1 Linear Algebra

1.1 Definitions

A *vector space*, often denoted by V, is a 'family' of vectors that obey some basic rules. A *vector* $\mathbf{v}$ is an element of a vector space. A vector space must always contain a zero vector.

A *subspace* of a vector space is a subset that obeys the rules of a vector space.

A *matrix* $\mathbf{A}$ is a linear operator that performs a linear transformation from one vector to another:

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

The matrix $\mathbf{A}$ maps the vector $\mathbf{x} \in \mathbb{C}^n$ to the vector $\mathbf{b} \in \mathbb{C}^m$:

$$\mathbf{A} : \mathbb{C}^n \to \mathbb{C}^m$$

An $m \times n$ matrix comes from the space $\mathbb{C}^{m\times n}$, and we will often write $\mathbf{A} \in \mathbb{C}^{m\times n}$.

The *conjugate transpose* of a matrix:

$$\mathbf{A}^H = \overline{\mathbf{A}^T} = \overline{\mathbf{A}}^T$$

A matrix $\mathbf{M} \in \mathbb{C}^{n\times n}$ is *Hermitian* if $\mathbf{M}^H = \mathbf{M}$.

The dot (or scalar) product is an operation between two equal-length vectors that yields a scalar.

$$\mathbf{x}^H\mathbf{y} = \sum_{i=1}^{n} \overline{x}_i y_i = \overline{\mathbf{y}^H\mathbf{x}}$$

The dot product is sometimes called the *inner product* $\langle\cdot,\cdot\rangle$.

For an $n\times n$ matrix $\mathbf{A}$, $(\lambda, \mathbf{x})$ is an eigenpair of $\mathbf{A}$ if $\mathbf{Ax} = \lambda\mathbf{x}$, where λ an eigenvalue of $\mathbf{A}$, and $\mathbf{x}$ is the corresponding eigenvector of $\mathbf{A}$.

The eigenvalues of a Hermitian matrix are *real* and the eigenvectors of a Hermitian matrix are *orthogonal*, i.e. $\mathbf{u}_i^H\mathbf{u}_j = \mathbf{u}_j^H\mathbf{u}_i = 0$ when $i \neq j$.

A matrix $\mathbf{Q} \in \mathbb{C}^{n\times n}$ is a *unitary* matrix if $\mathbf{Q}^H = \mathbf{Q}^{-1}$, i.e. $\mathbf{Q}^H\mathbf{Q} = \mathbf{Q}\mathbf{Q}^H = \mathbf{I}$. If $\mathbf{Q}$ was real, we would call it an *orthogonal* matrix.

A Hermitian matrix $\mathbf{M} \in \mathbb{C}^{n\times n}$ is *positive definite* if:

$$\mathbf{x}^H\mathbf{Mx} > 0 \quad \forall \mathbf{x} \in \mathbb{C}^n \backslash \mathbf{0}$$

The eigenvalues of a Hermitian positive definite matrix are strictly positive (this is a sufficient condition).

A Hermitian matrix $\mathbf{M} \in \mathbb{C}^{n\times n}$ is *semi-positive definite* if:

$$\mathbf{x}^H\mathbf{Mx} \geq 0 \quad \forall \mathbf{x} \in \mathbb{C}^n$$

The matrix $\mathbf{A}^H\mathbf{A}$ is positive semi-definite.

The *rank* of a matrix $\mathbf{A} \in \mathbb{C}^{m\times n}$ is the number of linearly independent rows or columns (the number is equal). It satisfies $\operatorname{rank}\mathbf{A} \leq \min(m,n)$. A matrix is *full rank* if $\operatorname{rank}\mathbf{A} = \min(m,n)$ and *rank deficient* if $\operatorname{rank}\mathbf{A} < \min(m,n)$.

A matrix in which most entries are zero is a *sparse matrix*.

Vector norms: A particular family of norms are known as l_p -norms:

$$\|\mathbf{x}\|_p = \left(\sum_{i=1}^{n} |x_i|^p\right)^{1/p}$$

The l_∞ norm $\|\mathbf{x}\|_\infty = \max_i |x_i|$.

We can define norms that involve a matrix $\mathbf{A}$,:

$$\|\mathbf{x}\|_A^2 = \langle \mathbf{x}, \mathbf{Ax}\rangle = \mathbf{x}^H\mathbf{Ax}$$

$\mathbf{A}$ must be positive definite. The above norm is often called the *energy norm*.

Operator norms: A norm of a matrix $\mathbf{A}$ is defined as:

$$\|\mathbf{A}\| = \max_{\mathbf{x}\in\mathbb{C}^n\backslash\mathbf{0}} \frac{\|\mathbf{Ax}\|}{\|\mathbf{x}\|}$$

This norm measures the *maximum amount* by which the matrix $\mathbf{A}$ can rescale a vector $\mathbf{x}$.

$$\|\mathbf{Ax}\| \leq \|\mathbf{A}\|\|\mathbf{x}\| \quad \forall \mathbf{x}$$

For $\|\mathbf{A}\|_2$(2 -norm):

$$\|\mathbf{A}\|_2^2 = \max_{\mathbf{x}\in\mathbb{C}^n\backslash 0} \frac{\mathbf{x}^H\mathbf{A}^H\mathbf{Ax}}{\mathbf{x}^H\mathbf{x}} = \lambda_{\max}\left(\mathbf{A}^H\mathbf{A}\right)$$

$\lambda_{\max}\left(\mathbf{A}^H\mathbf{A}\right)$ is the largest eigenvalue of $\mathbf{A}^H\mathbf{A}$. The norm $\|\mathbf{A}\|_2$ is therefore the square root of the largest eigenvalue of $\mathbf{A}^H\mathbf{A}$ (the largest *singular value* of $\mathbf{A}$). If $\mathbf{A}$ is Hermitian $\|\mathbf{A}\|_2 = |\lambda|_{\max}(\mathbf{A})$.

Frobenius norm:

$$\|\mathbf{A}\|_F = \sqrt{\sum_i \sum_j |a_{ij}|^2}$$

The Frobenius norm is also invariant under rotation, $\|\mathbf{QA}\|_F = \|\mathbf{A}\|_F$ where $\mathbf{Q}$ is a unitary matrix.

The condition number of a matrix $\mathbf{A}$ is:

$$\kappa(\mathbf{A}) = \|\mathbf{A}\|\left\|\mathbf{A}^{-1}\right\|$$

For the 2-norm, since the eigenvalues of $\mathbf{A}^{-1}$ are the reciprocal of the eigenvalues of $\mathbf{A}$:

$$\kappa_2(\mathbf{A}) = \frac{\sqrt{\lambda_{\max}\left(\mathbf{A}^H\mathbf{A}\right)}}{\sqrt{\lambda_{\min}\left(\mathbf{A}^H\mathbf{A}\right)}}$$

If A is Hermitian,

$$\kappa_2(\mathbf{A}) = \frac{|\lambda(\mathbf{A})|_{\max}}{|\lambda(\mathbf{A})|_{\min}}$$

1.2 Stability

Consider the problem $\mathbf{A}(\mathbf{x}+\delta\mathbf{x}) = \mathbf{b}+\delta\mathbf{b}$ where $\delta\mathbf{b}$ is the error in the RHS and $\delta\mathbf{x}$ is the consequent error in the solution.

$$\frac{\|\delta\mathbf{x}\|}{\|\mathbf{x}\|} \leq \|\mathbf{A}\|\left\|\mathbf{A}^{-1}\right\| \frac{\|\delta\mathbf{b}\|}{\|\mathbf{b}\|} = \kappa(\mathbf{A})\frac{\|\delta\mathbf{b}\|}{\|\mathbf{b}\|}$$

For large condition numbers we can expect small errors in $\mathbf{b}$ to cause large errors in $\mathbf{x}$. A matrix with a large condition number is said to be *ill-conditioned*.

1.3 Interpolation

Interpolation is fitting a function to a data set that passes through the data points. If we have n data points $\mathbf{f} = [f_1(x_1),\ldots,f_n(x_n)]^T$ in a one dimensional space, we can usually fit a polynomial with n coefficients of the form:

$$f(x) = c_0 + c_1P_1(x) + \ldots + c_{n-1}P_{n-1}(x)$$

We can solve the matrix equation $\mathbf{Ac} = \mathbf{f}$:

$$\begin{bmatrix} 1 & P_1(x_1) & \cdots & P_{n-1}(x_1) \\ \vdots & \vdots & \vdots & \vdots \\ 1 & P_1(x_1) & \cdots & P_{n-1}(x_n) \end{bmatrix}\begin{bmatrix} c_0 \\ \vdots \\ c_{n-1} \end{bmatrix} = [f_1(x_1),\ldots,f_n(x_n)]^T$$

The matrix $\mathbf{A}$ is known as the *Vandermonde* matrix. It is a notoriously ill-conditioned matrix, and the condition number grows with increasing polynomial degree using monomial base $1, x, x^2, \ldots, x^{n-1}$.

The *Legendre polynomials* on the interval $[-1,1]$:

$$(n+1)P_{n+1}(x) = (2n+1)xP_n(x) - nP_{n-1}(x)$$
$$P_0 = 1 \text{ and } P_0 = x$$

The special feature of Legendre polynomial is that:

$$\int_{-1}^{1} P_m(x)P_n(x)'dx = 0 \quad \text{if } m \neq n$$

Legendre polynomials of different degree are orthogonal to each other. The Legendre Vandermonde matrix is considerably better conditioned.

Polynomial interpolation can lead to oscillations towards the end of the interval known as the *Runge effect*. We can mitigate the Runge effect by using non-equispaced sampling points, and particularly good points are the roots of some orthogonal polynomials.

1.4 Data Fitting

To find a solution to the problem $\mathbf{Ax} = \mathbf{b}$, the vector $\mathbf{b}$ must lie in the column space of $\mathbf{A}$. If $\mathbf{A}$ is an $m \times n$ matrix and $m > n$ (a skinny matrix), we have more equations than unknowns and in general there will be no solution.

For some vector $\hat{x}$, we can define a *residual* vector $\mathbf{r}$:

$$\mathbf{r} = \mathbf{A}\hat{\mathbf{x}} - \mathbf{b}$$

Find $\hat{x}$ that minimises the residual $\mathbf{r}$ in the chosen norm:

$$\min_{\hat{\mathbf{x}}\in\mathbb{C}^n} \|\mathbf{r}(\hat{x})\| = \min_{\hat{\mathbf{x}}\in\mathbb{C}^n} \|\mathbf{A}\hat{\mathbf{x}} - \mathbf{b}\|$$

If we use the l_2-norm for the problem, we seek $\min_{\hat{\mathbf{x}}\in\mathbb{C}^n} \|\mathbf{A}\hat{\mathbf{x}} - \mathbf{b}\|_2$.

$$\hat{\mathbf{x}} = \left(\mathbf{A}^H\mathbf{A}\right)^{-1}\mathbf{A}^H\mathbf{b} = \mathbf{A}^+\mathbf{b}$$

The matrix $\mathbf{A}^+ = \left(\mathbf{A}^H\mathbf{A}\right)^{-1}\mathbf{A}^H$ is known as the *pseudoinverse* or the *Moore-Penrose inverse*. The inverse $\left(\mathbf{A}^H\mathbf{A}\right)^{-1}$ can be computed when $\mathbf{A}$ is full rank.

1.5 Iterative Methods for Linear Systems

Power iteration: An iterative method for finding the eigenvector associated with the largest absolute eigenvalue of a matrix. Since a vector $\mathbf{x} \in \mathbb{C}^n$ can be expressed in terms of the n eigenvectors of an $n\times n$ matrix $\mathbf{x} = \sum_{i=1}^{n} \alpha_i\mathbf{u}_i$, if we multiply $\mathbf{x}$ repeatedly by $\mathbf{A}$:

$$\mathbf{A}^k\mathbf{x} = \sum_{i=1}^{n} \alpha_i\lambda_i^k\mathbf{u}_i$$

If the largest eigenvalue is distinct the resulting vector will be aligned with the eigenvector of the largest eigenvalue. To find an estimate of the corresponding eigenvalue $\lambda^\star$, we could pose a minimisation problem in the l_2-norm $\min_{\lambda^\star\in\mathbb{C}} \left\|\mathbf{Ax} - \lambda^\star\mathbf{x}\right\|_2$. This is minimised when:

$$\lambda^\star = R(\mathbf{A},\mathbf{x}) = \frac{\mathbf{x}^H\mathbf{Ax}}{\mathbf{x}^H\mathbf{x}}$$

R is known as the *Rayleigh quotient*.

A family of *stationary methods* for finding approximate solutions to $\mathbf{Ax} = \mathbf{b}$ involves decomposing the matrix operator such that $\mathbf{A} = \mathbf{N} - \mathbf{P}$ and computing an approximate solution $\mathbf{x}_{k+1}$:

$$\mathbf{Nx}_{k+1} = \mathbf{b} + \mathbf{Px}_k$$

The process is then repeated to hopefully converge to the exact solution. Classic examples of splitting $\mathbf{A}$ include:

1. *Richardson iteration:* $\mathbf{N} = \mathbf{I}$.
2. *Jacobi method:* $\mathbf{N} = \operatorname{diag}(\mathbf{A})$.
3. *Gauss-Seidel:* $\mathbf{N} = L(\mathbf{A})$ is the lower triangular part of $\mathbf{A}$ (including the diagonal).

Defining the error at the kth iteration $\mathbf{e}_k = \mathbf{x}_{\text{exact}} - \mathbf{x}_k$:

$$\mathbf{e}_k = \left(\mathbf{N}^{-1}\mathbf{P}\right)^k\mathbf{e}_0$$

The method will converge only if the absolute value of every eigenvalue is less than one. The largest absolute eigenvalue of a matrix $\mathbf{A}$ is often denoted by $\rho(\mathbf{A})$ and is known as the *spectral radius*. The stationary methods based on splitting will converge if:

$$\rho\left(\mathbf{N}^{-1}\mathbf{P}\right) < 1$$

The *conjugate gradient* (CG) method is a *Krylov subspace method*. Consider that we have a set of n non-zero vectors $P = \{\mathbf{p}_0, \mathbf{p}_1, \ldots, \mathbf{p}_{n-1}\}$ that are A-conjugate:

$$\mathbf{p}_i^H\mathbf{Ap}_j = 0 \quad \text{if } i \neq j$$

Using P as a basis for the solution of $\mathbf{Ax} = \mathbf{b}$, where $\mathbf{A}$ is a $n \times n$ Hermitian positive-definite matrix:

$$\mathbf{x} = \sum_{i=0}^{n-1} \alpha_i\mathbf{p}_i$$

$$\alpha_j = \frac{\mathbf{p}_j^H \mathbf{b}}{\mathbf{p}_j^H \mathbf{A}\mathbf{p}_j}$$

A simple approach to generate the A-conjugate set is to pick n linearly independent vectors and apply the Gram-Schmidt process to build P.

For the error $\mathbf{e}_k = \mathbf{x}_{\text{exact}} - \mathbf{x}_k$, the CG method is monotone in the A-norm $\|\mathbf{y}\|_A^2 = \mathbf{y}^H\mathbf{A}\mathbf{y}$:

$$\|\mathbf{e}_{k+1}\|_A \le \|\mathbf{e}_k\|_A$$

The CG method will solve the problem exactly (in the absence of round-off error) in at most n iterations since $\|\mathbf{e}_k\|_A = 0$ for some $k \le n$.

The rate of convergence is affected by the condition number $\kappa_2(\mathbf{A})$:

$$\frac{\|\mathbf{e}_k\|_A}{\|\mathbf{e}_0\|_A} \le 2\left(\frac{\sqrt{\kappa_2}-1}{\sqrt{\kappa_2}+1}\right)^k$$

If the condition number of a matrix large, the CG method may be too slow to converge. *Preconditioning* with $\mathrm{P}^{-1} \approx \mathrm{A}^{-1}$:

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

1.6 Singular Value Decomposition

The *diagonalisation* of a Hermitian matrix $\mathbf{M} \in \mathbb{C}^{n\times n}$:

$$\mathbf{Q}^H\mathbf{M}\mathbf{Q} = \mathbf{\Lambda}$$

The columns of $\mathbf{Q}$ are the normalised (in l_2) eigenvectors of $\mathbf{M}$ and $\mathbf{\Lambda}$ is a diagonal matrix of the eigenvalues of $\mathbf{M}$ (which are real). Since the eigenvectors of a Hermitian matrix are orthogonal, $\mathbf{Q}$ is a unitary matrix:

$$\mathbf{M} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^H$$

The *singular value decomposition* (SVD) of an $m \times n$ matrix $\mathbf{A}$ is:

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^H$$

$\mathbf{\Sigma} \in \mathbb{R}^{m\times n}$ is a diagonal matrix, with diagonal entries σ_i (the *singular values*) sorted such that $\sigma_1 \ge \sigma_2 \ge \ldots \ge \sigma_p \ge 0$, where $p = \min(m,n)$. $\mathbf{U} \in \mathbb{C}^{m\times m}$ and $\mathbf{V} \in \mathbb{C}^{n\times n}$ are unitary matrices.

The columns of $\mathbf{V}$ are the (normalised) eigenvectors of $\mathbf{A}^H\mathbf{A}$ and the columns of $\mathbf{U}$ are the (normalised) eigenvectors of $\mathbf{A}\mathbf{A}^H$. The diagonal entries of $\mathbf{\Sigma}^H\mathbf{\Sigma}$ are the eigenvalues of $\mathbf{A}^H\mathbf{A}$, which are the same as the eigenvalues of $\mathbf{A}\mathbf{A}^H$.

$$\mathbf{A}\mathbf{V} = \mathbf{U}\mathbf{\Sigma}$$

Use $\mathbf{A}\mathbf{v}_i = \sigma_i\mathbf{u}_i$ to deduce the sign for $\mathbf{u}_i$ given $\mathbf{v}_i$ (or vice versa).

The terms below row n in $\mathbf{\Sigma}$ are always zero. Hence, the last $m-n$ columns of $\mathbf{U}$ make no contribution. In practice, the *reduced* SVD, in which redundant entries are removed, is typically used. If we expand the SVD with $\operatorname{rank}\mathbf{A} = r$:

$$\mathbf{A} = \sum_{i=1}^{r} \sigma_i \mathbf{u}_i \mathbf{v}_i^H$$

A *low rank* approximation of $\mathbf{A}$ is:

$$\mathbf{A}_k = \sum_{i=1}^{k} \sigma_i \mathbf{u}_i \mathbf{v}_i^H$$

The rank of $\mathbf{A}_k$ is $k < r$.

The SVD can construct optimal low-rank approximations of a matrix $\mathbf{A}$. It can be shown that for all matrices B of rank k or less:

$$\|\mathbf{A}-\mathbf{A}_k\|_F = \sqrt{\sigma_{k+1}^2 + \ldots + \sigma_r^2} \le \|\mathbf{A}-\mathbf{B}\|_F$$
$$\|\mathbf{A}-\mathbf{A}_k\|_2 = \sigma_{k+1} \le \|\mathbf{A}-\mathbf{B}\|_2$$

If $\mathbf{A}$ is an $m\times n$ matrix with $m > n$, we can partition the SVD as:

$$\mathbf{A} = \begin{bmatrix} \mathbf{U}_1 & \mathbf{U}_2 \end{bmatrix} \begin{bmatrix} \mathbf{\Sigma}_1 \\ \mathbf{0} \end{bmatrix} \mathbf{V}^H$$

The least squares solution to $\mathbf{A}\mathbf{x} = \mathbf{b}$ is:

$$\hat{\mathbf{x}} = \mathbf{V}\mathbf{\Sigma}_1^{-1}\mathbf{U}_1^H\mathbf{b} = \mathbf{V}\mathbf{\Sigma}^{+}\mathbf{U}^H\mathbf{b}$$

$\mathbf{\Sigma}^+$ is the pseudo inverse of $\mathbf{\Sigma}$.

If $\hat{\mathbf{x}} = \mathbf{A}^+\mathbf{b}\,(\hat{\mathbf{x}} \in \mathbb{C}^n)$ then $\|\hat{\mathbf{x}}\|_2^2 \ge \left|\mathbf{u}_n^H\mathbf{b}\right|^2/\sigma_{\min}^2$. If the smallest singular value $\sigma_{\min}$ is small, then the least squares solution will be large and very sensitive to changes in $\mathbf{b}$.

For a problem with zero singular values, consider a partitioning of the SVD:

$$\mathbf{A} = \begin{bmatrix} \mathbf{U}_1 & \mathbf{U}_2 \end{bmatrix} \begin{bmatrix} \mathbf{\Sigma}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{V}_1 & \mathbf{V}_2 \end{bmatrix}^H$$

For all $\mathbf{z} \in \mathbb{C}^{n-r}$ we have a solution to the least squares problem:

$$\hat{\mathbf{x}} = \mathbf{V}_1\mathbf{\Sigma}_1^{-1}\mathbf{U}_1^H\mathbf{b} + \mathbf{V}_2\mathbf{z}$$

$\hat{\mathbf{x}} = \mathbf{V}_1\mathbf{\Sigma}_1^{-1}\mathbf{U}_1^H\mathbf{b}$ is the minimiser to the least-squares problem with minimal l_2 norm.

2 Stochastic Processes

2.1 Finite-space Markov Chains

A Stochastic Process $X_0, X_1, X_2, \ldots$ is a *Markov chain* if and only if for all times $i \ge 1$:

$$P(X_{i+1} \mid X_0 = j_0, X_1 = j_1, \ldots, X_i = j_i) = P(X_{i+1} \mid X_i = j_i)$$

There are a set of random variables $X_0, X_1, X_2, \ldots$ indexed by time and each random variable takes a value from the *state-space*, $\mathcal{S}$.

For an initial distribution $\mathbf{x}^{(0)}$, the distribution at any time n is:

$$\mathbf{x}^{(n)} = \mathbf{x}^{(0)}\mathbf{P}^n$$

The *limiting distribution* is $\mathbf{x}^{(\infty)} = \mathbf{x}^{(0)}\mathbf{P}^{\infty}$. Any distribution that satisfies this expression is a *stationary distribution*.

A Markov Chain is *regular ergodic* if there exists a unique stationary distribution which is the limit point for all initial distributions and which puts positive mass on every element of the state-space $\mathcal{S}$. For an $n \times n$ transition matrix $\mathbf{P}$ then:

1. $\lambda = 1$ is an eigenvalue of $\mathbf{P}$
2. All eigenvalues satisfy $|\lambda| \le 1$

Find eigenvector for $\lambda = 1$ solving $\mathbf{P}^{\mathrm{T}}\mathbf{v} = \mathbf{v}$.

Two states of a Markov chain j and k *communicate* if, and only if, there exists integers m and n such that

$$P_{j,k}^m > 0 \text{ and } P_{k,j}^n > 0$$

A subset $\tilde{\mathcal{S}} \subseteq \mathcal{S}$ is a *recurrent set* if:

1. All pairs of states in $\tilde{\mathcal{S}}$ communicate.
2. If $j \in \tilde{\mathcal{S}}$ and $k \notin \tilde{\mathcal{S}}$ then $P_{j,k}^i = 0$ for all $i \ge 0$.

If a state belongs to a recurrent set then it is *recurrent*, otherwise it is *transient*. If all states in a Markov chain communicate with each other then the Markov chain is *irreducible*.

The waiting time until the next state X_d can be found using a series of linear equations:

$$q_j = \mathcal{E}[\text{ time to wait until } X_d \mid X_0 = j]$$
$$q_j = \sum_{k\in\mathcal{S}} P_{j,k}(1 + q_k)$$

The *period* of a state k of a Markov chain is the greatest common divisor of the set $\left\{i \ge 0 : P_{k,k}^i > 0\right\}$. A state k is *aperiodic* if it has period 1. A Markov chain is aperiodic if all states are aperiodic. If a chain is irreducible and aperiodic then it is regular ergodic and it will have a limiting distribution.

If $X_o, X_1, \ldots$ is a regular ergodic Markov chain with stationary distribution π, then for any function $f(x)$ as $N \to \infty$:

$$\frac{1}{N}\sum_{i=1}^{N} f(X_i) \to \sum_{k\in\mathcal{S}} \pi_k f(k)$$

A transition matrix P and a distribution π are in *detailed balance* if:

$$\pi_j P_{j,k} = \pi_k P_{k,j}$$

π is a stationary distribution of $\mathbf{P}$.

2.2 Continuous State-space Systems

Birth process: Consider when the birth rate of a cell is λ per unit time, $n(t)$ is the number of cells at time instance t and initially have $n(0) = n_0$ cells:

$$\frac{dn(t)}{dt} = \lambda n(t)$$
$$n(t) = n_0 \exp(\lambda t)$$

The number of births X in a time interval Δt follows a Poisson distribution:

$$P(X = k) = \frac{(\lambda\Delta t)^k e^{-\lambda\Delta t}}{k!}$$

The probability of n cells at time t is $P(N(t) = n) = P_n(t)$. Assuming small Δt and ignoring multiple events:

$$P_n(t+\Delta t) = P_n(t)(1 - n\lambda\Delta t) + P_{n-1}(t)((n-1)\lambda\Delta t)$$

Take the limiting condition $\Delta t \to 0$:

$$\frac{dP_n(t)}{dt} = -n\lambda P_n(t) + (n-1)\lambda P_{n-1}(t)$$

The *transition rate matrix* $\mathbf{Q}$ is:

$$\mathbf{Q} = \begin{bmatrix} -\lambda & \lambda & 0 & 0 & \cdots \\ 0 & -2\lambda & 2\lambda & 0 & \cdots \\ 0 & 0 & -3\lambda & 3\lambda & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

$$\frac{d\mathbf{x}(t)}{dt} = \mathbf{x}(t)\mathbf{Q}$$

$x_n(t) = P_n(t)$ and $\sum_{n=1}^{\infty} P_n(t) = 1$. If the birth process does reach a steady state:

$$\frac{d\mathbf{x}(\infty)}{dt} = \mathbf{x}(\infty)\mathbf{Q} = 0$$

This is only satisfied when $x_i(\infty) = 0$ for all i (not a valid pmf).

Birth-death process: Introduce *death rate* for a cell μ per unit time.

$$P_n(t+\Delta t) = P_n(t)(1 - n\lambda\Delta t - n\mu\Delta t) +$$
$$P_{n-1}(t)((n-1)\lambda\Delta t) + P_{n+1}(t)((n+1)\mu\Delta t)$$

Random walk: The probability of direction at time k, $\zeta_k \in \{-1, 1\}$, is uniform $P(\zeta_k = 1) = \frac{1}{2}$. After n steps the position, X_n is given by:

$$X_n = \sum_{k=1}^{n} \zeta_k$$

From the *central limit theorem*, the distribution of X_N is Gaussian, $\mathcal{N}(0, N)$.

Taking small step in direction ζ_k every δ seconds. From above location, W_t, is Gaussian distributed at time instance $t = N\delta$ where N is very large:

$$\mathcal{N}(0, N\delta) = \mathcal{N}(0, t)$$

In the limit $\delta \to 0$, this is *Brownian motion* which models the random motion of particles in a fluid resulting from collision. Brownian motion is also called a *Wiener process* in stochastic processes. Let the *particle density* at time t and position x be $f(x,t)$. Brownian motion is governed by simple diffusion equation:

$$\frac{\partial f(x,t)}{\partial t} = D\frac{\partial^2 f(x,t)}{\partial x^2}$$

Take initial condition as $f(x,0) = \delta(x)$, the final solution is:

$$f(x,t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} \exp\left(-Dk^2t\right)\exp(ikx)dk$$
$$= \frac{1}{\sqrt{4D\pi t}}\exp\left(-\frac{x^2}{4Dt}\right) = \mathcal{N}(0, 2Dt)$$

The properties of a one-dimensional Wiener process:

1. *Independence:* $W_t - W_s$ is independent of $\{W_\tau\}_{\tau\le s}$ for any $0 \le s \le t$.
2. *Stationarity:* The distribution of $W_{t+s} - W_s$ is independent of s.

3. *Gaussianity:* W_t is a Gaussian with $\mathcal{E}\{W_t\} = 0$ and $\mathcal{E}\{W_t W_s\} = 2D\min(t,s)$.
4. *Continuity:* W_t is a continuous function with t.

Wiener Process with Drift:

$$\frac{\partial p(x,t)}{\partial t} = m\frac{\partial p(x,t)}{\partial x} + D\frac{\partial^2 p(x,t)}{\partial x^2}$$

Ornstein-Uhlenbeck Process:

$$\frac{\partial}{\partial t}p(x,t) = \frac{\partial}{\partial x}(\beta x p(x,t)) + \frac{\partial^2}{\partial x^2}(Bp(x,t))$$

2.3 Monte Carlo Markov Chains

Numerical integration using histogram approach in d-dimensions:

$$\int h(\mathbf{x})d\mathbf{x} \approx \sum_{i=1}^{N} h\left(\mathbf{x}^{(i)}\right)\delta x_1 \delta x_2 \dots \delta x_d$$

Monte-Carlo Integration: Sample from a weighting function $w(\mathbf{x})$ in normalised form:

$$p(\mathbf{x}) = \frac{w(\mathbf{x})}{\int w(\mathbf{x})d\mathbf{x}}$$

$$\int h(\mathbf{x})d\mathbf{x} = \int \frac{h(\mathbf{x})}{p(\mathbf{x})}p(\mathbf{x})d\mathbf{x} \approx \frac{1}{N}\sum_{i=1}^{N}\frac{h\left(\mathbf{x}^{(i)}\right)}{p\left(\mathbf{x}^{(i)}\right)}$$

Consider uniform distribution over volume V, $p(\mathbf{x}) = 1/V$:

$$\int h(\mathbf{x})d\mathbf{x} = \frac{V}{N}\sum_{i=1}^{N} h\left(\mathbf{x}^{(i)}\right)$$

Importance sampling: Draw samples from the distribution $q(\mathbf{x})$:

$$\mathcal{E}\{f(\mathbf{x})\} = \int f(\mathbf{x})\frac{p(\mathbf{x})}{q(\mathbf{x})}q(\mathbf{x})d\mathbf{x} \approx \frac{1}{N}\sum_{i=1}^{N} f\left(\mathbf{x}^{(i)}\right)\frac{p\left(\mathbf{x}^{(i)}\right)}{q\left(\mathbf{x}^{(i)}\right)}$$

The ratio $\frac{p\left(\mathbf{x}^{(i)}\right)}{q\left(\mathbf{x}^{(i)}\right)}$ is the *importance* of the drawn sample. The closer that (scaled) $q(\mathbf{x})$ is to $p(\mathbf{x})$ the better.

Rejection sampling: Draw samples uniformly from from the 2-D box, accept those under the pdf curve and reject those above the line. Sampling in higher dimensions rapidly becomes highly wasteful (*curse of dimensionality*).

Metropolis-Hastings Algorithm: Current sample is $\mathbf{x}^{(i)}$, generate another sample, $\mathbf{x}^{(\star)}$, from $p\left(\mathbf{x} \mid \mathbf{x}^{(i)}\right)$:

$$\mathbf{x}^{(\star)} = \mathbf{x}^{(i)} + \mathbf{z}, \quad \mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$$

Accept the sample $\mathbf{x}^{(i+1)} = \mathbf{x}^{(\star)}$ with probability $\alpha = \min\left\{\frac{p\left(\mathbf{x}^{(\star)}\right)}{p\left(\mathbf{x}^{(i)}\right)}, 1\right\}$ else reject the sample $\mathbf{x}^{(i+1)} = \mathbf{x}^{(i)}$.

For this example the *proposal distribution* of $\mathbf{z}$ is symmetric $p\left(\mathbf{x}^{(i)} \mid \mathbf{x}^{(\star)}\right) = p\left(\mathbf{x}^{(\star)} \mid \mathbf{x}^{(i)}\right)$. The general form of the Metropolis-Hastings algorithm uses:

$$\alpha = \min\left\{\frac{p\left(\mathbf{x}^{(\star)}\right)p\left(\mathbf{x}^{(i)} \mid \mathbf{x}^{(\star)}\right)}{p\left(\mathbf{x}^{(i)}\right)p\left(\mathbf{x}^{(\star)} \mid \mathbf{x}^{(i)}\right)}, 1\right\}$$

Given a limiting distribution π, generate samples from a finite state-space Markov chain by choosing the proposal function to be a transition matrix $\mathbf{R}$:

$$r_{j,j} = 0, r_{j,k} > 0 \quad (j \neq k)$$

Given X_i, select $\hat{X}_{i+1}$ by sampling from the i^{th} row of $\mathbf{R}$ and accept this sample $\left(X_{i+1} = \hat{X}_{i+1}\right)$ with probability α else reject sample $(X_{i+1} = X_i)$:

$$\alpha = \min\left\{\frac{\pi_{\hat{X}_{i+1}} r_{\hat{X}_{i+1}, X_i}}{\pi_{X_i} r_{X_i, \hat{X}_{i+1}}}, 1\right\}$$

Initial samples often ignored (*burn-in* phase) and *thinning* is performed by taking only every n^{th} sample.

3 Optimization

3.1 Definitions

The quantity to be minimized or maximized is called the *objective function*, or *cost function*, or *utility function*, or *loss function*. The parameters that can be changed are called *control* or *decision variables*. The restrictions on the allowed parameter values are called *constraints*.

Mathematically, the optimization problem is minimize $f(x)$ subject to:

1. *Equality:*

$$c_i(x) = 0, \quad i = 1, \dots, m'$$

2. *Inequality:*

$$c_i(x) \geq 0, \quad i = m'+1, \dots, m$$

Inequality constraints that are restrictions on the allowed values of a single control variable are called *bounds*.

When minimizing $f(x)$ subject to constraints, $\mathcal{S}$ is the *feasible region* and any $x \in \mathcal{S}$ is a *feasible solution*. For an unconstrained problem, $\mathcal{S}$ is infinitely large.

The *gradient* is:

$$g(x) = \nabla f(x) = \left[\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \dots, \frac{\partial f}{\partial x_N}\right]^T$$

The *Hessian* is:

$$H(x) = \nabla(\nabla f(x)) = \nabla^2 f(x) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_N} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_N \partial x_1} & \cdots & \frac{\partial^2 f}{\partial x_N^2} \end{bmatrix}$$

At a feasible point x, a direction d is a *feasible direction* if an arbitrary small move from x in direction d remains feasible.

A function is *convex* if its graph at any point y is never below the tangent at any other point x:

$$f(y) \geq f(x) + \nabla f(x)^T (y - x)$$

A necessary condition for x^* to be a *local minimum* of $f(x)$ in S is $\nabla f(x^*) \cdot d \geq 0$ for all feasible directions d.

If x^* is an interior stationary point, $\nabla f(x^*) = 0$. Then $d^T H(x^*) d > 0$ (the Hessian is *positive definite*) is a sufficient condition of a strong local minimum.

At a stationary point $x^*, d^T H(x^*) d \geq 0$ (the Hessian is *positive semidefinite*) is a necessary condition of strong local minimum.

If $H(x)$ is positive definite everywhere, f is a convex function, and therefore the minimum is unique and a *global minimum*.

A matrix is positive definite if and only if all its eigenvalues are positive.

3.2 Search Methods

Iterative search methods:

1. Start with an initial guess, x_0, for the minimum of $f(x)$.
2. Propose a *search direction*, d_k.
3. Propose a step size α_k along d_k, typically, by an inner line search loop to find the lowest value of $f(x)$ along the direction d_k.
4. Update the estimate of the minimum location, $x_{k+1} = x_k + \alpha_k d_k$.
5. Repeat until convergence.

Convergence criteria:

1. Norm of the residual $\|f(x_{k+1}) - f(x_k)\| < \varepsilon_f$.
2. Norm of the error $\|x_{k+1} - x_k\| < \varepsilon_x$.
3. Norm of the gradient $\|\nabla f(x_k)\| < \varepsilon_g$

ε are user-defined tolerances.

The rate of convergence:

$$\lim_{k\to\infty} \frac{\|x_{k+1} - x^*\|}{\|x_k - x^*\|^p} = \beta$$

The rate of convergence is an *asymptotic* quantity. β is the *convergence ratio* and p is the *order of convergence*.

Line search: Interval reduction until the interval is smaller than a tolerance.

Golden section search: Impose a constant reduction factor $\beta \approx 0.618$ for the interval length. Convergence is linear.

Newton's method: Fit a quadratic function by matching the function value, first and second derivatives evaluated at a single point x_0:

$$f(x) \approx g(x) = f(x_0) + (x - x_0) f'(x_0) + \frac{1}{2}(x - x_0)^2 f''(x_0)$$

To calculate the minimum of $g(x)$, we set its derivative to zero:

$$x = x_0 - \frac{f'(x_0)}{f''(x_0)}$$

Newton's method has a quadratic convergence.

Quasi-Newton methods: The *secant method* takes the current and previous points to estimate the second derivative at the current point:

$$f''(x_1) \approx \frac{f'(x_1) - f'(x_0)}{x_1 - x_0}$$

The estimate for the minimum:

$$x = x_1 - f'(x_1)\frac{x_1 - x_0}{f'(x_1) - f'(x_0)}$$

Convergence is *superlinear*, $p \approx 1.618$.

3.3 Multidimensional Search

Steepest descent method: Set the search direction as the negative gradient $d_k \equiv -\nabla f(x_k)$. Determine by *line search* the step size α_k that minimizes $f(x)$ along d_k.

Ideally, successive search directions are orthogonal to each other $d_k^T d_{k+1} = 0$.

If we approximate the function with a second-order Taylor expansion, the step size is:

$$\alpha_k = -\frac{\nabla f(x_k)^T d_k}{d_k^T H(x_k) d_k}$$

Convergence is linear.

Newton-Raphson method: $f(x)$ is approximated by a quadratic function using the function value, gradient $\nabla f(x)$ and Hessian $H(x) = \nabla^2 f(x)$ at the current location x_k:

$$f(x) \approx q(x) = f(x_k) + \nabla f(x_k)^T (x - x_k) + \frac{1}{2}(x - x_k)^T H(x_k)(x - x_k)$$

The minimum of $q(x)$ occurs when $\nabla q(x) = 0$:

$$x = x_k - [H(x_k)]^{-1} \nabla f(x_k)$$

The search direction is $d_k = -[H(x_k)]^{-1}\nabla f(x_k)$ and the step size is $\alpha_k = 1$. Convergence is quadratic and it converges to the minimum in one iteration in a quadratic equation.

Conjugate Gradient method: We construct the new search direction as $d_k = -\nabla f(x_k) + \beta_k d_{k-1}$. We impose the *conjugacy* condition $d_{k-1}^T A d_k = 0$:

$$\beta_k = \frac{d_{k-1}^T A \nabla f(x_k)}{d_{k-1}^T A d_{k-1}} = \left[\frac{|\nabla f(x_k)|}{|\nabla f(x_{k-1})|}\right]^2$$

Convergence is linear. It converges after N iterations for an N dimensional quadratic form with exact line search.

3.4 Least Squares Fitting

The elements of the *residual* vector $r(x)$ are the errors in the model's predictions for each piece of data $r_j(x) = \phi_j(x) - y_j$. For all the pieces of data, the total error is:

$$f(x) = \sum_{j=1}^{m} r_j^2(x) = r(x)^T r(x)$$

The gradient of $f(x)$ is:

$$\begin{aligned}\nabla f(x) &= 2\sum_{j=1}^{m} \frac{\partial r_j(x)}{\partial x_i} r_j(x) \quad i = 1, 2, \ldots, m \\ &= 2J(x)^T r(x)\end{aligned}$$

$J(x)$ is the *Jacobian matrix* of $r(x)$:

$$J(x) = \begin{bmatrix} \frac{\partial r_1}{\partial x_1} & \cdots & \frac{\partial r_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial r_m}{\partial x_1} & \cdots & \frac{\partial r_m}{\partial x_n} \end{bmatrix}$$

The Hessian matrix $H(x)$ of $f(x)$ is expressed as:

$$H(x) = 2J(x)^T J(x) + 2\sum_{j=1}^{m} r_j(x) R_j(x)$$

$R_j(x)$ is the Hessian of the residual $r_j(x)$.

The Gauss-Newton method: Assume that the residuals at the optimum $r_j(x^*)$ are small. The Hessian $H(x)$ can be approximated as $\tilde{H}(x) = 2J(x)^T J(x)$. The search direction is then:

$$\begin{aligned} d_k &= -\left[\tilde{H}(x_k)\right]^{-1} \nabla f(x_k) \\ &= -\frac{1}{2}\left[J(x_k)^T J(x_k)\right]^{-1} 2J(x_k)^T r(x_k) \\ &= -J(x_k)^+ r(x_k) \end{aligned}$$

$J^+ = \left[J(x_k)^T J(x_k)\right]^{-1} J(x_k)^T$ is the *pseudoinverse* matrix. The Gauss-Newton methods converges nearly quadratically.

3.5 Constrained Optimization

Linear programming (LP) is an optimization problem in which both the objective and constraints are linear.

The m constraints $Ax = b$ define a subspace of $\mathbb{R}^n$. The bounds cut out a portion of this subspace which defines the *feasible region*. The feasible region has a number of corners called *extremal points*, which are called *basic* solutions. $n - m$ control variables are zero at these points and are called *free* variables.

The simplex algorithm:

1. Start with an extremal feasible solution, which is a vertex of the feasible region with m nonzero control variables.
2. Move along an edge of the feasible region to another vertex where f is smaller and continue until you find a vertex where every edge leading away would increase f.

3.6 Nonlinear Constrained Optimization

Minimise a non linear function $f(x)$ subject to m equality constraints $h_i(x) = 0$ and n inequality constraints $g_j(x)$. The set of equality and inequality constraints defines the *feasible region*.

At a location x, constraints are *active* if they do not allow x to infinitesimally change in at least one direction. Otherwise constraints are *inactive*.

At a minimum, the gradient of the function is parallel to the gradient of the constraint:

$$\nabla f(x^*) = -\lambda \nabla h(x^*)$$

The optimality condition can be recovered by defining an *unconstrained minimisation problem* with a new cost function:

$$L(x, \lambda_i) = f(x) + \sum_{i=1}^{m} \lambda_i h_i(x)$$

L is the *Lagrangian* and λ_i are the *Lagrange multipliers*, which are scalars.

If x^* is a stationary point of the Lagrangian then $\nabla L(x^*, \lambda) = 0$:

$$\begin{aligned} \nabla f(x^*) + [\nabla h(x^*)]^T \lambda &= 0 \\ h_i(x^*) = 0 \quad i &= 1, 2, \ldots, m \end{aligned}$$

A sufficient condition for a stationary point x^* to be a minimum is provided by the Hessian: $d^T \nabla^2 L(x^*) d > 0$ for $d \in$ space tangent to all constraints. Hence, $\nabla^2 L$ is positive definite in the *tangent* space.

For inequality constraints, we define the Lagrangian:

$$L\left(x, \mu_j\right) = f(x) + \sum_{j=1}^{n} \mu_j g_j(x)$$

If x^* is stationary point of $L(x)$, then $\nabla L\left(x^*, \mu_j\right) = 0$:

$$\begin{aligned} \nabla f(x^*) + [\nabla g(x^*)]^T \mu &= 0 \\ \mu_j g_j(x) &= 0 \\ \mu_j \geq 0, \quad j &= 1, 2, \ldots, n \end{aligned}$$

These conditions are called *Karush-Kuhn-Tucker* (KKT) conditions and μ_j are the *KKT multipliers*.

In the general case with both equality and inequality constraints:

$$L(x, \lambda, \mu) = f(x) + \sum_{i=1}^{m} \lambda_i h_i(x) + \sum_{j=1}^{n} \mu_j g_j(x)$$

Sensitivity:

$$\delta f(x^*) = -\delta h(x^*)^T \lambda - \delta g(x^*)^T \mu$$

$-\lambda_i$ and $-\mu_j$ are the sensitivities of f to an infinitesimal change in the constraint h_i and g_j, respectively.

Penalty functions: We can replace the constrained optimization problem with an approximated unconstrained optimization problem:

$$q(x, \kappa) = f(x) + \kappa \sum_{i=1}^{m} h(x)^2$$

κ is the user defined penalty parameter. For inequality constraints, $g(x) \leq 0$, we can use an asymmetric penalty funtion such as $\kappa \max[0, g(x)]^2$.

Battier functions:

1. Inverse barrier function:

$$q(x, \kappa) = f(x) - \frac{1}{\kappa} \sum_{j}^{m} \frac{1}{g_j(x)}$$

2. Logarithmic barrier function:

$$q(x, \kappa) = f(x) - \frac{1}{\kappa} \sum_{j}^{m} \ln\left[-g_j(x)\right]$$

For both, the larger κ, the sharper the divergence of the singularity near the constraint boundary.

3.7 Global Optimization

We want to sample the function $f(x)$ within an average range T in a way that we maximise the average information on the distribution of values of $f(x)$. Our goal is to minimize $f(x)$ which is called *energy* $f(x) = E(x)$.

The *Boltzmann distribution* is the maximum entropy distribution:

$$\Pr(E(x)) \propto \exp\left(-\frac{E(x)}{T}\right)$$

Simulated annealing: Change the configuration x using a *proposed distribution* U which is a uniform distribution centred at $x = 0$ and h is the width/shift:

$$x_{n+1} \overset{?}{\leftarrow} x_n + U\left(-\frac{1}{2}, \frac{1}{2}\right) h$$

Follow the *Metropolis-Hastings* rule for accepting or rejecting the proposal:

$$\begin{aligned} \Pr\left(\text{accept } x \to x'\right) &= \min\left(\frac{e^{-E(x')/T}}{e^{-E(x)/T}}, 1\right) \\ &= \min\left(e^{-\Delta E/T}, 1\right) \end{aligned}$$

$\Delta E = E(x') - E(x)$ is the energy change of the proposal. Annealing corresponds to lowering the temperature slowly as the simulation of the stochastic process proceeds.

$$T_{n+1} \leftarrow T_n \times (1 - \text{ rate })$$

(The End)