

Probabilistic Modeling in Computational Biology

Ian Holmes

Contents

1	Introduction and overview	4
1.1	Probability basics	5
1.2	Entropy and information	6
1.3	Conjugate priors	7
2	Molecular substitution processes	12
2.1	Overview of discrete Markov chains	12
2.2	Point substitution processes	13
2.3	Felsenstein’s pruning algorithm	16
2.4	Elston-Stewart peeling; sum-product algorithm and factor graphs	19
2.5	Explicit solution of substitution models	23
2.6	Simulating trajectories from Markov chains; Gillespie’s algorithm	24
3	Estimation of model parameters	26
3.1	Heuristic estimation of phylogenetic trees	26
3.2	Markov Chain Monte Carlo sampling	27
3.3	Expectation Maximization (EM)	33
3.4	EM for Gaussian mixture model	36

3.5	EM for substitution models	37
3.6	Priors over trees	43
4	Hidden-state models for biological sequences	45
4.1	Selected applications in the literature	45
4.2	Single-sequence hidden Markov models	45
4.3	Posterior probabilities for single-sequence HMMs	48
4.4	Pair Hidden Markov models	50
4.5	Evolutionary Hidden Markov models	54
4.6	Discriminative models and conditional random fields	55
5	Data compression	58
5.1	Coding and data compression	58
5.2	String compression in practice	60
6	Statistical alignment	61
6.1	Mechanisms of sequence mutation	61
6.2	The “links model” and string transducers	61
6.3	Multiple alignment with the links model	70
6.4	More realistic evolutionary models	70
7	Stochastic grammars for biological sequences	72
7.1	Overview of transformational grammars	72
7.2	RNA structure	73
7.3	Dynamic programming algorithms for SCFGs	76
7.4	Pair SCFGs, evolutionary SCFGs and tree transducers	78
7.5	Graph grammars	79
7.6	Discriminative grammars: conditional log-linear models	79

8	Continuous random variables	80
8.1	Review: properties of Gaussian distributions	80
8.2	Gaussian processes as stochastic processes	81
8.3	Gaussian processes as tools for machine learning	82
8.4	The Fokker-Planck equation	85
8.5	The Wiener process	89
8.6	The Ornstein-Uhlenbeck process	91
8.7	Phylogenetically related Brownian variables	93

1 Introduction and overview

- Class content:
 - Molecular biology (mostly assumed, some revision):
 - * Transcription, translation: DNA \rightarrow RNA \rightarrow protein
 - * Splicing; the spliceosome
 - * Ribosomal RNA and transfer RNA; the genetic code
 - * Noncoding RNA genes
 - * Pre-transcriptional regulation: transcription factors, initiation complex
 - * Post-transcriptional regulation: RNA localisation
 - * Structure of DNA, RNA; specific basepairing; protein-nucleic contacts
 - Probabilistic models:
 - * Bayes' theorem; common probability distributions; linear algebra (all assumed)
 - * Graphical models / factor graphs (tree, trellis...)
 - * State machines (HMMs, transducers)
 - * Linguistics-influenced grammars (SCFGs)
 - * Discriminative versions of the above generative models (CRFs, CLLMs...)
 - * Discrete-state continuous-time Markov chains
 - * Gaussian processes; stochastic differential equations (Langevin, Fokker-Planck...)
 - Statistical algorithms:
 - * Variants of dynamic programming / sum-product (Viterbi, Baum-Welch, CYK, Inside-Outside, Pruning, Peeling...)
 - * Posterior decoding
 - * Markov Chain Monte Carlo (MCMC)
 - * Parameter estimation: maximum likelihood/Expectation Maximization

* Simulation

- Not covered (beyond superficial mention):
 - Population genetics. Populations. Birth-death processes.
 - Brownian motion; Berg's model of *E.coli* motion (tumble/run).
 - Ecosystems. Lotka-Volterra equations (prey x , predators y)

$$\dot{x} = Ax - Bxy, \quad \dot{y} = -Cy + Dxy$$

- Ecological superprocesses: space and time. Lattices. Brownian birth/death.
- Stochastic gene regulation (McAdams-Arkin); signal transduction
- Reaction kinetics. Michaelis-Menten: $E + S \leftrightarrow ES \leftrightarrow EP \rightarrow E + P$
- Stochastic biophysics, e.g. protein folding& misfolding; diffusion through membranes.
- Instrumentation: image processing, microarrays, Markov random fields
- Clustering e.g. of microarray, proteomics data
- Natural language (e.g. in the biological literature)
- Biological databases; ontologies

1.1 Probability basics

- Review of probability theory; information theory and data compression
 - use $p(x)dx$ for both continuous & discrete distributions
 - conditional/joint probability; independence
 - expectation, variance, covariance; covariance matrix

- change of variables: if $p(x)$ is pdf of x , and $y = f(x)$, what is pdf $q(y)$ of y ? Clearly $dy = f'(x)dx$ and $p(x)dx = q(y)dy$; thus $q(y) = p(x)/f'(x)$ where $x = f^{-1}(y)$.
- expectation of a function, $\langle f(x) \rangle$; moments $\langle x^n \rangle$
- Characteristic function $\phi(s) = \langle \exp(\imath sx) \rangle$. NB
 - * $\phi(s)$ is Fourier transform of $p(x)$
 - * Fourier inverse: $p(x) = (2\pi)^{-1} \int ds \phi(s) \exp(-\imath xs)$
 - * Moments given by $\langle x^n \rangle = \left(-\imath \frac{d}{ds}\right)^n \phi(s)$
 - * If $y = ax$ then $G_y(k) = \langle \exp(\imath kax) \rangle = G_x(ka)$
 - * Convolution: if $y = \sum_i x_i$ where $\{x_i\}$ are independent RVs with c.f. $\phi_i(s)$, then $\phi_y(s) = \prod_i \phi_i(s)$
- For distributions over integers, generating function $G(s) = \sum_n p(s)s^n$ can also be useful

1.2 Entropy and information

- Shannon information content of an outcome, $h(x) = -\log_2 p(x)$
- entropy of a probability distribution, $H_p = \langle h(x) \rangle$
- relative entropy; Gibbs' inequality; proof thereof
 - relative entropy or Kullback-Leibler divergence is $D(p||q) = \sum_x p(x) \log \frac{p(x)}{q(x)}$
 - Gibbs' inequality: $D(p||q) \geq 0$ with equality iff $p = q$
 - Proof: start with Jensen's inequality for convex functions
 - * A function $f(x)$ is convex over (a, b) if every chord lies above the function. $f''(x) \geq 0$ is sufficient.
 - * Jensen's inequality: if x is an rv and $f(x)$ is convex, then $\langle f(x) \rangle \geq f(\langle x \rangle)$
 - * Physical version: let $p(x)$ be x -distribution of masses on curve $y = f(x)$. Then center of gravity $(\langle x \rangle, \langle f(x) \rangle)$ lies above curve.

- * If $f(x)$ is concave, then inequality is reversed
- Proof of Gibbs' inequality: apply Jensen to $u = q(x)/p(x)$, $f(u) = -\log u$
- Proof that $H_p \leq |\mathcal{X}|$: apply Jensen to $u = 1/p(x)$, $f(u) = -\log u$

1.3 Conjugate priors

- Two probability distributions $P(x|\theta)$ and $P(\theta)$ are said to be *conjugate* if the posterior $P(\theta|x)$ is of the same family as the prior $P(\theta)$
- The parameters of the conjugate prior distribution are called *hyperparameters*.
- Note that, in general,

$$P(\theta|x) = \frac{P(x|\theta)P(\theta)}{P(x)} = \frac{P(x|\theta)P(\theta)}{\int P(x|\theta')d\theta'}$$

where denominator is $P(x)$, the *Bayesian evidence*, which does not depend on θ . That is, the dependence of $P(\theta|x)$ on θ is same as that of $P(x|\theta)P(\theta)$, but to get an exact form for $P(\theta|x)$, we need to integrate out θ to find the evidence $P(x)$.

- Exponential, Poisson, Gamma distributions

- Exponential distribution: pdf for time to first event, T , given that mean event rate is μ

$$P(T|\mu) = \mu \exp(-\mu T)$$

- Poisson: probability distribution of number of events, n , in time T , given that mean event rate is μ

$$P(n|\mu) = \frac{(\mu T)^n \exp(-\mu T)}{n!}$$

(NB exponential distribution can be obtained by setting $n = 0$ and differentiating w.r.t. T)

- Gamma, conjugate to Poisson. Shape α , scale β .

$$P(\mu|\alpha, \beta) = \frac{\mu^{\alpha-1} \beta^\alpha \exp(-\mu\beta)}{\Gamma(\alpha)}$$

where $\Gamma()$ is the *gamma function*

$$\Gamma(z) = \int_{u=0}^{\infty} u^{z-1} \exp(-u) du$$

Clearly $\Gamma(1) = 1$. Integrating by parts for positive integer z ,

$$\begin{aligned} \Gamma(z+1) &= \int_{u=0}^{\infty} u^z \exp(-u) du \\ &= [-u^z \exp(-u)]_{u=0}^{\infty} + z\Gamma(z) \\ &= z! \end{aligned}$$

- Properties of gamma distribution: mean of μ is α/β and variance is α/β^2 .
- Conjugacy:

$$\begin{aligned} P(n) &= \frac{\Gamma(\alpha')}{\Gamma(n+1)\Gamma(\alpha)} \frac{T^n \beta^\alpha}{(\beta')^{\alpha'}} \\ P(\mu|n) &= \frac{\mu^{\alpha'-1} (\beta')^{\alpha'} \exp(-\mu\beta')}{\Gamma(\alpha')} \end{aligned}$$

i.e. posterior for μ is a gamma distribution with shape $\alpha' = \alpha + n$ and scale $\beta' = \beta + t$. (So α is like a “pseudocount”, and β a “pseudotime”.)

- Lengthier derivation:

$$\begin{aligned} P(n) &= \int_{\mu=0}^{\infty} P(n|\mu) P(\mu) d\mu \\ &= \int_{\mu=0}^{\infty} \frac{(\mu T)^n \exp(-\mu T)}{\Gamma(n+1)} \frac{\mu^{\alpha-1} \beta^\alpha \exp(-\mu\beta)}{\Gamma(\alpha)} d\mu \end{aligned}$$

$$\begin{aligned}
&= \frac{T^n \beta^\alpha}{\Gamma(n+1)\Gamma(\alpha)} \int_{\mu=0}^{\infty} \mu^{n+\alpha-1} \exp(-\mu(T+\beta)) d\mu \\
&= \frac{T^n \beta^\alpha}{\Gamma(n+1)\Gamma(\alpha)} \frac{1}{(T+\beta)^{n+\alpha}} \int_{u=0}^{\infty} u^{n+\alpha-1} \exp(-u) du \\
&= \frac{\Gamma(\alpha')}{\Gamma(n+1)\Gamma(\alpha)} \frac{T^n \beta^\alpha}{(\beta')^{(\alpha')}}
\end{aligned}$$

- Normal and Normal-gamma distributions

- Mean μ , precision τ (precision is reciprocal of variance). Data $\{x_i\}$ with moments $m_k = \sum_i x_i^k$. Likelihood

$$P(\mathbf{x}|\mu, \tau) = \prod_i \sqrt{\frac{\tau}{2\pi}} \exp\left(-\frac{\tau}{2}(x_i - \mu)^2\right) = \left(\frac{\tau}{2\pi}\right)^{m_0/2} \exp\left(-\frac{\tau}{2}(m_2 - 2\mu m_1 + \mu^2 m_0)\right)$$

- Conjugate prior: use a gamma prior (shape α , scale β) for $P(\tau)$, and a Normal prior (mean ϵ , precision $\lambda\tau$) for $P(\mu|\tau)$, so the full set of hyperparameters is $\{\alpha, \beta, \epsilon, \lambda\}$ and the prior is

$$P(\mu, \tau) = \frac{e^{-\beta\tau} \tau^{\alpha-1} \beta^\alpha}{\Gamma(\alpha)} \times \sqrt{\frac{\lambda\tau}{2\pi}} \exp\left(-\frac{\lambda\tau}{2}(\mu - \epsilon)^2\right)$$

- Conjugacy:

$$\begin{aligned}
P(\mathbf{x}) &= (2\pi)^{-m_0/2} \frac{\Gamma(\alpha')}{\Gamma(\alpha)} \frac{\beta^\alpha}{\beta'^{\alpha'}} \sqrt{\frac{\lambda}{\lambda'}} \\
P(\mu, \tau|\mathbf{x}) &= \frac{e^{-\beta'\tau} \tau^{\alpha'-1} \beta'^{\alpha'}}{\Gamma(\alpha')} \times \sqrt{\frac{\lambda'\tau}{2\pi}} \exp\left(-\frac{\lambda'\tau}{2}(\mu - \epsilon')^2\right)
\end{aligned}$$

where $\epsilon' = \frac{\lambda\epsilon + m_1}{\lambda + m_0}$, $\lambda' = \lambda + m_0$, $\alpha' = \alpha + \frac{m_0}{2}$ and $\beta' = \beta + \frac{1}{2}\left(\epsilon^2 + m_2 - \frac{(\lambda\epsilon + m_1)^2}{\lambda + m_0}\right)$.

- NB if we regard τ as fixed, then Gaussian is auto-conjugate.

- If we rescale the posterior distribution for τ , by dividing it by $(m_2/m_0 - m_1^2/m_0)^{-1}$ (the posterior mean estimate for τ), then the resultant gamma distribution with shape $m_0/2$ and scale $1/2$ is the “ χ^2 distribution”. (NB the canonical definition of the quantity χ^2 is in the special case where $\mu = 0$ and $\tau = 1$, in which case $\chi^2 = m_2$ and it does, indeed, follow this same gamma distribution.)

- Multinomial, Dirichlet distributions

- Multinomial: K possible outcomes, outcome probabilities \mathbf{p} , outcome frequencies \mathbf{n} in $N = \sum_k n_k$ trials

$$P(\mathbf{n}|\mathbf{p}) = \frac{N!}{\prod_k n_k!} \prod_k p_k^{n_k}$$

- Conjugate prior: let \mathbf{a} be a vector of *pseudocounts*, $a_1 \dots a_k$. The *Dirichlet distribution* for \mathbf{p} is

$$P(\mathbf{p}|\mathbf{a}) = \frac{\prod_i p_i^{a_i-1}}{\mathcal{B}(\mathbf{a})} \delta\left(\sum_i p_i - 1\right)$$

where $\mathcal{B}()$ is the *type one Dirichlet integral* or the *multinomial beta function*

$$\mathcal{B}(\mathbf{a}) = \int \left(\prod_i p_i^{a_i-1}\right) \delta\left(\sum_i p_i - 1\right) d\mathbf{p} = \frac{\prod_k \Gamma(a_k)}{\Gamma(\sum_k a_k)}$$

Properties: mean value of p_k is $a_k/\sum_j a_j$. Modal value is $p_k = (a_k - 1)/(\sum_j a_j - 1)$.

- Note the relationship between the multinomial coefficient and \mathcal{B}

$$\frac{N!}{\prod_k n_k!} = \frac{1}{\mathcal{B}(\mathbf{n} + \mathbf{1})} \times \frac{\Gamma(N + 1)}{\Gamma(N + K)}$$

- Conjugacy:

$$P(\mathbf{n}|\mathbf{a}) = \frac{\mathcal{B}(\mathbf{a}')}{\mathcal{B}(\mathbf{a})} \frac{N!}{\prod_k n_k!}$$

$$P(\mathbf{p}|\mathbf{n}, \mathbf{a}) = \frac{\prod_i p_i^{a'_i-1}}{\mathcal{B}(\mathbf{a}')} \delta\left(\sum_i p_i - 1\right)$$

where $\mathbf{a}' = \mathbf{a} + \mathbf{n}$ (hence the name “pseudocount”).

- Special case of {Multinomial,Dirichlet} when $K = 2$ is {Binomial,Beta}, and \mathcal{B} is the *beta function*.
- The following snippets from Mathworld point to a more rigorous solution of the type one Dirichlet integral
 - The beta integral may be obtained by writing $m!n!$ as a product of gamma functions with integrands u, v , transforming to $(x, y) = (\sqrt{u}, \sqrt{v})$ and then to polar co-ordinates $(x, y) = r(\cos \theta, \sin \theta)$. This yields

$$\mathcal{B}(m + 1, n + 1) = 2 \int_0^{\pi/2} \cos^{2m+1} \theta \sin^{2n+1} \theta d\theta = \frac{m!n!}{(m + n + 1)!}$$

See mathworld.wolfram.com/BetaFunction.html for details.

- More info on Dirichlet type one integral at mathworld.wolfram.com/DirichletIntegrals.html

2 Molecular substitution processes

2.1 Overview of discrete Markov chains

- Examples of finite & infinite discrete state spaces in biology:
 - Alphabets Ω : nucleotides, amino acids
 - Short strings Ω^N : codons, RNA basepairs, TF binding sites
 - Arbitrary-length strings Ω^* : proteins, viral genomes, chromosomes
 - * Periodic strings: circular plasmids, bacterial genomes
 - Vectors of strings $[\Omega^*]^N$, e.g. genomes
 - Integers: microsatellite repeat lengths, population size, allele frequency
 - Integer vectors: numbers of molecules by type/compartment, population by species/location (e.g. predator/prey)
 - Gating of ion channels (“open” or “closed”; conductance measurable by patch clamp)
 - Gene circuits
- NB in general (but not guaranteed) large/infinite chains are sparse w.r.t. allowed mutations
 - Ergodicity implies that all states are connected; for continuous chains, there are stronger implications about rates
- Matrix notation for equation of state
 - Discrete-time: $p_j^{[n+1]} = \sum_i p_i^{[n]} Q_{ij}$ where $p_j^{[n]} = P(x_n = j)$ and $Q_{ij} = P(x_{n+1} = j | x_n = i)$
 - * BTW, can write this as a graphical model $x_1 \rightarrow x_2 \rightarrow x_3 \rightarrow x_4 \dots$
 - Matrix form: $\mathbf{p}^{[n+1]} = \mathbf{p}^{[n]} \mathbf{Q}$
 - Discrete→continuous: suppose each step $n \rightarrow n + 1$ represents a time interval Δt
 - * We must now think in terms of the *rate* R_{ij} of mutation $i \rightarrow j$, i.e. the “probability per unit time” of the mutation.

- * Conventionally, define $R_{ii} = -\sum_{i \neq j} R_{ij}$, i.e. the negative “exit rate” from state i
- * Then $\mathbf{Q} = \mathbf{I} + \mathbf{R}\Delta t$ where $\sum_j R_{ij} = 0 \forall i$, so that $\Delta \mathbf{p} = \mathbf{p}\mathbf{R}\Delta t$
- * Continuous limit: $\frac{d}{dt}\mathbf{p}(t) = \mathbf{p}(t)\mathbf{R}$
- In general can have $\mathbf{R} \equiv \mathbf{R}(t)$ but will mainly consider *homogeneous* chains where \mathbf{R} is constant
- Equilibrium π : $\pi\mathbf{Q} = \mathbf{0}$ (discrete), $\pi\mathbf{R} = \mathbf{0}$ (continuous)
 - Finite chains must have an equilibrium; infinite chains, not necessarily
 - Practical issue: how to find the equilibrium? QR decomposition works
 - A chain at equilibrium \forall times t is called *stationary*
- Reversibility/detailed balance: $\pi_i R_{ij} = \pi_j R_{ji}$
 - Flow of probability from $i \rightarrow j$ exactly balances $j \rightarrow i$ at equilibrium
 - At instantaneous level, forward & backward mutations indistinguishable
 - Implies \exists symmetric matrix: $S_{ij} = R_{ij}\sqrt{\pi_i/\pi_j}$, so $\mathbf{S} = \mathbf{P}^{1/2} \mathbf{R} \mathbf{P}^{-1/2}$ where \mathbf{P} is diagonal and $\mathbf{P}_{kk} = \pi_k$
 - Equilibrium doesn't imply reversibility; e.g. unidirectional cycle $A \rightarrow C \rightarrow G \rightarrow T \rightarrow A$
- Origins of irreversibility; energy consumption (e.g. ATP)

2.2 Point substitution processes

- Examples: nucleotides, amino acids, RNA basepairs, codons (give example alignment in each case)
- Intuitive justification of the matrix exponential
 - Solution to $\mathbf{p}^{[t+1]} = \mathbf{p}^{[t]}\mathbf{Q}$ is clearly $\mathbf{p}^{[t]} = \mathbf{p}^{[0]}\mathbf{Q}^t$
 - * Analogy to corresponding one-dimensional difference equation: $p^{[t+1]} = p^{[t]}Q \Rightarrow p^{[t]} = p^{[0]}Q^t$

- Consider one-dimensional ODE, $\frac{dp}{dt} = pR$. Solution is $p(t) = p(0) \exp(Rt)$
- By analogy, would like solution of $\frac{d}{dt}\mathbf{p}(t) = \mathbf{p}(t)\mathbf{R}$ to be $\mathbf{p}(t) = \mathbf{p}(0)\mathbf{M}(t)$ where $\mathbf{M}(t) = \exp(\mathbf{R}t)$
- Here we've introduced (or wished for) the important *matrix exponential*, $\exp(\mathbf{A})$
 - Can define e.g. by Taylor series, $\exp(\mathbf{A}) = \sum_{n=0}^{\infty} \mathbf{A}^n/n!$
 - Note however that we have to be careful about transferring properties “blindly” over from scalar exponential
 - For example, if $\mathbf{A} \equiv \mathbf{A}(x, \dots)$ then, in general, $\frac{\partial}{\partial x}[\exp(\mathbf{A})] = \frac{\partial \mathbf{A}}{\partial x} \exp(\mathbf{A})$ does **NOT** hold
 - * this is because $\frac{\partial}{\partial x} \exp(\mathbf{A})$ contains terms of form $\sum_{k=0}^{n-1} [\mathbf{A}^k] \frac{\partial \mathbf{A}}{\partial x} [\mathbf{A}^{n-k-1}]$
 - * such terms are *not* in general equal to $n \frac{\partial \mathbf{A}}{\partial x} \mathbf{A}^{n-1}$, because \mathbf{A} and $\frac{\partial \mathbf{A}}{\partial x}$ may not commute
 - * exception is if these two matrices do, in fact, commute; then we recover the identity from the scalar case
 - * obvious case is when $\mathbf{A} \equiv \mathbf{R}t$ and $x \equiv t$, so $\frac{d}{dt} \exp(\mathbf{R}t) = \exp(\mathbf{R}t)\mathbf{R}$. This means our Taylor definition works!
 - Likewise, in general $\exp(\mathbf{A})\exp(\mathbf{B})$ does not equal $\exp(\mathbf{A} + \mathbf{B})$. However, $\mathbf{M}(t + t') = \mathbf{M}(t)\mathbf{M}(t')$. This is a form of the *Chapman-Kolmogorov equation*, often used as the principal definition of a Markov chain. It can also be used as an implementation shortcut/approximation to computing $M(t)$
 - * Strictly, Chapman-Kolmogorov is $\sum_j P(x_t = j | x_0 = i) P(x_{t+t'} = k | x_t = j) = P(x_{t+t'} = k | x_0 = i)$
 - For Markov chains, $\mathbf{M}(t) = \sum_{n=0}^{\infty} \mathbf{R}^n t^n/n!$. If t is small, then $t^n/n! \rightarrow 0$ for larger n , i.e. can approximate $\mathbf{M}(t)$ with truncated Taylor series or other finite polynomial. This is particularly time-efficient if \mathbf{R} is sparse, in which case \mathbf{R}^n will also be sparse for small n
 - For finite chains, can do exact calculations using eigenvalues & eigenvectors; see below
- Review of matrix diagonalisation
 - Matrix \mathbf{A} , eigenvalue λ : right (column) eigenvector $\mathbf{A}\mathbf{u} = \lambda\mathbf{u}$, left (row) eigenvector $\mathbf{v}\mathbf{A} = \lambda\mathbf{v}$
 - Eigenvalues are solutions of characteristic equation, $|\mathbf{A} - \lambda\mathbf{I}| = 0$
 - (Assuming matrix of right eigenvectors is invertible...) Can write $\mathbf{A} = \mathbf{U}\mathbf{D}\mathbf{U}^{-1}$ where \mathbf{D} is diagonal ($D_{kk} = \lambda_k$), $\mathbf{u}^{(k)}$ is k 'th column of \mathbf{U} , $\mathbf{v}^{(k)}$ is k 'th row of \mathbf{U}^{-1}

- Thus $\mathbf{A}^n = \mathbf{U}\mathbf{D}^n\mathbf{U}^{-1}$ and $\exp(\mathbf{A}) = \mathbf{U} \exp(\mathbf{D}) \mathbf{U}^{-1}$
- If \mathbf{A} is symmetric then left and right eigenvectors are the same, i.e. $\mathbf{U}^{-1} = \mathbf{U}^T$, and all λ_k are real

- Alternative definition of matrix exponential uses limit of discrete-time approximation:

$$\exp(\mathbf{R}t) = \lim_{\Delta t \rightarrow 0} (\mathbf{I} + \mathbf{R}\Delta t)^{t/\Delta t}$$

- Eigensolution of matrix exponential

- General properties of eigenvalues and eigenvectors

- * Zero eigenvalue(s) correspond to equilibrium distribution
 - More than one eval \Rightarrow orthogonal equilibria \Rightarrow noncommunicating state cliques
 - That at least one eval is zero can be seen by considering $\mathbf{R}\mathbf{x} = \mathbf{0}$ where $x_i = 1 \forall i$
- * All remaining eigenvalues of \mathbf{R} are negative; eigenvectors correspond to “modes of information loss”
 - Proof that evals are negative?
- * Eigenvalues are either real, or come in complex conjugate pairs

- Reversible processes

- * Algorithms to diagonalise symmetric matrices are simpler & stabler than those for asymmetric matrices
- * Recall, for reversible models, $\mathbf{R} = \mathbf{P}^{-1/2} \mathbf{S} \mathbf{P}^{1/2}$ where \mathbf{P} is diagonal and \mathbf{S} is symmetric
 - Not difficult to show that evals of \mathbf{S} are those of \mathbf{R} , and evects are trivially related by \mathbf{P}
- * Given π ($\Rightarrow \mathbf{P}$), we can work with evals & evects of \mathbf{S} rather than \mathbf{R}
- * In particular, $\mathbf{R}^N = \mathbf{P}^{-1/2} \mathbf{S}^N \mathbf{P}^{1/2}$ and $\exp(\mathbf{R}t) = \mathbf{P}^{-1/2} \exp(\mathbf{S}t) \mathbf{P}^{1/2}$

- Irreversible processes

- * Can group complex eigenvalues into conjugate pairs (corresponding to irregularly periodic behaviour)

- Proofs of selected theorems in linear algebra, quoted above

- Real-valued symmetric matrices have real eigenvalues; eigenvectors can be chosen to be orthonormal
 - * Proof that evals are real: conjugate of $\mathbf{A}\mathbf{u} = \lambda\mathbf{u}$ is $\bar{\mathbf{A}}\bar{\mathbf{u}} = \bar{\lambda}\bar{\mathbf{u}}$ (because, in general, $\overline{\bar{x}y} = x\bar{y}$)
 - * Thus $\bar{\mathbf{u}}^T\mathbf{A}\mathbf{u} = \lambda\bar{\mathbf{u}}^T\mathbf{u}$ and $\mathbf{u}^T\bar{\mathbf{A}}\bar{\mathbf{u}} = \bar{\lambda}\mathbf{u}^T\bar{\mathbf{u}}$. Subtracting these gives $\bar{\mathbf{u}}^T\mathbf{A}\mathbf{u} - \mathbf{u}^T\bar{\mathbf{A}}\bar{\mathbf{u}} = (\lambda - \bar{\lambda})\bar{\mathbf{u}}^T\mathbf{u}$
 - * Since \mathbf{A} is symmetric, $\bar{\mathbf{u}}^T\mathbf{A}\mathbf{u} = \mathbf{u}^T\bar{\mathbf{A}}\bar{\mathbf{u}}$
 - * Also, $\mathbf{u}^T\bar{\mathbf{u}} > 0$ unless $\mathbf{u} = 0$, because $\bar{x}x \geq 0$ unless $x = 0$. Thus $\lambda - \bar{\lambda} = 0$.
 - * Proof that \mathbf{u}_i and \mathbf{u}_j are orthogonal if $\lambda_i \neq \lambda_j$:
 - * Left-multiply $\mathbf{A}\mathbf{u}_i = \lambda_i\mathbf{u}_i$ by \mathbf{u}_j^T to obtain $\mathbf{u}_j^T\mathbf{A}\mathbf{u}_i = \lambda_i\mathbf{u}_j^T\mathbf{u}_i$ and likewise $\mathbf{u}_i^T\mathbf{A}\mathbf{u}_j = \lambda_j\mathbf{u}_i^T\mathbf{u}_j$.
 - * Subtracting gives $\mathbf{u}_j^T\mathbf{A}\mathbf{u}_i - \mathbf{u}_i^T\mathbf{A}\mathbf{u}_j = (\lambda_i - \lambda_j)\mathbf{u}_i^T\mathbf{u}_j$. LHS is zero by symmetry of \mathbf{A} . Hence $\mathbf{u}_i^T\mathbf{u}_j = 0$.
 - * Proof that one can find m orthonormal eigenvectors for an eigenvalue repeated m times is more involved
 - See e.g. <http://www.quandt.com/papers/basicmatrixtheorems.pdf>
 - Mirrored at <http://biowiki.org/BioE241>
- Eigenvalues of real-valued square matrices are either real, or occur in complex conjugate pairs
 - * Sketch of proof: complex conjugativity is distributive over multiplication and addition, therefore if λ is a zero of the characteristic equation then $\bar{\lambda}$ must also be a zero.
 - * If λ is an eigenvalue then $|\mathbf{R} - \lambda\mathbf{I}| = f(\lambda) = \sum_{n=0}^N a_n\lambda^n = 0$ where the a_n are all real
 - * Therefore $f(\bar{\lambda}) = \sum_{n=0}^N a_n(\bar{\lambda})^n = \sum_{n=0}^N a_n\bar{\lambda}^n = \bar{f(\lambda)} = \bar{0} = 0$, so $\bar{\lambda}$ is also an eigenvalue.

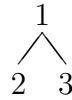
2.3 Felsenstein's pruning algorithm

- Notation and model
 - Let x_i be (imputed) state of node i and let y_i be (observed) character at leaf node i
 - * Let $X = \{x_i\}$ be the set of all node states, and let $Y = \{y_i\}$ be the set of all observed characters at leaf nodes
 - * Also let Y_n be the set of all y_i descended from node n , and \bar{Y}_n the set of all remaining y_i
 - Let t_{ij} be evolutionary “distance” (time) from i to j
 - Then if x_p is parent and x_c is child, $P(x_c|x_p) = M(t_{pc})_{x_px_c}$

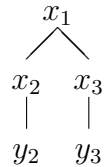
- * If c is a leaf node, then $P(y_c|x_c) = \delta_{x_c y_c}$, i.e. all observations y_c exactly reflect the corresponding state x_c
- * Thus $P(y_c|x_p) = \sum_{x_c} P(x_c|x_p)P(y_c|x_c) = M(t_{pc})_{x_p y_c}$
- Root node (always node 1) has probability $P(x_1) = \pi_{x_1}$ (assuming Ur-ancestor was at equilibrium)

• Felsenstein's pruning algorithm: first example of DP in this class

- Consider tree T_1 :



with the following dependencies between state variables X and observations Y :



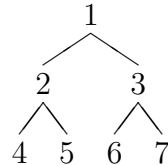
* Joint likelihood of all nodes:

$$P(X;Y) = P(x_1)P(x_2|x_1)P(x_3|x_1)P(y_2|x_2)P(y_3|x_3)$$

* Marginal likelihood of leaves:

$$\begin{aligned} P(Y) &= \sum_X P(X;Y) \\ &= \sum_{x_1} P(x_1)P(y_2|x_1)P(y_3|x_1) \end{aligned}$$

- Now consider tree T_2 :



* Joint likelihood of all nodes:

$$P(X;Y) = P(x_1)P(x_2|x_1)P(x_3|x_1)P(x_4|x_2)P(x_5|x_2)P(x_6|x_3)P(x_7|x_3) \prod_{n=4}^7 P(y_n|x_n)$$

* Marginal likelihood of leaves:

$$\begin{aligned}
P(Y) &= \sum_X P(X; Y) \\
&= \sum_{x_1} \sum_{x_2} \sum_{x_3} P(x_1)P(x_2|x_1)P(x_3|x_1)P(y_4|x_2)P(y_5|x_2)P(y_6|x_3)P(y_7|x_3) \\
&= \sum_{x_1} P(x_1) \left(\sum_{x_2} P(x_2|x_1)P(y_4|x_2)P(y_5|x_2) \right) \left(\sum_{x_3} P(x_3|x_1)P(y_6|x_3)P(y_7|x_3) \right)
\end{aligned}$$

* Rearranging sums changes complexity from $O(A^N)$ to $O(A^2N)$ where A is alphabet size, N is # of nodes in tree

* Note pattern: compute all conditional probs of form $F_n(x_n) \equiv P(\{y_i : i \text{ descended from } n\} | x_n) = P(Y_n | x_n)$

* Let C_n be the set of immediate children of n (so e.g. $C_1 = \{2, 3\}$). Rule to compute F_n is then

$$F_n(x_n) = \begin{cases} \prod_{c \in C_n} \left(\sum_{x_c} P(x_c|x_n)F_c(x_c) \right) & \text{if } n \text{ is an internal node} \\ P(y_n|x_n) & \text{if } n \text{ is a leaf node} \end{cases}$$

$$P(Y) = \sum_{x_1} P(x_1)F_1(x_1)$$

* This is the *pruning* algorithm (Felsenstein, 1981).

* Term after “ \prod ” sign in F_n can be written $E_c(x_p) = P(Y_c|x_p) = \sum_{x_c} P(x_c|x_p)F_c(x_c)$ where p is parent of c

* Can also be viewed as an instance of the *sum-product algorithm* for a factor graph (to be covered later)

• “Pulley principle” for reversible models. Consider tree T_1 again:



– Using $\pi_i M(t)_{ij} = \pi_j M(t)_{ji}$ and $\mathbf{M}(t)\mathbf{M}(t') = \mathbf{M}(t+t')$

$$P(y_2, y_3) = \sum_{x_1} \pi_{x_1} M(t_{12})_{x_1 y_2} M(t_{13})_{x_1 y_3}$$

$$\begin{aligned}
&= \sum_{x_1} \pi_{x_2} M(t_{12})_{y_2 x_1} M(t_{13})_{x_1 y_3} \\
&= \pi_{y_2} M(t_{12} + t_{23})_{y_2 y_3} \\
&= \pi_{y_3} M(t_{12} + t_{23})_{y_3 y_2}
\end{aligned}$$

- Note $P(y_2, y_3)$ depends only on $t_{12} + t_{23}$, so can “slide” root node (like a pulley) without affecting likelihood
- Corollary: can slide all the way to 2 or 3 (or to any leaf, in a tree with more nodes)
- Probability of an alignment conditioned on tree, $P(A|T)$, is product of column probabilities.
 - Denote maximum likelihood tree by $T_{ML} = \operatorname{argmax}_T P(A|T)$

2.4 Elston-Stewart peeling; sum-product algorithm and factor graphs

- Motivation: probability distribution over an ancestral state, $P(x_n|Y)$, or $p \rightarrow c$ branch, $P(x_p, x_c|Y)$
 - Recall $Y = \{y_i\}$ is the set of all leaf states, Y_n contains all y_i descended from node n , and \bar{Y}_n the remaining y_i . (Note $Y_1 = Y$, since node 1 is the root node.)
 - Felsenstein’s pruning algorithm computes $F_n(x_n) = P(Y_n|x_n)$ and thus $P(Y) = \sum_{x_1} \pi(x_1)F_1(x_1)$
 - We will now give recursions for $G_n(x_n) = P(x_n, \bar{Y}_n)$. With these we can easily get posterior probabilities for...
 - * State of ancestral node:

$$P(x_n|Y) = \frac{P(x_n, Y)}{P(Y)} = \frac{1}{P(Y)} P(x_n, \bar{Y}_n) P(Y_n|x_n) = \frac{1}{P(Y)} G_n(x_n) F_n(x_n)$$

- * Joint state of ancestral branch (parent p , child c , sibling s):



$$P(x_p, x_c|Y) = \frac{P(x_p, x_c, Y)}{P(Y)}$$

$$\begin{aligned}
&= \frac{1}{P(Y)} P(x_p, \bar{Y}_p) P(x_c|x_p) P(Y_c|x_c) P(Y_s|x_p) \\
&= \frac{1}{P(Y)} G_p(x_p) P(x_c|x_p) F_c(x_c) E_s(x_p)
\end{aligned}$$

where $E_s(x_p) = \sum_{x_c} P(x_c|x_p) F_s(x_c)$ as before

– Recursion for G_c is

$$\begin{aligned}
G_c(x_c) &= \sum_{x_p} G_p(x_p) P(x_c|x_p) E_s(x_p) \quad \text{if } c > 1 \text{ (not root)} \\
&= P(x_c) \quad \text{if } c = 1 \text{ (root)}
\end{aligned}$$

– aka Elston-Stewart peeling algorithm

- Handling degenerate characters; wildcards *vs* ambiguities; incorporating observation error into the graphical model

– IUPAC “ambiguous” characters:

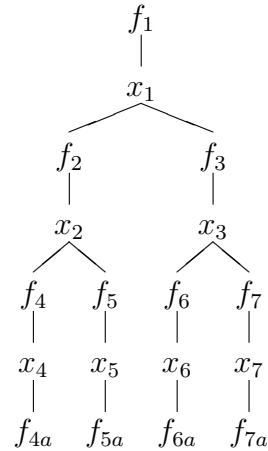
Ambiguous code	Possibilities
R	A G
Y	C T
M	A C
K	G T
S	C G
W	A T
H	A C T
B	C G T
V	A C G
D	A G T
N or X	A C G T

– Can handle this by introducing an “ambiguity model”, $P(y_i|x_i) = A_{x_i y_i}$, for observation y_i given state x_i

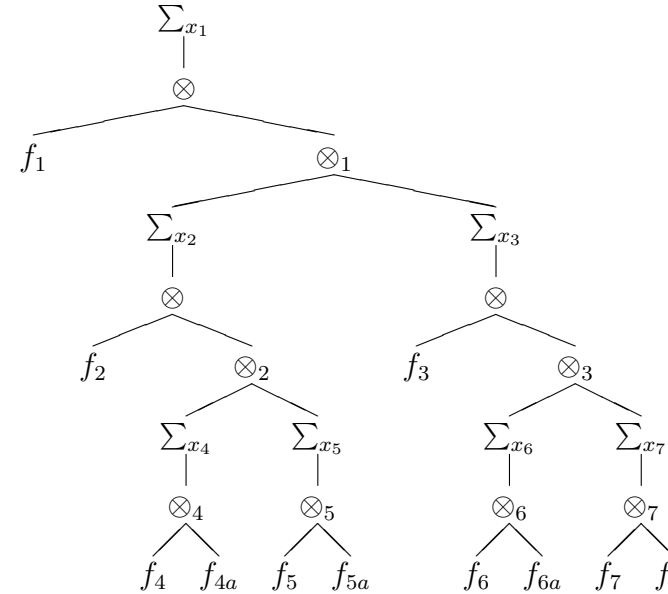
– Formula for $P(X; Y)$ is unchanged, but now factorises into a product of π_i 's, M_{ij} 's and A_{ij} 's. Can express this using a *factor graph*

• Factor graphs and the sum-product algorithm

– T_2 : joint factor graph



and expression tree for marginal $P(Y)$



(Nodes labeled \otimes_n in expression tree correspond to F_n in pruning algorithm.)

$$\begin{aligned}
 f_1(x_1) &= \pi_{x_1} \\
 f_2(x_1, x_2) &= M(t_{12})_{x_1 x_2} \\
 f_3(x_1, x_3) &= M(t_{13})_{x_1 x_3} \\
 f_4(x_2, x_4) &= M(t_{24})_{x_2 x_4} \\
 f_5(x_2, x_5) &= M(t_{25})_{x_2 x_5} \\
 f_6(x_3, x_6) &= M(t_{36})_{x_3 x_6}
 \end{aligned}$$

$$\begin{aligned}
f_7(x_3, x_7) &= M(t_{37})_{x_3x_7} \\
f_{4a}(x_4) &= A_{x_4y_4} \\
f_{5a}(x_5) &= A_{x_5y_5} \\
f_{6a}(x_6) &= A_{x_6y_6} \\
f_{7a}(x_7) &= A_{x_7y_7} \\
P(x_1 \dots x_7; y_4 \dots y_7) &= f_1(x_1)f_2(x_1, x_2)f_3(x_1, x_3)f_4(x_2, x_4)f_5(x_2, x_5)f_6(x_3, x_6)f_7(x_3, x_7)f_{4a}(x_4)f_{5a}(x_5)f_{6a}(x_6)f_{7a}(x_7) \\
&= g(x_1 \dots x_7)
\end{aligned}$$

- We must first introduce the idea of the “summary” or “not-sum” operator, $\sum_{\sim\{\}} f(\cdot)$
 - * The summary of $f(X')$ wrt some set X'' , where $X'' \subseteq X' \subseteq X$, is obtained by summing $f(X')$ over all $x_i \in (X' \cup \bar{X}'')$. The result is a function of X'' .
 - * e.g. $\sum_{\sim\{x_3, x_4\}} f(x_2, x_3, x_4, x_5, x_6) = \sum_{x_2} \sum_{x_5} \sum_{x_6} f(x_2, x_3, x_4, x_5, x_6)$
 - * Also define $\sum_{\sim\{X'\}} f(X') = f(X')$, i.e. the summary of f wrt its own arguments is f itself
 - * If $g(X) = P(X)$ is a joint likelihood, then the summaries are marginal likelihoods: $\sum_{\sim\{X'\}} g(X) = P(X')$
- Factor graphs contain “function nodes” (f_i) and “variable nodes” (x_i)
 - * An edge $f_i - x_j$ indicates x_j is a parameter of f_i
 - * Computation proceeds by “message-passing”. The message from node a to node b is written $\mu_{a \rightarrow b}$
 - * Message $\mu_{f_i \rightarrow x_j}$ is $\sum_{\sim\{x_j\}} f_i \prod_{k \neq j} \mu_{x_k \rightarrow f_i}$ where $\{\mu_{x_k \rightarrow f_i}\}$ are the incoming messages to f_i
 - * Message $\mu_{x_i \rightarrow f_j}$ is $\prod_{k \neq j} \mu_{f_k \rightarrow x_i}$ where $\{\mu_{f_k \rightarrow x_i}\}$ are the incoming messages to x_i
 - * **The message sent from a node n on an edge e is the product of the local function at n (or the unit function, $x_n \rightarrow 1$, if n is a variable node) with all messages received at n on edges *other* than e , summarized for the variable associated with e .**
- For a phylogenetic tree, the messages are as follows ($c - p - s$ denoting child—parent—sibling)
 - * Leaves-to-root: $\mu_{f_{na} \rightarrow x_n} = f_{na}(x_n)$, $\mu_{x_n \rightarrow f_n} = F_n(x_n)$, $\mu_{f_c \rightarrow x_p} = E_c(x_p)$
 - * Root-to-leaves: $\mu_{f_n \rightarrow x_n} = G_n(x_n)$, $\mu_{x_p \rightarrow f_c} = G_p(x_p)E_s(x_p)$

- See Kschichang, Frey & Loeliger, IEEE 47:2, February 2001. “Factor Graphs and the Sum-Product Algorithm”.
- Parsimony as ML/Viterbi history reconstruction
 - The sum-product algorithm works by factorising terms like $\sum_{x_1} \sum_{x_2} f(x_1)f(x_2) = (\sum_{x_1} f(x_1)) (\sum_{x_2} f(x_2))$
 - This in turn relies on the distributive law, $a \times (b + c) = a \times b + a \times c$ and can be applied to any semiring with operators (\otimes, \oplus)
 - For example, consider $a \oplus b = \max(a, b)$ and $a \otimes b = ab$. Then the pruning algorithm calculates the likelihood of the ML imputation of the $\{x_i\}$.
 - Also consider $a \oplus b = \max(a, b)$, $a \otimes b = a + b$ and set $M(t)_{ij} = A_{ij} = 1 - \delta_{ij}$ and $\pi_i = 0$. Then M counts the number of substitutions, and pruning returns the most *parsimonious* imputation of $\{x_i\}$
 - * NB there is a branch-and-bound algorithm to find the most parsimonious tree consistent with observed sequences. However this is not really probabilistic modeling & we won't cover it on this course

2.5 Explicit solution of substitution models

1. Jukes and Cantor, 1969. Evolution of protein molecules. In "Mammalian Protein Metabolism", pp21-132, Academic Press, New York.
 - What is the variance of a Jukes-Cantor distance estimate?
2. Dayhoff, Schwartz and Orcutt, 1978. A Model of Evolutionary Change in Proteins.
3. Kimura M. A simple method for estimating evolutionary rates of base substitutions through comparative studies of nucleotide sequences. J Mol Evol 1980 Dec;16(2):111-20.
 - Illustration that eigenvectors are modes of information loss
4. Felsenstein J. Evolutionary trees from DNA sequences: a maximum likelihood approach. J Mol Evol 1981;17(6):368-76.

5. Hasegawa M, Kishino H, Yano T. Dating of the human-ape splitting by a molecular clock of mitochondrial DNA. *J Mol Evol* 1985;22(2):160-74.
6. Yang Z. Maximum likelihood phylogenetic estimation from DNA sequences with variable rates over sites: approximate methods. *J Mol Evol* 1994 Sep;39(3):306-14. pdf here
7. Yang Z. Estimating the pattern of nucleotide substitution. *J Mol Evol* 1994 Jul;39(1):105-11.
8. Goldman N, Whelan S. A novel use of equilibrium frequencies in models of sequence evolution. *Mol Biol Evol* 2002 Nov;19(11):1821-31.
9. Soyer O, Dimmic MW, Neubig RR, Goldstein RA. Using evolutionary methods to study G-protein coupled receptors. *Pac Symp Biocomput* 2002;:625-36. See also Koshi JM, Goldstein RA. Analyzing site heterogeneity during protein evolution. *Pac Symp Biocomput* 2001;:191-202.

2.6 Simulating trajectories from Markov chains; Gillespie's algorithm

- Simulation by discrete-time approximation
 - Break time interval t into $N = t/\Delta t$ discrete steps; probability matrix for each step is $\mathbf{Q} \simeq \mathbf{I} + \mathbf{R}\Delta t$
 - At each step x_n , sample x_{n+1} using $P(x_{n+1} = j | x_n = i) = Q_{ij}$
 - Drawbacks: accurate only if $\Delta t \ll \max_i(-1/R_{ii})$
- Algorithm for simulating trajectories on a discrete Markov chain
 - Let $r_i = -R_{ii}$ be *exit rate* from state i , and let $s_j^{(i)} = R_{ij}/r_i$ be *jump probability* for $i \rightarrow j$ (with $i \neq j$)
 - Each row of \mathbf{R} is specified by an exit rate r_i and a vector $\mathbf{s}^{(i)}$ of jump probabilities
 - Simulating the process can be conceptualised as follows:
 - * Wait for time t before exiting current state. The pdf of t is $p(t) = r_i \exp(-r_i t) = r_i \times P(\text{no mutation at time } t)$
 - * Randomly pick the next state using weights $\mathbf{s}^{(i)}$

- Sampling the wait time: let v be an r.v. uniformly distributed from 0 to 1 (so pdf is $q(v) = 1$ for $0 \leq v \leq 1$) and let $t = -\frac{1}{r_i} \log[1 - v]$, so that $v = 1 - \exp(-r_i t)$. Then $p(t) = q(v) \frac{dv}{dt} = r_i \exp(-r_i t)$ as required.
 - * More generally, we can use this method to sample from any distribution $p(t)$ whose cumulative distribution $v(t) = \int_0^t p(t') dt'$ can be inverted to give $t = t(v)$
- This algorithm allows us to simulate efficiently while keeping track of precise trajectories, including event times, as well as various summary statistics such as time spent in each state (w_i), number of substitution events (u_{ij}), etc.
- A special case is *Gillespie's algorithm* from computational chemistry (J. Phys. Chem. 1977)
 - Let $x_i(t)$ represent the number of molecules of species i at time t . Suppose that reaction μ has instantaneous rate c_μ and requires $n_i(\mu)$ molecules of species i . The total rate of this reaction is then

$$a_\mu = c_\mu \prod_i^{x_i} C_{n_i(\mu)}$$

- Thus the total reaction rate is $r = \sum_\mu a_\mu$. The time to the next reaction is exponentially distributed with rate r . The probability that the next reaction is of type μ is a_μ/r .
- Assuming all molecular collisions result in a reaction, the individual reaction rates c_μ are determined by the physical properties of the molecules (mass, volume) and the temperature of the system via Boltzmann distributions. (Of course the above assumption is probably false: only some proportion of collisions will result in a reaction.)
 - A couple of ways to generate an ultrametric phylogenetic tree by simulation (NB in an ultrametric tree, all leaves are same distance from root):
 - Yule process (forwards in time). Speciation rate $N\lambda$ where N is number of species. Stop after fixed time.
 - Coalescent process (backwards in time). Coalescence rate $\frac{1}{2}N(N-1)\kappa$.
 - Can also generate non-ultrametric trees, e.g. with a birth-death process.

3 Estimation of model parameters

3.1 Heuristic estimation of phylogenetic trees

- Fast tree construction methods estimate T_{ML} . Not strictly probabilistic modeling, but useful to know.
 - Start by constructing a *distance matrix*: d_{ij} = ML distance from seq i to seq j
 - **UPGMA** (Sokal & Michener, 1958) iteratively joins nearest-neighbor pairs and replaces them with a “common ancestor” node. Creates clock-like trees.
 - * Start by putting each node i into its own cluster C_a . The distance between clusters C_a and C_b is $d_{ab} = \frac{1}{|C_a||C_b|} \sum_{i \in C_a} \sum_{j \in C_b} d_{ij}$
 - * Pick the two clusters i, j for which d_{ij} is minimal. Define a new cluster $C_k = C_i \cup C_j$. Remove C_i and C_j from the list of clusters.
 - * Create new node k with daughters i, j and place at height $d_{ij}/2$. Repeat until only one cluster remains.
 - **Neighbor-joining** (Saitou & Nei, 1987) is like UPGMA but attempts to reconstruct unrooted trees.
 - * If parent of i, j is k then $d_{km} = (d_{im} + d_{jm} - d_{ij})/2$, assuming the d_{ij} are “additive”
 - * Can therefore exactly reconstruct tree, and branch lengths, if we can identify neighbouring nodes
 - * It is not sufficient to choose i, j with minimal d_{ij} because long branches get penalised
 - e.g. in tree ((A:4,B:1):1,(C:1,D:4):1) where $d_{BC} = 4, d_{AB} = 5$
 - * Instead, subtract averaged distances to other nodes: define $D_{ij} = d_{ij} - r_i - r_j$ where $r_i = \frac{1}{L-2} \sum_{k \in L} d_{ik}$ where L is set of leaves
 - * Proceed as with UPGMA: pick $\text{argmin}_{i,j} D_{ij}$, create ancestral k , calculate d_{km} for all m , remove i, j ; repeat.
 - **Weighted neighbor-joining** (Bruno, Succi & Halpern, 2000) takes account of the variances (errors) in the d_{ij}
 - * If i, j are neighbors then the following should hold:
 - Additivity: $d_{ik} - d_{jk}$ is independent of k
 - Positivity: $d_{ik} + d_{jl} - d_{ij} - d_{kl} = 2d_{PQ} \geq 0$, where $P-Q$ is the internal branch separating i, j and any two other nodes k, l

- * Deviations from these ideals are modeled as Gaussian variables (assuming underlying Jukes-Cantor model), yielding a likelihood criterion for joining taxa
- * Implementation in the program `weighor`
- Measures of confidence in a given tree:
 - Tree sampling. Explore the posterior distribution $P(T|A)$ (e.g. by MCMC; see elsewhere in this class).
 - Bootstrapping. From set of N alignment columns, randomly sample N (with replacement).

3.2 Markov Chain Monte Carlo sampling

- A method for generating representative samples from posterior distribution $P(x)$ where x is a high-dimensional variable
 - For example, suppose x is a tree topology. The number of unrooted bifurcating trees over n taxa is

$$(2n - 5)!! = 1 \times 3 \times 5 \times \dots \times (2n - 5) = \frac{(2n - 5)!}{2^{n-2}(n - 2)!}$$

as can be seen by considering that an unrooted binary tree over n taxa has $2n - 3$ branches and therefore $2n - 3$ ways to add an $n + 1$ 'th taxon. (See “Inferring Phylogenies”, Felsenstein, 2004.)

- Alternatively consider the set of possible branch lengths for such a tree, which is $[0, \infty)^{2n-1}$
- MCMC generates a series of samples $X = \{x_i\}$ such that $P(x \in X) \propto P(x)$
- Applications: Monte Carlo integration: $\int f(x)P(x)dx \simeq \frac{1}{|X|} \sum_{x \in X} f(x)$; model comparison; decision theory; inference; any other situation where full posterior distribution can be approximated by finite sample
- General idea: construct a Markov process (discrete-time, homogeneous, stationary) with transition matrix $Q^*(x, x') = P(x_{n+1} = x' | x_n = x)$, whose equilibrium distribution is $P(x)$ (so $\sum_{x'} P(x')Q^*(x', x) = P(x)$).
- Each time-step, $x_n \rightarrow x_{n+1}$, is called a *move*

- It's common (but **not** compulsory) to make the chain reversible, i.e. satisfying detailed balance:

$$P(x)Q^*(x, x') = P(x')Q^*(x', x)$$

See e.g. “Improving Asymptotic Variance of MCMC Estimators: Non-reversible Chains are Better” (Neal, 2004) for examples of irreversible MCMC.

- Desiderata:

- given x , should be easy to sample x' from $P(x'|x) = Q^*(x, x')$;
- Q should be irreducible (i.e. paths exist between any two states)
 - * Issues: correlation time (and thus “burn-in time” from initial x); good “mixing”

- Metropolis-Hastings algorithm: start with a proposal density, $\mathcal{Q}(x, x')$. Given $x^{(t)}$, sample x' from $P(x'|x^{(t)}) = \mathcal{Q}(x^{(t)}, x')$. The proposed move to x' is accepted with probability $h(x^{(t)}, x')$, where

$$h(x, x') = \min\left(\frac{\mathcal{Q}(x', x)P(x')}{\mathcal{Q}(x, x')P(x)}, 1\right)$$

is the *Hastings ratio*. If move is accepted, then $x^{(t+1)} = x'$; if move is rejected, then $x^{(t+1)} = x^{(t)}$.

- Effective move rate is $Q^*(x, x') = \mathcal{Q}(x, x')h(x, x')$ (for $x \neq x'$) which satisfies detailed balance
- If proposal distribution is good, Hastings ratios will be (reasonably) close to 1
- $P(x)$ is only used via ratio $P(x')/P(x)$, thus sufficient to compute $P(x)$ to within constant factor
- If \mathcal{Q} is symmetrical ($\mathcal{Q}(x, x') = \mathcal{Q}(x', x)$), then h depends only on $P(x')/P(x)$
- Application of Metropolis-Hastings to Monte Carlo integration is called *Importance Sampling*
- Invented by Metropolis, Teller and Rosenbluth, 1953, supposedly at a Los Alamos dinner party
- Special case: *Gibbs sampling* for multivariate x . Useful when conditional distributions for components of x are known.

- Suppose \mathbf{x} is multidimensional and can be written $\mathbf{x} = (x_1, x_2 \dots x_D)$.
- Proposal distribution is obtained as follows: pick a random dimension d (or cycle through the dimensions). Then

$$Q(\mathbf{x}, \mathbf{x}') = \begin{cases} P(x'_d | x_1 \dots x_{d-1}, x_{d+1} \dots x_D) & \text{if } x_i = x'_i \text{ for all } i \neq d \\ 0 & \text{otherwise} \end{cases}$$

i.e. sample x_d from its conditional posterior distribution, given all the other x_i .

- Hastings ratio is always 1, so moves always accepted.

- Example: stochastic k -means algorithm; stochastic EM

- Holmes-Bruno joint sequence-expression model

- Example: MCMC sampling of phylogenetic trees (MrBayes, Handel)

- Moves include branch-sliding; branch-length rescaling (either one branch, or all branches); branch-swapping; resampling of rate parameters
- The rescaling moves are tricky. Suppose that we generate x' by sampling some parameter u from a p.d.f. $g(u)$, so $x' = x'(x, u)$. Let $u' = u'(x, u)$ be the "inverse" parameter that takes us back from $x' \rightarrow x$. According to Green (1995), the Hastings ratio then depends on the Jacobian, as follows

$$\frac{Q(x', x)}{Q(x, x')} = \frac{g(u')}{g(u)} \left| \frac{\delta(x', u')}{\delta(x, u)} \right|$$

For branch-rescaling moves, we have $x' = ux$ and $u' = 1/u$ for some randomly-sampled multiplier u . The Jacobian is

$$\begin{aligned} \left| \frac{\delta(x', u')}{\delta(x, u)} \right| &= \left| \begin{array}{cc} \frac{\delta x'}{\delta x} & \frac{\delta x'}{\delta u} \\ \frac{\delta u'}{\delta x} & \frac{\delta u'}{\delta u} \end{array} \right| \\ &= \left| \begin{array}{cc} u & x \\ 0 & -u^{-2} \end{array} \right| \\ &= u^{-1} \end{aligned}$$

- Example: ungapped sequence alignments (Lawrence *et al*)
 - Suppose you have K sequences, $\{S^{(1)} \dots S^{(K)}\}$, and that $S_i^{(k)}$ is the i 'th residue of the k 'th sequence.
 - Let x_k be the indentation of the k 'th sequence. The motif (of length M) runs from x_k to $x_k + M - 1$.
 - Let \mathcal{U} denote the null hypothesis that the sequences are unrelated. Then

$$\frac{P(\mathbf{x})}{P(\mathcal{U})} = \prod_{m=0}^{M-1} \frac{f(\mathbf{n}(m))}{\prod_{\omega} q(\omega)^{n_{\omega}(m)}}$$

where $q(\omega)$ is a background distribution over alphabet symbols ω , and $n_{\omega}(m)$ is the number of times symbol ω appears in column m of the motif

$$n_{\omega}(m) = \sum_{k=1}^K \delta(S_{x_k+m} = \omega)$$

The $f(\mathbf{n})$ function should reward conserved columns. Lawrence *et al* use an entropy-like measure

$$f(\mathbf{n}) = \prod_{\omega} p_{\omega}^{n_{\omega}} = \exp(-KS[p])$$

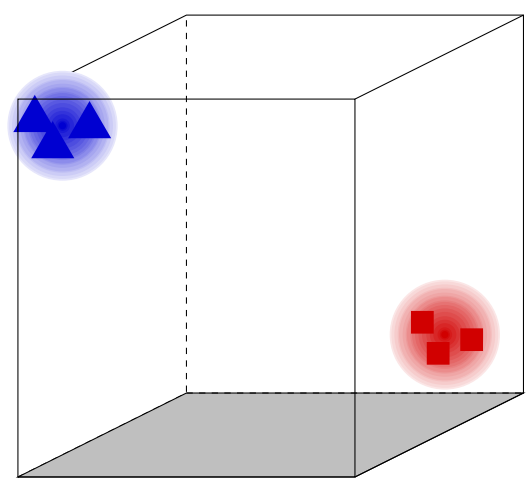
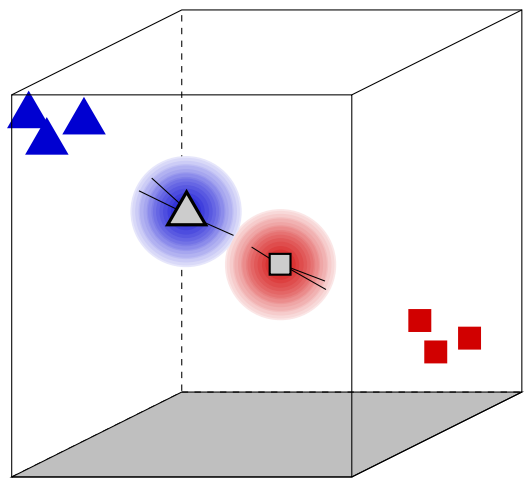
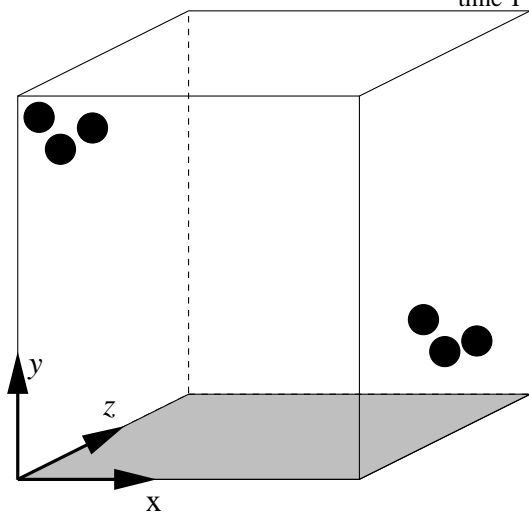
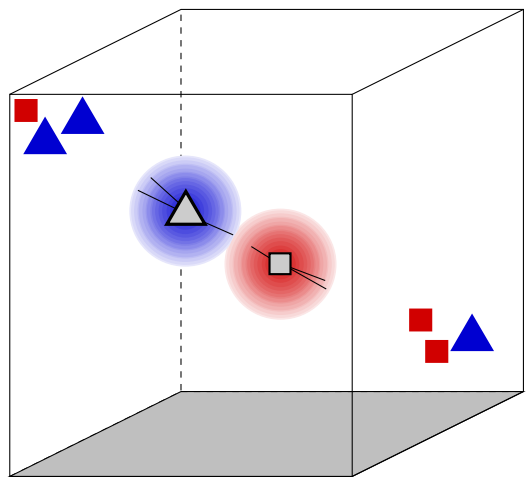
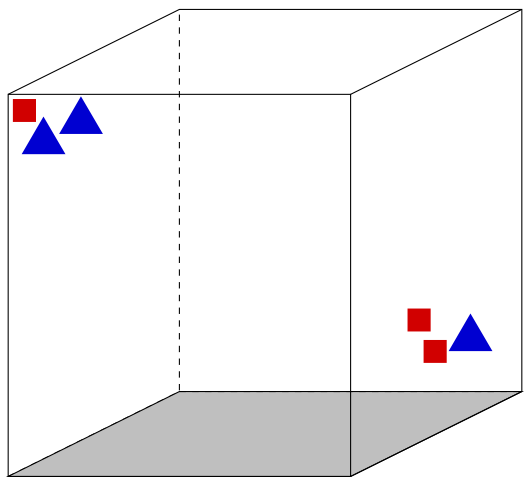
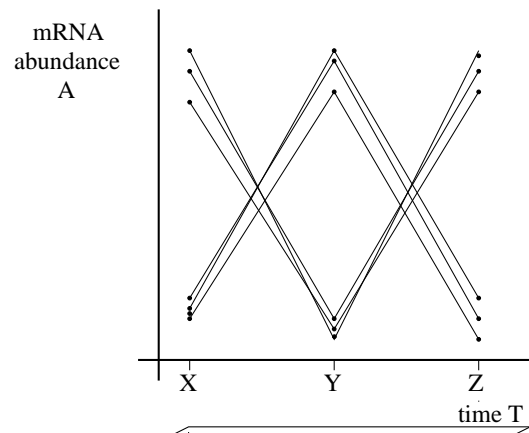
where $p_k = n_k/K$. Note that this is the distribution \mathbf{p} that maximizes $\prod_{\omega} p_{\omega}^{n_{\omega}}$ for a given \mathbf{n} . Due to this implicit maximization, the above $f(\mathbf{n})$ is not strictly a probability distribution for \mathbf{n} .

- Typically a vector of pseudocounts \mathbf{a} is added to \mathbf{n} . This suggests the Dirichlet evidence as an alternative form for f that does not involve any implicit maximization:

$$f(\mathbf{n}) = \frac{\mathcal{B}(\mathbf{n} + \mathbf{a})}{\mathcal{B}(\mathbf{a})\mathcal{B}(\mathbf{n})}$$

- Another way to formulate this problem is as a chain over $(\mathbf{x}, \{\mathbf{p}(m)\})$ with Gibbs-sampling steps that alternate between resampling one of the x_k 's and resampling all the \mathbf{p} 's.
- For good mixing it's also useful to allow moves that slide the entire motif window, i.e. $\mathbf{x} \leftarrow \mathbf{x} \pm \mathbf{1}$

- Sometimes you want to sample over the motif width M as well.
 - * Consider however the following subtlety that arises if you keep track of the position-specific probabilities as part of the chain (instead of using the Dirichlet-evidence approach).
 - * If the chain state is $(\mathbf{x}, M, \mathbf{p}(1) \dots \mathbf{p}(M))$ and you move from $M \rightarrow M + 1$, then you need a new vector of probabilities $\mathbf{p}(M + 1)$ and so the dimensionality of your state space has changed, which is a Bad Thing for reversibility.
 - * The idea of *reversible jump MCMC* is to set a ceiling for M (let's say M_{\max}) and to let the state be $(\mathbf{x}, M, \mathbf{p}(1) \dots \mathbf{p}(M_{\max}))$.
 - * For a particular $M < M_{\max}$, the parameters $\mathbf{p}(M + 1) \dots \mathbf{p}(M_{\max})$ are unused “dummy” parameters that are simply there to pad out your “real” parameters, so that the dimensionality of your state space is fixed.
- Example: multiple alignments with evolutionary HMM
- Example: trajectory through continuous-time Markov chain
 - Arbitrary rate grammars; concatenation equations
- Example: random walk in potential surface $V(\mathbf{x})$ over \mathfrak{R}^D .
 - Let $P(X) = \exp(-\beta V(\mathbf{x}))/Z$ where $\beta = 1/kT$ and $Z = \int \exp(-\beta V(\mathbf{x})) dx$.
 - Proposal distribution for \mathbf{x}' is a Gaussian centered on \mathbf{x} . Thus $Q(\mathbf{x}, \mathbf{x}') = Q(\mathbf{x}', \mathbf{x})$
 - Hastings ratio is $h(\mathbf{x}, \mathbf{x}') = \min(\exp(-\beta(V(\mathbf{x}') - V(\mathbf{x}))), 1)$.
- Example: lattice models for (i) polymer melts, (ii) protein folding, (iii) RNA folding
- Example: molecular dynamics simulations (Langevin force)
- Example: genealogical histories using the coalescent
- Example: genome rearrangement models



3.3 Expectation Maximization (EM)

- Maximum likelihood (ML) point estimates: inevitable at some point (can't put priors on priors on priors forever)
 - General idea is to find $\operatorname{argmax}_{\theta} P(x|\theta)P(\theta)$ and pretend that this is the “true” value of θ
- Useful algorithm for ML is EM = Expectation Maximization (Dempster, Laird and Rubin, Journal of the Royal Statistical Society, 1977)
 - “Greedy ascent” algorithm for estimating $\theta_{ML} = \operatorname{argmax}_{\theta} P(y, \theta)$ where $P(y, \theta) = \sum_x P(x, y|\theta)$
 - Here x is missing data and y is observed data. General idea is to alternate between estimating x and θ .
 - Rather than fixing x , insight of EM is to estimate a probability distribution $P(x)$ and use this to estimate θ
- Example of iterative re-estimation: K -means algorithm.
 - N points $\{\mathbf{y}^{(i)}\}$ in D dimensions; K clusters; point i has cluster label $x^{(i)}$
 - Start by randomising all $x^{(i)}$
 - Re-estimate cluster centroids; set each $x^{(i)}$ to closest cluster; iterate to convergence
 - K -means is commonly used for clustering, e.g. (in bioinformatics) microarray data
 - We will see that it's very similar to EM on a mixture-of-Gaussians model
- Derivations of EM
 - The following argument follows Anders Krogh's in Durbin *et al*, chapter 11.6
 - * Suppose we have some estimate $\theta^{(n)}$ and we want to choose $\theta^{(n+1)}$ such that $P(y|\theta^{(n+1)}) \geq P(y|\theta^{(n)})$
 - * Since $P(x, y|\theta) = P(y|\theta)P(x|y, \theta)$ we can write $\log P(y|\theta) = \log P(x, y|\theta) - \log P(x|y, \theta)$ for some θ
 - * Multiplying by $P(x|y, \theta^{(n)})$ and summing over x gives

$$\log P(y|\theta) = \sum_x P(x|y, \theta^{(n)}) \log P(x, y|\theta) - \sum_x P(x|y, \theta^{(n)}) \log P(x|y, \theta)$$

* Let first term on RHS be $\mathcal{E}(\theta|\theta^{(n)}) = \sum_x P(x|y, \theta^{(n)}) \log P(x, y|\theta)$. Then

$$\log P(y|\theta) = \mathcal{E}(\theta|\theta^{(n)}) - \sum_x P(x|y, \theta^{(n)}) \log P(x|y, \theta)$$

Likewise

$$\log P(y|\theta^{(n)}) = \mathcal{E}(\theta^{(n)}|\theta^{(n)}) - \sum_x P(x|y, \theta^{(n)}) \log P(x|y, \theta^{(n)})$$

* Subtracting gives

$$\begin{aligned} \log P(y|\theta) - \log P(y|\theta^{(n)}) &= \mathcal{E}(\theta|\theta^{(n)}) - \mathcal{E}(\theta^{(n)}|\theta^{(n)}) + \sum_x P(x|y, \theta^{(n)}) \log \frac{P(x|y, \theta^{(n)})}{P(x|y, \theta)} \\ &= \mathcal{E}(\theta|\theta^{(n)}) - \mathcal{E}(\theta^{(n)}|\theta^{(n)}) + D(P(x|y, \theta^{(n)}) || P(x|y, \theta)) \end{aligned}$$

* Since last term on RHS is always ≥ 0 , we have $P(y|\theta^{(n+1)}) \geq P(y|\theta^{(n)})$ as long as

$$\theta^{(n+1)} = \operatorname{argmax}_{\theta} \mathcal{E}(\theta|\theta^{(n)})$$

* If $\theta^{(n+1)} = \theta$, then a maximum has been reached and so $P(y|\theta^{(n+1)}) = P(y|\theta^{(n)})$

* Note that $\mathcal{E}(\theta|\theta^{(n)})$ is the **expected joint log-likelihood of the missing data x and observed data y as a function of θ** , with the expectation taken over the posterior distribution of x as estimated using $\theta^{(n)}$

* Also note if $P(x, y|\theta)$ has the form $\prod_i \theta_i^{x_i}$, where x_i is the number of times an event occurs and θ_i is the probability of that event, then $\mathcal{E}(\theta|\theta^{(n)})$ has the form $\sum_i \langle x_i \rangle_{P(x|y, \theta^{(n)})} \log \theta_i$

- In other words, computing the expected log-likelihood \mathcal{E} typically involves computing **expected counts** $\langle x_i \rangle$ for the missing data

- Note also that, in this case, EM is making use of the first derivatives:

$$\begin{aligned} \frac{\partial(\log P(y|\theta))}{\partial(\log \theta_i)} &= \frac{\theta_i}{P(y|\theta)} \frac{\partial P(y|\theta)}{\partial \theta_i} \\ &= \frac{\theta_i}{P(y|\theta)} \frac{\partial}{\partial \theta_i} \sum_x P(x, y|\theta) \end{aligned}$$

$$\begin{aligned}
&= \frac{\theta_i}{P(y|\theta)} \sum_x \left(\frac{x_i}{\theta_i}\right) P(x, y|\theta) \\
&= \frac{\theta_i}{P(y|\theta)} \sum_x \left(\frac{x_i}{\theta_i}\right) P(y|\theta) P(x|y, \theta) \\
&= \langle x_i \rangle_{P(x|y, \theta)}
\end{aligned}$$

- * Likewise if $P(x, y|\theta)$ contains terms of the form $\exp(-\theta_i x_i)$, where θ_i is an event rate and x_i is the time that elapses before the event occurs, the corresponding terms in \mathcal{E} will be $-\theta_i \langle x_i \rangle$ involving the **expected times** $\langle x_i \rangle$
- * Computing $P(x|y, \theta^{(n)})$ is called the **E-step** of EM. Computing $\theta^{(n+1)}$ is the **M-step**.
- Neal and Hinton’s variational view of EM (Learning in Graphical Models, M.Jordan, 1998)
 - * Let $\tilde{P}(x)$ be a probability distribution over the missing data and let $H(\tilde{P})$ be the entropy of \tilde{P} . Define

$$\begin{aligned}
F(\theta, \tilde{P}) &= \langle \log P(x, y|\theta) \rangle_{\tilde{P}} + H(\tilde{P}) \\
&= \sum_x \tilde{P}(x) (\log P(x, y|\theta) - \log \tilde{P}(x)) \\
&= \sum_x \tilde{P}(x) (\log P(y|\theta) + \log P(x|y, \theta) - \log \tilde{P}(x)) \\
&= \log P(y|\theta) - D(\tilde{P}(x) || P(x|y, \theta))
\end{aligned}$$

where $D(\tilde{P}(x) || P(x, y, \theta))$ is the Kullback-Leibler divergence.

- * Suppose we fix $\theta = \theta^{(n)}$ and maximise $F(\theta^{(n)}, \tilde{P})$ w.r.t. \tilde{P} . Then the latter expression for F shows that the maximum is at $\tilde{P}(x) = P(x|y, \theta^{(n)})$ (due to Gibbs’ inequality). This is the *E*-step of EM.
- * If we then fix \tilde{P} at this value, we have $F(\theta, \tilde{P}) = F(\theta, P(x|y, \theta^{(n)})) = \mathcal{E}(\theta|\theta^{(n)}) + H(\tilde{P})$. Maximising this w.r.t. θ is the *M*-step of EM.
- * Thus EM can be viewed as a two-step maximization of F . If $[-\log P(x, y|\theta)]$ is analogous to the “energy” of state x , then $[-F]$ is like a “free energy” (energy minus entropy).

3.4 EM for Gaussian mixture model

- Mixture-of-Gaussians example

- Again, suppose we have N datapoints $\{y_i\}$ (restricted to one dimension for simplicity)
- Probabilistic model: mixture of K Gaussian components; component k has mean m_k and variance v_k . Parameters $\theta = \{m_k, v_k\}$. Each component has equal probability $1/K$.
- If component label of point i is x_i , then joint likelihood for point i is

$$P(x_i, y_i | \theta) = \frac{1}{K} (2\pi v_{x_i})^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(y_i - m_{x_i})^2 / v_{x_i}\right)$$

- Marginal likelihood for observed data is

$$P(y_i | \theta) = \sum_{x_i} P(x_i, y_i | \theta)$$

- Joint likelihood for all observed data and missing component labels is

$$P(x, y | \theta) = \prod_{i=1}^N P(x_i, y_i | \theta)$$

- Posterior probability of i 'th component label (the E-step) is

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

- Expected log-likelihood, and partial derivatives thereof: let $W_i(x_i) = P(x_i | y_i, \theta^{(n)})$. Then

$$\begin{aligned} \mathcal{E}(\theta | \theta^{(n)}) &= \langle \log P(x, y | \theta) \rangle_{P(x|y, \theta^{(n)})} \\ &= \sum_{i=1}^N \langle \log P(x_i, y_i | \theta) \rangle_{P(x_i|y_i, \theta^{(n)})} \end{aligned}$$

$$\begin{aligned}
&= -N \log K - \frac{N}{2} \log(2\pi) - \frac{1}{2} \sum_{i=1}^N \sum_{k=1}^K W_i(k) \left(\log(v_k) + \frac{(y_i - m_k)^2}{v_k} \right) \\
\frac{\partial \mathcal{E}}{\partial m_k} &= \sum_{i=1}^N W_i(k) \frac{y_i - m_k}{v_k} \\
\frac{\partial \mathcal{E}}{\partial v_k} &= -\frac{1}{2} \sum_{i=1}^N W_i(k) \left(\frac{1}{v_k} - \frac{(y_i - m_k)^2}{v_k^2} \right)
\end{aligned}$$

– Setting the partial derivatives to zero gives

$$\begin{aligned}
m_k &= \frac{\sum_i W_i(k) y_i}{\sum_i W_i(k)} \\
v_k &= \frac{\sum_i W_i(k) (y_i - m_k^2)}{\sum_i W_i(k)} \\
&= \frac{\sum_i W_i(k) y_i^2}{\sum_i W_i(k)} - \left(\frac{\sum_i W_i(k) y_i}{\sum_i W_i(k)} \right)^2
\end{aligned}$$

which are immediately recognisable as the mean and variance of the y_i , weighted by $W_i(k)$.

- The K -means algorithm is equivalent to fixing the v_k and taking the limit $v_k \rightarrow \infty \forall k$ so that $W_i(k) \rightarrow 1$ for the most probable cluster and 0 for all other clusters. Neal and Hinton refer to this as a “winner-take-all” variant of EM.
- Note that this EM algorithm can get stuck in an infinite-likelihood “trap” if a single point gets assigned to a cluster and $v_k \rightarrow 0$. This can be fixed by putting a prior distribution on the parameters (m_k, v_k) .

- Algorithms that make use of second derivatives (indirectly or directly): conjugate gradient, quasi-Newton, Gauss-Newton, Newton-Raphson...

3.5 EM for substitution models

- Review approach of Dayhoff *et al.*: (roughly) take a pairwise alignment of two closely related sequences, count the number of instances C_{ij} of each aligned residue-pair (i, j) , estimate the evolutionary distance Δt separating the two sequences,

and then set $R_{ij} \leftarrow C_{ij}/\Delta t$.

- Drawback: ignores multiple substitutions. We seek a maximum likelihood version, with the likelihood implicitly taking multiple substitutions into account.
 - We will see that this amounts (at least for the discrete-time approximation) to getting an “unbiased” estimate of \mathbf{C} . These correspond to the *expected* number of times that each $i \rightarrow j$ transition occurred.
 - Our unbiased estimate of \mathbf{C} depends on our current estimate of the rate matrix: if we think that the R_{ij} are small, there will be few multiple substitutions, but if the R_{ij} are large, there will be many. Thus the two things that we are trying to estimate are inter-related, but that’s how EM works: we fix one and estimate the other, then do it the other way round, then iterate to convergence.
 - We start with a discrete-time approximation (breaking the time interval into small, finite steps). We then consider the limit where the time-steps get infinitely small. In this continuous-time limit, there are an infinite number of $i \rightarrow i$ transitions. It then makes more sense to consider the amount of *time* spent in state i .
- Take the pairwise case first. The derivation is laborious but the result we’re aiming for is that we can get our expected counts \mathbf{C} by conditioning on, then summing over, all possible times at which a substitution can occur.
 - Use discrete-time approximation: break T into discrete timesteps of time Δt with the discrete-time transition matrix $\mathbf{Q} = \mathbf{I} + \mathbf{R}\Delta t$ (later we’ll take the limit $\Delta t \rightarrow 0$). Let x_n be the state at time $t = n\Delta t$. Let the probability distribution over x_0 be π_{x_0} . Suppose that $x_0 = a$ and $x_N = b$ are observed (where $N = T/\Delta t$), while states $x_1 \dots x_{N-1}$ are missing data. Thus

$$\begin{aligned}
 P(x_0 \dots x_N | \theta) &= \pi_{x_0} \prod_{n=0}^{N-1} Q_{x_n x_{n+1}} \\
 P(x_0, x_N | \theta) &= \pi_{x_0} [\mathbf{Q}^N]_{x_0 x_N} \\
 P(x_1 \dots x_{N-1} | x_0, x_N, \theta) &= \frac{P(x_0 \dots x_N | \theta)}{P(x_0, x_N | \theta)}
 \end{aligned}$$

- Let $\theta' = (\pi', \mathbf{Q}')$ be the old parameters and $\theta = (\pi, \mathbf{Q})$ be the new parameters. The EM function $\mathcal{E}(\theta|\theta')$ is

$$\mathcal{E}(\theta|\theta') = \log \pi_{x_0} + \sum_i \sum_j C_{ij} \log Q_{ij}$$

where C_{ij} is the **expected number of times that the transition $i \rightarrow j$ occurred**. This can be seen immediately from the fact that the joint likelihood for observed & missing data can be written in the form

$$P(x_0 \dots x_N | \theta) = \pi_{x_0} \prod_i \prod_j Q_{ij}^{\xi_{ij}}$$

where ξ_{ij} counts the usage of transition $i \rightarrow j$. We have already seen in the general case that the above form for \mathcal{E} holds with $C_{ij} = \langle W_{ij} \rangle$.

- Here is a longer derivation, showing exactly how the C_{ij} are computed

$$\begin{aligned} \mathcal{E}(\theta|\theta') &= \sum_{x_1} \sum_{x_2} \dots \sum_{x_{N-1}} P(x_1 \dots x_{N-1} | x_0, x_N, \theta') \log P(x_0 \dots x_N | \theta) \\ &= \sum_{x_1} \sum_{x_2} \dots \sum_{x_{N-1}} P(x_1 \dots x_{N-1} | x_0, x_N, \theta') \log \left(\pi_{x_0} \prod_{n=0}^{N-1} Q_{x_n x_{n+1}} \right) \\ &= \sum_{x_1} \sum_{x_2} \dots \sum_{x_{N-1}} P(x_1 \dots x_{N-1} | x_0, x_N, \theta') \left(\log \pi_{x_0} + \sum_{n=0}^{N-1} \log Q_{x_n x_{n+1}} \right) \\ &= \log \pi_{x_0} + \sum_{n=0}^{N-1} \sum_{x_n} \sum_{x_{n+1}} \left[\sum_{\{x_k: 1 \leq k < n, n+1 < k \leq N\}} P(x_1 \dots x_{N-1} | x_0, x_N, \theta') \right] \log Q_{x_n x_{n+1}} \\ &= \log \pi_{x_0} + \sum_{n=0}^{N-1} \sum_{x_n} \sum_{x_{n+1}} \left[\sum_{\sim \{x_n, x_{n+1}\}} P(x_1 \dots x_{N-1} | x_0, x_N, \theta') \right] \log Q_{x_n x_{n+1}} \\ &= \log \pi_{x_0} + \sum_{n=0}^{N-1} \sum_{x_n} \sum_{x_{n+1}} P(x_n, x_{n+1} | x_0, x_N, \theta') \log Q_{x_n x_{n+1}} \end{aligned}$$

$$\begin{aligned}
&= \log \pi_{x_0} + \sum_i \sum_j \left[\sum_{n=0}^{N-1} P(x_n = i, x_{n+1} = j | x_0, x_N, \theta') \right] \log Q_{ij} \\
&= \log \pi_{x_0} + \sum_i \sum_j C_{ij} \log Q_{ij}
\end{aligned}$$

Clearly C_{ij} is the expected number of transitions $i \rightarrow j$ that occurred among the missing data. Recalling that the start and end states are $x_0 = a$ and $x_N = b$, and observing that the expectation C_{ij} is additive over all timepoints n at which the transition could occur, we have

$$\begin{aligned}
C_{ij} &= \sum_{n=0}^{N-1} P(x_n = i, x_{n+1} = j | x_0 = a, x_N = b, \theta') \\
&= \sum_{n=0}^{N-1} \frac{P(x_n = i | x_0 = a, \theta') P(x_{n+1} = j | x_n = i, \theta') P(x_N = b | x_{n+1} = j, \theta')}{P(x_N = b | x_0 = a, \theta')} \\
&= \sum_{n=0}^{N-1} \frac{[\mathbf{Q}^n]_{ai} Q_{ij} [\mathbf{Q}^{N-n-1}]_{jb}}{[\mathbf{Q}^N]_{ab}}
\end{aligned}$$

Eigenvector decomposition $\mathbf{Q} = \mathbf{U}\mathbf{D}\mathbf{U}^{-1}$, where \mathbf{D} is diagonal (with $D_{kk} = \lambda_k$), gives us $\mathbf{Q}^n = \mathbf{U}\mathbf{D}^n\mathbf{U}^{-1}$ and so

$$\begin{aligned}
C_{ij} &= \frac{Q_{ij}}{[\mathbf{U}\mathbf{D}^N\mathbf{U}^{-1}]_{ab}} \sum_{n=0}^{N-1} [\mathbf{U}\mathbf{D}^n\mathbf{U}^{-1}]_{ai} [\mathbf{U}\mathbf{D}^{N-n-1}\mathbf{U}^{-1}]_{jb} \\
&= \frac{Q_{ij}}{\sum_m U_{am} \lambda_m^N U_{mb}^{-1}} \sum_{n=0}^{N-1} \left(\sum_k U_{ak} \lambda_k^n U_{ki}^{-1} \right) \left(\sum_l U_{jl} \lambda_l^{N-n-1} U_{lb}^{-1} \right) \\
&= \frac{Q_{ij}}{\sum_m U_{am} \lambda_m^N U_{mb}^{-1}} \sum_k U_{ak} U_{ki}^{-1} \sum_l U_{jl} U_{lb}^{-1} \sum_{n=0}^{N-1} \lambda_k^n \lambda_l^{N-n-1} \\
&= \frac{Q_{ij}}{\sum_m U_{am} \lambda_m^N U_{mb}^{-1}} \sum_k U_{ak} U_{ki}^{-1} \sum_l U_{jl} U_{lb}^{-1} \Lambda_{kl}
\end{aligned}$$

where

$$\Lambda_{kl} = \begin{cases} \frac{\lambda_l^N - \lambda_k^N}{\lambda_l - \lambda_k} & \text{if } \lambda_k \neq \lambda_l \\ N\lambda_l^{N-1} & \text{if } \lambda_k = \lambda_l \end{cases}$$

To obtain Λ_{kl} we have used the identity $\sum_{k=0}^{N-1} a^k = (a^N - 1)/(a - 1)$.

- Hopefully it should also be clear that if we have two sequences X, Y of length L then the expected counts matrix \mathbf{C} can be obtained by summing over all sites X_l, Y_l , so

$$C_{ij} = \sum_{l=1}^L C_{ij}^{[X_l \xrightarrow{N} Y_l]}$$

where $C_{ij}^{[a \xrightarrow{N} b]}$ is the C_{ij} derived above for a process beginning in state $x_0 = a$ and ending in $x_N = b$.

- To summarise: the EM algorithm for parameterising a discrete-time Markov chain from a pairwise alignment is
 1. Start with some initial estimate of the probability matrix \mathbf{Q} .
 2. Estimate the transition counts C_{ij} by summing over all sites X_l, Y_l as above.
 3. Update $Q_{ij} \leftarrow C_{ij} / \sum_k C_{ik}$ (that this maximizes \mathcal{E} can be seen using Lagrange multipliers)
 4. Repeat until the algorithm converges on a fixed matrix \mathbf{Q} (i.e. until the likelihood $P(Y|X, \mathbf{Q})$ stabilises).
- Continuous limit: as $\Delta t \rightarrow 0$, so $N \rightarrow \infty$. The upshot of this is that C_{ij} converges and is meaningful for $i \neq j$, but $C_{ii} \sim 1/N \rightarrow 0$ because during most of the short time intervals, the chain stays in the same state.
 - More detailed argument: the N -dependent terms in the expression for C_{ij} are $Q_{ij}\Lambda_{kl}/\mathbf{Q}_{ab}^N$ with $Q_{ij} = \delta_{ij} + R_{ij}/N$. The factors of λ^N approximately cancel in Λ_{kl} and \mathbf{Q}_{ab}^N , and the factor of N in Λ_{kk} cancels the $1/N$ in Q_{ij} as long as $i \neq j$. However, when $i = j$, there is an unaccounted factor of N from Λ_{kk} (indeed from all Λ_{kl} where $\lambda_k = \lambda_l$).
 - To get round this, we can define $W_i = C_{ii}T/N = C_{ii}\Delta t$ to be the expected *time* spent waiting in state i . This then converges to a meaningful, finite value as $\Delta t \rightarrow 0$.

– In fact, recalling that $\mathbf{M}(t) = \exp(\mathbf{R}t)$, we have by analogy to the discrete case

$$\begin{aligned}
C_{ij} &= \frac{1}{M(T)_{ab}} \int_0^T M(t)_{ai} (R_{ij} dt) M(T-t)_{jb} \\
&= \frac{R_{ij}}{\sum_m U_{am} \exp(\lambda_m T) U_{mb}^{-1}} \sum_k U_{ak} U_{ki}^{-1} \sum_l U_{jl} U_{lb}^{-1} \mathcal{J}_{kl}(T) \\
W_i &= \frac{1}{M(T)_{ab}} \int_0^T M(t)_{ai} (dt) M(T-t)_{jb} \\
&= \frac{1}{\sum_m U_{am} \exp(\lambda_m T) U_{mb}^{-1}} \sum_k U_{ak} U_{ki}^{-1} \sum_l U_{jl} U_{lb}^{-1} \mathcal{J}_{kl}(T)
\end{aligned}$$

where

$$\mathcal{J}_{kl}(T) = \int_0^T \exp(\lambda_k t) \exp(\lambda_l (T-t)) dt = \begin{cases} \frac{\exp(\lambda_l T) - \exp(\lambda_k T)}{\lambda_l - \lambda_k} & \text{if } \lambda_k \neq \lambda_l \\ T \exp(\lambda_k T) & \text{if } \lambda_k = \lambda_l \end{cases}$$

- (Note that the estimates for C_{ij} and W_i can be tested by simulation.)
- (Note also that the λ_k in the above expression are the eigenvalues for \mathbf{R} , not \mathbf{Q} . Since $\mathbf{Q} = \mathbf{I} + \mathbf{R}$, they are related by $\lambda_k^{(\mathbf{Q})} = 1 + \lambda_k^{(\mathbf{R})}$. The eigenvectors are the same.)
- To estimate R_{ij} , note that $R_{ij} = \lim_{\Delta t \rightarrow 0} \frac{Q_{ij}}{\Delta t} = \lim_{\Delta t \rightarrow 0} \frac{C_{ij}}{\sum_k C_{ik} \Delta t} = \frac{C_{ij}}{W_i}$ since $\lim_{\Delta t \rightarrow 0} C_{ik} \Delta t = \delta_{ik} W_i$.
- To summarise, the EM algorithm for *continuous-time* Markov chains and pairwise alignments is as follows:
 1. Start with some initial estimate of the probability matrix \mathbf{R} .
 2. Estimate the transition counts C_{ij} by summing over all sites X_l, Y_l .
 3. Update $R_{ij} \leftarrow C_{ij}/W_i$.
 4. Repeat until the algorithm converges on a fixed matrix \mathbf{R} (i.e. until the likelihood stabilises).
- Strictly speaking, the expected likelihood \mathcal{E} probably needs to be recast as an expected likelihood *density* w.r.t. the W_i (which are continuous variables) if this EM algorithm is to be made rigorous, but life is too short.

- EM for a multiple alignment (with a known phylogenetic tree): sum over all alignment columns, all tree branches, and all possible states $a \rightarrow b$ of each branch. Use peeling algorithm to find posterior probabilities of each $a \rightarrow b$ state.
 - Can accumulate counts in eigenvector space to save time.
 - Detailed derivation: recall that, for a parent-node-sibling triplet (p, n, s) :

$$P(x_p = a, x_n = b|Y) = \frac{G_p(a)M(t_{pn})_{ab}F_n(b)E_s(a)}{P(Y)}$$

where Y represents the observed states at the tree leaves, and $\{F_n, G_p, E_s\}$ are the pruning and peeling likelihoods corresponding to messages on the factor graph.

Thus, summing over alignments A , columns C and branches (p, n, s) :

$$\begin{aligned} C_{ij} &= \sum_A \sum_C \sum_{(p,n,s)} \sum_{a,b} P(x_p = a, x_n = b|Y_{AC}) C_{ij}(a, b, t_{pn}) \\ &= \sum_A \sum_C \sum_{(p,n,s)} \sum_{a,b} \left(\frac{G_p(a)M(t_{pn})_{ab}F_n(b)E_s(a)}{P(Y_{AC})} \right) \left(\frac{1}{M_{ab}(t_{pn})} \sum_k U_{ak}U_{ki}^{-1} \sum_l U_{jl}U_{lb}^{-1} \mathcal{J}_{kl}(t_{pn}) \right) \\ &= \sum_k U_{ki}^{-1} \sum_l U_{jl} \sum_A \sum_C \frac{1}{P(Y_{AC})} \sum_{(p,n,s)} \left(\sum_a U_{ak}G_p(a)E_s(a) \right) \left(\sum_b U_{lb}^{-1}F_n(b) \right) \mathcal{J}_{kl}(t_{pn}) \end{aligned}$$

- The terms $\sum_a U_{ak}G_p(a)E_s(a)$ and $\sum_b U_{lb}^{-1}F_n(b)$ are projections of the peeling and pruning messages onto the eigenvector basis.

3.6 Priors over trees

- Phylogenetic tree: nodes represent species, edges labeled with time (or, more accurately, “evolutionary distance”)
 - Molecular clock-like tree: leaves are temporally situated. Distance from leaf to root = time leaf was sampled
 - The *coalescent* gives a probability distribution over trees within populations

- * Hardy-Weinberg: alleles A and a , frequencies p and $1 - p$, infinite population, no mutation or selection. Diploid frequencies:

$$\begin{aligned} f(AA) &= p^2 \\ f(Aa) &= 2p(1 - p) \\ f(aa) &= (1 - p)^2 \end{aligned}$$

- * Wright-Fisher: as Hardy-Weinberg, but finite population of size N . Frequency X_t of allele A is a discrete-state discrete-time Markov chain with transition matrix

$$p_{ij} = {}^{2N}C_j \left(\frac{i}{2N}\right)^j \left(1 - \frac{i}{2N}\right)^{2N-j}$$

- * Kingman coalescent: for k individuals, the probability that no two of them *coalesce* (i.e. share a parent) in the previous generation is

$$P_{\text{coalescence}} = 1 - \prod_{i=1}^{k-1} \left(1 - \frac{i}{2N}\right) = \sum_{i=1}^{k-1} \frac{i}{2N} + O(N^2) = \frac{k(k-1)}{4N} + O(N^2)$$

Number of generations to coalescence is geometrically distributed with mean $\frac{4N}{k(k-1)}$

- * Continuous approximation: time to coalescence is exponentially distributed with mean $\frac{4N}{k(k-1)}$
- * Many variations of Kingman coalescent, e.g. varying population size: use rescaled time $\tau = \int_0^t \frac{N_0}{N_s} ds$
- * Other modifications consider selection, recombination

- Not all trees are clock-like (concession to varying generation times & mutation rates). A weak exponential prior over total branch lengths may be a suitable alternative

4 Hidden-state models for biological sequences

4.1 Selected applications in the literature

1. Churchill GA. Stochastic models for heterogeneous DNA sequences. Bull Math Biol 1989;51(1):79-94.
2. Brown M, Hughey R, Krogh A, Mian IS, Sjolander K, Haussler D. Using Dirichlet mixture priors to derive hidden Markov models for protein families. Proc Int Conf Intell Syst Mol Biol 1993;1:47-55.
 - Bailey TL, Elkan C. Fitting a mixture model by expectation maximization to discover motifs in biopolymers. Proc Int Conf Intell Syst Mol Biol 1994;2:28-36.
3. Burge C, Karlin S. Prediction of complete gene structures in human genomic DNA. J Mol Biol 1997 Apr 25;268(1):78-94.
4. Goldman N, Thorne JL, Jones DT. Using evolutionary trees in protein secondary structure prediction and other comparative sequence analyses. J Mol Biol 1996 Oct 25;263(2):196-208. **This is an “Evolutionary Hidden Markov model” which we will revisit in section 4.5.**

Alternate examples: signal peptide, transmembrane protein prediction.

4.2 Single-sequence hidden Markov models

- Early motivation: isochores
 - Long regions of uniform GC content (which is correlated with gene density, recombination frequency...)
 - * e.g. Major Histocompatibility Complex (MHC) class II and class III sequences on human chromosome 6
 - * Lengths 900.9 kb, 642.1 kb; GC-content 41%, 52%
 - Gary Churchill: first Hidden Markov Model for isochore detection (1989)
 - Earliest non-thermodynamic hit to “isochore” on PubMed is 1986, Alonso *et al*
 - HMM analogy: occasionally dishonest casino (Durbin *et al*)

- Hidden Markov model: notation
 - Let x denote hidden state, y observed symbol. State space includes START and END
 - Let $e(x, y)$ be probability of emitting character y in state x
 - Let $t(i, j)$ be probability of transition to state j if currently in state i
- Idea of a particular partitioning as a “path” through the HMM
 - Let y_n be observed nucleotide at position n and let x_n be hidden state of position n
 - * Let L be length of sequence
 - * For convenience, set $x_0 = \text{START}$ and $x_{L+1} = \text{END}$
 - * Let $Y = \{y_1 \dots y_L\}$ represent entire observed sequence, $X = \{x_0 \dots x_{L+1}\}$ hidden state sequence
 - * Each $X = \{x_k\}$ represents a “path” (sketch); there are $\sim K^L$ possible paths for K states
 - Joint likelihood is $P(X, Y) = t(x_0, x_1) \prod_{k=1}^L e(x_k, y_k) t(x_k, x_{k+1})$
 - * Note that $P(X, Y) \equiv P(X, Y | x_0)$. Conditioning on x_0 is assumed throughout
- Two questions, two algorithms. (i) Viterbi: what X maximises $P(X, Y)$? (ii) Forward: what is $P(Y)$?
- Viterbi algorithm for finding ML hidden state path

$$\begin{aligned}
 \max_X P(X, Y) &= \max_{x_L} \max_{x_{L-1}} \dots \max_{x_1} t(x_0, x_1) \prod_{k=1}^L e(x_k, y_k) t(x_k, x_{k+1}) \\
 &= \max_{x_L} t(x_L, x_{L+1}) e(x_L, y_L) \max_{x_{L-1}} t(x_{L-1}, x_L) e(x_{L-1}, y_{L-1}) \dots \max_{x_1} t(x_1, x_2) e(x_1, y_1) t(x_0, x_1) \\
 &= \max_{x_L} t(x_L, \text{END}) V_L(x_L)
 \end{aligned}$$

where

$$V_n(x_n) = \begin{cases} e(x_n, y_n) \max_{x_{n-1}} t(x_{n-1}, x_n) V_{n-1}(x_{n-1}) & \text{if } n > 0 \\ 1 & \text{if } n = 0 \end{cases}$$

Note that

$$V_n(x_n) = \max_{x_1 \dots x_{n-1}} P(x_1 \dots x_n, y_1 \dots y_n | x_0)$$

or, in words, $V_n(x)$ is the maximum likelihood of any path ending in state x and emitting symbols $y_1 \dots y_n$.

The ML state sequence, $\operatorname{argmax}_X P(X, Y)$, is recovered by **traceback** from V_L to V_1 .

- Forward algorithm for summing over all possible hidden state paths

$$\begin{aligned} P(Y) &= \sum_X P(X, Y) \\ &= \sum_{x_L} \sum_{x_{L-1}} \dots \sum_{x_1} t(x_0, x_1) \prod_{k=1}^L e(x_k, y_k) t(x_k, x_{k+1}) \\ &= \sum_{x_L} t(x_L, x_{L+1}) e(x_L, y_L) \sum_{x_{L-1}} t(x_{L-1}, x_L) e(x_{L-1}, y_{L-1}) \dots \sum_{x_1} t(x_1, x_2) e(x_1, y_1) t(x_0, x_1) \\ &= \sum_{x_L} t(x_L, \text{END}) F_L(x_L) \end{aligned}$$

where

$$F_n(x_n) = \begin{cases} e(x_n, y_n) \sum_{x_{n-1}} t(x_{n-1}, x_n) F_{n-1}(x_{n-1}) & \text{if } n > 0 \\ 1 & \text{if } n = 0 \end{cases}$$

Note that

$$F_n(x_n) = P(x_n, y_1 \dots y_n | x_0) = \sum_{x_1 \dots x_{n-1}} P(x_1 \dots x_n, y_1 \dots y_n | x_0)$$

or, in words, $F_n(x)$ is the sum of likelihoods of all paths ending in state x and emitting symbols $y_1 \dots y_n$.

By analogy to Viterbi traceback, a ML state sequence can be sampled by stochastic traceback from F_L to F_1 .

- Implementation issues
 - probability underflow for long sequences

- slow floating-point multiply
- motivation for (possibly discretized) log-space scores

4.3 Posterior probabilities for single-sequence HMMs

- Definition of the posterior probability that position n is in state k : sum over paths

$$P(x_n = k|Y) = \frac{\sum_X P(X, Y)\delta(x_n = k)}{P(Y)}$$

- Splitting the path into three parts: $< n$, $= n$ and $> n$

$$P(x_n|Y) = \sum_{x_1 \dots x_{n-1}} \sum_{x_{n+1} \dots x_L} \frac{P(x_1 \dots x_n, y_1 \dots y_n|x_0) P(x_{n+1} \dots x_L, y_{n+1} \dots y_L|x_n)}{P(Y)} = \frac{F_n(x_n) B_n(x_n)}{P(Y)}$$

where

$$B_n(x_n) = P(x_{L+1}, y_{n+1} \dots y_L|x_n) = \sum_{x_{n+1} \dots x_L} P(x_{n+1} \dots x_L, y_{n+1} \dots y_L|x_n)$$

Likewise,

$$P(x_n, x_{n+1}|Y) = \frac{F_n(x_n)t(x_n, x_{n+1})e(x_{n+1}, y_{n+1})B_{n+1}(x_{n+1})}{P(Y)}$$

and (useful for compression)

$$P(y_{n+1} = k|y_1 \dots y_n) = \frac{\sum_i \sum_j F_n(i)t(i, j)e(j, k)}{\sum_i F_n(i)}$$

- The Backward algorithm

$$B_n(x_n) = \begin{cases} \sum_{x_{n+1}} t(x_n, x_{n+1})e(x_{n+1}, y_{n+1})B_{n+1}(x_{n+1}) & \text{if } n < L \\ t(x_n, \text{END}) & \text{if } n = L \end{cases}$$

- The Baum-Welch training algorithm

- Likelihood can be written in the form

$$P(X, Y) = \left(\prod_{i,j} t(i, j)^{u(i,j)} \right) \left(\prod_{i,k} e(i, k)^{f(i,k)} \right)$$

where $u(i, j)$ is the number of transitions $i \rightarrow j$ and $f(i, k)$ is the number of emissions of character k from state i

- Sufficient statistics for EM algorithm are therefore

$$\hat{t}(i, j) = \sum_{n=0}^L P(x_n = i, x_{n+1} = j | Y)$$

$$\hat{e}(i, k) = \sum_{n=1}^L P(x_n = i | Y) \delta(y_n = k)$$

where \hat{t} and \hat{e} are the posterior expectations of u and f . Dirichlet (mixture) priors can be used for t and e .

- Note that $\hat{t}(i, j) = \frac{\partial \log P(X)}{\partial \log t(i, j)}$ and $\hat{e}(i, k) = \frac{\partial \log P(X)}{\partial \log e(i, k)}$

- Implementation issues: log-space probability addition

$$\log(e^a + e^b) = \max(a, b) + \oplus(|a - b|) \quad \oplus(|a - b|) = \log(1 + e^{-|a-b|})$$

If scores are discretized, can implement \oplus as a lookup table for speed.

- Null states

- Sparse transition matrices are good (Forward/Backward and Viterbi time complexities are \propto no. of transitions)
 - Convenience: reduce number of transitions (e.g. delete states of profile HMM)
 - Need to do a *topological sort* of null states so that they're filled in the correct order

- Awkwardness: null cycles. Can always eliminate null states as follows

$$\mathbf{t} = \mathbf{a} + \mathbf{b} + \mathbf{c} + \mathbf{d} \quad \text{where} \quad \mathbf{a} = \begin{pmatrix} * & 0 \\ 0 & 0 \end{pmatrix}, \mathbf{b} = \begin{pmatrix} 0 & * \\ 0 & 0 \end{pmatrix}, \mathbf{c} = \begin{pmatrix} 0 & 0 \\ * & 0 \end{pmatrix}, \mathbf{d} = \begin{pmatrix} 0 & 0 \\ 0 & * \end{pmatrix}$$

$$\mathbf{t}' = \begin{pmatrix} * & 0 \\ 0 & 0 \end{pmatrix} = \mathbf{a} + \sum_{n=0}^{\infty} \mathbf{b} \mathbf{d}^n \mathbf{c} = \mathbf{a} + \mathbf{b}(\mathbf{I} - \mathbf{d})^{-1} \mathbf{c}$$

where \mathbf{a} (and \mathbf{t}') contain emit→emit transitions, \mathbf{b} emit→null, \mathbf{c} null→emit and \mathbf{d} null→null.

- Factor graph representation of HMM; similarity to pruning/peeling and parsimony (Forward-Backward \subset Sum-Product)
- General-purpose HMM implementations: DART library (C++)

4.4 Pair Hidden Markov models

- Motivation: pairwise sequence alignment, pairwise genefinding, etc.
- Let x denote hidden state, y character in sequence Y , z character in sequence Z
- Let $\Delta y(x)$ be 1 if state x emits a character to Y , and 0 otherwise; likewise $\Delta z(x) = 1$ iff x emits to Z
- Emission probability $e(x, y, z)$ is defined as follows:
 - If $\Delta y(x) = 1$ and $\Delta z(x) = 0$, then x is called a **delete** state and $e(x, y, z) \equiv e_d(x, y)$ is a function of x and y only
 - If $\Delta y(x) = 0$ and $\Delta z(x) = 1$, then x is called an **insert** state and $e(x, y, z) \equiv e_i(x, z)$ is a function of x and z only
 - If $\Delta y(x) = 1$ and $\Delta z(x) = 1$, then x is called a **match** state and $e(x, y, z) \equiv e_m(x, y, z)$ is a function of x, y and z
 - If $\Delta y(x) = 0$ and $\Delta z(x) = 0$, then x is called a **null** state and $e(x, y, z)$ is a function of x only (typically just 1)
 - We will assume for now that there are no null states (apart from START and END).
- As before, $t(i, j)$ is the probability of transition to state j if currently in state i

- Suppose sequence lengths are K, L so observed data are $Y = \{y_1 \dots y_K\}$ and $Z = \{z_1 \dots z_L\}$
- Again we have a state path $x_1, x_2 \dots x_N$ and for convenience we set $x_0 = \text{START}$ and $x_{N+1} = \text{END}$.
 - Denote by Λ_{kl} the event that there exists a *break* at (k, l) :

$$\Lambda_{kl} \Rightarrow \exists n : \sum_{i=1}^n \Delta y(x_i) = k, \sum_{i=1}^n \Delta z(x_i) = l$$

So Λ_{kl} means that, at some point n on the state path, the model has emitted k symbols to Y and l symbols to Z .

- Viterbi

$$V_{kl}(x_n) = \max_{x_1 \dots x_{n-1}} P(\Lambda_{kl}, x_1 \dots x_n, y_1 \dots y_k, z_1 \dots z_l | x_0)$$

Recursion (assuming no null states)

$$V_{kl}(x_n) = \begin{cases} e(x_n, y_k, z_l) \max_{x_{n-1}} t(x_{n-1}, x_n) V_{k-\Delta y(x_n), l-\Delta z(x_n)}(x_{n-1}) & \text{if } k > 0 \text{ or } l > 0 \\ 1 & \text{if } k = l = 0 \text{ and } x_n = \text{START} \\ 0 & \text{if } k = l = 0 \text{ and } x_n \neq \text{START} \\ 0 & \text{if } k < 0 \text{ or } l < 0 \end{cases}$$

- Forward

$$F_{kl}(x_n) = P(\Lambda_{kl}, x_n, y_1 \dots y_k, z_1 \dots z_l | x_0) = \sum_{x_1 \dots x_{n-1}} P(\Lambda_{kl}, x_1 \dots x_n, x_{N+1}, y_1 \dots y_k, z_1 \dots z_l | x_0)$$

Recursion (assuming no null states)

$$F_{kl}(x_n) = \begin{cases} e(x_n, y_k, z_l) \sum_{x_{n-1}} t(x_{n-1}, x_n) F_{k-\Delta y(x_n), l-\Delta z(x_n)}(x_{n-1}) & \text{if } k > 0 \text{ or } l > 0 \\ 1 & \text{if } k = l = 0 \text{ and } x_n = \text{START} \\ 0 & \text{if } k = l = 0 \text{ and } x_n \neq \text{START} \\ 0 & \text{if } k < 0 \text{ or } l < 0 \end{cases}$$

- Backward

$$B_{kl}(x_n) = P(\Lambda_{kl}, x_{N+1}, y_{k+1} \cdots y_K, z_{l+1} \cdots z_L | x_n) = \sum_{x_{n+1} \cdots x_N} P(\Lambda_{kl}, x_{n+1} \cdots x_{N+1}, y_{k+1} \cdots y_K, z_{l+1} \cdots z_L | x_n)$$

Recursion (assuming no null states)

$$B_{kl}(x_n) = \begin{cases} \sum_{x_{n+1}} t(x_n, x_{n+1}) e(x_{n+1}, y_{k+1}, z_{l+1}) B_{k+\Delta y(x_{n+1}), l+\Delta z(x_{n+1})}(x_{n+1}) & \text{if } k < K \text{ or } l < L \\ t(x_n, \text{END}) & \text{if } k = K \text{ and } l = L \\ 0 & \text{if } k > K \text{ or } l > L \end{cases}$$

- Evidence, posterior probabilities & EM counts

$$\begin{aligned} P(Y, Z) &= \sum_x F_{KL}(x) t(x, \text{END}) \\ P(\Lambda_{kl}, x_n | Y, Z) &= \frac{F_{kl}(x_n) B_{kl}(x_n)}{P(Y)} \\ P(\Lambda_{kl}, x_n, x_{n+1} | Y) &= \frac{F_{kl}(x_n) t(x_n, x_{n+1}) e(x_{n+1}, y_{k+1}, z_{l+1}) B_{k+\Delta y(x_{n+1}), l+\Delta z(x_{n+1})}(x_{n+1})}{P(Y)} \\ \hat{t}(i, j) &= \sum_{k=0}^K \sum_{l=0}^L P(\Lambda_{kl}, x_n = i, x_{n+1} = j | Y, Z) \\ \hat{e}_m(x, y, z) &= \sum_{k: y_k=y} \sum_{l: z_l=z} P(\Lambda_{kl}, x_n = x | Y, Z) \\ \hat{e}_d(x, y) &= \sum_{k: y_k=y} \sum_{l=0}^L P(\Lambda_{kl}, x_n = x | Y, Z) \\ \hat{e}_i(x, z) &= \sum_{k=0}^K \sum_{l: z_l=z} P(\Lambda_{kl}, x_n = x | Y, Z) \end{aligned}$$

- Decision theory (“optimal accuracy”).

- Decision theory: maximise expected “reward”, making use of the posterior distribution
- Overlap score: an objective function (i.e. reward) that compares predicted alignment α with true alignment α'
 - * Overlap score is $|\alpha \cap \alpha'|$, where an alignment is viewed as a set of match co-ords $\alpha = \{(k_1, l_1), (k_2, l_2) \dots\}$
 - * Several other good objective functions (e.g. “Cline shift score”); overlap is simpler, albeit less realistic
 - NB also $\delta(\alpha = \alpha')$ which only rewards perfect alignments, yielding a multiplicative, Viterbi-like recursion
 - * Example criteria: how good is alignment for structure prediction? homology detection? benchmark of choice?
 - e.g. PROBCONS (Batzoglou *et al*) uses the sum-of-pairs score, same as the BALiBASE benchmark
- Posterior expectation of overlap score for an alignment (NB only match states have $\Delta y(x)\Delta z(x) \neq 0$)

$$A[\alpha] = \sum_{(k,l) \in \alpha} P(\text{match}, k, l) \quad P(\text{match}, k, l) = \sum_x \Delta y(x)\Delta z(x)P(\Lambda_{kl}, x_n = x)$$

- Optimal accuracy recursion: let α_{kl} be any alignment up to (k, l) , so $k' \leq k$ and $l' \leq l$ for all $(k', l') \in \alpha$. Then

$$O_{kl} = \max_{\alpha_{kl}} A[\alpha_{kl}] = \begin{cases} \max \begin{pmatrix} O_{k-1, l-1} + P(\text{match}, k, l), \\ O_{k, l-1}, \\ O_{k-1, l} \end{pmatrix} & \text{if } k \geq 0 \text{ and } l \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

- Optimal alignment α recovered by traceback from O_{KL} .

- Dynamic programming algorithms whose finite state automata are almost or exactly Pair HMMs

- Needleman-Wunsch; Smith-Waterman; Gotoh; Altschul, Proteins 1998
- General implementations: DART library (C++), Exonerate (C)

4.5 Evolutionary Hidden Markov models

- Can readily extend the Pair HMM to a multi-sequence HMM for multiple sequence alignment
 - Arbitrary number N of output sequences $Y^{(1)}, Y^{(2)}, Y^{(3)} \dots Y^{(N)}$ of lengths $L_1 \dots L_N$ (see e.g. Holmes 2003)
 - Dynamic programming time/memory complexity is $O(\prod_n L_n)$ —not cheap
 - Ultimately, would like to structure $\Delta Y^{(n)}(x)$, $t(x, x')$ and $e(x, y^{(1)} \dots y^{(N)})$ according to some underlying phylogenetic tree
 - * The DP algorithms can also be tree structured, c.f. “progressive alignment”
 - For now, we ignore phylogenetic structure of indels (Δ, t) and concentrate on substitution model (e)
- Initial, simplistic, restrictive concept of Evolutionary HMM, lacking a good gap model:
 - **A single-sequence HMM that emits phylogenetically-correlated multiple alignment columns, instead of single characters.**
 - Follows e.g. Goldman, Thorne & Jones, 1996 (“*Using evolutionary trees in protein secondary structure prediction...*”)
 - For now, gaps will be glossed over heuristically (disallowed/treated as wildcards/excessively gappy columns discarded/etc.)
- For a tree with N leaves, the emitted symbol alphabet is Ω^N where Ω is the single-character alphabet
- Now the emission function $e(x, \mathbf{y})$ for $\mathbf{y} \in \Omega^N$ is expensive (and unnecessary) to tabulate
 - We can implement it as an instance of pruning instead: DP within DP
 - Assume an underlying continuous-time discrete-state Markov chain with rate matrix $\mathbf{R}^{(x)}$ and initial distribution $\pi^{(x)}$
 - Let $e(x, \mathbf{y})$ be probability of observing characters \mathbf{y} at (leaf) nodes of a phylogenetic tree, with this process
 - * Tree, like alignment, will be specified as an input to our DP recursions
- The counts $\hat{t}(x, x')$ are still relevant, but $\hat{e}(x, \mathbf{y})$ are used to accumulate EM update counts for $\mathbf{R}^{(x)}$

- Can update t 's and $\mathbf{R}^{(x)}$'s simultaneously or asynchronously; c.f. Neal and Hinton
- Many applications in genomic biology (see class projects)
 - Annotation of multiple alignments of DNA, RNA or protein sequences
 - Isochores; gene prediction; DNA-protein binding site modeling and analysis; protein transmembrane structure, signal peptide, domain profiles... etc.
- Implementation: `xrate` (distributed with the DART library)
 - S-expressions for the underlying alignment grammar
 - * Here “alignment grammar” means the alphabet Ω , the HMM transition matrix t , the Δ 's, the \mathbf{R} 's and the π 's
 - * More generally, can use stochastic context-free grammars as well as HMMs, & states can emit several co-evolving columns
 - Stockholm format for alignment, tree, Viterbi annotation, likelihoods and posterior probabilities
 - * Allows internal nodes as well as leaves to be specified
 - By default, gaps in the multiple alignment are treated as wildcards (unobserved character is summed over)
 - * A smarter handling of gaps soon leads us to indel rate models and so-called “**Statistical Alignment**”
 - Log messages (type `xrate -help` and `xrate -loghelp`)
 - Command-line options (type `xrate -help`)

4.6 Discriminative models and conditional random fields

- HMMs model $P(Y) = \sum_X P(X, Y)$ (*generative* modeling). ML training maximizes this probability.
- Intuitively, since we are interested in predicting X correctly, it may make more sense to model conditional probability $P(X|Y)$ (*discriminative* modeling)

- Consider the conditional likelihood for an HMM, expressed in terms of the *feature vector* $\{u, f\}$ implied by X :

$$\log P(X|Y) = \frac{1}{P(Y)} \exp \left(\sum_{i,j} u(i, j) \log t(i, j) + \sum_{i,k} f(i, k) \log e(i, k) \right)$$

where $P(Y)$ is computed by the Forward algorithm.

- We can write down a likelihood $P(X|Y)$ for a similarly trellis-structured graphical model as follows

$$P(X|Y) = \frac{1}{Z} \exp \left(\sum_{i,j} u(i, j) a(i, j) + \sum_{i,k} f(i, k) b(i, k) \right)$$

where Z is a partition function (computed by a Forward-like sum-product algorithm)

$$Z = \sum_{X'} \exp \left(\sum_{i,j} u_{X'}(i, j) a(i, j) + \sum_{i,k} f_{X'}(i, k) b(i, k) \right)$$

For equivalence with the HMM, set $a(i, j) = \log t(i, j)$ and $b(i, k) = \log e(i, k)$.

- A *linear-chain conditional random field* is essentially such a trellis-structured model, lacking the normalization constraints on the parameters $\theta = \{a, b\}$ that are implicit in the generative HMM
 - Sutton & McCallum, “An Introduction to Conditional Random Fields for Relational Learning”
 - Lafferty, McCallum & Pereira, “Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data”
- To avoid overfitting, and in place of the normalization constraints, it is useful to add a *regularization* term, e.g. a Gaussian prior on $a()$ and $b()$ with variance σ^2 , penalizing large weights. Then the function to be optimized is

$$\ell(\theta) = \log P(X|Y) - \sum_{i,j} \frac{a(i, j)^2}{2\sigma^2} + \sum_{i,k} \frac{b(i, k)^2}{2\sigma^2}$$

- Parameter estimation proceeds by **numerical optimization** of $\ell(\theta)$, typically using quasi-Newton algorithms
 - e.g. the BFGS algorithm: Dimitri P. Bertsekas. *Nonlinear Programming*. Athena Scientific, 2nd edition, 1999
- The partial derivatives $\frac{\partial \ell}{\partial a^{(i,j)}}$ and $\frac{\partial \ell}{\partial b^{(i,k)}}$ are computed by direct analogy to $\hat{t}(i, j)$ and $\hat{u}(i, k)$ in Baum-Welch.
- In the absence of normalization constraints, we are free to add more “features” without having to directly account for them by subdividing the state space of the HMM. For example: a run of T’s, a palindromic motif, etc. In the generative (HMM) view, all such features must be explicitly identified with a path through the model, so that nothing is counted more than once. In a discriminative framework, it doesn’t matter if a residue is counted twice.
- In the trellis factor graph view, functions relating x_i and y_i are $P(x_i|Y)$ rather than $P(y_i|x_i)$
- HMMs and linear CRFs form what Ng and Jordan (2002) call a “generative-discriminative pair”.
 - Other such pairs include naive Bayes *vs* logistic regression.

5 Data compression

5.1 Coding and data compression

- Kraft-McMillan inequality and optimal compression (MacKay pp92+)
 - Let \mathcal{A}^+ be the set of all finite-length strings over the alphabet \mathcal{A} . A binary symbol code C is a mapping from \mathcal{A} to $\{0, 1\}^+$. The extended code C^+ is a mapping from \mathcal{A}^+ to $\{0, 1\}^+$ obtained by concatenation of the corresponding codewords:

$$c^+(x_1x_2 \dots x_N) = c(x_1)c(x_2) \dots c(x_N)$$

The code C is *uniquely decodable* if, under the extended code C^+ , no two strings have the same encoding. (NB if no codeword is a prefix of any other codeword, then C is also a *prefix code* with a corresponding tree.)

- Let $l(x)$ be the length of the codeword $C(x)$. Let $z = \sum_{x \in \mathcal{A}} 2^{-l(x)}$. Consider

$$z^N = \left[\sum_{x \in \mathcal{A}} 2^{-l(x)} \right]^N = \sum_{x_1 \in \mathcal{A}} \sum_{x_2 \in \mathcal{A}} \dots \sum_{x_N \in \mathcal{A}} 2^{-(l(x_1)+l(x_2)+\dots+l(x_N))}$$

The quantity $l(x_1) + l(x_2) + \dots + l(x_N)$ is the length of the C^+ -encoding of the string $\mathbf{x} = x_1x_2 \dots x_N$. For every string $\mathbf{x} \in \mathcal{A}^N$, there is one term in the above sum for z^N .

- Let A_l be the number of strings \mathbf{x} having encoded length l . Let $l_{\min} = \min_x l(x)$ and let $l_{\max} = \max_x l(x)$. Then

$$z^N = \sum_{l=Nl_{\min}}^{Nl_{\max}} 2^{-l} A_l$$

- If C is uniquely decodable, then $A_l \leq 2^l$. Therefore

$$z^N = \sum_{l=Nl_{\min}}^{Nl_{\max}} 2^{-l} A_l \leq \sum_{l=Nl_{\min}}^{Nl_{\max}} 1 \leq Nl_{\max}$$

Since this holds for arbitrarily large N , it must be the case that $z \leq 1$. This is the **Kraft-McMillan inequality** (or, strictly, McMillan's part of it; Kraft showed that if the inequality holds for some set of $l(x)$, then there is a corresponding prefix code.)

- Assume a probability distribution $p(x)$ over observed symbols and let the *expected length* of C be $L_p(C) = \langle l(x) \rangle_p$. **The entropy H_p is a lower bound on $L_p(C)$.**
- Proof: define the implicit probabilities $q(x) = 2^{-l(x)}/z$ where $z = \sum_x 2^{-l(x)}$. By Gibbs' inequality, $\langle \log 1/q(x) \rangle_p \geq \langle \log 1/p(x) \rangle_p$, and by Kraft-McMillan, $z \leq 1$. Thus

$$L_p(C) = \langle \log 1/q(x) \rangle_p - \log z \geq \langle \log 1/p(x) \rangle_p - \log z \geq H_p$$

- **The expected length $L_p(C)$ is minimized if and only if the codelengths $l(x)$ are equal to the Shannon information contents $h(x)$.** This is one intuitive justification of the term “information content”.
- Note that any compression scheme has an implicit probability distribution $q(x)$ for which it is optimal.
- **Huffman tree** generates optimal prefix codes with $L_p(C) < H_p + 1$ (but NB overhead of 1 bit/symbol).
 - Bottom-up tree construction algorithm: combine the two least frequent symbols into a single symbol, and repeat.
- **Arithmetic coding** is a block (not symbol) compression scheme that achieves asymptotically perfect efficiency.
 - Suppose x_k is the k 'th symbol in \mathcal{A} . Let x_k be associated with the subinterval $[r_k, r_k + p_k)$ of the unit interval $[0, 1)$, where $p_i = p(x_i)$ and $r_k = \sum_{i < k} p_i$.
 - Consider the two-symbol string $x_j x_k$. This can be encoded by taking the subinterval $[r_k, r_k + p_k)$ of the subinterval $[r_j, r_j + p_j)$; that is, the interval $[r_j + r_k p_j, r_j + (r_k + p_k) p_j)$.
 - Successively longer strings can be encoded by further subdividing the interval. Suppose $[a, a + b)$ is the current interval; then, to transmit the next character x_k , we set $a' \leftarrow a + r_k b$ and $b' \leftarrow (r_k + p_k) b$.
 - (The **Dasher** program is an amazing, interactive illustration of this concept put to a novel & practical use.)
 - Assuming that \mathcal{A} contains an “end-of-file” character (or the string to be encoded is of fixed length), the string can be uniquely identified by transmitting any binary real number c in the final interval $[a, b)$.

- To avoid the encoder having to track numbers to high precision, whenever a and b share a common first bit, this encoded bit can be transmitted and the interval rescaled. Likewise, as bits are received by the decoder, whenever the real number c lands within an interval $[r_k, r_k + p_k)$ the corresponding decoded symbol x_k can be output and c can be rescaled.
- Examples involving composition of DNA and protein sequences
- Order- N models for
 - text (e.g. Shannon): $P(x_{N+k}|x_k \dots x_{N+k-1})$
 - sequences (e.g. $CG \rightarrow TG$ effects)

5.2 String compression in practice

- Heuristic methods
 - Run-length encoding: any repeated character is followed by an integer *run-length*
 - Lempel-Ziv compression: substrings from a recent window can be reinserted. Used by GIFs (patented by Compuserve)
 - Burrows-Wheeler Transform or “block-sorting” compression. Used by **bzip**
 - * Sort all “rotations” of the text; transmit final column
 - The n 'th “rotation” of the string $x_1x_2 \dots x_N$ is $x_n \dots X_Nx_1 \dots x_{n-1}$
 - * Replaces repeated substrings by repeated characters
- Probabilistic models
 - Prediction by Partial Matching (PPM). Uses finite context strings. Can “escape” to shorter context string
 - Markov chains

6 Statistical alignment

6.1 Mechanisms of sequence mutation

- Point substitution: the canonical models
- Context-dependent and multiple-nucleotide substitutions
 - e.g. CpG \rightarrow 5-methyl-cytosine (5mC) \rightarrow C-deamination \rightarrow TpG
- Insertions and deletions; mechanisms
 - DNA foldback, cruciforms, etc; polymerase stutter; microsatellites
 - Transposition (DNA cut'n'paste; retrotransposition; rolling-circle); horizontal transfer
- Duplications (local, nonlocal); inversions; whole-chromosome and -genome duplications, polyploidy
- Rearrangement
- Recombination; gene conversion

6.2 The “links model” and string transducers

- Motivation: how to handle gaps? Gaps as a fifth nucleotide in standard point substitution model: advantages (simplicity), drawbacks (irreversibility, “ghost” bases)
- Whole-sequence models: state space Ω^* . Rate grammar notation: write mutation rules
 - Point substitution with rate matrix \mathbf{Q} : rule is $AxB \rightarrow AyB : Q_{xy}$ where $A, B \in \Omega^*$ and $x, y \in \Omega$
 - Insertion: rule can be written $AC \rightarrow ABC$, etc.

- How can we calculate pairwise likelihoods (i.e. matrix exponential) or multiple alignment likelihoods for such a model? At first glance, appears impossible (or very difficult): infinite-dimensional (albeit sparse) rate matrix where total mutation rate (and number of mutations) scales with sequence length.
- Under point substitution model, likelihood for whole alignment factorises into product of column likelihoods. This works because each column can be thought of as an independently evolving zone. Extension to indel models follows same idea of independent zones.
- Thorne, Kishino, Felsenstein 1991. **Links model.**

– First consider the following rate grammar:

Event	Rule	Rate
Substitution	$AxB \rightarrow AyB$	Q_{xy}
Insertion	$AB \rightarrow AyB$	$\lambda\pi_y$
Deletion	$AxB \rightarrow AB$	μ

– Here $A, B \in \Omega^*$, $x, y \in \Omega$, \mathbf{Q} is a reversible Markov process on Ω with equilibrium π , λ is a point insertion rate and μ is a point deletion rate

- Reversibility and equilibrium of the model

– Number of links in sequence evolves independently from actual nucleotides themselves

– Let n be sequence length (i.e. number of mortal links)

- * n evolves according to a classical “linear birth-death process with constant immigration” (in fact immigration rate = birth rate)
- * Total insertion rate (i.e. rate of $n \rightarrow n + 1$) is $\lambda(n + 1)$
- * Total deletion rate (i.e. rate of $n \rightarrow n - 1$) is μn
- * If reversibility holds, then $P(n) \times \text{Rate}(n \rightarrow n + 1) = P(n + 1) \times \text{Rate}(n + 1 \rightarrow n)$ where $P(n)$ is equilibrium length distribution
- * Thus $P(n)\lambda(n + 1) = P(n + 1)\mu(n + 1)$ so that $P(n) = \kappa^n(1 - \kappa)$ where $\kappa = \frac{\lambda}{\mu}$ i.e. geometric

- * NB for normalisation, $\kappa < 1 \Rightarrow \lambda < \mu$
- * Expected sequence length at equilibrium is $\frac{\kappa}{1-\kappa}$. As $\lambda \rightarrow \mu$, equilibrium sequence length $\rightarrow \infty$
- * Equation of state for $n(t)$. Here $P_t(n') = P(n(t) = n')$

$$\frac{d}{dt}P_t(n) = \lambda n P_t(n-1) + \mu(n+1)P_t(n+1) - (\lambda n + \mu(n+1))P_t(n)$$

Same as equation for immortal zone fates (see below)

- Clearly individual nucleotides are distributed according to π at equilibrium (since newly-inserted nucleotides are also sampled from this distribution). So equilibrium probability distribution over sequences X is

$$P(X) = \kappa^{|X|}(1-\kappa) \prod_{i=1}^{|X|} \pi_{X_i}$$

NB this is also the likelihood for generating X from a single-state HMM with self-loop transition probability κ and emit vector π .

- TK&F introduced the following construction to help analyse this model (following Bishop & Thompson, 1986)
 - A biological sequence is modeled as a sequence of links: one “immortal link” followed by zero or more normal or “mortal links”
 - Mortal and immortal links spawn new “child links” to their right (rate λ). Mortal links can also die (rate μ).
 - Each mortal link corresponds to an observed, independently evolving nucleotide (or amino acid). The immortal link is invisible. (Without the immortal link, if the sequence ever reached zero length it would get irreversibly stuck. This might or might not be realistic, but one goal of Thorne *et al* was a reversible model.)
 - From the TKF 1991 paper:

“The insertion-deletion process is framed in terms of a birth-death process of these links. Each link evolves independently from all other link; a birth or death of one link does not affect the probability of a birth or death of any other link. Both types of links can be associated with births. The birth rate per normal link is equal to the birth rate per immortal link (λ). A newborn link is always a normal link. We adopt the convention that it appears immediately to the right of its parent link. Accompanying the birth of a normal link is the birth of a DNA base immediately to the left of the newborn link. The probabilities that the newborn DNA base will be A, G, T, or C are π_A , π_G , π_T and π_C , respectively. Normal links are subject to death (μ is the death rate per normal link) but immortal links are not.”

- Solving for conditional probabilities $P(\text{descendant}|\text{ancestor})$ in TKF91
 - Consider an ancestral sequence with n mortal links (and one immortal link)
 - Each ancestral link defines a zone that evolves independently from all other zones
 - Zones corresponding to mortal ancestral links can acquire new links, lose links (including the original mortal link) and even die off (i.e. lose all its links, and become inert)
 - * NB this implies the process w.r.t. zones is irreversible. Apparent paradox arises because we’ve introduced new information in the form of the zone (alignment) co-ordinates, and our original model did not promise to be reversible with respect to zones (alignments).
 - * A specific consequence of fixing the zone co-ordinates is that the alignment $\begin{matrix} X- \\ -X \end{matrix}$ is distinct from the alignment $\begin{matrix} -X \\ X- \end{matrix}$, since the latter implies an insertion followed by a deletion in the same zone, whereas the former does not imply any ordering on the insertion & deletion events, since they occurred in different zones. This asymmetry can, in fact, be “fixed” by permuting certain alignment columns when reversing the arrow of time, but this is something of a technical detail.
 - The zone corresponding to the immortal link can acquire and lose new links, but never dies off
 - Differential equations for zone fates. Suppose time t has elapsed since ancestral sequence observed. Let

- * $p_n(t)$ be the probability that n mortal links are descended from a mortal link **and that one of them is the original**;
- * $q_n(t)$ be the probability that n mortal links are descended from a mortal link **and that the original died**;
- * $r_n(t)$ be the probability that n mortal links are descended from an immortal link.

Differential equations:

$$\begin{aligned}\frac{d}{dt}p_n(t) &= \lambda(n-1)p_{n-1}(t) + \mu np_{n+1}(t) - (\lambda + \mu)np_n(t) \\ \frac{d}{dt}q_n(t) &= \lambda(n-1)q_{n-1}(t) + \mu(n+1)q_{n+1}(t) + \mu p_{n+1}(t) - (\lambda + \mu)nq_n(t) \\ \frac{d}{dt}r_n(t) &= \lambda nr_{n-1}(t) + \mu(n+1)r_{n+1}(t) - (\lambda(n+1) + \mu n)r_n(t)\end{aligned}$$

Boundary conditions:

$$\begin{aligned}p_n(0) &= \delta(n=1) \\ q_n(0) &= 0 \\ r_n(0) &= \delta(n=0)\end{aligned}$$

Solutions:

$$\begin{aligned}p_n(t) &= \alpha\beta^{n-1}(1-\beta) \\ q_0(t) &= (1-\alpha)(1-\gamma) \\ q_n(t) &= (1-\alpha)\gamma\beta^{n-1}(1-\beta) \quad \text{for } n > 0 \\ r_n(t) &= \beta^n(1-\beta)\end{aligned}$$

where

$$\alpha(t) = \exp(-\mu t)$$

$$\beta(t) = \frac{1 - \alpha(t) \exp(\lambda t)}{\kappa^{-1} - \alpha(t) \exp(\lambda t)}$$

$$\gamma(t) = 1 - \frac{\beta(t)}{\kappa(1 - \alpha(t))}$$

- Thorne *et al* quote these results without derivation (they were obtained by comparison with formulae for similar generic birth-death processes). They can readily be verified.
- Metzler argument: if n is the no. of surviving links at time t , then $P(n \geq k + 1 | n \geq k)$ must be independent of k , since $n \geq k$ implies that there has been an available insertion site for time t
 - * Holmes used this to derive numerical estimates of posterior expectations for number of indels
- Feller sketches a systematic approach that can be used to solve for $r_n(t)$ and guess forms of p_n, q_n :
 - * **Linear birth-death process with birth rate λ , death rate μ and (constant) immigration rate λ**
 - * Write down p.d.e. for generating function
 - Introduce generating function $G(s, t) = \sum_{n=0}^{\infty} s^n r_n(t)$. The r_n are recoverable as $r_n(t) = \left. \frac{\partial^n G}{\partial s^n} \right|_{s=0}$

· Let $\Delta = \frac{\partial}{\partial s}$ and use the following operator table to build a p.d.e.:

	Operator L	Coefficient of s^n in LG
	1	r_n
$s\Delta s$	$s(1 + s\Delta)$	nr_{n-1}
	Δ	$(n+1)r_{n+1}$
Δs	$1 + s\Delta$	$(n+1)r_n$
	$s\Delta$	nr_n

$$\begin{aligned} \frac{\partial G}{\partial t} &= [\lambda s(1 + s\Delta) + \mu\Delta - \lambda(1 + s\Delta) - \mu s\Delta] G \\ &= \lambda(s-1)G + (\lambda s - \mu)(s-1) \frac{\partial G}{\partial s} \end{aligned}$$

Can rewrite this as

$$\frac{1}{\lambda(s-1)} \frac{\partial G}{\partial t} + (\mu/\lambda - s) \frac{\partial G}{\partial s} = G$$

Boundary condition is $G(s, 0) = 1$.

* Use method of characteristics to rewrite this p.d.e. as o.d.e.'s.

• Suppose $s = s(u)$ and $t = t(u)$. Then $\frac{dG}{du} = \frac{\partial G}{\partial t} \frac{dt}{du} + \frac{\partial G}{\partial s} \frac{ds}{du}$ which looks like our p.d.e. if

$$\begin{aligned}\frac{dt}{du} &= \frac{1}{\lambda(s-1)} \\ \frac{ds}{du} &= \mu/\lambda - s \\ \frac{dG}{du} &= G\end{aligned}$$

The general solution for G is

$$G(s, t) = g(v)e^u$$

where $g(\cdot)$ is any function and v is constant along a characteristic, so $dv/du = 0$.

Solving the o.d.e. for $s(u)$ we obtain $s = e^{-u} + \mu/\lambda$.

Furthermore, on a characteristic curve, the following is true

$$\frac{dt}{ds} = \frac{dt/du}{ds/du} = \frac{1}{\lambda(s-1)(\mu/\lambda - s)}$$

and hence

$$\log \left| \frac{s - \mu/\lambda}{s - 1} \right| + (\mu - \lambda)t = \text{const.}$$

The general solution for G can thus be written

$$G(s, t) = g \left(\frac{s - \mu/\lambda}{s - 1} e^{(\mu - \lambda)t} \right) (s - \mu/\lambda)^{-1}$$

where $g(\cdot)$ is an arbitrary function, to be determined by the boundary condition – which is $G(s, 0) = 1$, so

$$\begin{aligned}
 g\left(\frac{s - \mu/\lambda}{s - 1}\right) &= s - \mu/\lambda \\
 g(v) &= (\mu/\lambda - 1) \left(\frac{1}{1 - v} - 1\right) \\
 G(s, t) &= \frac{\mu/\lambda - 1}{\mu/\lambda - e^{(\lambda-\mu)t} - s(1 - e^{(\lambda-\mu)t})} \\
 r_n(t) &= \left. \frac{\partial^n G}{\partial s^n} \right|_{s=0} \\
 &= \left. \frac{(\mu/\lambda - 1) (1 - e^{(\lambda-\mu)t})^n}{(\mu/\lambda - e^{(\lambda-\mu)t} + s(e^{(\lambda-\mu)t} - 1))^{n+1}} \right|_{s=0} \\
 &= (1 - \beta(t)) \beta(t)^n
 \end{aligned}$$

as expected.

- Karlin and McGregor analysed linear birth-death processes in detail, and found a series of orthogonal polynomials associated with transition probabilities of the process
 - * Can this be used to get exact formulae posterior counts of the numbers of insertions and deletions?
- TKF91 as a transducer and a Pair HMM
 - Transducer: stochastic finite state machine that “eats” input symbols and “emits” output symbols, representing the action of a finite time interval t (ancestor=input, descendant=output)
 - * States **START**, **INSERT**, **WAIT**, **MATCH**, **DELETE**, **END**
 - **WAIT** is a null state introduced for later convenience; it means “wait for input symbol”
 - * A transducer is similar to a Pair HMM, but normalised differently
 - Forward likelihood is conditional $P(\text{des}|\text{anc})$ rather than joint $P(\text{anc}, \text{des})$

- * Emission probabilities $\exp(\mathbf{Q}t)_{xy}$ (MATCH state), π_y (INSERT state), 1 (DELETE state)

From/To	S	I	W	M	D	E
S	.	β	$1 - \beta$.	.	.
I	.	β	$1 - \beta$.	.	.
W	.	.	.	α	$1 - \alpha$	1 (dots are zeroes)
M	.	β	$1 - \beta$.	.	.
D	.	γ	$1 - \gamma$.	.	.
E

- Can obtain joint Pair HMM for likelihood $P(\text{anc}, \text{des})$ by “left-multiplying” transducer by single-state HMM for ancestor

From/To	S	I	E
S	.	κ	$1 - \kappa$
I	.	κ	$1 - \kappa$
E	.	.	.

- Joint anc-des states SS, SI, SW, IM, ID, II, IW, EE

- Emission probabilities $\pi_x \exp(\mathbf{Q}t)_{xy}$ (IM state), π_y (SI, II states), π_x (ID state)

From/To	SS	SI	SW	IM	ID	II	IW	EE
SS	.	β	$1 - \beta$
SI	.	β	$1 - \beta$
SW	.	.	.	$\kappa\alpha$	$\kappa(1 - \alpha)$.	.	$1 - \kappa$
IM	β	$1 - \beta$.
ID	γ	$1 - \gamma$.
II	β	$1 - \beta$.
IW	.	.	.	$\kappa\alpha$	$\kappa(1 - \alpha)$.	.	$1 - \kappa$
EE

- Here, in constructing transitions for the expanded transducer, we have used the general rule: “*update the rightmost transducer that is not in a WAIT state; if this transducer outputs a symbol, then recursively update any transducers to the right of it that can eat an input symbol coming from the left (where end-of-sequence is counted as a symbol)*”

- * All transitions update one transducer only, except those from $\{\text{SW}, \text{IW}\}$ to $\{\text{IM}, \text{ID}, \text{EE}\}$
 - Note redundancy: can eliminate **SW** and **IW**, collapse **SI** and **II** together...
 - * ...in fact, for this model [TKF91], can collapse DP right down to one variable; see e.g. Miklòset *al*
- Consider also multiplying transducer by itself and summing out missing states
 - Sequences $X \xrightarrow{t_1} Y \xrightarrow{t_2} Z$
 - Markov property: $P(Z|X, t_1 + t_2) = \sum_Y P(Y|X, t_1)P(Z|Y, t_2)$
 - Can write an expanded transducer for $P(Y|X, t_1)P(Z|Y, t_2)$
 - * Joint $XY - YZ$ states: **SS**, **SI**, **SW**, **IM**, **ID**, **II**, **IW**, **WW**, **MM**, **MD**, **MI**, **DW**, **EE**
 - * Exercise: write out transition matrix & emit probabilities

6.3 Multiple alignment with the links model

- Evolutionary HMMs
 - Exhaustive DP (c.f. Hein 2001)
- MCMC and evolutionary HMMs
 - Branch sampling; node sampling; aunt displacement; (parent sampling); sequence sampling
 - Chaining programs together via MCMC; `tkfalign` and propose/accept/reject

6.4 More realistic evolutionary models

- The long indel model
 - Knudsen-Miyamoto transducer
 - Miklòs-Lunter-Holmes transducer

- Short-range context-sensitive substitution models
 - Siepel-Haussler approach: model k -mer as Ω^k -state stochastic process $x_1(t), x_2(t) \dots x_k(t)$
 - * Conditional probability of next aligned pair, given previous $k-1$ aligned pairs: $P(x_k(0), x_k(t) | x_1(0) \dots x_{k-1}(0), x_1(t) \dots x_{k-1}(t))$
 - Lunter-Hein approach: model full Ω^L process. Rate matrix is sum of k -mer terms, some of which commute, some don't. Taylor series for matrix exponential can be factorised and solved by DP.
 - Short-range context-dependence: relevant for protein? Probably not for substitution (eg recent Brenner *et al* study) but maybe for indels (which might look like local duplications)
- Short-range context-sensitive indel models
 - Motivation: many indels appear, empirically, to be miniature local duplications
- Mutation-selection models
 - Description of model
- Rearrangement models: combinatorial explosion in histories, MCMC slow
 - Most authors write the genome in bigger units (e.g. identifiable genes or blocks of synteny, rather than individual nucleotides)
 - Hannenhalli & Pevzner, 1999; Siepel, 2002; Miklòs, Bioinformatics 2003

7 Stochastic grammars for biological sequences

7.1 Overview of transformational grammars

- Overview: HMM profiles, HMM genefinders, SCFGs for RNA, repeats, beta-sheets; Natural Language Processing
- What is a transformational grammar? Formal definition: terminals Ω , nonterminals Φ , transformation rules
 - “Language” = set of strings generated by the grammar
 - “Parser” = computer program to **decide** if a given input string is in the language (returns “true” or “false”)
 - More generally, we’re interested in parsers that compute scores (energies, probabilities) for a given input string
 - These scores are associated with the transformation rules. The grammar is said to be *score-attributed* (Knuth)
- The Chomsky hierarchy of grammars and their associated parsers
 - Regular grammars: finite-state machines (HMMs)
 - Context-free grammars: pushdown automata (SCFGs)
 - * The **parse tree**; the **inside sequence** and **outside sequence**
 - * Chomsky Normal Form; Eddy *et al*’s “RNA Normal Form”
 - * Transformations to & consequent universality of Chomsky Normal Form
 - * History: Panini’s “Ashtadhyayi”: a CFG for Sanskrit (3959 rules (sutras), circa.300-700BC). Bishop Robert Lowth’s “A Short Introduction to English Grammar” (1762). Chomsky’s theory of universal generative grammars (1956 and onwards).
 - Context-sensitive grammars: Turing machines with bounded tape = linear-bounded automata
 - * Big category: low-complexity sequence repeats, tandem repeats, other bounded correlations e.g. pseudoknots
 - * Lempel-Ziv compression algorithm (allowing local stutter) can be viewed as one of these
 - * Limited context sensitivity (e.g. basepair stacking) can be effected using a *lexicalised* SCFG

– Unrestricted grammars: complete Turing machines

- Summary of Chomsky hierarchy (NB regular \subset context-free \subset context-sensitive \subset unrestricted)

Class	Example rule	Automaton	Complexity	Related models	Max-product	Sum-product	EM
Regular	$S \rightarrow xS$	Finite-state	$O(L)$	HMM, GHMM	Viterbi	Forward Backward	Baum-Welch
Context-free	$S \rightarrow TU$	Pushdown	$O(L^3)$	SCFG	CYK	Inside Outside	(Inside-Outside)
Context-sensitive	$ST \rightarrow TS$	Linear-bounded	$O(\Phi ^L)$	TAG, RNRG	-	-	-
Unrestricted	$ST \rightarrow U$	Turing machine	Undecidable	-	-	-	-

Here S, T, U are nonterminals and x is a terminal. “Complexity” refers to the time complexity of parsing a sequence of length L .

7.2 RNA structure

- Why RNA is important in evolution and cell biology
 - RNA world; pre- and post-transcriptional regulation; Crick’s idea of studying simple examples; ribotechnology
- RNA structure terminology: basepairs, stems, loops, pseudoknots, kissing loops
- Terms contributing to the free energy of a folded RNA structure (scor.berkeley.edu)
 - Hydrogen bonding between bases. Canonical and noncanonical pairs.
 - Stacking energies due to overlap of π -orbitals of adjacent planar basepairs
 - * H-bonding and stacking terms can be combined and measured by direct experiment.
 - Unusual configurations: tetraloops, triloops, triple-A platforms
 - * Finite number of cases, so also amenable to experimental measurement.
 - Entropic cost of closing loops
 - * Theory: rods and Gaussian springs. Integrate out displacements, get likelihood ratio (Doi-Edwards, Isambert)
 - * Statistics of random walk, $\langle |\Delta \mathbf{x}|^2 \rangle \propto t$, and self-avoiding walk, $\langle |\Delta \mathbf{x}|^2 \rangle \propto t^{1+\epsilon}$

- Renormalisation (Edwards, de Gennes)
 - * Empirical scaling laws fit experimental measurements
- Ligands: solvation, metal ions, small molecules
 - * General rules not yet known. Specific small-molecule binding requires conserved motifs (riboswitches).
- Unlike proteins (where amino acid sidechains make multiple contacts), a “**basepair stacking + convex loop penalty**” picture of the free energy seems reasonably accurate as a first approximation (Zuker, Turner, Mathews...)
- The Nussinov algorithm: Finds *strictly nested* foldback structures, i.e. excluding pseudoknots.
 - RNA sequence X , length L , nucleotides $x_1 \dots x_L$
 - Let $H(x, x') = 1$ if xx' is a canonical Watson-Crick basepair, and 0 otherwise
 - Nussinov recursion finds the structure for $x_i \dots x_j$ that has the most strictly nested canonical basepairs

$$S(i, j) = \max \left(S(i+1, j), S(i, j-1), S(i+1, j-1) + H(x_i, x_j), \max_{i \leq k \leq j} (S(i, k) + S(k, j)) \right)$$

Best structure for X is found by traceback from $S(1, L)$

	Rule	Score
$S \rightarrow$	$S x$	0
	$x S$	0
	$x S x'$	$H(x, x')$
	$S S$	0
	ϵ	0

– Equivalent to the following **score-attributed grammar**

- If we allow $H(x, x')$ to be the free energy of basepair formation for xx' (and assume a zero energy cost for any structural feature except basepairs), then this becomes an **energy-attributed grammar**. Same recursion for S now calculates “ground state” energy for a very simple energy model ignoring all but hydrogen bonding terms for strictly nested basepairs. The partition function for this highly simplified model is

$$Z(i, j) = Z(i+1, j) + Z(i, j-1) + Z(i+1, j-1) \exp(-\beta H(x_i, x_j)) + \sum_{i \leq k \leq j} Z(i, k) Z(k, j)$$

where $\beta = 1/kT$ is an “inverse temperature”. NB: have simply changed $H \rightarrow \exp(-\beta H)$, $+ \rightarrow \times$ and $\max \rightarrow +$ in equation for S

- We can also have **probability-attributed grammars** or **stochastic grammars**

	Score	Form
- Relationship between scoring schemes:	Bayes	$P(x)$
	Shannon	$h(x) = -\log_2 P(x)$
	Boltzmann	$E(x) \sim -\log_e P(x)$

Specifically the Boltzmann probability uses a scaling factor (inverse temperature) and a partition function, $P(x) = \frac{1}{Z} \exp[-\beta E(x)]$, where $Z = \sum_x \exp[-\beta E(x)]$.

- Zuker’s energy-attributed grammar. All basepairs xy are represented by parse subtrees rooted in the “lexicalized” state V_{xy} . (“Lexicalized” means there are $4 \times 4 = 16$ of these states, for the 16 possible basepairs. However, we don’t have to store all 16 of these possibilities when doing DP on a given sequence X . Why? For any given inside subsequence $X_i \dots X_j$, the only accessible states are W and $V_{X_i X_j}$.)

LHS	RHS	Energy
W	$\rightarrow W x$	0
	$ x W$	0
	$ x V_{xy} y$	$\alpha(x, y)$
	$ W W$	0
	$ x y$	$\alpha(x, y)$
	$ \epsilon$	0
V_{ab}	$\rightarrow z^n$	$h(n)$
	$ x V_{xy} y$	$\alpha_S(x, y a, b)$
	$ x V_{xy} y z^n$	$\alpha(x, y) + b(n)$
	$ z^n x V_{xy} y$	$\alpha(x, y) + b(n)$
	$ z^m x V_{xy} y z^n$	$\alpha(x, y) + i(m + n)$
	$ x V_{xy} y W x' V_{x'y'} y'$	$\alpha(x, y) + \alpha(x', y')$
	$ \epsilon$	0

Here x, y, x', y', z_i are terminals (nucleotides), and z^n is shorthand for an n -nucleotide emission $z_1 \dots z_n$.

The scoring scheme is as follows:

Free energy term	Meaning
$\alpha(x, y)$	Energy of basepair xy (at the beginning of a stem, i.e. stacked on nothing)
$\alpha_S(x, y a, b)$	Energy of basepair xy stacked on top of basepair ab
$h(n)$	Hairpin loop enclosing n bases
$b(n)$	Asymmetric bulge of n bases
$i(n)$	Interior loop (symmetric bulge) with total of n bases

This scoring scheme was historically formulated in terms of Sankoff's "k-loop decomposition".

NB Zuker's algorithm (grammar), like Nussinov's, excludes pseudoknots.

Strictly, Zuker described the algorithm that finds the lowest-energy parse using the above grammar (CYK). McCaskill described the algorithm for calculating the partition function (Inside) and thus the posterior probabilities of individual basepairs (Outside).

7.3 Dynamic programming algorithms for SCFGs

- Chomsky normal form. (RNA normal form is more useful in practise, but CNF is easier to present.)

Rule	Name
$A \rightarrow BC$	Bifurcation
$A \rightarrow a$	Emission
$A \rightarrow \epsilon$	Termination

For nonterminals $A, B, C \in \Phi$ and terminals $a \in \Omega$:

Probabilities denoted by $P(\text{rule})$, e.g. $P(A \rightarrow BC)$

- Inside algorithm.

Let $I_A(i, k) = P(x_i \dots x_{i+k} | A)$ be sum of probabilities for parse trees rooted in A generating sequence $x_i \dots x_{i+k}$.

$$I_A(i, k) = \left(\sum_B \sum_C \sum_{j=0}^k P(A \rightarrow BC) I_B(i, j) I_C(i + j, k - j) \right) + \begin{cases} 0 & \text{if } k > 1 \\ P(A \rightarrow x_i) & \text{if } k = 1 \\ P(A \rightarrow \epsilon) & \text{if } k = 0 \end{cases}$$

NB loopy dependencies, e.g. if $P(A \rightarrow AA) \neq 0$ and $P(A \rightarrow \epsilon) \neq 0$. It's common to try to avoid these when designing the grammar.

- Cocke-Younger-Kasami (CYK) algorithm.

Let $Y_A(i, k)$ be probability of ML parse tree rooted in A generating sequence $x_i \dots x_{i+k}$.

$$Y_A(i, k) = \max \left(\max_B \max_C \max_{j=0}^k P(A \rightarrow BC) Y_B(i, j) Y_C(i + j, k - j), \begin{cases} 0 & \text{if } k > 1 \\ P(A \rightarrow x_i) & \text{if } k = 1 \\ P(A \rightarrow \epsilon) & \text{if } k = 0 \end{cases} \right)$$

NB similar to Nussinov.

- Outside algorithm.

Let $O_A(i, k) = P(x_1 \dots x_{i-1}, A, x_{i+k+1} \dots x_L | S)$ be sum of probabilities for incomplete parse trees rooted in S , ending in A and generating outside sequence $x_1 \dots x_{i-1}$ and $x_{i+k+1} \dots x_L$.

$$O_A(i, k) = \sum_B \sum_C \left(\sum_{j=0}^i O_C(i - j, j + k) P(C \rightarrow BA) I_B(i - j, j) + \sum_{j=0}^{L-i-k} O_C(i, j + k) P(C \rightarrow AB) I_B(i + k, j) \right)$$

with boundary condition $O_A(0, L) = \delta(A = S)$.

- Inside-Outside and posterior probabilities.

Posterior probability that parse tree contains a subtree with inside sequence $x_i \dots x_{i+k}$ rooted in state A :

$$P(A_{i,i+k} | X) = \frac{O_A(i, k) I_A(i, k)}{I_S(0, L)}$$

Posterior probability of bifurcation $A_{i,j+k} \rightarrow B_{i,j} C_{i+j,k}$:

$$P(A_{i,j+k} \rightarrow B_{i,j} C_{i+j,k} | X) = \frac{O_A(i, j + k) P(A \rightarrow BC) I_B(i, j) I_C(i + j, k)}{I_S(0, L)}$$

etc.

- Parameter estimation: we can write the parse tree likelihood as $\prod_i \theta_i^{n_i}$, where θ_i is the probability of rule i and n_i is the number of times it was applied. As with HMMs, the EM algorithm for SCFGs thus involves computing posterior expectations $\langle n_i \rangle$ for these counts, and setting $\theta_i \propto \langle n_i \rangle$. The posterior expectations are computed using the Inside-Outside algorithm, as shown for rules of the form $P(A \rightarrow BC)$.
- There is a KYC-like analogue to Outside, that can be used to find ML parse tree including a particular subsequence.
- Implementation issues: time complexity is $O(|\Phi|^3 L^3)$, memory is $O(|\Phi| L^2)$.
 - Cubic factors in time complexity arise due to bifurcations, so minimize number of bifurcations for max efficiency (c.f. RNA normal form).

7.4 Pair SCFGs, evolutionary SCFGs and tree transducers

- Evolutionary SCFGs: PFOLD (xfold, evofold, etc.)
 - As with Evolutionary HMMs, we can let the terminals be alignment columns
 - Again, terminal emission likelihood $P(A \rightarrow a)$ is implemented as Felsenstein pruning
- Pair SCFGs: Evoldoer. Version of TKF that describes evolution of RNA secondary structure (Holmes 2005).
 - Two kinds of TKF91 links model, recursively nested in a tree
 - Stem sequences rooted in S nonterminals; basepair alphabet Ω^2 ; ends in an L
 - Loop sequences rooted in L nonterminals; nucleotide alphabet Ω ; S 's also allowed in sequence
 - * Really need to adapt TKF91 model to allow deletion prob to depend on symbol, otherwise S substructures get deleted at same rate as nucleotides
- Pair SCFGs (e.g. Stemloc, QRNA). Heuristic, but with lots of go-faster stripes (pre-aligning & pre-folding sequences).
- Rules for composing Pair SCFGs exist (c.f. string transducers; can view conditionally normalized pair SCFGs as “tree transducers”).

7.5 Graph grammars

- Tree-adjoining grammars
 - Aravind Joshi (1975, 1985)
- Rivas-Eddy papers
 - A dynamic programming algorithm for RNA structure prediction including pseudoknots. JMB 1999.
 - The language of RNA: a formal grammar that includes pseudoknots. Bioinformatics 2000.
- Graph grammars: easy to describe and simulate, attractive for biology; but how does their DP work?
- Other grammars whose marginals are easy to compute by sum-product DP, e.g.
 - stochastic tree grammars (Abe and Mamitsuka, ISMB 1994)
 - context-sensitive HMMs with finite memory of last N emitted characters (Yoon and Vaidyanathan, 2004)

7.6 Discriminative grammars: conditional log-linear models

- Recall HMMs and linear CRFs form a “generative-discriminative pair”
- The analogous discriminative model for SCFGs is a Conditional Log-Linear Model
- This allows a great deal of physics-like parameterization (loop entropies, stacking free energies, terminal mismatch...) without having to introduce many new nonterminals
 - Do, Woods and Batzoglou. “CONTRAFold: RNA secondary structure prediction without physics-based models.” Bioinformatics 22:14, pp e90-e98

8 Continuous random variables

8.1 Review: properties of Gaussian distributions

- Review of salient facts about Gaussian distributions (Gardiner p36-37)

– Multivariate Gaussian: if \mathbf{x} is a vector of n Gaussian r.v.s, then

$$P(\mathbf{x}) = [2\pi \det(\sigma)]^{-1/2} \exp\left(-\frac{1}{2}(\mathbf{x} - \bar{\mathbf{x}})^T \sigma^{-1}(\mathbf{x} - \bar{\mathbf{x}})\right)$$

where $\bar{\mathbf{x}}$ is the mean and σ is the (symmetric) covariance matrix.

– Characteristic function

$$\phi(\mathbf{s}) = \langle \exp(i\mathbf{s}^T \mathbf{x}) \rangle = \exp(i\mathbf{s}^T \bar{\mathbf{x}} - \frac{1}{2}\mathbf{s}^T \sigma \mathbf{s})$$

– General formulae for moments when $\bar{\mathbf{x}} = 0$: odd moments are zero, higher moments satisfy

$$\langle x_i x_j x_k \dots \rangle = \frac{2N!}{N! 2^N} \{\sigma_{ij} \sigma_{kl} \sigma_{mn} \dots\}_{\text{sym}}$$

where “sym” means the symmetrized form of the product of σ 's, and $2N$ is the order of the moment, e.g.

$$\begin{aligned} \langle x_i x_j \rangle &= \sigma_{ij} \\ \langle x_1 x_2 x_3 x_4 \rangle &= \frac{4!}{2! 2^2} \left\{ \frac{1}{3} [\sigma_{12} \sigma_{34} + \sigma_{13} \sigma_{24} + \sigma_{14} \sigma_{23}] \right\} \\ &= \sigma_{12} \sigma_{34} + \sigma_{13} \sigma_{24} + \sigma_{14} \sigma_{23} \\ \langle x_i^4 \rangle &= 3\sigma_{ii}^2 \end{aligned}$$

– Central limit theorem (van Kampen p26): consider arbitrary $P_X(x)$ with $\langle x \rangle = 0$, $\langle x^2 \rangle = \sigma$ and let $z = n^{-1/2} \sum_n x_n$

* Characteristic function for P_X is

$$G_X(k) = \int \exp(ikx) P_X(x) dx = 1 - \frac{1}{2} k^2 \sigma + O(k^4)$$

Thus characteristic function for P_Z is

$$G_Z(k) = \left[G_X \left(\frac{k}{\sqrt{n}} \right) \right]^n = \left[1 - \frac{\sigma k^2}{2r} + O \left(\frac{k^4}{r^{3/2}} \right) \right]^n \rightarrow \exp \left(-\frac{1}{2} \sigma k^2 \right)$$

(using the limit $\lim_{n \rightarrow \infty} (1 + y/n)^{-n} = \exp(-y)$).

- * Therefore, in the limit $n \rightarrow \infty$, z is Gaussian-distributed.
- Normal-gamma posterior for Gaussian mean/precision: see section 1.3
- Joint models for sequence and expression data
 - * Barash & Friedman

8.2 Gaussian processes as stochastic processes

- Definition of a Gaussian process (van Kampen p63-64)
 - “Hierarchy of Distribution Functions” (van Kampen p61+). Consider timepoints $t_1 < t_2 < t_3 \dots t_n$. Define

$$P_n(x_1, t_1; x_2, t_2; \dots; x_n, t_n) \equiv P(x(t_1) = x_1, x(t_2) = x_2, \dots, x(t_n) = x_n)$$
 - If P_n is an n -dimensional Gaussian $\forall n, \{t_1 \dots t_n\}$, then $x(t)$ is a *Gaussian process*
 - The covariance matrix is $\sigma_{ij} = \langle x(t_i)x(t_j) \rangle$
 - Marginals of a multivariate Gaussian are themselves multivariate Gaussians. The full distribution $P(x(t))$ can be thought of as an infinite-dimensional Gaussian, P_∞
 - A Gaussian process is effectively a prior over functions, that can be fully specified by the covariance function
- The *characteristic functional*, $G([k])$, plays a role analogous to the characteristic function for discrete processes. Define an arbitrary auxiliary test function, $k(t)$. Then $G([k])$ is the following functional of $k(t)$

$$G([k]) = \langle \exp \left[i \int_{-\infty}^{\infty} k(t)x(t)dt \right] \rangle = \exp \left[i \int k(t_1)\langle x(t_1) \rangle dt_1 - \frac{1}{2} \int \int k(t_1)k(t_2)\langle \langle x(t_1)x(t_2) \rangle \rangle dt_1 dt_2 \right]$$

8.3 Gaussian processes as tools for machine learning

- Inference, prediction, clustering with GPs (MacKay chapter 45, p535-548; MacKay 1998, “Introduction to Gaussian Processes”)
 - Suppose we have N datapoints, $\{\mathbf{x}^{(n)}, t_n\}_{n=1}^N$. The input variables $\mathbf{x}^{(n)}$ are I -dimensional vectors. The target variables t_n will be assumed real scalars (corresponding to interpolation or regression problems).
 - Goal: fit some (nonlinear) function $y(\mathbf{x})$. Posterior probability of $y(\mathbf{x})$ is

$$P(y(\mathbf{x})|\mathbf{t}_N, \mathbf{X}_N) = \frac{P(\mathbf{t}_N|y(\mathbf{x}), \mathbf{X}_N)P(y(\mathbf{x}))}{P(\mathbf{t}_N|\mathbf{X}_N)}$$

(Typically $P(\mathbf{t}_N|y(\mathbf{x}), \mathbf{X}_N)$ is assumed to be separable Gaussian noise.) In parametric approaches, $y(\mathbf{x}) \equiv y(\mathbf{x}; \mathbf{w})$ where \mathbf{w} is a set of parameters over which we place some prior. In nonparametric approaches (e.g. Gaussian processes), we place a prior directly on $P(y(\mathbf{x}))$.

- A Gaussian process can be defined as a probability distribution over functions, $P(y(\mathbf{x}))$, of the form

$$P(y(\mathbf{x})|\mu(\mathbf{x}), \mathbf{A}) = \frac{1}{Z} \exp \left[-\frac{1}{2} (y(\mathbf{x}) - \mu(\mathbf{x}))^T \mathbf{A} (y(\mathbf{x}) - \mu(\mathbf{x})) \right]$$

where \mathbf{A} is a linear operator and the inner product of two functions is

$$y(\mathbf{x})^T z(\mathbf{x}) = \int y(\mathbf{x})z(\mathbf{x})d\mathbf{x}$$

The operator \mathbf{A} must be *positive definite*, i.e. $y(\mathbf{x})^T \mathbf{A} y(\mathbf{x}) > 0$ for all functions except $y(\mathbf{x}) = 0$.

- Parametric approaches; fixed, adaptive basis functions; neural nets (MacKay p536-537)
 - Consider a set of basis functions, $\{\phi_h(\mathbf{x})\}_{h=1}^H$.

– Case #1: fixed basis functions (parameters independent of \mathbf{w})

$$y(\mathbf{x}; \mathbf{w}) = \sum_{h=1}^H w_h \phi_h(\mathbf{x})$$

e.g. radial basis functions

$$\phi_h(\mathbf{x}) = \exp \left[-\frac{(\mathbf{x} - \mathbf{c}_h)^2}{2r^2} \right]$$

In this model, y is a linear function of \mathbf{w} .

- * Let $R_{nh} = \phi_h(\mathbf{x}^{(n)})$. Then $y^{(n)} = \sum_h R_{nh} w_h$. Let $\mathbf{y} = (y^{(1)}, y^{(2)} \dots y^{(N)})$ be the vector of y -values and let $\mathbf{t} = (t^{(1)}, t^{(2)} \dots t^{(N)})$ be the vector of corresponding t -values. Thus $\mathbf{y} = \mathbf{R}\mathbf{w}$.
- * If \mathbf{w} is Gaussian-distributed

$$P(\mathbf{w}) = \mathcal{N}(\mathbf{0}, \sigma_w^2 \mathbf{I})$$

then \mathbf{y} is also Gaussian with covariance matrix

$$\langle \mathbf{y}\mathbf{y}^T \rangle = \langle \mathbf{y}\mathbf{y}^T \rangle = \langle \mathbf{R}\mathbf{w}\mathbf{w}^T \mathbf{R}^T \rangle = \mathbf{R} \langle \mathbf{w}\mathbf{w}^T \rangle \mathbf{R}^T = \sigma_w^2 \langle \mathbf{R}\mathbf{R}^T \rangle$$

- * Additive noise: if $\mathbf{t} = \mathbf{y} + \mathbf{v}$ where $v_k \sim \mathcal{N}(0, \sigma_v^2)$ then

$$P(\mathbf{w}) = \mathcal{N}(\mathbf{0}, \sigma_w^2 \mathbf{R}\mathbf{R}^T + \sigma_v^2 \mathbf{I})$$

– Case #2: adaptive basis functions (parameters dependent on \mathbf{w})

$$y(\mathbf{x}; \mathbf{w}) = \sum_{h=1}^H w_h^{(2)} \tanh \left(\sum_{i=1}^I w_{hi}^{(1)} x_i + w_{h0}^{(1)} \right) + w_0^{(2)}$$

This is equivalent to a two-layer feedforward neural network with nonlinear hidden units and a linear output. The input weights are $\{w_{hi}^{(1)}\}$, the hidden unit biases $\{w_{h0}^{(1)}\}$, the output weights $\{w_h^{(2)}\}$ and the output bias $w_0^{(2)}$. In this model, y is a nonlinear function of \mathbf{w} .

- Nonparametric approaches: the spline smoothing method (MacKay p538-541) attempts to minimize the functional

$$M(y(x)) = \frac{1}{2}\beta \sum_{n=1}^N (y(x^{(n)}) - t_n)^2 + \frac{1}{2}\alpha \int \left[\frac{d^k y}{dx^k} \right]^2 dx$$

(If $k = 2$ then $y = \operatorname{argmin} M$ is a *cubic spline* with discontinuities in $\frac{d^2 y}{dx^2}$ at the $x^{(n)}$.) The term involving α is equivalent to the following prior over $y(x)$

$$P(y(x)|\alpha) = \text{const.} \times \exp \left(-\frac{1}{2}\alpha \int \left[\frac{d^k y}{dx^k} \right]^2 dx \right)$$

which is a Gaussian process prior with $\mathbf{A} = [D^k]^T D^k$. Combined with linearly independent Gaussian noise on each measurement, this gives a Gaussian process model with MAP estimates identical to those produced by splines.

To show the equivalence between splines and parametric models explicitly, we can develop Fourier-series basis functions for splines. Suppose the range of x -values of interest is within $[0, 2\pi]$ (the x -axis can be rescaled if necessary). Keeping it real, write

$$\begin{aligned} y(x) &= \sum_{h=0}^{\infty} c_h \cos(hx) + \sum_{h=1}^{\infty} s_h \sin(hx) \\ \frac{d^k y}{dx^k} &= \begin{cases} \sum_{h=1}^{\infty} c_h h^k (-1)^{k/2} \cos(hx) + \sum_{h=1}^{\infty} s_h h^k (-1)^{k/2} \sin(hx) & (k \text{ even}) \\ \sum_{h=1}^{\infty} c_h h^k (-1)^{(k+1)/2} \sin(hx) + \sum_{h=1}^{\infty} s_h h^k (-1)^{(k-1)/2} \cos(hx) & (k \text{ odd}) \end{cases} \\ P(y(x)|\alpha) &\sim \exp \left(-\frac{1}{2}\alpha \int_0^{2\pi} \left[\frac{d^k y}{dx^k} \right]^2 dx \right) \\ &\sim \exp \left(-\frac{1}{2}\alpha \times 2\pi \sum_{h=1}^{\infty} h^{2k} (c_h^2 + s_h^2) \right) \end{aligned}$$

which is a Gaussian prior on the $\{c_h, s_h\}$. (NB MacKay (p539) has a different, incompatible form for this prior, with $h^{k/2}$ instead of $2\pi h^{2k}$; not sure where the error is, but the fundamental point (it's Gaussian) is unaffected.)

–

- Gaussian process regression (MacKay p542-543)
- Example covariance functions (MacKay p543-545; Abrahamsen, 1997); Wiener, Ornstein-Uhlenbeck
- Adaptive inference: learning the hyperparameters of the covariance function (MacKay p545-546)
- Gaussian process classifiers (MacKay p547)

8.4 The Fokker-Planck equation

- Kramers-Moyal expansion (treatment follows van Kampen p197-198; see also Gillespie p74+)
 - The most general form of the *master equation* for a continuous-time stochastic process can be written

$$\frac{\partial}{\partial t}p(x, t) = \int W(x - r; r)p(x - r, t)dr - p(x, t) \int W(x; r)dr$$

where $W(x; r)$ is the rate from x to $x + r$. In the notation we used for discrete state spaces, $W(x; r) \equiv R_{x, x+r}$

- Assuming that $W(x; r)$ varies smoothly in x and is sharply peaked in r , we can write the term $W(x - r; r)p(x - r, t)$ in the first integral as a Taylor expansion in x :

$$\frac{\partial}{\partial t}p(x, t) = \sum_{n=0}^{\infty} \int \frac{(-r)^n}{n!} \frac{\partial^n}{\partial x^n} \{W(x; r)p(x, t)\} dr - p(x, t) \int W(x; r)dr$$

(Note that we're only allowed to expand $W(x; r)$ in x , not in r , since it varies smoothly in x but rapidly in r .)

- We then rewrite the terms in the expansion using the *jump moments*

$$a_n(x) = \int_{-\infty}^{\infty} r^n W(x; r)dr$$

so that the master equation becomes the Kramers-Moyal equation

$$\begin{aligned}\frac{\partial}{\partial t}p(x, t) &= \sum_{n=0}^{\infty} \frac{(-1)^n}{n!} \frac{\partial^n}{\partial x^n} \{a_n(x)p(x, t)\} - p(x, t) \int W(x; r)dr \\ &= \sum_{n=1}^{\infty} \frac{(-1)^n}{n!} \frac{\partial^n}{\partial x^n} \{a_n(x)p(x, t)\}\end{aligned}$$

– Truncating the Taylor expansion to second order gives

$$\frac{\partial}{\partial t}p(x, t) = -\frac{\partial}{\partial x} \{a_1(x)p(x, t)\} + \frac{1}{2} \frac{\partial^2}{\partial x^2} \{a_2(x)p(x, t)\}$$

which is a form of the Fokker-Planck equation; see below.

– Consider the discrete-time process x_n where $t = n\tau$. We have

$$x_{n+1} = x_n + \Xi_n$$

where the Ξ_n are random variables distributed $\sim W(x_n; \Xi)\tau$. Whatever the precise form of $W(x; r)$, we're effectively assuming that we can characterize it (and hence Ξ_n) by its first two moments, a_1 and a_2 . Since $x_n = \sum \Xi_n$, the process x_n and hence $x(t)$ tends towards a Gaussian, by the central limit theorem.

– Gillespie uses different terminology: the continuous-time version of what we have called Ξ_n is the “propagator” and is written explicitly as a function of dt , i.e. $\Xi(dt; x, t)$; $W(x; r)$ is the “propagator density function” and is written $\Pi(r|dt; x, t)$ (Gillespie p67); and the $a_n(x)$ are the *propagator moment functions* and are written B_n (Gillespie p68). Gillespie makes the argument that the propagator density function is a Gaussian to first order in dt (Gillespie p114-115).

• Fokker-Planck equation (Gillespie p121; van Kampen p193+)

– Fokker-Planck describes the time evolution of the probability density for a continuous stochastic process

$$\frac{\partial}{\partial t}p(x, t) = -\frac{\partial}{\partial x}A(x, t)p(x, t) + \frac{1}{2} \frac{\partial^2}{\partial x^2}B(x, t)p(x, t)$$

- By comparison with the Kramers-Moyal expansion we see that $A = a_1$ and $B = a_2$, so A and B are the mean and variance of the drift (i.e. the jump rate $W(x; r)$).
 - * When $B = 0$, we have a (deterministic) Liouville process.
 - * When $A = 0$ and B is constant, we have Brownian motion, aka the Wiener process.
 - * When $A = -kx$ and B is constant, we have Brownian motion with exponential decay, aka the Ornstein-Uhlenbeck process.
- Note that the terms $A(x, t)$ and $B(x, t)$ are time-dependent, unlike our earlier treatment of the Kramers-Moyal expansion. This is just because we didn't allow the jump rate $W(x; r)$ to be a function of t . It's straightforward to repeat the Kramers-Moyal expansion using time-dependent jump rates and moments, $W(x; r; t)$ and $a_n(x; t)$.

- Moment evolution equations (Gillespie p81-87)

- Using the (time-dependent) propagator we have $x(t + dt) = x(t) + \Xi(dt; x(t), t)$ and hence

$$x^n(t + dt) = [x(t) + \Xi(dt; x(t), t)]^n = x^n(t) + \sum_{k=1}^n {}^n C_k \times x^{n-k}(t) \Xi^k(dt; x(t), t)$$

To find the expectation of this we use the following result (see Gillespie p68,82-83 for justification)

$$\langle x^j(t) \Xi^k(dt; x(t), t) \rangle = \langle x^j(t) b_k(t) \rangle dt + O(dt) \quad (1)$$

(where $b_k(t) \equiv a_k(x(t))$ is just an alternate notation for the k 'th jump moment). Hence we arrive at the *moment evolution equations*

$$\frac{d}{dt} \langle x^n(t) \rangle = \sum_{k=1}^n {}^n C_k \langle x^{n-k}(t) b_k(t) \rangle$$

with the initial conditions $\langle x^n(0) \rangle = x_0^n$.

- A further application of (1) gives the evolution of the autocorrelation function

$$\begin{aligned} \langle x(t_1) x(t_2 + dt_2) \rangle &= \langle x(t_1) x(t_2) + x(t_1) \Xi(dt; x(t_2), t_2) \rangle \\ &= \langle x(t_1) x(t_2) \rangle + \langle x(t_1) b_1(t_2) \rangle \\ \frac{d}{dt_2} \langle x(t_1) x(t_2) \rangle &= \langle x(t_1) b_1(t_2) \rangle \end{aligned}$$

- Putting these together, we obtain the following equations for the evolution of the mean, variance and covariance

$$\begin{aligned}\frac{d}{dt}\langle x(t) \rangle &= \langle b_1(t) \rangle \\ \frac{d}{dt}\text{var}\{x(t)\} &= \frac{d}{dt} (\langle x^2(t) \rangle - \langle x(t) \rangle^2) \\ &= 2(\langle x(t)b_1(t) \rangle - \langle x(t) \rangle \langle b_1(t) \rangle) + \langle b_2(t) \rangle \\ \frac{d}{dt_2}\text{cov}\{x(t_1)x(t_2)\} &= \frac{d}{dt_2} (\langle x(t_1)x(t_2) \rangle - \langle x(t_1) \rangle \langle x(t_2) \rangle) \\ &= \langle x(t_1)b_1(t_2) \rangle - \langle x(t_1) \rangle \langle b_1(t_2) \rangle\end{aligned}$$

with the initial conditions

$$\begin{aligned}\langle x(t_0) \rangle &= x_0 \\ \text{var}\{x(t_0)\} &= 0 \\ \text{cov}\{x(t_1)x(t_2 = t_1)\} &= \text{var}\{x(t_1)\}\end{aligned}$$

These equations are closed iff (for $n = 1, 2$) $b_n(t) = a_n(x(t))$ is a polynomial in x of degree $\leq n$ (Gillespie p86).

- The covariance is all we need to do inference with Gaussian processes, but it's useful to look deeper
- Liouville processes (Gillespie p126)

- When $B(x, t) = 0$, the jump rate $W(x; r)$ has no variance. It must therefore be a delta function in r , and we can write

$$x(t + dt) - x(t) = A(x(t), t)dt$$

that is, $\frac{d}{dt} = A(x, t)$.

- This is a completely deterministic process, or *Liouville process*.
- The Liouville process guides intuition as to the role of $A(x, t)$ in the Fokker-Planck equation.

8.5 The Wiener process

- The Wiener process (undamped Brownian motion, diffusive drift, limit of random walk...)
 - Derivation of Fick's equations for one-dimensional diffusion (Berg, "Random Walks in Biology", p18-20)
 - * Discrete random walk: $x(n) = \sum_{i=1}^n d_i$ where $P(d_i = +\delta) = P(d_i = -\delta) = 1/2$
 - Implies that $\langle x(n) \rangle = 0$ and $\langle x(n)^2 \rangle = n\delta^2$
 - If each step takes time τ then $n = t/\tau$, so $\langle x(n)^2 \rangle = \frac{\delta^2}{\tau} t = 2Dt$ where $D = \delta^2/2\tau$ is the diffusion constant
 - Let $r(x, t) = P(x(t) = x)$. In time τ , a particle at x has probability 1/2 of drifting to $x + \delta$, and a particle at $x + \delta$ has probability 1/2 of drifting to x . The net flux of probability mass from x to $x + \delta$ is

$$J(x) = \frac{1}{\tau} \left(\frac{r(x, t)}{2} - \frac{r(x + \delta, t)}{2} \right) = D \frac{1}{\delta} \left(\frac{r(x, t)}{\delta} - \frac{r(x + \delta, t)}{\delta} \right)$$

The continuous limit of $r(x, t)/\delta$ is the probability density $p(x, t)$ and so

$$J(x) = -D \frac{\partial p}{\partial x}$$

This is a version of *Fick's first equation* (Berg p18).

Consider the interval from x to $x + \delta$. Flux from the left is $J(x)$ and from the right $-J(x + \delta)$. Since probability mass is conserved, we have

$$[p(x, t + \tau) - p(x, t)] \times \delta = [J(x) - J(x + \delta)] \times \tau$$

Dividing through by $\delta\tau$, taking the continuous limit and substituting the expression for $J(x)$, we have

$$\frac{\partial p}{\partial t} = D \frac{\partial^2 p}{\partial x^2}$$

This is *Fick's second equation* (Berg p20), aka the master equation (or Fokker-Planck equation) for Brownian motion.

- In the continuous limit ($\tau \rightarrow 0$), $x(t)$ is the sum of a large number ($n = t/\tau$) of IID rvs (the d_i 's). By the central limit theorem, the finite-time increments $x(t_2) - x(t_1)$ must be Gaussian-distributed, so $x(t)$ is a Gaussian process.
 - Furthermore, $x(t_2) - x(t_1)$ is independent of $x(t_3) - x(t_2)$. This is called *independence of increments*.
- Fokker-Planck equation for Wiener process (Gardiner p66-70). Choose units such that $\delta^2 = \tau, D = \frac{1}{2}$. Then

$$\frac{\partial}{\partial t} p(x, t) = \frac{1}{2} \frac{\partial^2}{\partial x^2} p(x, t) \quad p(x, 0) = \delta(x)$$

- Fourier-transforming the Fokker-Planck equation gives for the characteristic function $\phi(s, t) = \langle \exp(isx(t)) \rangle_{x(t)}$

$$\frac{\partial}{\partial t} \phi = -\frac{1}{2} s^2 \phi \quad \phi(s, 0) = 1$$

which has the solution $\phi(s, t) = \exp(-\frac{1}{2} s^2 t)$. The Fourier inversion gives

$$p(x, t) = (2\pi t)^{-1/2} \exp\left(-\frac{x^2}{2t}\right)$$

- Autocorrelation function:

$$\langle x(t_1)x(t_2) \rangle = \langle x(t_1)^2 \rangle + \langle x(t_1)(x(t_2) - x(t_1)) \rangle = t_1$$

since the second expectation vanishes due to independence of increments.

- Non-differentiable/fractal nature of sample trajectories:

$$P(|x(t+\epsilon) - x(t)| > k\epsilon) = 2 \int_{k\epsilon}^{\infty} p(x, \epsilon) dx = 2 \int_{k\epsilon}^{\infty} (2\pi\epsilon)^{-1/2} \exp\left(-\frac{x^2}{2\epsilon}\right) dx$$

In the limit $\epsilon \rightarrow 0$, this is one (intuitively, $|x(\epsilon)|_{\text{rms}} \sim \sqrt{\epsilon}$, so ϵ approaches zero faster than x does). This means that the derivative of $x(t)$ is infinite “almost surely” (in probabilistic terminology).

8.6 The Ornstein-Uhlenbeck process

- The Ornstein-Uhlenbeck process: Brownian motion with exponential decay (van Kampen p83-85)
 - Originally constructed to describe the *velocity* of a Brownian particle (van Kampen p84)
 - Fokker-Planck equation (Gardiner p74-77)

$$\frac{\partial}{\partial t}p(x, t) = \frac{\partial}{\partial x}(kxp(x, t)) + \frac{1}{2}D \frac{\partial^2}{\partial x^2}p(x, t)$$

Boundary condition is $p(x, 0) = \delta(x - x_0)$.

- Characteristic equation for $\phi(s, t) = \langle \exp(\imath sx) \rangle$

$$\frac{\partial}{\partial t}\phi(s, t) + ks \frac{\partial}{\partial s}\phi(s, t) = -\frac{1}{2}Ds^2\phi(s, t) \quad (2)$$

Boundary condition is $\phi(s, 0) = \exp(\imath sx_0)$.

- * Method of characteristics: see Gardiner p75. Let $t = t(r)$ and $s = s(r)$. Then

$$\frac{d\phi}{dr} = \frac{\partial\phi}{\partial t} \frac{dt}{dr} + \frac{\partial\phi}{\partial s} \frac{ds}{dr}$$

which is the same as (2) if

$$dr = \frac{dt}{1} = \frac{ds}{ks} = -\frac{d\phi}{\frac{1}{2}Ds^2}$$

(which Gardiner calls the “subsidiary equation”). Integrating the equation involving ds and dt , and the equation involving ds and $d\phi$, gives

$$s = a \exp(kt) \quad \phi = b \exp(-Ds^2/4k)$$

where a, b are arbitrary constants. This can be rearranged to give the “characteristic directions”

$$\begin{aligned} u(s, t, \phi) &= s \exp(-kt) &= a \\ v(s, t, \phi) &= \phi \exp(Ds^2/4k) &= b \end{aligned}$$

By the subsidiary equation, $du = dv = 0$. Thus a general solution to (2) is $f(u, v) = 0$ with f an arbitrary function, or (equivalently) $v = g(u)$ with g an arbitrary function. Therefore

$$\phi(s, t) = \exp(-Ds^2/4k)g[s \exp(-kt)]$$

The boundary condition $\phi(s, 0) = \exp(\imath sx_0)$ requires that $g(s) = \exp(Ds^2/4k + \imath sx_0)$. Hence

$$\phi(s, t) = \exp\left(-\frac{Ds^2}{4k}(1 - \exp(-2kt)) + \imath sx_0 \exp(-kt)\right)$$

which is the characteristic function of a Gaussian with

$$\begin{aligned}\langle x(t) \rangle &= x_0 \exp(-kt) \\ \text{var}\{x(t)\} &= \frac{D}{2k} (1 - \exp(-2kt))\end{aligned}$$

Thus

$$p(x, t) = \left(\frac{\pi D}{k}(1 - \exp(-2kt))\right)^{-1/2} \exp\left[-\frac{k}{D} \frac{(x - x_0 \exp(-kt))^2}{1 - \exp(-2kt)}\right]$$

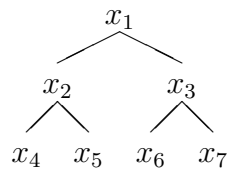
The canonical form has $D = 2k$ (van Kampen p83).

- Doob’s Theorem (van Kampen p84)
- Moments, autocorrelation functions
- Spectral density of fluctuations; Wiener-Khinchin theorem (van Kampen p58-61)
- Fluctuation-Dissipation Theorem (e.g. van Kampen p220-221)
- Eigenfunctions (Gardiner p134)
- Langevin/stochastic differential equation (Gardiner p106-107; van Kampen p219-221)
 - * Ito vs Stratonovich interpretations (“Ito-Stratonovich dilemma”: van Kampen p232-237)
 - * Ito’s formula; corollaries

- * Formulation of OU process as SDE
 - Solution of OU SDE by substitution
- * Simulation of Langevin equations (Gardiner chapter 10, p373-392)
- Case study of an Ornstein-Uhlenbeck process in stochastic systems biology: the enzyme futile cycle
 - Samoilov M, Plyasunov S, Arkin AP. Stochastic amplification and signaling in enzymatic futile cycles through noise-induced bistability with oscillations. Proc Natl Acad Sci U S A. 2005 Feb 15;102(7):2310-5.
- Multivariate Ornstein-Uhlenbeck process (Gardiner p109-112)
- Case study of inference using a multivariate OU process: relationship between CD4 and beta-2-microglobulin in AIDS patients
 - Sy JP, Taylor JM, Cumberland WG. A stochastic model for the analysis of bivariate longitudinal AIDS data. Biometrics. 1997 Jun;53(2):542-55.

8.7 Phylogenetically related Brownian variables

- Felsenstein, chapter 23 (p391-414)
- Consider tree



where x_n are Brownian variables.

- For a (parent,child) pair (p, c) let t_c be distance from p to c and let $d_c = x_c - x_p$. We have $\langle d_c \rangle = 0$ and $\langle d_c^2 \rangle = Dt_c$.
- Covariance matrix
- Pruning algorithm