally defined as follows:

$$\begin{array}{lcll}
\text{Set } \mathbf{v}_1 & = & \mathbf{a}_1. & \text{Then} \quad \mathbf{q}_1 = \frac{\mathbf{v}_1}{|\mathbf{v}_1|}. \\
\text{Set } \mathbf{v}_2 & = & \mathbf{a}_2 - \left(\mathbf{a}_2^\mathsf{T}\mathbf{q}_1\right)\mathbf{q}_1. & \text{Then} \quad \mathbf{q}_2 = \frac{\mathbf{v}_2}{|\mathbf{v}_2|}. \\
\vdots & & \vdots & \qquad\quad \vdots \qquad \vdots \\
\text{Set } \mathbf{v}_j & = & \mathbf{a}_j - \sum_{i=1}^{j-1}\left(\mathbf{a}_j^\mathsf{T}\mathbf{q}_i\right)\mathbf{q}_i. & \text{Then} \quad \mathbf{q}_j = \frac{\mathbf{v}_j}{|\mathbf{v}_j|}.
\end{array}$$

Note that $|\mathbf{v}| = \mathbf{v}^\mathsf{T}\mathbf{v}$ is the norm of $\mathbf{v}$.

Having found $\mathbf{Q}$, find $\mathbf{R}$ by setting $\mathbf{R} = \mathbf{Q}^\top\mathbf{A}$. $\mathbf{R}$ is indeed upper triangular since the ith column of $\mathbf{Q}$ is, by construction, orthogonal to first $i-1$ columns of $\mathbf{A}$.

Producing $\mathbf{Q}$ and $\mathbf{R}$ takes twice as long as the mn^2 steps to form $\mathbf{A}^\mathsf{T}\mathbf{A}$, but that extra cost gives a more reliable solution.

There is another method of orthogonalisation that we don't cover here which has better numerical stability using so-called Householder reflectors.

Example: Use the Gram-Schmidt process to orthogonalise the matrix

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

Solution: Let $\mathbf{v}_1 = \mathbf{a}_1 = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$ and then normalise to get $\mathbf{q}_1 = \frac{\mathbf{v}_1}{|\mathbf{v}_1|} = \begin{bmatrix} 0.5 \\ 0.5 \\ 0.5 \\ 0.5 \end{bmatrix}$. Now set $\mathbf{v}_2 = \mathbf{a}_2 - (\mathbf{a}_2^\top \mathbf{q}_1)\mathbf{q}_1 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} - \left(\begin{bmatrix} 0 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} 0.5 \\ 0.5 \\ 0.5 \\ 0.5 \end{bmatrix} \right) \begin{bmatrix} 0.5 \\ 0.5 \\ 0.5 \\ 0.5 \end{bmatrix} = \begin{bmatrix} -0.5 \\ -0.5 \\ 0.5 \\ 0.5 \end{bmatrix}$.

Since $\mathbf{v}_2 = 1$, $\mathbf{q}_2 = \frac{\mathbf{v}_2}{|\mathbf{v}_2|} = \mathbf{v}_2$.

Finally, to get $\mathbf{q}_3$, set

$$\begin{aligned}
\mathbf{v}_3 &= \mathbf{a}_3 - (\mathbf{a}_3^\top \mathbf{q}_1)\mathbf{q}_1 - (\mathbf{a}_3^\top \mathbf{q}_2)\mathbf{q}_2 \\
&= \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \end{bmatrix} - \left(\begin{bmatrix} 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0.5 \\ 0.5 \\ 0.5 \\ 0.5 \end{bmatrix} \right) \begin{bmatrix} 0.5 \\ 0.5 \\ 0.5 \\ 0.5 \end{bmatrix} - \left(\begin{bmatrix} 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} -0.5 \\ -0.5 \\ 0.5 \\ 0.5 \end{bmatrix} \right) \begin{bmatrix} -0.5 \\ -0.5 \\ 0.5 \\ 0.5 \end{bmatrix} \\
&= \begin{bmatrix} 0.5 \\ -0.5 \\ 0.5 \\ -0.5 \end{bmatrix}
\end{aligned}$$

which is also normalised, so $\mathbf{q}_3 = \mathbf{v}_3$ and

$$\mathbf{Q} = \frac{1}{2} \begin{bmatrix} 1 & -1 & 1 \\ 1 & -1 & -1 \\ 1 & 1 & 1 \\ 1 & 1 & -1 \end{bmatrix}.$$

We find $\mathbf{R}$ as follows:

$$\mathbf{R} = \mathbf{Q}^\mathsf{T}\mathbf{A} = \frac{1}{2} \begin{bmatrix} 1 & 1 & 1 & 1 \\ -1 & -1 & 1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix}.$$

$\square$

9.4 Computing $\mathbf{u}^*$ via SVD: the Pseudoinverse

The most stable computation to find the solution to the normal equation is given by singular value decomposition (SVD).
Recall that SVD decomposes the $m \times n$ matrix $\mathbf{A}$ as $\mathbf{A} = \mathbf{U}\mathbf{D}\mathbf{V}^\mathsf{T}$ where:

- $\mathbf{U}$ is a column-orthonormal $n \times m$ matrix, so $\mathbf{U}^\mathsf{T}\mathbf{U} = \mathbf{I}_n$,
- $\mathbf{V}$ is an orthonormal $n \times n$ matrix so $\mathbf{V}^\mathsf{T}\mathbf{V} = \mathbf{I}_n$ (indeed, $\mathbf{V}^\mathsf{T} = \mathbf{V}^{-1}$), and
- $\mathbf{D} = \mathrm{diag}\{\sigma_1, \ldots, \sigma_n\}$ is a diagonal $n \times n$ matrix of singular values. Since $\mathbf{D}$ is diagonal, $\mathbf{D}^\mathsf{T} = \mathbf{D}$.

Now consider the product $\mathbf{A}^\mathsf{T}\mathbf{A}$ that arises in the normal equation. Substituting $\mathbf{A} = \mathbf{U}\mathbf{D}\mathbf{V}^\mathsf{T}$ in this product gives:

$$\mathbf{A}^\mathsf{T}\mathbf{A} = (\mathbf{U}\mathbf{D}\mathbf{V}^\mathsf{T})^\mathsf{T}\mathbf{U}\mathbf{D}\mathbf{V}^\mathsf{T} = \mathbf{V}\mathbf{D}^\mathsf{T}\mathbf{U}^\mathsf{T}\mathbf{U}\mathbf{D}\mathbf{V}^\mathsf{T} = \mathbf{V}\mathbf{D}^\mathsf{T}\mathbf{D}\mathbf{V}^\mathsf{T} = \mathbf{V}\mathbf{D}^2\mathbf{V}^\mathsf{T}.$$

We can thus express the normal equation in a much simplified form:

$$\begin{aligned}
\mathbf{A}^\mathsf{T}\mathbf{A}\mathbf{u} &= \mathbf{A}^\mathsf{T}\mathbf{b} \\
\implies \mathbf{V}\mathbf{D}^2\mathbf{V}^\mathsf{T}\mathbf{u}^* &= \mathbf{V}\mathbf{D}\mathbf{U}^\mathsf{T}\mathbf{b} \\
\implies \mathbf{D}^2\mathbf{V}^\mathsf{T}\mathbf{u}^* &= \mathbf{D}\mathbf{U}^\mathsf{T}\mathbf{b} \\
\implies \mathbf{V}^\mathsf{T}\mathbf{u}^* &= \underbrace{\left(\mathbf{D}^2\right)^{-1}\mathbf{D}}_{\mathbf{D}^+}\mathbf{U}^\mathsf{T}\mathbf{b} \\
\implies \mathbf{u}^* &= \mathbf{V}\mathbf{D}^+\mathbf{U}^\mathsf{T}\mathbf{b}.
\end{aligned}$$

The matrix $\mathbf{D}^+$ is called the "*pseudoinverse*" of $\mathbf{D}$ and is defined as follows: $\mathbf{D}^+ = \mathrm{diag}\left\{\sigma_1^+, \ldots, \sigma_n^+\right\}$ where

$$\sigma_i^+ = \begin{cases} \sigma_i^{-1} = \frac{1}{\sigma_i} & \text{if } \sigma_i > 0 \\ 0 & \text{otherwise.} \end{cases}$$

Thus if $\mathrm{rank}(\mathbf{A}) = n$, then all the singular values of $\mathbf{A}$ are non-zero, in which case $\mathbf{D}^+ = \mathbf{D}^{-1} = \mathrm{diag}\left\{\frac{1}{\sigma_1}, \ldots, \frac{1}{\sigma_n}\right\}$. In this case, In the former case, $\mathbf{D}\mathbf{D}^+ = \mathbf{D}^+\mathbf{D} = \mathbf{I}_n$.
However, if $\mathrm{rank}(\mathbf{A}) = r < n$, there are only $r < n$ non-zero singular values and $\mathbf{D}^+ = \mathrm{diag}\{\frac{1}{\sigma_1}, \ldots, \frac{1}{\sigma_k}, \underbrace{0, \ldots, 0}_{n-r \text{ zeros}}\}$. In this case, $\mathbf{D}\mathbf{D}^+ = \mathbf{D}^+\mathbf{D} = \mathrm{diag}\{1, \ldots, 1, \underbrace{0, \ldots, 0}_{n-r \text{ zeros}}\}$ — which is very close to, but not quite, the identity matrix.
We call the product matrix $\mathbf{V}\mathbf{D}^+\mathbf{U}^\mathsf{T}$ the *pseudoinverse* of $\mathbf{A}$, written $\mathbf{A}^+$. That is ,

$$\mathbf{A}^+ = \mathbf{V}\mathbf{D}^+\mathbf{U}^\mathsf{T}.$$

If $\mathrm{rank}(\mathbf{A}) = n$, then $\mathbf{A}^+\mathbf{A} = \mathbf{V}\mathbf{D}^+\mathbf{U}^\mathsf{T}\mathbf{U}\mathbf{D}\mathbf{V}^\mathsf{T} = \mathbf{V}\mathbf{D}^+\mathbf{D}\mathbf{V}^\mathsf{T} = \mathbf{V}\mathbf{V}^\mathsf{T} = \mathbf{I}_n$, and $\mathbf{A}^+ = \mathbf{A}^{-1}$.

Recall that the matrix $\mathbf{A}^{\mathrm{T}}\mathbf{A}$ is ill-conditioned when the smallest singular value, σ_n, is very small. This leads to instability in computing the solution to the normal equation. The pseudo-inverse method provides a way of removing this instability to get an approximate but stable solution by simply removing the smallest singular value or values.

9.4.1 Properties of the pseudo inverse $\mathbf{A}^+$

The pseudo-inverse of $\mathbf{A}$ always exists (all we need to do is calculate the SVD and form the product described above).

The SVD of $\mathbf{A}$ gives $\mathbf{A} = \mathbf{U}\mathbf{D}\mathbf{V}^{\mathrm{T}}$ from which we get $\mathbf{A}\mathbf{V} = \mathbf{U}\mathbf{D}$ or, considering the individual columns, $\mathbf{A}\mathbf{v}_i = \sigma_i \mathbf{u}_i$.

- If $\mathbf{A}$ is a square matrix such that $\mathbf{A}^{-1}$ exists, then the singular values for $\mathbf{A}^{-1}$ are $\sigma^{-1} = \frac{1}{\sigma}$ and $\mathbf{A}^{-1}\mathbf{u}_i = \frac{1}{\sigma_i}\mathbf{v}_i$
- If $\mathbf{A}^{-1}$ does not exist, then the pseudoinverse matrix $\mathbf{A}^+$ does exist such that:

$$\mathbf{A}^+\mathbf{u}_i = \begin{cases} \frac{1}{\sigma_i}\mathbf{v}_i & \text{if} \quad i \le r = \operatorname{rank}(\mathbf{A}) \text{ i.e. if } \sigma_i > 0 \\ 0 & \text{for} \quad i > r. \end{cases}$$

- Pseudoinverse matrix $\mathbf{A}^+$ has the same rank r as $\mathbf{A}$
- The matrices $\mathbf{A}\mathbf{A}^+$ and $\mathbf{A}^+\mathbf{A}$ are also as near as possible to the $m \times m$ and $n \times n$ identity matrices, respectively
- $\mathbf{A}\mathbf{A}^+$ – the $m \times m$ projection matrix onto the column space of $\mathbf{A}$
- $\mathbf{A}^+\mathbf{A}$ – the $n \times n$ projection matrix onto the row space of $\mathbf{A}$

Example: Find the pseudo-inverse of $\mathbf{A} = \begin{bmatrix} 0 & 1 \\ 1 & 1 \\ 1 & 0 \end{bmatrix}$.

Solution: The singular value decomposition of $\mathbf{A}$ is

$$\mathbf{A} = \mathbf{U}\mathbf{D}\mathbf{V}^{\top} = \underbrace{\begin{bmatrix} \frac{1}{\sqrt{6}} & \frac{1}{\sqrt{2}} \\ \frac{2}{\sqrt{6}} & 0 \\ \frac{1}{\sqrt{6}} & -\frac{1}{\sqrt{2}} \end{bmatrix}}_{\mathbf{U}} \underbrace{\begin{bmatrix} \sqrt{3} & 0 \\ 0 & 1 \end{bmatrix}}_{\mathbf{D}} \underbrace{\begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix}}_{\mathbf{V}^{\mathrm{T}}}.$$

So the pseudo-inverse of $\mathbf{A}$ is

$$\begin{aligned} \mathbf{A}^+ = \mathbf{V}\mathbf{D}^+\mathbf{U}^{\top} &= \underbrace{\begin{bmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix}}_{\mathbf{V}} \underbrace{\begin{bmatrix} \frac{1}{\sqrt{3}} & 0 \\ 0 & 1 \end{bmatrix}}_{\mathbf{D}^+} \underbrace{\begin{bmatrix} \frac{1}{\sqrt{6}} & \frac{2}{\sqrt{6}} & \frac{1}{\sqrt{6}} \\ \frac{1}{\sqrt{2}} & 0 & -\frac{1}{\sqrt{2}} \end{bmatrix}}_{\mathbf{U}^{\mathrm{T}}} \\ &= \begin{bmatrix} -\frac{1}{3} & \frac{1}{3} & \frac{2}{3} \\ \frac{2}{3} & \frac{1}{3} & -\frac{1}{3} \end{bmatrix}. \end{aligned}$$

Now let's check the products $\mathbf{AA}^+$ and $\mathbf{A}^+\mathbf{A}$:

$$\mathbf{AA}^+ = \begin{bmatrix} 0 & 1 \\ 1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} -\frac{1}{3} & \frac{1}{3} & \frac{2}{3} \\ \frac{2}{3} & \frac{1}{3} & -\frac{1}{3} \end{bmatrix} = \begin{bmatrix} \frac{2}{3} & \frac{1}{3} & -\frac{1}{3} \\ \frac{1}{3} & \frac{2}{3} & \frac{1}{3} \\ -\frac{1}{3} & \frac{1}{3} & \frac{2}{3} \end{bmatrix}$$

while

$$\mathbf{A}^+\mathbf{A} = \begin{bmatrix} -\frac{1}{3} & \frac{1}{3} & \frac{2}{3} \\ \frac{2}{3} & \frac{1}{3} & -\frac{1}{3} \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

□

We end our discussion on computational methods in linear algebra here. We have only touched on a small number of the wide variety of techniques used in this area, but we have tried to direct our attention to some of the more common and useful techniques. This is a rich and extremely useful area of study and we encourage interested students to look into fields where these tools and ideas are applied and explored further including computer vision, engineering, physics, applied mathematics and statistics.

10 Introduction to stochastic processes and probability

Models and methods that have been considered so far in the course have been deterministic — a single input produces the same answer every time while solutions to problems are aimed at finding one correct answer or a close approximations to it. These deterministic models, methods and approximations can be very accurate, particularly in engineering and physical applications.

Many systems, however, are inherently random. Apparently identical inputs may produce radically different outputs and no two realisations of the system are exactly the same. Whether that randomness is the result of our imprecise measurements (were the inputs exactly the same?) or there is a fundamental randomness built into the system (quantum uncertainty?), we should attempt to model and quantify this uncertainty. We regularly refer to non-deterministic systems as *stochastic* rather than random to avoid the common usage of random (where it is often used to mean uniformly random where every outcome is equally likely).

The framework we use to make these models is probability theory and, ideally, we use statistical inference to find the relationship between our models and the system we are studying. Simulation is one of the tools we use to understand how our models behave and is often used where exact statistical inference is prohibitively difficult. Both statistical inference and simulation rely heavily on computational power and algorithms. Model building is typically more of an art than a science and is done by hand (or mind).

In this section, we look at some of the basic terminology of probability theory, introduce the fundamental ideas behind statistical inference and see how we can simulate stochastic processes *in silico*.

11 Primer on Probability

The basic challenge of probability theory and applied probability is to understand and describe the laws according to which events occur.

A event can be pretty much anything. Commonly used examples in probability are rolling dice, picking balls out of an urn or tossing a coin — these are commonly used because they are simple, easy to understand and aid our intuition. But all sorts of events can be thought of as random: the amount of rain falling in an area in a given period, the number of mutations that occur when a cell splits, the age of the person currently reading this sentence. Indeed, if we consider randomness to a a property of our state of knowledge of an event, any event can be considered random.

Formally, we define a *probability* to be a number between 0 and 1 assigned to a set of outcomes of a random process called an *event*. This number is typically interpreted as the chance of the event occurring or as the degree of plausibility we place on the event occurring.

The set of all outcomes that the random process can take is known as the *state space* and is often denoted Ω.

An *event*, A, is a subset of the state space: $A \subseteq \Omega$. When Ω is finite or countable, probability can be viewed as a function from the set of all subsets of Ω, written $\mathcal{A}$, to the interval $[0, 1]$, that is $P : \mathcal{A} \to [0, 1]$. $\mathcal{A}$, the set of all subsets of Ω, is called the *power set* of Ω. That is, for some event or collection of events, we view the function P as giving the probability of that event occurring.

This interpretation is not mathematically correct when Ω is uncountable (for example, when our random process can take any value in continuous interval) but it will be sufficient to guide our intuition here.

Example: Tossing a coin 2 times and recording the result of each toss. The random process is tossing the coin twice. The state space is the set of all possible outcomes: $\Omega = \{HH, HT, TH, TT\}$.

An example of an event is that we throw a tails first: in this case $A = \{TH, TT\}$.

There are $2^4 = 16$ possible events that we could consider here as that is the size of the power-set (the set of all possible subsets) of Ω. For completeness, we write down all possible events: $\mathcal{A} = \{\emptyset, \{HH\}, \{HT\}, \{HT\}, \{TT\}, \{HH, HT\}, \{HH, TH\}, \{HH, TT\}, \{HT, TH\}, \{HT, TT\}, \{TH, TT\}, \{HH, HT, TH\}, \{HH, HT, TT\}, \{HH, TH, TT\}, \{HT, TH, TT\}, \Omega\}$. □

Note that sometimes we distinguish between simple and compound events. In the above example, the simple events are $\{HH\}, \{HT\}, \{HT\}, \{TT\}$ while compound events are combinations of simple events.

Example: Length of time waiting for bus, measured from arrival at bus stop until bus arrives. Supposing the buses come every 15 mins. Then $\Omega = [0, 15]$ (that is, any time in the interval between 0 and 15 minutes). An example of an event is $A = [0, 1]$ being the event that the wait for the bus is at most 1 minute. □

Let A and B be events. Then the event C that A and B occur is given by $C = A \cap B$ while the event F that A or B occurs is given by $D = A \cup B$. The event A does not occur is given by $A^c = \bar{A} = \Omega - A = \Omega \backslash A = \{\omega \in \Omega : \omega \notin A\}$.

Example: Suppose we roll a fair die and record the value. Then $\Omega = \{1, 2, 3, 4, 5, 6\}$. Let A be the event that the roll is even, B be the event that we roll a 3 or a 6. Then $A = \{2, 4, 6\}$ and $B = \{3, 6\}$. The event that A and B occur is $A \cap B = \{6\}$ while the event that A or B happens is $A \cup B = \{2, 3, 4, 6\}$. The event that A does not occur is $A^c = \{1, 3, 5\}$ (A does not occur when the roll is odd). $\square$

11.1 Conditional probability and independence

For events A and B, if we know that B occurred, what can we say about the probability of A given that knowledge? This is captured by the concept of the *conditional probability of A given B* is written $P(A|B)$ and defined by

$$P(A|B) = \frac{P(A \cap B)}{P(B)}.$$

This is only defined where $P(B) > 0$.

Example: Suppose we roll a fair die and record the value. Let A be the event that 2 is rolled and B be the event that the roll is an even number. What is the probability that the roll is a two given that we know it is even? This is just $P(A|B)$. We calculate it as follows. $P(B) = 1/2$ and $P(A \cap B) = P(A) = 1/6$ since $A \cap B = A$. Thus

$$P(A|B) = \frac{P(A \cap B)}{P(B)} = \frac{1/6}{1/2} = \frac{1}{3}.$$

$\square$

We say that events A and B are *independent* when $P(A \cap B) = P(A)P(B)$. From the definition of conditional probability, it is clear that if A and B are independent, then $P(A|B) = P(A)$ and $P(B|A) = P(A)$.

Note that we usually write $P(A, B)$ instead of $P(A \cap B)$. More generally, we write $P(A_1, \ldots, A_k)$ for $P(\bigcap_{i=1}^k A_i)$.

Rearranging the definition of conditional probability, we see that $P(A, B) = P(A|B)P(B) = P(B|A)P(B)$. Repeated applications of this result gives

$$P(A_1, \ldots, A_k) = P(A_1|A_2, \ldots, A_k)P(A_2|A_3, \ldots, A_k) \ldots P(A_{k-1}|A_k)P(A_k).$$

11.2 Bayes' Theorem

From the definition of conditional probability, we can prove the following result, known as Bayes' theorem.

$$P(B|A) = \frac{P(A|B)P(B)}{P(A)}.$$

This simple result is important because it tells us how the forward probability $P(A|B)$ is related to the backward probability $P(B|A)$. We'll see that this relationship is crucial to statistical inference.

11.3 Random variables

A *random variable* (r.v.) X is a variable whose value results from the measurement of a random process. That is, a random variable is a measurement of some random event. We use capital letters to denote random variables while lower-case letters to denote particular observations or *realisations* of the random variable. So $X = x$ is the event that the random variable X takes the particular value x.

A **discrete** random variable takes a finite or countably infinite number of values, while a **continuous** random variable can take an uncountable number of possible values.

Random variables are most commonly real valued (that is, their value is a real number) but they can take any value. For example, we could consider random sequences, random graphs or random trees. For now, lets stick with real valued random variables. Formally, a real-valued random variable is a map from events to the real numbers: $X : \Omega \to \mathbb{R}$.

For discrete random variables, the **probability distribution function** (pdf) or **probability mass function** (pmf) is a function (rule, table) that assigns probabilities to each possible value of X.

$P(X = x) = p(x)$ is the probability that $X = x$. Sometimes write $P(X = x) = p_x$ or $P(X = x) = f(x)$.

As usual, we have $0 \leq P(X = x) \leq 1$ and $\sum_x P(X = x) = 1$.

For continuous random variables, the probability that a random variable takes any one exact value is zero, that is $P(X = x) = 0$, so we consider instead the **probability density function**, $p_X(x)$ (also written $f_X(x)$ or $f(x)$) from which we can calculate the probability that X lies in the interval $[a, b]$:

$$Pr(a \leq X \leq b) = \int_a^b p_X(x)dx.$$

$p_X(x)$ is real-valued, non-negative and normalised, i.e.,

$$\int_{-\infty}^{\infty} p_x(x)dx = 1.$$

Note that the integral $\int_a^b p_X(x)dx$ gives the area under the curve $p_X(x)$ between $x = a$ and $x = b$.

The **cumulative distribution function** (cdf), or simply the *distribution function*, is defined, for both discrete and continuous random variables, by $F(x) = P(X \leq x)$. This is a function that is monotonically increasing from 0 to 1. For continuous random variables, the cdf is continuous, while for discrete random variables, it is a step function with dis-continuities.

These ideas immediately extend to multiple random variables, so that the **joint probability density function** of n random valuables $X_1, \dots, X_n$ takes n arguments, $p_{X_1,\dots,X_n}(x_1, \dots, x_n)$ that is real-valued, non-negative and normalised. The probability that the point $(X_1, \dots, X_n)$ lies in some region is just the multiple integral over that region.

Given a joint probability density function, we obtain the probability density for a subset of the variables by integrating over the ones not in the subset. For example, given $p_{XY}(x, y)$, we have

$$p_X(x) = \int_{-\infty}^{\infty} p_{XY}(x, y) dy.$$

This process is known as **marginalization**. The process is the same for a discrete variable, if we replace the integral with a sum:

$$P(x) = \sum_{y \in \mathcal{Y}} P(x, y)$$

Two random variables X and Y are **independent** when

$$p_{XY}(x, y) = p_X(x) p_Y(y).$$

Equivalently, X and Y are independent when

$$p_{Y|X}(y|x) = p_Y(y).$$

The **expected value** of a random variable X is called the mean and is given by

$$E[X] = \int_{-\infty}^{\infty} x p_X(x) dx.$$

For discrete random variables, this is written

$$E[X] = \sum_{x \in \mathcal{X}} x p_x.$$

The symbol μ is often used for the mean.

The **variance** of a random variable is $\mathrm{Var}(X) = E[(X - E[X])^2] = E[X^2] - (E[X])^2$. The variance is a measure of the spread of a random variable about its mean.

The **expected value of a function f of X** is

$$E[f(X)] = \int_{-\infty}^{\infty} f(x) p_X(x) dx.$$

11.4 Commonly used distributions

In this course, we'll primarily be discussing bioinformatics where some commonly used discrete probability distribution functions are: Bernoulli, geometric, binomial, uniform and Poisson. Commonly used continuous distributions are uniform, normal (Gaussian), exponential, and gamma. Those are briefly described below. For more thorough descriptions, refer to any decent statistics text or, more simply, the relevant Wikipedia entries.

11.4.1 Bernoulli distribution

A random variable X with a *Bernoulli distribution* takes values 0 and 1. It takes the value 1 on a 'success' which occurs with probability p where $0 \leq p \leq 1$. It takes value 0 on a failure with probability $q = 1 - p$. Thus it has the single parameter p. If X is Bernoulli, $E[X] = q \cdot 0 + p \cdot 1 = p$ and $\mathrm{Var}(X) = E[X^2] - E[X]^2 = q \cdot 0^2 + p \cdot 1^2 - p^2 = pq$.

11.4.2 Geometric distribution

X has a *geometric distribution* when it is the number of Bernoulli trials that fail before the first success. It therefore takes values in $\{0, 1, 2, 3, \ldots\}$. If the Bernoulli trials have probability p of success, the pdf for X is $P(X = x) = (1 - p)^x p$. If X is geometric,

$$E[X] = \frac{q}{p} \text{ and } \mathrm{Var}(X) = \frac{q}{p^2}.$$

Note that the Geometric distribution can be defined instead as the total number of trials required to get a single success. This version of the geometric can only take values in $\{1, 2, 3, \ldots\}$. The pdf, mean and variance all need to be adjusted accordingly.

11.4.3 Binomial distribution

X has a *binomial distribution* when it represents the number of successes in n Bernoulli trials. There are thus two parameters required to describe a binomial random variable: n, the number of Bernoulli trials undertaken, and p, the probability of success in the Bernoulli trials. The pdf for X is

$$f(x) = P(X = x) = \binom{n}{x} p^x (1 - p)^{n-x} \text{ for } x = 0, 1, 2, \ldots, n.$$

where $\binom{n}{x} = \frac{n!}{x!(n-x)!}$. For a binomial variable X,

$$E[X] = np \text{ and } Var[X] = np(1 - p) = npq.$$

We write $X \sim Bin(n, p)$ when X has a binomial distribution with parameters n and p.

11.4.4 Poisson distribution

The *Poisson distribution* is used to model the number of rare events that occur in a period of time. The events are considered to occur independently of each other. The distribution has a single parameter, λ, and probability density function

$$f(x) = \exp(-\lambda)\frac{\lambda^x}{x!} \text{ for } x = 0, 1, 2, 3, \ldots.$$

If X is Poisson,

$$E[X] = \lambda \text{ and } Var[X] = \lambda.$$

We write $X \sim Poiss(\lambda)$ when X has a Poisson distribution with parameter λ.

11.4.5 Uniform distribution (discrete or continuous)

Under the *uniform distribution*, all possible values are equally likely. So if X is discrete and takes n possible values, $P(X = x_i) = 1/n$ for all x_i.

If X is continuous and uniform over the interval $[a, b]$, the density function is $f(x) = \frac{1}{b-a}$. In this case, write $X \sim U([a, b])$.

11.4.6 Normal distribution

The *Normal*, or Gaussian, distribution, with mean μ and variance σ^2, $(\mu \in \mathbb{R}; \sigma > 0)$ has density function

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left\{-\frac{1}{2\sigma^2}(x-\mu)^2\right\}$$

We write $X \sim N(\mu, \sigma^2)$.

The normal distribution is a widely used distribution in statistical modelling for a number of reasons. A primary reason is that it arises as a consequence of the central limit theorem which says that (under a few weak assumptions) the sum of a set of identical random variables is well approximated by a normal distribution. Thus when random effects all add together, they often result in a normal distribution. Measurement error terms are typically modelled as normally distributed.

11.4.7 Exponential distribution

The *Exponential* distribution describes the time between rare events so always takes non-negative values. It has a single parameter, λ known as the rate and has density function

$$f(x) = \lambda e^{-\lambda x},$$

where $x \geq 0$. If X is exponentially distributed,

$$E[X] = \frac{1}{\lambda} \text{ and } \mathrm{Var}(X) = \frac{1}{\lambda^2}.$$

Write $X \sim Exp(\lambda)$.

11.4.8 Gamma distribution

The *Gamma* distribution arises as the sum of a number of exponentials. It has two parameters, k and θ, called the shape and scale, respectively. These parameters can be used to specify the mean and variance of the distribution.

$$f(x) = \frac{1}{\theta^k \Gamma(k)} x^{k-1} \exp(-x/\theta) \text{ for } x > 0,$$

where $\Gamma(k) = \int_0^\infty t^{k-1} e^{-t}\, dt$ is the gamma function (the extension of the factorial function, $k!$, to all real numbers). The mean and variance of a gamma distributed random variable X is

$$E[X] = k\theta \text{ and } \mathrm{Var}(X) = k\theta^2.$$

Write $X \sim \mathit{Gamma}(k, \theta)$.

Note that the gamma distribution has different parametrisations which result in different looking (but mathematically equivalent) expressions for the density, mean and variance — be sure to check which parametrisation is being used.

12 Inference

Let's consider how we pose and tackle problems in a statistical framework. Suppose we have a statistical model for some real process (from biology, physics, sociology, psychology etc...). By having a model, we mean that given a set of control parameters, θ, we can predict the outcome of the system, D. For example, our model may be that each element of D is a draw from one of the distributions we described above so the control parameters θ are just the parameter(s) of that distribution. Note that the model of the process may include our (imperfect or incomplete) method of measuring the outcome.

In an abstract sense, then, we can consider the model as a black box with input vector θ (the parameters) and output vector D, the data.

$$\begin{tikzcd}
\begin{array}{c}\text{Parameter vector}\\ \theta\end{array} \arrow[r] & \boxed{\text{Model of system}} \arrow[r] & \begin{array}{c}\text{Data}\\ \text{D}\end{array}
\end{tikzcd}$$

The model gives us the forward probability density of the outcome given the parameter, that is, $P(D|\theta)$. This density is the called the likelihood, although, as we see below, we don't usually consider it as a density in the usual way.

This model is not deterministic. The data D can be seen as a random sample from the probability distribution defined by the model (and parameters). Changing the value of the parameters typically does not change the possible outcomes of the model but it will change the shape of the probability distribution, making some outcomes more likely, others less likely.

Example: Suppose we are interested the number of buses stopping at a bus stop over the course of an hour. We watch for the hour between 8am and 9am every weekday morning for 2 weeks. We observe the outcomes $D = (10, 7, 5, 6, 12, 9, 10, 5, 14, 7)$. A sensible model here might be the Poisson distribution where we say that the number of bus arrivals in an interval is Poisson distributed with parameter λ. Our parameter vector contains just the single parameter $\theta = (\lambda)$ and our data vector contains the 10 observed outcomes $D = (D_1, D_2, \ldots, D_{10}) = (10, 7, 5, 6, 12, 9, 10, 5, 14, 7)$.

We derive the likelihood as follows.

The probability of observing the data D for a given value of λ is $P(D|\lambda)$. Let's assume that each observation is independent of others then $P(D|\lambda) = \prod_i P(D_i|\lambda)$. That is, the probability of observing this series of outcomes is just the product of the probabilities of observing each particular outcome.

The likelihood of a single observation is given by the probability distribution function for the Poisson since $D_i \sim Poiss(\lambda)$ so:

$$P(D_i|\lambda) = \frac{\lambda^{D_i}}{D_i!} e^{-\lambda}.$$

And so the likelihood of observing the full data D is just

$$P(D|\lambda) = \prod_i P(D_i|\lambda) = \prod_i \frac{\lambda^{D_i}}{D_i!} e^{-\lambda}.$$

$\square$

Note that the likelihood is a probability density function for D. But D is typically fixed in the sense that we make the observations which remain fixed through-out the analysis. We will be interested in considering the likelihood as a function of the parameters θ. The likelihood is *not* a probability density function for θ since, in general $\int_{\theta\in\Theta} P(D|\theta)\, d\theta \neq 1$.

12.1 Bayesian inference

The statistical problem essentially comes down to one of observing the outcome, D and wanting to recover the parameters θ.

That is, we want to estimate θ given D. We summarise our estimate of θ as a probability distribution, conditional on having observed D: $P(\theta|D)$. This is called the **posterior distribution** of θ.

From Bayes' theorem, we can express the posterior in terms of the likelihood:

$$P(\theta|D) = \frac{P(D|\theta)P(\theta)}{P(D)},$$

where $P(D|\theta)$ is the **likelihood**, $P(\theta)$ is the **prior distribution** of θ and $P(D)$ is a normalisation constant.

The prior $p(\theta)$ summarises what we know about a parameter before making any observations.

The posterior, $p(\theta|D)$ summarises what we know about θ after observing the data.

The likelihood tells us about the likeliness of the data under the model for any value of θ. Recall that we consider the likelihood a function of θ rather than a probability density for D; to emphasise this fact, people often write it explicitly as a function of θ: $L(\theta) = P(D|\theta)$.

Bayes' theorem tells us how we update our beliefs given new data. Our updated beliefs about θ are encapsulated in the posterior, while are initial beliefs are encapsulated in the prior. Bayes' theorem simply tells us that that we obtain the posterior by multiplying the prior by the likelihood (and dividing by $P(D)$ which we can think of as a normalisation constant).

Note that we need the normalisation constant as the posterior is a probability distribution for θ, so its density must integrate to 1, i.e., $\int_{\theta\in\Theta} f(D|\theta)\, d\theta = 1$. Thus the normalisation constant is $P(D) = \int_{\theta\in\Theta} P(D|\theta)P(\theta)\, d\theta$. Typically this integral is hard to calculate so we try to find that will avoid having to calculate it.

Example: In the example above, we found an expression for the likelihood. To find an expression for the posterior, we need to decide on a prior distribution. Suppose we had looked up general info about bus stops in the city and found that the busiest stop had an average of 30 buses an hour while the quietest had an average of less than 1 bus per hour. We use this prior information to say that any rate parameter λ producing an average of between 0 ($\lambda = 0$) and 30 ($\lambda = 30$) buses an hour is equally likely. This leads us to the prior $\lambda \sim U(0, 30)$. The density of this prior is $f(\lambda) = 1/30$ for $0 \leq \lambda \leq 30$.

To get the posterior density, we use the formula above:

$$f(\lambda|D) = \frac{f(D|\lambda)f(\lambda)}{P(D)} = \frac{\prod_i \frac{\lambda^{D_i}}{D_i!} e^{-\lambda} \frac{1}{30}}{P(D)}.$$

The normalisation constant $P(D)$ is the integral of the numerator over all possible values of λ:

$$P(D) = \int_0^{30} \prod_i \frac{\lambda^{D_i}}{D_i!} e^{-\lambda} \frac{1}{30} \, d\lambda.$$

$\square$

While it is possible to calculate this particular integral analytically, for most posterior distributions analytical integration is either very difficult or impossible. We'll investigate methods for avoiding calculating difficult integrals like this in later sections.

12.2 Maximum likelihood

It is often difficult or inconvenient to deal with the posterior distribution (when the prior is hard to specify or the normalisation constant is impossible to calculate). In these cases, we can still use our probabilistic model by concentrating solely on the likelihood function. The aim here is typically to find the parameters that maximise the likelihood function. That is, those parameters under which the observed data is most likely. We call this parameter estimate the maximum likelihood estimate and write it as

$$\hat{\theta} = \arg\max_{\theta} f(D|\theta) = \arg\max_{\theta} L(\theta; D)$$

This function can be maximise using standard tools from calculus (taking the derivative and setting it to zero – it is often easier to work with the log of the likelihood function as they both share a maximum) or using numerical techniques such as hill-climbing.

Many methods in statistics are based on maximum likelihood including regression, χ^2-tests, $t-$tests, ANOVA and so on.

Example: In the bus example above, we could find the maximum likelihood estimator for λ by differentiating the log-likelihood, $\log(L(\lambda; D))$ with respect to λ, setting the result to zero and solving. Note that we often work with the log-likelihood rather than the likelihood for a couple of reasons: it is often easier algebraically and it helps avoid numerical under-flow when the likelihood itself is very small.

13 Simulation

In a statistical setting there are a number of reasons we may wish to simulate from a distribution or a stochastic process. We may wish to get a feeling for how the process behaves or estimate some quantity that we cannot calculate analytically. An example of the latter case arises in Bayesian statistics, where the aim is to find the posterior distribution $f(\theta|x)$. This can be very difficult for two main reasons:

- The normalising constant ($P(D)$, above) involves a $p-$dimensional integral (where p is the number of parameters of the model, that is, $\theta = (\theta_1, \theta_2, \ldots, \theta_p)$) which is often impossible to calculate analytically.
- Even if we are able to find $f(\theta|x)$, if we want to find the marginal distribution for some part of θ this may again involve a high-dimensional integral.

Both of these reasons boil down to the fact that integration is hard.

To get around this problem, our approach will be to obtain a sample of values, $\{\theta^{(i)}\}$ for $i = 1, \ldots, n$, from the distribution of interest and use this sample to estimate properties of the distribution.

For example, the mean of the distribution is $E[\theta] = \sum_{\theta\in\Omega} \theta \Pr(\theta)$. We can estimate this from the sample set by $E[\theta] \approx \bar{\theta} = \frac{1}{n}\sum_1^n \theta^{(i)}$. This is called the sample mean of θ, and is indicated by the bar over the variable. The sample mean is an *estimator* of the mean. More generally, we estimate the mean of a function of θ by $E[g(\theta)] \approx \bar{g}(\theta) = \frac{1}{n}\sum_1^n g(\theta^{(i)})$.

How good are these estimates? Since each sample $\theta^{(i)}$ is a random variable, $\bar{g}$ is a random variable. That is, each time we obtain a different sample of values of θ, we will get a different value for $\bar{g}$. Clearly, as n, the size of the sample, increases our estimate will become more accurate but by how much?

It turns out that under quite general conditions, the main being that the samples, $\theta^{(i)}$ are independent of each other, $\bar{g}$ is normally distributed with mean $E[g]$ and variance $\mathrm{var}(\bar{g}) = \mathrm{var}(g)/n$. When stated formally, this is known as a central limit theorem.

Thus, if we have a method of simulating lots of independent samples, we can quickly get extremely accurate estimates of the quantities we are interested in.

Note that we can estimate more complex things than simple means using these methods. For example, we can estimate the shape of distributions by drawing a histogram of the sampled points.

So our attention turns to how we can generate this random sample. First we consider how we can generate or simulate randomness at all using (deterministic) computers.

13.1 Random number generation

All simulation relies on a ready supply of random numbers. There are currently no known methods to generate truly random numbers with a computer without measuring some physical process. There are, however, many fast and efficient methods for generating pseudo-random numbers that, for most applications, are completely sufficient. The fact that these are based on algorithms that are repeatable makes them superior to physically based rngs for scientific simulation purposes.

We do not go into the mathematical details of pseudo-random number generators here as most major languages have libraries that implement perfectly adequate algorithms. It is worth considering briefly what we want in a RNG. The following quality criteria are taken from L'Ecuyer in the Handbook of Computational Statistics, 2004.

The RNG must:

- have a very long period so that no repetitions of the cycle occur;
- be efficient in memory and speed;
- repeatable so that simulations can be reproduced;
- portable between implementations;
- have efficient jump-ahead methods so that a number of virtual streams of random numbers can be created from a single stream for use in parallel computations; and,
- have good statistical properties approximating the uniform distribution on $[0, 1]$.

It is relatively simple to come up with rngs that satisfy the first of these criteria, yet the last is where the difficulties occur. The performance of rngs can be tested via the diehard test suite (or more recently, the dieharder suite). See `http://www.phy.duke.edu/~rgb/General/dieharder.php`.

13.2 Simulating from univariate distributions via Inversion sampling

Simulation from discrete or continuous distributions with cumulative density function $F(X)$ relies on the following result which tells us that all we need to simulate draws from an arbitrary univariate distribution is a draw from $U(0, 1)$ and use the inverse of the cdf:

Result (Inversion method): If $U \sim U(0, 1)$, then $X = F^{-1}(U)$ produces a draw from X where F^{-1} is the inverse of X.

Thus when the cdf is known and we can find the inverse, sampling from the distribution is easy, as the following example shows.

Example (simulating an exponential random variable): if $X \sim \text{Exp}(\lambda)$, then $F(x) = \int_{-\infty}^{x} f(t)dt = \int_{-\infty}^{x} \lambda e^{-\lambda t}dt = 1 - e^{-\lambda x}$.

It is simple to see (by setting $u = 1-e^{-\lambda y}$ and solving for y) that $F^{-1}(u) = -\log(1-u)/\lambda$.

Since $1 - U \sim U(0, 1)$ when $U \sim U(0, 1)$, we can use this expression to generate exponential random variables by generating the uniform random variable u and setting $x = -\log(u)/\lambda$. $\square$

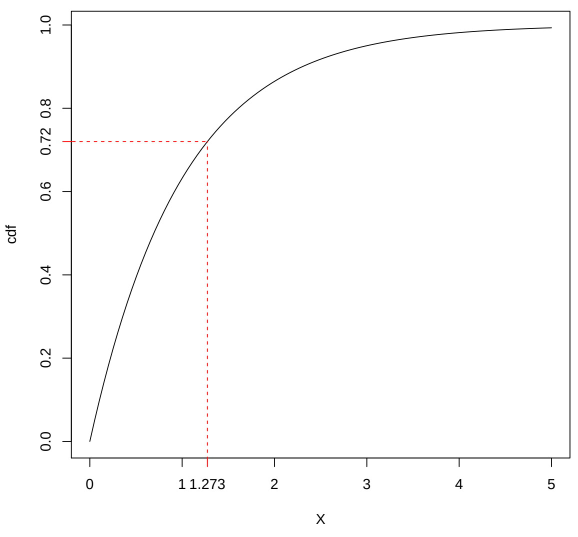


Figure 4: Sampling an $Exp(1)$ random variable using the inversion method. A uniform sample $u \sim U(0,1)$ is drawn. Here $u = 0.72$ shown on the vertical axis. This is mapped, via the cdf, to $x \sim Exp(1)$ to produce $x = -\log(1-u) = 1.272$ shown on the horizontal axis.

Note that with discrete random variables, a inverse of F is ambiguously defined (since F is a step function). It is possible to extend the definition of an inverse to derive the following method for simulating discrete random variables.

Inversion sampling from a discrete distribution: If X is discrete with $P(X = x_i) = p_i$, we generate $U \sim U(0,1)$ and set $X = x_1$ if $u < p_1$ and $X = x_i$ if $\sum_{j=1}^{i-1} p_j < u < \sum_{j=1}^{i} p_j$.

Example: Use the inversion method to obtain samples from $X \sim Binomial$(n = 5, p = 0.3).

Solution: The possible values X can take are $(0,1,2,3,4,5)$ with probabilities $f(x) = (0.168, 0.360, 0.309, 0.132, 0.028, 0.002)$, respectively (from Section 11.5.3). Obtain the cdf by taking the cumulative sum of these probabilities: $F(X) = (0.168, 0.528, 0.837, 0.969, 0.998, 1.000)$. Now obtain samples from X by sampling $u \sim U(0,1)$ and finding the index of the smallest value of the cdf which is larger than u. E.g. if $u = 0.439$, $x = 1$ since $F(0) = 0.168 < u < F(1) = 0.528$, Similarly, if $u = 0.972$, $x = 4$ since $F(3) < u < F(4)$. $\square$

13.3 Stochastic processes

So far we have talked about random variables which can be pretty much any object but are only observed at one time. A stochastic process is a collection of random variables that describes the evolution of a system that is subject to randomness. A stochastic process could, for example, describe the position of a particle that is being buffeted by other particles, the state of a genetic sequence that is subject to copying with mutation, or the shape and size of a land mass that is subject to geological forces.

Mathematically, we consider a random process X as the set of random variables $\{X_t : t \in T\}$ where T is some index set, such as discrete time ($T = 0, 1, 2, \ldots$) or continuous time ($T = [0, \infty)$).

We will try to get an understanding of these process by studying a few examples.

13.3.1 Random walk

One of the simplest stochastic processes is known as the simple symmetric random walk, or drunkard's walk. Imagine a person leaves the pub so drunk that their method of getting home consists of taking random steps, with probability 0.5 the step is in the direction of home with probability 0.5 it is in the other opposite direction. We can model the drunk's position after the ith step as a random variable X_i where $X_0 = 0$ (that is, the pub is the origin). Then $X_{i+1} = X_i + S_i$ where

$$S_i = \begin{cases} +1 & \text{with probability } 1/2 \\ -1 & \text{with probability } 1/2. \end{cases}$$

with is the direction of the ith step. Equivalently, $X_i = X_0 + \sum_{j=0}^{i-1} S_j$. The process, while amenable to analytic techniques, is extremely simple to simulate: we just need to be able to simulate Bernoulli random variables.

The random walk has many variations: instead of looking at a symmetric walk, consider $S_i = 1$ with probability p; we can consider the random walk in higher dimensions, choosing from 2^d possible directions in d dimensions; and choosing a different step size ($S_i = \pm c$, say).

The process has some very nice, and often surprising, properties. For example, the simple symmetric random walk crosses every point an infinite number of times (this is known as Gambler's ruin, as if X models the amount a gambler is winning when betting \$1 on toss of a coin, the gambler will certainly eventually lose all their money if they play for long enough against a casino with infinite resources).

Secondly, the random walk in d dimensions returns to the origin with probability 1 for $d = 1, 2$, but for $d \geq 3$, that probability is below 1 (about 0.34 in for $d = 3$, 0.19 for $d = 4$ etc.).

13.3.2 Poisson process

The Poisson process is a simple yet incredibly useful model for events that occur independently of each other and randomly in time (or space). It is commonly used to model events such as:

- Genetic mutations
- arrival times of customers
- radioactive decay
- occurrences of earthquakes
- industrial accidents
- errors in manufacturing processes (e.g. silicon wafers or cloth)

We will consider processes in time although the concepts extend readily to space (the last of the examples above could be spatial).

A Poisson process is sometimes called a *point process.* It is counts the number of events in the time interval $[0, t]$. Let $N(t)$ be the number of points in the interval, so that $N(t)$ is a counting process.

Define a *Poisson process with intensity* λ, where $\lambda > 0$ (also called the *rate*) to be a process $\mathbf{N} = \{N(t), t \geq 0\}$ that takes values in $S = \{0, 1, 2, \ldots\}$ that satisfies the following properties:

1. $N(0) = 0$ and if $s < t$ then $N(s) < N(t)$.
2. If $s < t$ then $N(t) - N(s)$ is the number of arrivals in $(s, t]$ which is independent of the number (and times) of the arrivals in $(0, s]$.
3. $$\Pr(N(t+h) = n + m | N(t) = n) = \begin{cases} \lambda h + o(h) & \text{if } m = 1 \\ o(h) & \text{if } m > 1 \\ 1 - \lambda h + o(h) & \text{if } m = 0. \end{cases}$$

Here the notation $o(h)$ indicates that, as h gets small the bit of the expression that is $o(h)$ disappears. A strict definition is that function f is $o(h)$ ('of order little oh of h') if

$$\lim_{h \to 0} \frac{f(h)}{h} = 0.$$

Examples: Check that $f(h) = h^2$ is $o(h)$ while $f(h) = h^{-\frac{1}{2}}$ is not. □

The Poisson process is related to the Poisson distribution by the fact that $N(t)$ has a Poisson distribution with parameter λt so that

$$\Pr(N(t) = k) = \frac{(\lambda t)^k}{k!} \exp(-\lambda t)$$

for $k \in \{0, 1, 2, \ldots\}$.

Now look at the times between events in a Poisson process. Let T_i denote the time of the ith event of the process and $T_0 = 0$. Then the ith *inter-arrival time*, $X_i = T_{i+1} - T_i$, is exponentially distributed with parameter λ, that is $X_i \sim \text{Exp}(\lambda)$, $i = 1, 2, 3, \ldots$.

Poisson processes have some very nice properties.

Splitting: Let $\{N(t), t \geq 0\}$ be a Poisson process with rate λ. Suppose each event is of type i, for $i \in \{1, \ldots, k\}$ with probability p_i and suppose that this is independent of other events.

If we observe just the events of type i, they form a Poisson process with rate λp_i independently of the remaining types of events.

For example, if we look at the request for different types of data in a network or arrivals of different types of customer, we get the large Poission process separated out as multiple smaller (lower rate) Poisson processes.

Merging: The converse of splitting is merging: Let $N = \{N(t), t \geq 0\}$ be a Poisson process with rate λ and $M = \{M(t), t \geq 0\}$ be a Poisson process with rate μ independent of N. Then $L = \{L(t) = N(t) + M(t), t \geq 0\}$ is a Poisson process of rate $\lambda + \mu$.

Together, these results tell us how to model multiple Poisson processes using a single large process: Suppose we have n independent Poisson processes where process i has rate λ_i. Then the merged process has events of n different types. If we observe an event in the merged process, let p_i be the probability of the event being of type i. What is p_i? According to the splitting theorem, the type i process has rate $\lambda_i = \lambda p_i$ where $\sum_i p_i = 1$ so that $\lambda = \sum_i \lambda_i$. So p_i is given by

$$p_i = \frac{\lambda_i}{\lambda} = \frac{\lambda_i}{\lambda_1 + \ldots + \lambda_n} = \frac{\text{rate of type } i}{\text{total rate}}.$$

Example: We model arrivals at a bus stop as a Poisson process. Some people arriving are students and some are office workers. Students arrive at rate λ_1, office workers arrive at rate λ_2. Merging these processes tells us the total rate of arrivals is $\lambda_1 + \lambda_2$. The probability that any given arrival is a student is $\frac{\lambda_1}{\lambda_1+\lambda_2}$ while the probability that they are an office worker is $\frac{\lambda_2}{\lambda_1+\lambda_2}$. $\square$

14 Markov chains

We think of a random process as a sequence of random variables $\{X_t : t \in T\}$ where T is an index set. T can be thought of as time. If T is discrete, the process $X(t)$ is called a *discrete time random process* while if T is continuous, $X(t)$ is called a *continuous time random process*. The random walk example is an example of a discrete time process (each time unit corresponds to a single step in the process) while the Poisson process is a continuous time process (arrivals happen at any time). We will consider only discrete time processes until further notice.

The random walk and Poisson processes described above both share an important property, known as the Markov property. Intuitively, this is the property of memorylessness in that future states depend only on the current state and not any past states. That is, to propagate the process forward, we need only be told the current state to generate the next state.

Formally, the sequence of random variables $X_1, X_2, X_3, \ldots$ is a *Markov chain* if it has the *Markov property*:

$$P(X_{n+1} = x | X_n = x_n, X_{n-1} = x_{n-1}, \ldots, X_2 = x_2, X_1 = x_1) = P(X_{n+1} = x | X_n = x_n).$$

Markov chains are commonly used to model processes that are sequential in nature and where the future state only depends on the current state. This limited dependence property is called the Markov property (after Andrey Andreyevich Markov, a Russian mathematician from the late 19th century).

Example 1: The random walk. Start at $X_0 = 0$. If X_n is the current state, $P(X_{n+1} = X_n + 1 | X_n) = 1/2 = P(X_{n+1} = X_n - 1 | X_n)$. This is a Markov chain on an infinite state space. A realisation of this chain: 0 1 0 -1 0 1 2 3 2 1 2 1 2 1 0 1

$$\begin{tikzcd}
\cdots \arrow[r, bend left, "0.5"] & -2 \arrow[l, bend left, "0.5"] \arrow[r, bend left, "0.5"] & -1 \arrow[l, bend left, "0.5"] \arrow[r, bend left, "0.5"] & 0 \arrow[l, bend left, "0.5"] \arrow[r, bend left, "0.5"] & 1 \arrow[l, bend left, "0.5"] \arrow[r, bend left, "0.5"] & 2 \arrow[l, bend left, "0.5"] \arrow[r, bend left, "0.5"] & \cdots \arrow[l, bend left, "0.5"]
\end{tikzcd}$$

Example 2: Weather. The weather tomorrow depends on the weather today. If it is sunny today, tomorrow it is sunny with probability 0.7 and rainy otherwise. Rain clears a bit faster, so if it is rainy today, it is rainy tomorrow with probability 0.6 and sunny otherwise. This is a Markov chain with state space $\{R, S\}$ (for rainy and sunny, respectively). The following is a simulated realisation: S S S S S S R S R R R

$$\begin{tikzcd}
R \arrow[loop left, "0.6"] \arrow[r, bend left, "0.4"] & S \arrow[loop right, "0.7"] \arrow[l, bend left, "0.3"]
\end{tikzcd}$$

Example 3: The following is **not** a Markov chain. Recall our random walk is called a drunkard's walk. Imagine someone occasionally helps the drunkard on the way home by carrying him 10 paces either to the left or the right. This person has limited patience, though, so will help at most 3 times. When the person has not yet reached the limit of their patience, possible transitions include $X_{n+1} = X_n \pm 10$ or $X_{n+1} = X_n \pm 1$. After the person has intervened to help 3 times, the only possible transitions are $X_{n+1} = X_n \pm 1$. So to see if this large movement is possible, we need to look back in history to see how many interventions have occurred. Thus the distribution of the next state depends on more than just the current state and the chain is not a Markov chain. $\square$

The chain is *homogeneous* if $\Pr(X_{n+1} = j|X_n = i) = \Pr(X_1 = j|X_0 = i)$. If a chain is homogeneous, we write $P_{ij} = \Pr(X_1 = j|X_0 = i)$. The transition probabilities are normalised so that $\sum_j P_{ij} = 1$.

The matrix $P = [P_{ij}]$ is called a *stochastic matrix* as all its entries are non-negative and its rows sum to 1, so that $\sum_j P_{ij} = 1$.

Example: The transition matrix for the weather example given above is $\begin{bmatrix} 0.7 & 0.3 \\ 0.4 & 0.6 \end{bmatrix}$ where rows and columns 1 and 2 are indexed by S and R, respectively. $\square$

A homogeneous Markov chain is completely defined by specifying an initial distribution $\Pr(X_0 = i)$ for X_0 and the transition probabilities X_{n+1} given X_n, P_{ij}.

The $m-$*step transition probability* is the probability of going from state i to state j in exactly m steps, $P_{ij}(m) = \Pr(X_{n+m} = j | X_n = m)$. Hence the $m-$step transition matrix is $P_m = [P_{ij}(m)]$.

A result known as the *Chapman-Kolmogorov equations* tells us $P_{m+n} = P_m P_n$ (where the right-hand side is just standard matrix multiplication). In particular, this result tells us that $P_n = P^n$, that is, the n-step transition matrix is just the nth power of the (one-step) transition matrix.

15 Introduction to genetics and genetic terminology

The history of life can be viewed, in a rather mundane way, as a long running and very complex stochastic (or random) process.

At a very basic level, and after many simplifying assumptions, we can think of the historical process explaining the relationships between species as a tree. The points where the tree splits are speciations and the leaves of the tree are different species. The past is back at the base or root of the tree and time increases from the root to the tips. Information is passed along the tree (away from the root) from one generation to the next via genetic material.

Genetic material is thought to be the only means by which biological information is passed from parent to offspring. The process of copying genetic material is imperfect, so that children will differ slightly from the parent. These imperfections consist of errors in the copying, known as mutations, and can be thought of as a stochastic process.

The fundamental objects we will be studying are sequences of characters representing biological macromolecules: DNA (Deoxyribonucleic), RNA (Ribonucleic acid) and proteins. DNA are RNA are the primary forms of genetic material. The characters in DNA and RNA sequences are drawn from 4 letter alphabets: DNA has $\Omega = \{A, C, G, T\}$ while RNA has $\Omega = \{A, C, G, U\}$. The A stands for adenine, C for cytosine, G for guanine, T for thymine and U for uracil. These are known as nucleobases or simply bases, with C, T, U being *pyramidines* and A, G being *purines*. Protein sequences consist of the twenty amino acids that are represented by the alphabet $\{A, R, N, D, C, E, Q, G, H, I, L, K, M, F, P, S, T, W, Y, V\}$ (that is, all the letters except $\{B, J, O, U, X, Z\}$). We will refer to the bases in an DNA/RNA sequence or the amino acids in a protein sequence as *residues*.

In eukaryotes (organisms with cells that have a nucleus), the three types of sequences related to each other by the Central Dogma of Molecular Biology that states, DNA makes RNA makes Protein. Or, more prosaically, DNA is *transcribed* into a type of RNA called mRNA that is then *translated* into protein.

There are some good animations showing how translation and transcription work at `www.hhmi.org/biointeractive/animations/index.html`, in particular see the DNA transcription and translation animations. A Japanese anime style film of the central dogma is also worth a look: `http://www.youtube.com/watch?v=-ygpqVr7_xs`.

Parts of the the DNA sequence encode information for proteins. These regions are known as genes and must be transcribed to RNA before being built into proteins. When the DNA is transcribed to RNA, all bases are copied exactly except that T (thymine) is transcribed as U (uracil). Once copied, the RNA is edited at splice sites so that only exons remain (the introns are edited out). This leaves the *messenger RNA*, mRNA, which is then translated to a protein sequence (poly-peptide chain). This translation occurs via the genetic code which translates consecutive triples of RNA bases (known as a codon) into one of the 20 amino acids. There are $4^3 = 64$ possible codes since there is an alphabet of 4 bases. 60 of these code for proteins, 1 (AUG) is a start codon and 3 (UAA, UGA and UAG) are stop codons signalling the start or finish of a protein. A particular amino acid may be encoded by just one codon (e.g. AUG$\rightarrow$Methionine(M)) or up to 6 (e.g. any of UUA, UUG, CUU, CUC, CUA, CUG$\rightarrow$Leucine (L)). Once the poly-peptide chain is formed it folds into three dimensional molecule, taking on a particular structure.

Example: The sequence `atgaggttgacgctactttgttgcacctggagggaa` can be split into codons `atg agg ttg acg cta ctt tgt tgc acc tgg agg gaa` which translate into the protein sequence `MRLTLLCCTWRE`. □

In this course, we are only interested in the primary structure of sequences, that is, the order in which residues occur along the sequence. We will ignore the secondary, tertiary and quaternary structure of proteins — secondary structure is the name for the regular substructures such as alpha helices and beta sheets, the tertiary structure are the three dimensional structures of single molecules while quaternary structure are the complex forms taken by collections of single protein molecules. The study of these more complex structures is known as structural bioinformatics.

When DNA is passed from one generation to the next, the copy made is not exact. There are a number of processes that cause differences to arise between the parent and child. Recombination is one such process and involves the mixing of the maternal and paternal copies of DNA when the gametes (eggs or sperm) are produced. Other processes are generally thought of as mutations. The simplest are *point mutations* where the offspring sequence differs from the parent sequence by a single base (residue). This type of mutation is called a *single nucleotide polymorphism*, abbreviated as SNP and pronounced 'snip'. *Insertions* (or deletions) refer to the child sequence gaining (losing) one or more base than the parent. Larger scale mutations include: *gene duplication* which is a large scale insertion where the child inherits extra copy of a region containing a whole gene. Other large scale mutations include inversions (part of the sequence is reversed end to end) and translocations (a piece of the sequence is copied out of order).

Examples of mutations: Consider the short sequence `cgctcaccatgaagcgtttcactaat`. We demonstrate types of mutations showing the original sequence and a mutated version of it below with X marking the mutation.

- Single nucleotide polymorphism (SNP)
  ```
  cgctcaccatgaagcgtttcactaat
  cgctcgccatgaagcgtttcactaat
  ```

```
.....X...................
```

- Insertion
```
cgctcacc----atgaagcgtttcactaat
cgctcacctgatatgaagcgtttcactaat
........XXXX..................
```

- Deletion
```
cgctcaccatgaagcgtttcactaat
cgct----atgaagcgtttcactaat
....XXXX..................
```

- Duplication (the copied region is marked with parentheses). Note that duplication usually refers to gene duplication where whole genes are copied.
```
cgctcaccatgaagcgtttcacta-----------at
cgctcaccatgaagcgtttcactacaccatgaagcat
....(.........).........XXXXXXXXXXX..
```

- Inversion (again, this typically happens at a larger scale than shown here)
```
cgctcaccatgaagcgtttcactaat
cgctctaccagaagcgtttcactaat
.....XXXXX................
```

□

All these processes can be modelled and studied, with varying degrees of difficulty. We'll focus primarily on the question of how to align the sequences, how to identify regions of interest in sequences (for example, genes), and given aligned sequences, how can we reconstruct the evolutionary history (the tree) of those sequences. This last problem will require us to model the the mutation process where we restrict ourselves to looking at how point mutations arise.

The models we use will use are relatively simple, sometimes to the point of being downright crude. It is good to keep in mind the quote from the famous statistician George Box who said, "All models are wrong but some are useful".

15.1 Summary of above

- We model genetic sequences: think of them as strings of letters.
- There are 3 types of sequence, DNA, RNA or Protein.
- DNA sequences are composed of the 4 letters, or bases, $\{A, C, G, T\}$, RNA is made of the bases $\{A, C, G, U\}$ while protein sequences are made up of the 20 amino acids.
- The three types of sequence are related by the central dogma of molecular biology: DNA is transcribed to RNA and then translated to protein.

- Protein sequences fold up into more complex structures. We will ignore this structure in this introductory course.
- DNA is copied from parent to child.
- At copying, mutations are introduced.
- Mutations may be single nucleotide polymorphisms (SNPs), insertions, deletions or of other types.
- We use a tree to model the history of relationships between individuals (which are represented by their sequences).

To model the complex random process of genetic mutation and inheritance, we will need tools from applied probability and statistics. The next few sections are concerned with introducing the main tools and concepts that we will use for our study. All of you will have previously encountered at least some of the ideas we discuss here but, as with the linear algebra sections, it helps to review the main points before plunging in to new material.

16 Alignment

16.1 Homology

Homology (from the Greek, to agree) is a crucial concept in biology referring to traits or, in the case of sequences, sequence regions that share a common ancestry. We expect homologous regions to be similar to each other where the level of similarity will depend on how recently they shared a common ancestor.

Thus to say two regions are homologous is an evolutionary hypothesis. Mrs Darwin (in Carol Ann Duffy's poem from the collection The World's Wife) was making an evolutionary hypothesis of homology:

7 April 1852
Went to the Zoo.
I said to Him —
Something about that Chimpanzee over there reminds me of you.

The claim does not imply that the regions share a similar function now or, depending on the time since divergence, that they even particularly similar, just that they share a common ancestor. Therefore, sequences are either homologous or not, there are no degrees of homology. We often infer homology between two sequences when they are similar but we must be careful as we can get similarity without homology. Homologous sequences are sometimes referred to as *homologs*.

There are two main ways that we get similarity without homology: either by chance or by convergent evolution. Similarity by chance will occur even in completely random sequences on a finite alphabet. In two random sequences of four letters, we would expect similarity by chance of 25%.

Convergent evolution occurs when similar functions evolve independently of each other. An example of this are the wings of birds and insects. We don't believe these two very different creatures had a common ancestor that evolved wings but that wings evolved indecently of each other in the insect and bird lineages. Thus, while wings in a fly and a sparrow may be superficially similar, they are not considered homologous. The same applies to sequences that code for similar proteins (i.e., have similar function) but have evolved independently. Such traits/regions are called analogous.

We distinguish between two types of homology: orthology and paralogy:

Orthology occurs when two genes are separated by a speciation event and evolve independently from there on.

Paralogy occurs when a region of the genome is duplicated in the same genome (a duplication event) and they evolve in parallel in the same genome. The two copies are said to be paralogs.

16.2 Pairwise alignment

Given two sequences, if they are homologues, how do they align with each other? That is, exactly which sites in the sequence are homologous with each other?

We consider pairs of sequences, x and y of length m and n, respectively. x_i is the ith symbol of x. These symbols are usually the 4 DNA or RNA bases or the 20 amino acids. We refer to the symbols as *residues.*

We will allow *gaps* to be introduced in either sequence to allow them to align better. Biologically, gaps correspond to insertions or deletions in the sequence.

Clearly, there are many ways of aligning a pair of sequences (how many?), but what is the best alignment?

Example: x = GAATTC and y = GATTA

```
GAATTC or GAATTC-
GA-TTA    -GATT-A
```

are two possible alignments. □

16.3 Scoring alignments

The best alignment will depend on how we score alignments. It is easy to come up with different scoring regimes (e.g., score 1 for a match, -1 for a mismatch) but we really want to compare two models — that the similarity we see is just chance vs. that the sequences are homologs.

We initially consider alignments without gaps.

16.3.1 Model of non-homologous sequences

The most basic model is that each letter appears with some probability, letter a appears with probability q_a (note that the probabilities summed over the alphabet are 1), that each site is independent and that the each sequence is independent. Then the probability of seeing sequence x is

$$P(x) = \prod_{i=1}^{n} q_{x_i}$$

and the joint likelihood of an alignment is just the joint probability of the sequences x and y,

$$P(x,y) = P(x)P(y) = \prod_{i=1}^{n} q_{x_i} \prod_{i=1}^{m} q_{y_i} = \prod_{i=1}^{n} q_{x_i} q_{y_i}.$$

16.3.2 Model of homologous sequences

An alternative model is that the two sequences are related and the probability of seeing the pair of residues a (from x) and b (from y) aligned at a locus is p_{ab}. The probability of the alignment is then the product of the loci,

$$P(x,y) = \prod_{i=1}^{n} p_{x_i y_i}.$$

To compare these two models, we take ratio of these likelihoods:

$$\frac{\prod_{i=1}^{n} p_{x_i y_i}}{\prod_{i=1}^{n} q_{x_i} q_{y_i}} = \prod_{i=1}^{n} \frac{p_{x_i y_i}}{q_{x_i} q_{y_i}}.$$

It is easier to work with log-likelihoods (and addition) than likelihoods and multiplication, so we take the log of this quantity to get

$$S = \sum_i s(x,y_i)$$

where

$$s(a,b) = \log\left(\frac{p_{ab}}{q_a q_b}\right).$$

The score of the alignment then is the sum over the local score $s(a,b)$. s can be thought of as a matrix and is often referred to as a *score matrix* or *substitution matrix.*

There are various methods for deciding reasonable values for the entries of the matrix, discussed below. Note that even when ad hoc values are chosen for the matrix, the underlying probabilities, p_{ab} and q_a can be derived by reversing the above argument. That is, when a score matrix is selected we are making implicit assumptions about the q_as and p_{ab}s.

Example: If x = GAATTC and y = GGATTA are aligned as

```
GAATTC
GGATTA
```

where $s(a,b) = 2$ if $a = b$ and $s(a,b) = -1$ if $a \neq b$, the alignments scores $2 - 1 + 2 + 2 + 2 - 1 = 6$. □

16.3.3 Scoring gaps

To make sequences align fully, we add gaps to one sequence or the other. A gap in x corresponds to an insertion in y with respect to x or a deletion in x with respect to y. Adding gaps comes with a penalty, so reduces the score for the match.

For a gap of length k, write $\gamma(k)$ for the penalty. We consider two forms for γ.

A *linear penalty* is defined by $\gamma(k) = -dk$ for some $d > 0$. That is, each deleted base adds a penalty of d.

An *affine penalty* is defined by $\gamma(k) = -d - (k-1)e$ where $d > e > 0$. d is the gap open penalty and e is the gap extension penalty. The affine penalty is more biologically appropriate as insertions or deletions are typically created in a single event rather than building up one residue at a time.

More complex gap penalties can be used, for example, we may wish to have different penalties for gaps matched with different residues, or non-linear functions of gap length. Such penalties come at the cost of more difficult implementation.

In the algorithms below, we'll first consider the simple case of the linear penalty.