239 Discrete Mathematics for the Life Sciences

Lior Pachter
Notes by Qiaochu Yuan

Spring 2013

1 Introduction

There are 20 (actually 21) amino acids which are coded from a 4-letter alphabet. This is the largest possible size of a comma-free code on 4 letters, so it was tempting to conclude that amino acids were coded via a comma-free code. However, this is not true. In fact, amino acids are coded by multiple 3-letter words (codons); usually 4, but sometimes $3, 2$, or 1 (as in tryptophan). This genetic code is very nearly universal (there are small variations in a handful of species).

Codons which code for the same amino acid are called synonymous. Synonymous codons are usually the same in the first 2 letters and differ only in the third; this seems like it might be useful because mutations in the third letter don't affect the resulting amino acid.

Codon explorer (http://bmf.coloradu.edu/codonexplorer/) is a tool for describing frequencies of codons in various genomes. It returns, among other things, the Codon Adaptation Index, which roughly speaking describes how close the codons are to being the most common synonymous codon.

The mystery of the two straight lines is the following. If we measure how frequent different nucleotides are in different positions in codons, this correlates linearly very well with GC-content (percentage of bases that are either guanine or cytosine).

How do we fit lines to points in general? One way is the least-squares approximation. (It is easy to do such things with R.) Least-squares can be done by minimizing the sum of the squares of the vertical or horizontal distances (if we want to fit lines to points in a plane). Another is using principal component analysis, which minimizes perpendicular distances to the data points instead. One way to interpret the PCA line is that, after projecting all the points onto the line, it maximizes the remaining variance in the data.

The human genome consists of 2.8 billion sites and there are 6.8 billion humans. This can be combined into an enormous matrix on which it would then be possible to perform PCA. Part of this matrix has been collected (for Europeans) and the two largest principal components have been plotted; the result agrees very well with the geography of Europe.

Codons not only determine what amino acids are made but how many. This was tested using a synthetic library of GFP (green fluorescent protein) genes, whose codons were randomly replaced by synonyms and then let loose to observe how often the corresponding proteins were produced. It turns out that replacing codons with synonyms determines how RNA molecules (which are single-stranded) fold in on themselves. Different folding structures have different free energies, and this correlates to fluorescence (hence how much of the protein is created).

Nucleotides can be modified by a process called methylation, which adds a methyl group to either cytosine or adenine. It's known that methylation affects transcription and therefore translation rates. Methylated cytosines also occasionally become thymine.

2 The genome as a random object

Suppose we have a collection of N sequences (of either nucleotides or genes). We suspect that they may have a subsequence in common of length w, and the problem is to determine how likely this is. We don't demand that the subsequences are exactly the same, but only that they are similar. This is relevant to binding site motifs.

Probabilistically, we will think of a genome as randomly determined by probabilities p_i of a nucleotide appearing in a particular position, as well as probabilities $q_{j,A}, q_{j,C}, q_{j,G}, q_{j,T}$ (the position weight matrix or position-specific scoring matrix) where j indexes positions in the motif. These probabilities determine how likely nucleotides are to appear when the genome switches to expressing the motif. This is a vaguely reasonable model because, although mutations occur randomly, binding sites need to continue to exist. (This isn't a great model but a lot of people use it.)

Weblogo is an online resource for visualizing binding sites. There are columns of nucleotides for each binding site, and the height of a given letter, say A, is $q_{i,A}R_i$ where $R_i = \log_2 4 - (H_i + e_i)$ and where $H_i = -\sum q_{i,A} \log_2 q_{i,A}$ is the Shannon entropy and e_i is a correction for small sample size.

We want an algorithm that will learn the p_i and the $q_{j,\cdot}$, which we do as follows (assuming that we know w). We first choose a random collection of starting positions a_k (where k runs over the N sequences). Next, we choose a random sequence z and learn the p_i and $q_{j,\cdot}$ by counting starting from the a_k in every sequence other than z. Given these learned probabilities, we then compute the probability of every possible starting position a_z in sequence z and pick a random start site in z according to that distribution. We then repeat. This algorithm converges to the correct probabilities and is a special case of the Metropolis-Hastings algorithm.

If w is not known, the problem of determining w is the problem of model selection. It is necessary to penalize for longer w because of the existence of additional parameters (which leads to overfitting), but none of the standard method for doing this appear to work in this case.

One drawback of this algorithm is that it assumes that motifs appear one in each sequence. This is usually false. We can address this by adding new parameters determining how often motifs appear in each sequence.

A variant of the algorithm involves cycling through the sequences rather than choosing them at random.

(Lecture on Metropolis-Hastings and Gibbs sampling not given)

3 Significance

How do you tell if a genome feature is significant?

A *k-mer* is a sequence of k base pairs. We want to compute probabilities like the probabilities of seeing copies of some k-mer in a row given that we are looking at a random genome in such a way that preserves k-mer frequencies. This is important because k-mer frequencies are not uniform in real genomes.

(In a plot where x describes a given k-mer frequency and y describes how many k-mers appear at a given frequency, we expect something like a normal distribution and don't see that; instead the distribution for 11-mers in the human genome has two peaks to the left of the naive mean and has heavy tails.)

One way to model the genome is using Markov chains. Let Σ be a finite alphabet (here it will just be $\{0, 1\}$). Let θ be a set of parameters (here $\theta_{00}, \theta_{01}, \theta_{10}, \theta_{11}$) between 0 and 1; these describe transition probabilities. Let ℓ be a length (here 4). Our Markov model will be

$$p_{ijkl} = \frac{\theta_{ij}\theta_{jk}\theta_{kl}}{z} \tag{1}$$

where

$$z = \sum_{i,j,k.l} \theta_{ij}\theta_{jk}\theta_{kl}. \tag{2}$$

The distributions we get in this way satisfy various relations. For example, $p_{0010} = p_{0100} = p_{1001}$. Less obviously, we have quadratic relations like $p_{0011}^2 = p_{0001}p_{0111}$, and nine additional relations like $p_{0111}p_{1010} = p_{0101}p_{1110}$ which turn out to generate the ideal of all polynomial relations and which describe all probability distributions we get in this way. This is a special case of the Hammersley-Clifford theorem.

The above is an example of a log-linear model. In general a log-linear model takes the form

$$(\theta_1, ...\theta_d) \mapsto \frac{(\theta^{a_1}, ...\theta^{a_m})}{\sum \theta^{a_j}} \tag{3}$$

where

$$\theta^{a_j} = \prod_{i=1}^{d} \theta_i^{a_{ij}} \tag{4}$$

and a_{ij} is an arbitrary matrix. Markov models are log-linear, and log-linear models have the property that maximum likelihood estimation is easy, which is nice. (This is made precise by Birch's theorem, which will be on the homework.)

Here is an algorithm for generating random sequences from a given sequence in a way that preserves 2-mer counts. The idea will be to randomly swap subwords in a way that preserves 2-mers. If s is a sequence and k a constant, then we denote by $X_k(s)$ the set of sequences of the same length as s with the same k-mer frequencies as s, and the goal is to sample from $X_k(s)$ more or less uniformly.

We will do this by choosing four locations $a < b < c < d$ in the sequence and testing whether it is possible to swap the substring between a and b with the substring between c and d without altering k-mer counts.

We also denote by $Y_k(s)$ the subset of $X_k(s)$ with the same first $k-1$-mer. We see that

$$X_k(s) = \bigcup_{s'} Y_k(s') \tag{5}$$

where s' runs over all cyclic permutations of s. We will sample from $Y_k(s)$ for simplicity. Define a graph $G_k(s)$ whose vertices are the elements of $Y_k(s)$ and whose edges are given by shuffles (if the shuffle is invalid, the edge is a loop). The first question is whether this graph is connected. The second question is what the mixing time of a random walk is on this graph.

Exercise 3.1. *$G_k(s)$ is connected. Its diameter is at most length$(s) - 2k + 2$.*

Hint: let $j(u,v)$ be the first position from the left where the sequences u and v disagree and let $\rho(u,v) = n - k + 2 - j(u,v)$. Then $\rho(u,v) = 0$ if $u = v$.

$G_k(s)$ is regular of degree $\binom{n-k+1}{4}$. We want to perform a random walk on this graph and determine the mixing time. If T is the corresponding transition matrix, the answer depends on the second largest eigenvalue of T.

Conjecturally, for fixed k and alphabet size, there is a polynomial $p(n)$ such that such that for any sequence s of length n and any m, if we perform $mp(n)$ steps, then the final distribution is less than 2^{-m} from uniform in total variation distance.

An alternative approach is the following. Let $D_k(s)$ be the graph whose vertices are the $k-1$-mers in s and whose edges are of the form $s_i \ldots s_{i+k-2} \to s_{i+1} \ldots s_{i+k-1}$. This is a de Brujin graph. It has the advantage over $G_k(s)$ that its size is not exponential in n.

1. A walk on this graph from $s_1 \ldots s_{k-1}$ to $s_{n-k+2} \ldots s_n$ is called an Eulerian trail.

2. Any such walk selects a sequence from $Y_k(s)$.

3. If $f_1, \ldots f_m$ are the frequencies of the various k-mers in s, then a given sequence in $Y_k(s)$ corresponds to $\prod f_i!$ Eulerian trails.

4. Given an Eulerian trail E, for any vertex $\gamma \neq \beta$, let $e(\gamma)$ be the last edge exited from γ in E. The collection of such edges $T(E)$ is acyclic (a forest).

5. Every tree (forest?) rooted at β corresponds to $d^+(\beta) \prod_{\gamma \neq \beta} (d^+(\gamma) - 1)!$ Eulerian trails (where d^+ denotes the outdegree).

6. There is a nice algorithm for picking random trees rooted at β.

4 Genome alignments

There is a genome browser available at genome.ucsc.edu. Among other things, it produces alignments (aligns corresponding sections of different genomes). There are occasionally errors; one place to start looking for them is places where a number of nucleotides is missing that doesn't make up a collection of codons.

Darwin wrote down a tree whose leaves were extant species and whose root is their common ancestor, with intermediate vertices corresponding to intermediate common ancestors. This leads to the notion of a *hierarchy* or a *laminar family*. For a set X, a hierarchy on X is a collection H of subsets of X such that for every $e_1, e_2 \in H$, the intersection $e_1 \cap e_2$ is either empty, e_1, or e_2.

Example $\{\{A\}, \{B\}, \{C\}, \{D\}, \{B, C, D\}, \{A, B, C, D\}\}$ is a hierarchy on $\{A, B, C, D\}$.

We can visualize such a thing using a *rooted X-forest*, which is a rooted forest (a disjoint union of rooted trees) some of whose vertices are labeled by the elements of X (say $\varphi : X \to V$ is the labeling function) and such that all unlabeled vertices have outdegree at least 2.

Exercise 4.1. *There is a natural bijection between hierarchies on X and rooted X-forests.*

(It may be necessary to require that every element of X appears in an element of the hierarchy.)

Let $\sigma^1, ...\sigma^k$ be a collection of genomes, with σ^i_j the j^{th} nucleotide of the genome σ^i. From such a genome we want to construct a *homology forest* describing which nucleotides share a common ancestor.

Define a *partial global alignment* as follows. Given a homology forest on the nucleotides in the σ^i, there is an equivalence relation with m equivalence classes $c_1, ...c_m$ where two nucleotides are in the same component if they are in the same tree. The c_i are also equipped with a partial order where $c_r < c_s$ if $(i, j_1) \in c_r, (i, j_2) \in c_s$ and $j_1 < j_2$ (here (i, j) is the nucleotide σ^i_j). A simple example is the null alignment, where nothing is related to anything else.

Recall that a linear extension or topological order on a partial order is a refinement to a total order. A *global alignment* is a linear extension of the partial order in a partial global alignment.

Exercise 4.2. *The number of partial global alignments of two sequences of length n, m is $\binom{n+m}{n}$.*

Exercise 4.3. *The number of global alignments for two sequences of length n, m is*

$$\sum \binom{m}{i} \binom{n}{i} 2^i. \tag{6}$$

In the special case $n = m$ these are the central Delannoy numbers.

For example, one possible global alignment between the sequences GATTACA and CACACA is

```
G  A  T  T  A  C  A  -  -
-  -  C  -  A  C  A  C  A
```

which has 9 components, the columns, in the given order.

Restrict to the case of two sequences. There is a criterion to decide whether a given global alignment is good. There are three important things:

1. matches, which are $(i_1, j_1) \sim (i_2, j_2)$ such that $\sigma^{i_1}_{j_1} = \sigma^{i_2}_{j_2}$
2. mismatches, which are as above but not equal
3. spaces (sometimes called gaps), which are (i, j) that aren't equivalent to anything.

Note that if the sequences have length n, m, then twice the number of (mismatches plus matches) plus the number of spaces is $n + m$.

To determine a score for a given global alignment we should choose a weight (say -1) describing mismatches and a weight (say -2) describing spaces, then choose an alignment with the highest score. Any such scoring procedure gives an element on the boundary of the convex hull of the space of global alignments (plotted according to their mismatches and spaces).

4.1 Probability distributions on global alignments

We will describe a Markov chain model for two-sequence global alignments. There are states called start, S, end, 1, 2, H (for homology), with a transition from start to S of probability 1, a transition from S to end of probability τ, transitions from S to 1, 2, H of probabilities $\frac{1-\tau}{3}$, and transitions from 1, 2, H to S of probability 1.

In state 1, we print A, C, G, T on top and a dash on the bottom, each with probability $\frac{1}{4}$. In state 2, we print a dash on the top and A, C, G, T on bottom, eahc with probability $\frac{1}{4}$. And in state H, we print either a matching A, C, G, T on both the top and bottom, each with probability $\frac{\mu}{4}$, or we print a mismatch, each with probability $\frac{1-\mu}{12}$. The log likelihood of a given alignment is

$$\log \tau + mM + xX + sS \tag{7}$$

where $M = \log\left(\frac{1-\tau}{3}\frac{\mu}{4}\right), X = \log\left(\frac{1-\tau}{3}\frac{1-\mu}{12}\right)$, and $S = \log\left(\frac{1-\tau}{3}\frac{1}{4}\right)$. To find a likely alignment we therefore want to maximize this sum.

Here are some questions we can now ask given two sequences σ^1, σ^2

1. What's the probability that two entries σ^1_i and σ^2_j match?
2. What's the most likely alignment for fixed $\theta = (\mu, \tau)$?
3. What's the most likely alignment over all θ?

Recall that a subset of $\mathbb{R}^n$ is convex if any line drawn between two points in the set lies in the set. On the collection of convex polytopes there are two operations, $\oplus$ (convex hull of the union) and $\odot$ (Minkowski sum, the collection of pointwise sums). This is the *polytope algebra*. When $n = 1$ it is closely related to the tropical semiring $(\mathbb{R}, \max, +)$.

Consider the following algorithm (Needleman-Wunsch) for finding the most likely alignment. Construct a table whose rows are labeled by a space and then the entries of σ^1 and whose columns are labeled by a space and then entries of σ^2. The entries of the table are recursively determined by the initial conditions

$$M(i,0) = iS, M(0,j) = jS, M(0,0) = 1_\odot \tag{8}$$

and the recurrence relation

$$M(i,j) = [M(i-1,j) \odot S] \oplus [M(i,j-1) \odot S] \oplus [M(i-1,j-1) \odot (X \text{ or } M)] \tag{9}$$

where the last entry is X if $\sigma^1_i \neq \sigma^2_j$ and M if $\sigma^1_i = \sigma^2_j$ (and where $\oplus = \max, \odot = +$). Tracing back from the last entry to the corresponding maximal entries constructs the most likely alignment.

To find the optimal alignment over all values of the parameters θ, we will run the above algorithm but with the polytope semiring instead of the tropical semiring. We initialize $M(i,0)$ to be the point $(i,0)$, $M(0,j)$ to be the point $(0,j)$, $M(0,0)$ to be the empty polygon, and we take $M = (0,0), X = (1,0), S = (0,1)$. The result of this algorithm will be the convex hull of the space of alignments.

The complexity of this algorithm is as follows. $\oplus, \odot$ are both linear in the number of lattice points involved, so the question is to bound this.

Theorem 4.4. *(Andrews) In d dimensions, there exists a constant C_d such that the number of vertices in the convex hull of a lattice polytope P is at most*

$$C_d\left(Vol(P)\right)^{\frac{d-1}{d+1}}. \tag{10}$$

If n, m are the lengths of our two sequences, then furthermore we have $\mathrm{Vol}(P) \leq (n+m)^d$, so when $d = 2$ the complexity is $O\left(nm(n+m)^{2/3}\right)$.

4.2 Evolutionary models

We want to think about how ancestral nucleotides give rise to extant nucleotides. Over time, some number of copying events will occur. Copying will be modeled by a 4×4 stochastic matrix, the *copying matrix*, describing how likely a nucleotide is to be incorrectly copied as another nucleotide. The distribution of the copying events themselves will be given by a Poisson process, so that

$$\mathbb{P}(X_t = k) = \frac{e^{-\lambda t}(\lambda t)^k}{k!} \tag{11}$$

where λ is a fixed parameter. We will also have *substitution matrices* $\theta(t)$ where $\theta(0) = I$ and

$$\theta(t) = \sum_{k=0}^{\infty} C^k \mathbb{P}(X_t = k) = \exp\left(\lambda t(C - I)\right). \tag{12}$$

Let $Q = C - I$ be the *rate matrix*. Q has the following properties:

1. $q_{ij} \geq 0$ if $i \neq j$,
2. $q_{ii} < 0$,
3. $\sum_j q_{ij} = 0$ (all row sums are zero).

Theorem 4.5. *The following hold:*

1. *$\theta(s + t) = \theta(s)\theta(t)$ (Chapman-Kolmogorov)*
2. *$\theta(t)$ is the unique solution to the differential equation $\theta'(t) = \lambda Q\theta(t)$ (or $\theta'(t) = \lambda\theta(t)Q$) with initial condition $\theta(0) = I$.*

3. $\theta^{(k)}(0) = \lambda^k Q^k$.

4. A matrix Q *is a rate matrix iff* $\theta(t)$ *is stochastic for every* $t \geq 0$.

Exercise 4.6. *Check the above and also check that* $\theta(t) = I + \lambda Q t + O(|t|^2)$.

Example (Jukes-Cantor, 1969) Let C have all entries $\frac{1}{4}$. Then $\theta(t)$ has diagonal entries $\frac{1+3e^{-\lambda t}}{4}$ and off-diagonal entries $\frac{1-e^{-\lambda t}}{4}$.

Exercise 4.7. *If* $A = UDU^{-1}$ *(where* D *is diagonal) then* $e^A = Ue^DU^{-1}$*; moreover,* $(e^D)_{ii} = e^{D_{ii}}$. *Use this to show how to compute* $\theta(t)$ *if there exists a matrix* Π *such that* ΠQ *is symmetric.*

Example (Hasegawa, 1985) Take

$$C = \begin{bmatrix} 1 - \pi_G\alpha - \pi_Y\beta & \pi_C\beta & \pi_A\alpha & \pi_T\beta \\ \pi_A\beta & 1 - \pi_G\alpha - \pi_R\beta & \pi_G\beta & \pi_T\alpha \\ \pi_A\alpha & \pi_C\beta & 1 - \pi_A\alpha - \pi_Y\beta & \pi_T\beta \\ \pi_A\beta & \pi_C\alpha & \pi_G\beta & 1 - \pi_C\alpha - \pi_R\beta \end{bmatrix} \tag{13}$$

where

1. $\alpha, \beta \geq 0$
2. $0 \leq \alpha + 2\beta < 1$,
3. $\pi_A + \pi_C + \pi_G + \pi_T = 1$,
4. $\pi_Y = \pi_C + \pi_T$,
5. $\pi_R = \pi_A + \pi_G$.

Here C, T are the pyrimidines (Y) and A, G are the purines (R). The matrix above is supposed to model the existence of two different kinds of mutations, namely *transition mutations* (within purines / pyrimidines) and *transversion mutations* (from purine to pyrimidine or vice versa).

Example (GTR/Rev) (Felsenstein hierarchy) Take $Q = \Pi S$ where

$$\Pi = \begin{bmatrix} \pi_A & 0 & 0 & 0 \\ 0 & \pi_C & 0 & 0 \\ 0 & 0 & \pi_G & 0 \\ 0 & 0 & 0 & \pi_T \end{bmatrix} \tag{14}$$

and

$$S = \begin{bmatrix} \frac{\pi_C\alpha-\pi_G\beta-\pi_T\gamma}{-\pi_A} & \alpha & \beta & \gamma \\ \alpha & \frac{\pi_A\alpha-\pi_G\delta-\pi_T\epsilon}{-\pi_C} & \delta & \epsilon \\ \beta & \delta & \frac{\pi_A\beta-\pi_C\delta-\pi_T\epsilon}{-\pi_G} & \xi \\ \delta & \epsilon & \xi & \frac{\pi_A\delta-\pi_C\epsilon-\pi_G\xi}{-\pi_T} \end{bmatrix}. \tag{15}$$

Let T_n^+ be the set of transition matrices (stochastic matrices with non-negative entries and positive determinant) under multiplication.

Theorem 4.8. *(Tuffley) Let $\psi : T_n^+ \to \mathbb{R}$ be a continuous homomorphism. Then ψ is a scalar multiple of* $\log\det$.

With $\theta(t) = e^{\lambda t Q}$ we have

$$\log\det\theta(t) = \lambda t\,\mathrm{tr}(Q). \tag{16}$$

Such functions ψ are relevant to determining branch lengths in phylogenetic trees, which describe how many mutations on average occurred. In the case of Jukes-Cantor, the logarithm of the determinant is $-3\lambda t$ but the actual branch length is $\frac{3}{4}\lambda t$.

Suppose that an ancestor gives rise to two descendants. For one of the descendants, there is a probability μ_1 that a nucleotide stays the same and a probability τ_1 that it becomes a given different nucleotide (so $\mu_1 + 3\tau_1 = 1$) and similarly for the other. Then the probability that the two descendants share a given nucleotide is

$$\mu_1\mu_2 + 3\tau_1\tau_2 \tag{17}$$

and the probability that the two descendants do not share a given nucleotide is

$$3\mu_1\tau_2 + 3\mu_2\tau_1 + 6\tau_1\tau_2. \tag{18}$$

We can obtain data about descendants: suppose we observe n nucleotides and that k of them are different. To fit the parameters to the observation we therefore want

$$(1-3\tau_1)(1-3\tau_2)+3\tau_1\tau_2 = \frac{n-k}{n} \tag{19}$$

which gives

$$(1-4\tau_1)(1-4\tau_2) = 1-\frac{4}{3}\frac{k}{n}. \tag{20}$$

The Jukes-Cantor model gives $\tau_i = \frac{1-e^{-\lambda_i t_i}}{4}$, from which it follows that

$$\frac{3}{4}(\lambda_1 t_1 + \lambda_2 t_2) = -\frac{3}{4}\log\left(1-\frac{4}{3}\frac{k}{n}\right). \tag{21}$$

Hence from data we can estimate the sum of the branch lengths. (The RHS here is the Jukes-Cantor correction to the naive estimate of $\frac{k}{n}$ itself, which it reduces to as k becomes small relative to n.)

For a fixed phylogenetic tree with more than two leaves, we generally can't estimate the branch lengths in closed form. We also need to do or approximate maximum likelihood estimates to determine the tree topology. In fact, this problem is NP-complete.

Here is a basic question: are humans closer to dogs or are they closer to rats? Phylogenetic analysis using a wide variety of models suggests that humans are closer to dogs. However, other kinds of evidence suggest that humans are closer to rats, e.g. fossil evidence and DNA evidence coming from insertions and deletions, as well as evidence coming from transposable elements. This suggests that the models used in phylogenetic analysis are flawed.

A *dissimilarity map* on a set X is a function $\delta : X \times X \to \mathbb{R}$ satisfying $\delta(x,y) = \delta(y,x)$ and $\delta(x,x) = 0$. (We also write $\delta(x,y) = \delta_{xy}$.) We can represent such a thing as an $|X| \times |X|$ symmetric matrix with zeroes along the diagonal. We can construct dissimilarity maps on X from weighted trees T some of whose vertices are labeled by the elements of X by taking $\delta(x,y)$ to be the weighted distance from x to y.

From the underlying tree of a weighted tree, we can construct an edge-incidence matrix S_T such that $(S_T)_{ij,e}$ is equal to 1 if the edge e is on the path from i to j and equal to 0 otherwise. If ℓ is the vector of weights, it follows that

$$\delta = S_T \cdot \ell. \tag{22}$$

In this case δ is said to be *T-additive.* If in addition ℓ has only positive entries, then δ is a *tree metric.*

For any $i, j, k, \ell \in X$, consider the three quantities

$$\delta_{ij} + \delta_{k\ell}, \delta_{ik} + \delta_{j\ell}, \delta_{i\ell} + \delta_{jk}. \tag{23}$$

The *weak four-point condition* is that at least two of the above are equal. The *four-point condition* is that at least two of the above are equal and greater than (or equal to?) the third. T-additive implies the weak four-point condition and tree metric implies the four-point condition.

Exercise 4.9. *(Theorem 1) δ is T-additive iff it satisfies the weak four-point condition.*

Exercise 4.10. *(Theorem 2) δ is a tree metric iff it satisfies the four-point condition.*

An interesting question for biology is the following: given δ, find a tree metric δ^T that minimizes

$$\sum_{ij} (\delta_{ij} - \delta_{ij}^T)^2 \tag{24}$$

(over all T). This is hard. However, if we relax δ^T to be T-additive instead of a tree metric, this becomes least-squares for a fixed tree, and moreover the least-squares answer is very clean. (We sometimes instead minimize a weighted sum of squares.)

Some hints for the above exercises. An *ultrametric* is a dissimilarity map satisfying the strong triangle inequality

$$\delta_{ij} \le \max(\delta_{ik}, \delta_{kj}). \tag{25}$$

This implies that two of the lengths must be equal and greater than (or equal to?) the third. There is an operation, the *Gromov transform*, given by

$$\delta_r(x, y) = \frac{1}{2} \left(\delta(x, y) - \delta(x, r) - \delta(y, r) \right) \tag{26}$$

for some $r \in X$.

Exercise 4.11. *If δ satisfies the four-point condition, then δ_r is an ultrametric for any r.*

If δ satisfies the four-point condition, we can choose a leaf r and get an ultrametric δ_r on $X - \{r\}$. This lets us build a rooted weighted tree giving rise to the ultrametric by induction. We can then attempt to invert the Gromov transform.

An example δ satisfying the four-point condition is

$$\begin{bmatrix} 0 & 3 & 5 & 5 & 5 \\ 3 & 0 & 6 & 6 & 6 \\ 5 & 6 & 0 & 6 & 6 \\ 5 & 6 & 6 & 0 & 2 \\ 5 & 6 & 6 & 2 & 0 \end{bmatrix}. \tag{27}$$

Consider a collection of cities. Let δ_{ij} be the distance between city i and city j. The traveling salesman problem asks for a way to tour all of the cities such that the total distance traveled is minimized. It is NP-complete. A simple greedy algorithm for solving it is the following; we first join the two closest cities, then the next two closest cities, and so forth. This is actually a very bad algorithm. In some cases it can give the worst tour.

Neighbor joining is a slightly different algorithm. Starting from a particular method of joining the cities, we produce a triangulation, and then from this triangulation we produce a phylogenetic tree linking the cities.

Theorem 4.12. *If δ is a* Kalmanson metric*, then neighbor joining produces an optimal tour.*

Let T be a tree and denote by $\mathcal{T}_T$ the space of weighted trees (not necessarily with positive weights) of combinatorial type T. The weighted least squares tree metric for a dissimilarity map d is

$$\operatorname{argmin}_{d^* \in \mathcal{T}_T} \sum_{i,j=1}^{n} \frac{1}{v_{ij}} (d_{ij} - d^*_{ij})^2. \tag{28}$$

Theorem 4.13. *(Desper and Gascuel, Mihaescu and Pachter) For any trivalent tree T, if v_{ij} is proportional to $(1 + A(p_{ij} - 1))2^{p_{ij}}$, then*

$$\ell(d^*) = \sum_{e \in E} \ell_e(d^*) = \sum_{i,j=1}^{n} 2^{1-p_{ij}} d_{ij}. \tag{29}$$

This is the balanced minimum evolution length. *Here p_{ij} is the number of edges between i and j.*

There are other combinatorial formulas for lengths of least squares trees, e.g. Rzhetsky and Desper and Gascuel. Some of these are IIP, which means that the formula for the length of an edge only depend on distances using that edge. This turns out to be true if and only if the v_{ij} are *semi-multiplicative* with respect to T.

Theorem 4.14. *(Gauss-Markov) Suppose that a dissimilarity map is given by $d = S_t\ell + \epsilon$ where ϵ is a vector of random variables with expectation 0 and variance v_{ij}. Then the estimator*

$$\ell_e(d^*) = (0, ...1, ...0) \cdot (S_t^\dagger V^{-1} S_t)^{-1} S_t^\dagger V^{-1} D \tag{30}$$

has minimum variance among the linear unbiased estimators of $(0, ...1, ...0) \cdot \ell$ (where the 1 is in the e^{th} position).

Finding the minimum length tree for $v_{ij} = 2^{p_{ij}}$ is equivalent to minimizing the linear function $\sum 2^{1-p_{ij}} d_{ij}$ over the *balanced minimum evolution polytope* (the convex hull of the vectors of the form $2^{1-p_{ij}^T}$ ove rall trivalent trees T). This is NP-hard. The neighbor-joining algorithm is a greedy algorithm for doing this. It proceeds by minimizing

$$Q_{ij} = d_{ij} - \frac{1}{n-2}\left(\sum_k d_{ik} + \sum_\ell d_{j\ell}\right). \tag{31}$$

Theorem 4.15. *If d^* is a tree metric for T and*

$$\|d^* - d\|_\infty \leq \frac{e_{min}}{2} \tag{32}$$

then the tree with minimal BME length for d is T.

Theorem 4.16. *(Kleinman) BME is the only minimum evolution method that is consistent and has radius $\frac{1}{2}$ in the sense that the above holds.*

5 Sequencing

Recall that DNA is transcripted into RNA, which is translated into protein, which then does stuff. Suppose we would like to determine the RNA content of a sample.

One way to do this is RNA-Seq: roughly speaking, we cut them up into sscDNA fragments, which then become dscDNA fragments, which are then selected for their size.

If we want to compute gene abundances, we can use fragments per kilobase per million mapped. This lets us compare across genes and also across experiments.

There are issues which we will attempt to fix using an expectation-maximization algorithm. This algorithm has two steps, an E-step and an M-step.

1. Call the set of transcripts T, $|T| = n$.
2. Transcripts have lengths $l_1, ..., l_n$
3. and abundances $\rho_1, ..., \rho_n$ with $0 \leq \rho_i \leq 1$, $\sum_i \rho_i = 1$.
4. Reads are always unique to the transcripts. $M_t :=$ # of reads sampled from $t \in T$.

We have a likelihood

$$\mathcal{L}(\rho_1, ..., \rho_n) = \prod_{t \in T} \prod_{f \in t} (\frac{\rho_t l_t}{(\sum_r \rho_r l_r) l_t}) = \prod_t (\frac{\rho_t l_t}{(\sum_r \rho_r l_r) l_t})^{u_i}$$

where $f \in T$ are the fragments in segment t. $\alpha_t = \frac{\rho_t l_t}{\sum_r \rho_r l_r}$ - can rewrite $\mathcal{L}$ in terms of α. To find parameters, compute

$$\max_{\alpha_1 + ... + \alpha_n = 1} \mathcal{L}(\alpha)$$

This is equivalent to $\prod_t (\alpha_t / l_t)^{u_t}$ and we can take a log, and use lagrange multipliers.

$$\frac{\partial \log \mathcal{L}(\alpha)}{\partial \alpha_i} = \sum_t u_t \log(\alpha_t) + \lambda(\alpha_1 + ... + \alpha_n - 1) = \frac{u_i}{\alpha_i} + \lambda = 0$$

Thus $\hat{\alpha}_i = \frac{\mu_i}{\sum_j \mu_j}$ as one might expect. $\hat{\rho}_t = \frac{\mu_t}{n l_t}(\frac{1}{\sum_r \frac{u_r}{N l_r}})$.

In our model, reads map contiguously. Suppose $S \subseteq T$ and we have observations μ_S. We want to maximize our likelihood again, but over S

$$\mathcal{L}(\alpha_1, ..., \alpha_n) = \prod_S (\sum_{t \in S} \frac{\alpha_t}{l_t})^{\mu_s}$$

This is a harder optimization problem. Idea: there is 'hidden data" C_t. If we have C_t (counts)

$$\prod_t (\alpha_t/l_t)^{C_t}$$

to yield $\hat{\alpha}_t = \frac{C_t}{\sum_r C_r}$, somehow by analogy to the other case. Odd!

Way to formulate this idea: $V : \mathbb{Z}^T \to \mathbb{R}$, $V(S) = \sum_{s' \in S} \mu(s')$. $\phi_i(v) :=$ expected number of counts for transcript i. The constraints are

1. $\sum_i \phi_i(v) = V(T)$
2. $v(S \cup \{i\} = V(S)$ for all S. $\phi_i(V) = 0$
3. $\phi_i(V + W) = \phi_i(V) + \phi_i(W)$, $\phi_i(aV) = a\phi_i(V)$
4. If $v(S \cup \{i\}) = v(S \cup \{j\})$ then $\phi_i(V) = \phi_j(V)$ for all S not containing i, j

These conditions imply a unique solution (HW).

$$\phi_i(V) = \sum_{S \subseteq T - \{i\}} \frac{|S|!(n - |S| - 1)!}{n!} [v(S \cup \{i\}) - v(S)]$$

Comment: this is essentially a result of game theory about some sort of "fair distribution" of value.

The *EM* algorithm: $f_{ij}(\theta)$, θ are the parameters. $f_i(\theta) = \sum_{j=1}^M f_{ij}(\theta)$. Now

$$\mathcal{L}_f(\theta) = f_1(\theta)^{\mu_1} ... f_M(\theta)^{\mu_N}$$

and

$$\mu_{ij} = \frac{\mu_i f_{ij}(\theta)}{\sum_{i=1}^M f_{ij}(\theta)}$$

Then the steps are

1. Compute μ_{ij}.
2. Compute θ^*.
3. Iterate

There is an associated theorem.

Theorem 5.1. *$\mathcal{L}_f(\theta^*) \geq \mathcal{L}_f(\theta)$ and $\mathcal{L}_f(\theta^*) = \mathcal{L}_f(\theta)$ when θ is a critical point.*

Proof. Difference =

$$\sum_{i=1}^{N}[\log f_i(\theta^*) - \log f_i(\theta)] = \sum_{i=1}^{N}\sum_{j=1}^{M}\mu_{ij}(\log f_{ij}(\theta^*) - \log f_{ij}(\theta)) + \sum_{i=1}^{N}\mu_i(\log\frac{f_i(\theta^*)}{f_i(\theta)} - \sum_{j=1}^{M}\frac{\mu_{ij}}{\mu_i}\log\frac{f_{ij}(\theta^*)}{fij(\theta^*)}$$

Interpret something as a KL divergence to finish the proof. □

6 Datasets

Here is a list of useful datasets for projects.

1. http://genome.ucsc.edu/ENCODE/ has a lot of data that can be analyzed for various projects.
2. http://sra.dnanexus.com/ has a lot of data as well.
3. http://megasun.bch.umontreal.ca/ogmp/projects/other/mt_list.html has mitochondrial DNA (it would be interesting to build a tree from this).
4. http://www.smgf.org/mtdna/database_stats.jspx also has mitochondrial DNA.
5. http://www.1000genomes.org/ has human genomes.
6. https://genome10k.soe.ucsc.edu/ has non-human genomes.
7. http://hgdownload.soe.ucsc.edu/downloads.html has non-human genomes.

7 Hypothesis testing

Consider Simpson's paradox (more properly the Yule-Simpson paradox). When we divide up the number of Democrats vs. Republicans voting for the Civil Rights Act by north vs. south and by House vs. Senate, we find that more Democrats voted for the act than Republicans in each category. However, overall we find that more Republicans voted for the act than Democrats.

This is a serious statistical issue. For example, it may appear that a drug is more effective than the absence of the drug when a population is divided into men and women, but less effective overall. To the extent that this is a paradox, it is because we are failing to distinguish correlation from causation.

Exercise 7.1. *See if you can find Simpson's paradox in Mendel's paper.*

8 Hierarchical organization of RNA

RNA has primary structure (nucleotides), secondary structure (list of base pairs), and tertiary structure (interaction of secondary structures). More formally, we will identify secondary structures with noncrossing partitions (with blocks of size 2). Given a sequence $s_1, ...s_n$ of nucleotides, we can write down a pairing matrix p_{ij} whose entries are equal to 1 if i can pair with j and 0 otherwise. Let $N^i_{\ell,n}$ denote the number of such structures on $s_\ell, ...s_n$ with i pairings. This satisfies the recurrence

$$N^{i+1}_{\ell,n+1} = N^{i+1}_{\ell,n} + \sum_{k=\ell}^{n-m} \sum_{j=0}^{i} N^j_{k,n+1} N^{i-j}_{\ell,k}. \tag{33}$$

RNA secondary structure looks like a curled-up line and may have bulges, internal loops, or hairpins. (Tertiary structure includes, for example, pseudoknots, kissing hairpins, and hairpin-bulges.)

To predict secondary structure, we can use covariation analysis (conserved patterns of basepairs during evolution) or the minimum free-energy method (structures that are energetically stable).

Assume (Tinoco-Uhlenbeck) that the free energy of each base pair is independent of all the other pairs and loop structures. Then the total free energy is the sum of the energies of each base pair. This gives a dynamic programming approach to computing free energies which has reasonable complexity (N^3).

How do we obtain structural information about RNA? One option is SHAPE (selective 2'-hydroxyl acylation analyzed by primer extension). Another is DMS-seq.

9 Networks

There are many kinds of networks (regulatory, mtetabolic, signaling, protein interaction, functional) in biology. These are often abstracted into graphs. Metabolic networks are related to chemical reaction networks, which are mathematically well-developed. In regulatory networks there is a lot of data. In signaling networks things are complicated. In protein interaction networks there is a lot of graph theory. Functional networks have been inaccessible until very recently.