CS 581, Feb 1, 2018
Today

• Review of previous lecture
• Maximum Likelihood optimization problem
• Felsenstein’s pruning algorithm
• Bayesian MCMC methods
• Research opportunities
Summary of Tuesday’s lecture

• How to solve Maximum Parsimony (MP) on small datasets
• MP is statistically consistent on some CFN model trees.
• However, there are some other CFN model trees in which MP is not statistically consistent.
• Worse, MP is positively misleading on some CFN model trees (what does this mean?).
• This phenomenon is called “long branch attraction”, and the trees for which MP is not consistent are referred to as “Felsenstein Zone trees” (after the paper by Felsenstein).
• The problem is not limited to 4-leaf trees, and it’s also not limited to CFN. (True also for all standard DNA models.)
Cavender-Farris-Neyman (CFN)

- Models binary sequence evolution
- For each edge $e$, there is a probability $p(e)$ of the property “changing state” (going from 0 to 1, or vice-versa), with $0 < p(e) < 0.5$ (to ensure that unrooted CFN tree topologies are identifiable).
- State at the root is 0 or 1 with equal probability.
- Every position evolves under the same process, independently of the others.
Statistical Consistency
Proving Statistical Consistency under CFN

• To prove that a method is statistically consistent under CFN, you have to show:
  – For all model CFN trees, as the sequence length increases the probability that the method returns the model tree converges to 1.0.

• To prove that a method is not statistically consistent, you only need to find one model CFN tree for which this statement doesn’t hold.
Proving Statistical Consistency under CFN

• Some distance-based methods (e.g., the Naïve Quartet Method) are statistically consistent under the CFN model:
  – For all model CFN trees, as the sequence length increases the probability that the method returns the model tree converges to 1.0.

• But this isn’t true for Maximum Parsimony or UPGMA (as we showed earlier).

• It is true for Maximum Likelihood (we won’t prove this).
Maximum likelihood (ML) under Cavender-Farris-Neyman

• Given a set $S$ of binary sequences, find the Cavender-Farris-Neyman model tree (tree topology and substitution probabilities on the edges) that maximizes the probability of producing the input data $S$.

ML, if solved exactly, is statistically consistent under Cavender-Farris (and under the DNA sequence models, and more complex models as well).

The problem is that ML is hard to solve.
“Solving ML”

• Technique 1: compute the probability of the data under each model tree, and return the best solution.

• Problem: Exponentially many trees on n sequences, and infinitely many ways of setting the numerical parameters on each of these trees!
Solving NP-hard problems exactly is ... unlikely

- Number of (unrooted) binary trees on \( n \) leaves is \((2n-5)!!\)
- If each tree on 1000 taxa could be analyzed in 0.001 seconds, we would find the best tree in 2890 millennia

<table>
<thead>
<tr>
<th>#leaves</th>
<th>#trees</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>15</td>
</tr>
<tr>
<td>6</td>
<td>105</td>
</tr>
<tr>
<td>7</td>
<td>945</td>
</tr>
<tr>
<td>8</td>
<td>10395</td>
</tr>
<tr>
<td>9</td>
<td>135135</td>
</tr>
<tr>
<td>10</td>
<td>2027025</td>
</tr>
<tr>
<td>20</td>
<td>(2.2 \times 10^{20})</td>
</tr>
<tr>
<td>100</td>
<td>(4.5 \times 10^{190})</td>
</tr>
<tr>
<td>1000</td>
<td>(2.7 \times 10^{2900})</td>
</tr>
</tbody>
</table>
“Solving ML”

• Technique 2: For each of the tree topologies, find the best numerical parameters (i.e., substitution probabilities \( p(e) \) for every edge \( e \)).

• Problem: Exponentially many trees on \( n \) sequences, and calculating the best setting of the parameters on any given tree is hard!

Even so, there are hill-climbing heuristics for both of these calculations (finding parameter settings, and finding trees).
Maximum Likelihood

• Input: sequence data $S$,
• Output: the model tree $(T,\Theta)$ (where $\Theta$ denotes the set of substitution probabilities on each of the branches of the tree $T$) s.t. $\Pr(S|(T,\Theta))$ is maximized.

NP-hard to find best tree.
Important in practice.
Good heuristics (RAxML, FastTree, IQTree, PhyML, and others)
“Solving ML”

• Given model tree:
  – Change tree topology and/or substitution probabilities
  – Score new model tree under likelihood (i.e., calculate the probability of the data given the model tree)
  – Accept new model tree if probability goes up
  – Perhaps accept anyway with some small probability
“Solving ML”

• Given model tree:
  – Change tree topology and/or substitution probabilities
  – Score new model tree under likelihood (i.e., calculate the probability of the data given the model tree)
  – Accept new model tree if probability goes up
  – Perhaps accept anyway with some small probability
Designing methods for large-scale phylogeny estimation

However, otherwise the relationship between the sets of neighboring trees is more complicated (i.e., neither neighborhood contains the other). Other techniques for moving through treespace that have been used successfully on large datasets are described in Goloboff (1999).

One of the desirable properties of a local search strategy is that all possible trees should be reachable by a path that uses only the specified move. Therefore, we may ask whether every two trees are connected by a path of NNI moves. As shown in Gordon et al. (2013), not only is the space of trees connected via NNI moves (so that all two trees can be visited by a path of NNI moves), but there is a path that visits every tree exactly once (i.e., a Hamiltonian path).

When a local optimum is reached, then the search begins at a new tree, often obtained using a strategy that employs randomness. For example, the parsimony ratchet (Nixon, 1999) operates by (1) producing a bootstrap alignment (i.e., a modification of the original multiple sequence alignment produced by sampling with replacement from the sites of the alignment), (2) running the heuristic search strategy anew on this bootstrap alignment until a local optimum is obtained, and then (3) continuing the search from the new tree, but now based on scores computed using the original alignment. This alternation between searches based on the original alignment and the bootstrap replicate alignment can be repeated several times.

This process repeats until a desired stopping criterion is met (e.g., a time limit, or evidence that the search strategy is not likely to find a better scoring tree). At the end, the set of all best trees is returned; this set can be very large for maximum parsimony, but searches for maximum likelihood trees or for minimum evolution trees usually produce a single “best” tree.

Figure 3.13 In the star-decomposition heuristic, one repeatedly pairs off two edges that are incident to the central node that provide the best improvement of the parsimony score, until a resolved phylogenetic tree is obtained.

(a) Phylogenetic tree  (b) One NNI tree  (c) Other NNI tree

Figure 3.14 Nearest neighbor interchange (NNI). (a) An unrooted phylogenetic tree \( T \). The four subtrees attached to the two ends of the edge marked “\( \ast \)” can each interchange in one of two possible ways, leading to two distinct phylogenetic trees, shown in (b) and (c).

Tree \( T \) is transformed to another \( T' \) by applying a branch-swapping operation that rearranges a part of \( T \). More precisely, the idea is to generate a number of neighboring phylogenetic trees in this way and then to move to the one that has the best parsimony score. This is repeated a number of times until no further improvement can be achieved.

The simplest branch-swapping operation is the nearest neighbor interchange (NNI) \([178, 203]\), in which two of four subtrees that are attached to a common edge are swapped in one of two possible ways, as illustrated in Figure 3.14.

A more general branch-swapping method is called subtree prune and regraft (SPR), in which a subtree is pruned from the given phylogenetic tree and regrafted at a different location of the phylogenetic tree, as shown in Figure 3.15 [191].

Figure 11.1 (Figure 3.14 in Huson et al. (2010)) Trees related by Nearest Neighbor Interchange (NNI) moves. In an NNI move, an internal edge in the tree is selected, and the four subtrees around the edge are identified. Then, two of these subtrees around the edge (one from each side) are swapped. Thus, trees (b) and (c) are each obtained by one NNI move applied to tree (a).

The challenge in using heuristics to find good solutions for NP-hard problems is that even the best heuristics are only ensured to find good local optima, and local optima are not necessarily global optima. Furthermore, these heuristic search strategies can take a long time (and use large amounts of memory) on large datasets, meaning days, weeks, or even...
Figure 11.3 (Figure 3.16 in Huson et al. (2010)) Trees related by a Tree Bisection and Reconnection (TBR) move. In a TBR move, an edge is deleted from the tree, creating two trees; the two trees are then re-attached by the introduction of an edge between them. Note that every NNI or SPR move is a TBR move. Tree (c) is obtained by applying one TBR move to tree (a).
Approaches for “solving” ML

1. Hill-climbing heuristics (which can get stuck in local optima)
2. Randomized algorithms for getting out of local optima
“Solving ML”

• Given model tree:
  – Change tree topology and/or substitution probabilities
  – Score new model tree under likelihood (i.e., calculate the probability of the data given the model tree)
  – Accept new model tree if probability goes up
  – Perhaps accept anyway with some small probability
Calculating probability of data, given CFN model tree

Basic challenge:

• Given set S of binary sequences and a CFN model tree T, what is the probability that T generates S?
Computing the probability of the data

• Given a model tree (with all the parameters set) and character data at the leaves, you can compute the probability of the data.

• Small trees can be done by hand (brute-force technique, see lecture from January 30).
Brute Force Technique

• Assume you have the CFN model tree on n leaves (with substitution probabilities on the edges).

• To calculate the probability of a sequence alignment with k sites, you can use the Brute Force technique (all ways of labelling internal nodes):
  – Running time is $O(k2^n)$. (Question: what would this be for Jukes-Cantor?)
  – Not polynomial.
Computing the probability of the data

- Given a model tree (with all the parameters set) and character data at the leaves, you can compute the probability of the data.
- Small trees can be done by hand (brute-force technique, see lecture from January 30).
Computing the probability of the data

• Given a model tree (with all the parameters set) and character data at the leaves, you can compute the probability of the data.

• Small trees can be done by hand (brute-force technique, see lecture from January 30).

• Large examples are computationally intensive - but still polynomial time (using Felsenstein’s Pruning Algorithm).
Felsenstein’s Pruning Algorithm

- FPA(v,b) is the probability of the single-site data at the leaves below v in CFN model tree T, given that the state at v is b.
- The probability of the observed single-site data given CFN model tree T is:
  
  \[ \frac{1}{2}(\text{FPA}(r,0)+\text{FPA}(r,1)) \]
  
  where r is the root.

Questions:
- Why is there a factor of \( \frac{1}{2} \)?
- How do we extend this to binary sequences of length k?
- How do we compute FPA(v,b) for each v in V(T) and b in \{0,1\}?
Calculating FPA(v,b)

• If v is a leaf, it’s trivial.
• Otherwise v has two children, w₁ and w₂.
• \[ \text{FPA}(v,b) = \sum_{a \in \{0,1\}} \text{FPA}(w₁,a) \cdot \Pr(w₁=a \mid v=b) \times \sum_{a \in \{0,1\}} \text{FPA}(w₂,a) \cdot \Pr(w₂=a \mid v=b) \]

We know the probabilities \( \Pr(w_j = a \mid v=b) \) (for \( j=1,2 \)), since they are defined by the model tree substitution probabilities on each edge.

Therefore, we can calculate these values from the “bottom up”, just as in Sankoff’s algorithm for maximum parsimony.
Maximum Likelihood

- Input: sequence data $S$,
- Output: the model tree $(T, \Theta)$ (where $\Theta$ denotes the set of substitution probabilities on each of the branches of the tree $T$) s.t. $\Pr(S|(T,\Theta))$ is maximized.

NP-hard to find best tree.
Important in practice.
Good heuristics (RAxML, FastTree, IQTree, PhyML, and others)
Statistical Methods

- **Maximum Likelihood**: find model tree most likely to have generated the observed data.
Statistical Methods

- **Maximum Likelihood**: find model tree most likely to have generated the observed data

- **Bayesian Estimation**: output distribution of tree topologies in proportion to their likelihood for having generated the observed data (marginalizing over all the possible numeric parameters)
Bayesian analyses

• Algorithm is a random walk through space of all possible model trees (trees with substitution matrices on edges, etc.).

• From your current model tree, you perturb the tree topology and numerical parameters to obtain a new model tree.

• Compute the probability of the data (character states at the leaves) for the new model tree.
  – If the probability increases, accept the new model tree.
  – If the probability is lower, then accept with some probability (that depends upon the algorithm design and the new probability).

• Run for a long time...
Bayesian estimation

After the random walk has been run for a very long time...

• Gather a random sample of the trees you visit
• Return:
  – Statistics about the random sample (e.g., how many trees have a particular bipartition), OR
  – Consensus tree of the random sample, OR
  – The tree that is visited most frequently

Bayesian methods, if run *long enough*, are statistically consistent methods (the tree that appears the most often will be the true tree with high probability).

*MrBayes* is standard software for Bayesian analyses in biology.
Figure 8.3 The x-axis indicates the numeric parameters for the GTR model trees that can equip the different tree topologies, Tree 1 and Tree 2; the y-axis indicates the likelihood score. Note that the likelihood curve Tree 1 has a larger peak than that for Tree 2, but Tree 2 has a larger area under its curve than Tree 1. Therefore, GTR maximum likelihood would pick Tree 1, but marginalizing over the GTR numeric parameters (as in a Bayesian analysis) would favor Tree 2.

As a result, there is an important distinction between maximum likelihood and Bayesian estimation. As seen in Figure 8.3, Tree 1 has a higher maximum likelihood score than Tree 2, because for at least one set of numeric parameters Tree 1 has higher probability of generating the input sequence alignment than Tree 2. However, the total probability of producing the input sequence alignment is higher for Tree 2 than Tree 1, when all the possible numeric parameters are taken into account. Hence, for this input, Tree 2 would be preferred over Tree 1 in a Bayesian framework.

There are several challenges in using Bayesian methods. For example, Bayesian methods require priors, and the choice of priors can affect the accuracy of the resultant trees; hence, the selection of priors requires great care (Alfaro and Holder, 2006; Yang, 2009). The second is that Bayesian MCMC methods need to be run until they have reached the stationary distribution, and it is not at all straightforward to assess whether stationarity has been reached. Furthermore, on large datasets, it can require very long running times to reach stationarity. As a result, Bayesian MCMC analyses are typically not used for very large datasets.
Extensions to molecular phylogenetics

• DNA and RNA sequence evolution models typically have four states (nucleotides)
• “Protein-coding” sequence evolution models have 64 states (3 nucleotides in a row make a codon, which makes an amino acid).
• Protein sequence evolution models have 20 states (one for each amino acid).
Classical Sequence Evolution
Jukes-Cantor DNA model

• Character states are A,C,T,G (nucleotides).
• All substitutions have equal probability.
• On each edge $e$, there is a value $p(e)$ indicating the probability of change from one nucleotide to another on the edge, with $0<p(e)<0.75$ (to ensure that JC trees are identifiable).
• The state (nucleotide) at the root is random (all nucleotides occur with equal probability).
• All the positions in the sequence evolve identically and independently.
Phylogeny Estimation under Jukes-Cantor

• Maximum Likelihood: Given a set S of nucleotide sequences, find the Jukes-Cantor model tree (tree topology and substitution probabilities) that maximizes the probability of producing the input data S.

• Maximum parsimony: Given set S, find tree with minimum number of changes

• Distance-based estimation: Given set S, first compute Jukes-Cantor distances (see text), then apply distance-based method (e.g., Naïve Quartet Method, Neighbor joining, etc.)

ML, if solved exactly, is statistically consistent under Jukes-Cantor
MP, if solved exactly, is not statistically consistent and can be positively misleading (nearly same proof as for CFN)
Distance-based estimation can be statistically consistent
Statistical gene tree estimation methods either with a substitution probability \( p(e) \) for edge \( e \) or with a branch length \( l(e) \) on edge \( e \) indicating the expected number of changes on edge \( e \), just as we did for the CFN model. In that formulation, \( l(e) = \frac{3}{4} \ln \left( \frac{1}{4} p(e) \right) \). However, we can also write this in terms of a rate matrix and branch lengths (using the same lengths as given above), where \( Q_{xy} = Q_{uv} \) for \( x \neq y \) and \( u \neq v \). We also need to specify the distribution of states at the root, which is given by \( p_x = \frac{1}{4} \) for all nucleotides \( x \). Thus, the JC69 model is an example of a model that can be expressed in terms of a common rate matrix across the tree.

The Generalised Time Reversible (i.e., GTR) model (Tavaré, 1986) makes the fewest constraints on the rate matrix of all the time-reversible stationary models, and is the most commonly used model for phylogenetic inference on DNA sequences. Intermediate models, some of which are shown in Figure 8.1, can be obtained by relaxing the constraint given in the JC69 model in various ways. The models in the figure are all identifiable, and estimation under these models is generally computationally feasible. These are all examples of standard DNA site evolution models, with GTR the most complex of the standard models; see Hillis et al. (1996); Li (1997); Yang (2014) for more information.

### Figure 3.9 from Huson et al., 2010

**Standard DNA site evolution models**

```
        JC
       /  \  \
   F81   K2P
      /  \  /  \
HKY =F84 K3SP
   /  \  /  \
TrN   SYM
   /  \  \
GTR
```

In summary, the Jukes-Cantor model of DNA evolution assumes that all four bases (A, C, G, and T) occur with the same frequency (0.25) and that changes from one base to another occur at the same rate between all bases. There are many ways to relax these conditions to obtain more general models. For example, if we let the bases occur with different and arbitrary frequencies (although they have to sum to 1), and allow two different rates of change, one for transitions (that is, changes between A and G or between C and T) and a second one for transversions (all other changes), then we obtain the so-called Hasegawa-Kishino-Yano model. Both the Jukes-Cantor model and the Hasegawa-Kishino-Yano model are special cases of the general time reversible model, see Figure 3.9.

Another important way to obtain more general models of DNA evolution is to allow different substitution rates at different positions in the sequence. For example, this is often done by defining a discrete collection of rate classes using a Gamma distribution, which is popular because it has one main parameter, \( \alpha \), that determines the shape of the distribution [246]. When \( \alpha < 1 \), then the distribution is exponentially shaped and asymptotic to both the vertical and horizontal axes. When \( \alpha = 1 \), then the result is an exponential distribution. For \( \alpha > 1 \), the Gamma distribution assumes a unimodal, but skewed shape.
Phylogeny estimation under GTR

• Generalized Time Reversible model of nucleotide sequence evolution:
  – Allows fairly general 4x4 substitution matrix
  – All the sites evolve $i.i.d.$ down the tree
  – Most general DNA sequence evolution model used in practice (and contains all the submodels)
  – “logdet” distances makes distance-based estimation statistically consistent
  – ML still consistent, MP still inconsistent
Summary so far

- **Distance-based methods:**
  - many are statistically consistent under standard sequence evolution models
  - most are polynomial time
- **Maximum Parsimony:**
  - is not statistically consistent under standard sequence evolution models, but it can be consistent on some model trees.
  - is NP-hard and computationally intensive in practice.
- **Maximum Likelihood:**
  - is statistically consistent under standard sequence evolution models
  - is NP-hard and computationally intensive in practice
- **Bayesian MCMC:**
  - is statistically consistent under standard sequence evolution models
  - is computationally intensive in practice
Performance on data

- Statistical consistency or inconsistency is an asymptotic statement, and requires a proof; it really has nothing much to say about performance on finite data.

- To evaluate performance on finite data, we use simulations.

- MP is surprisingly good for many model conditions (even sometimes better than some statistically consistent methods).

- Typically ML outperforms all other methods on simulated data in terms of topological accuracy.
Figure 5: Accuracy as a function of the diameter under the K2P+Gamma model for fixed sequence length and two numbers of taxa.

We reported all results so far under the K2P+Gamma model only, due to space limitations. However, we explored performance under the JC (Jukes-Cantor) model as well. The relative performance of the methods we studied was the same under the JC model as under the K2P+Gamma model. However, throughout the experiments, the error rate of the methods was lower under the JC model (using the JC distance-correction formulas) than under the K2P+Gamma model of evolution (using the K2P+Gamma distance-correction formulas). This might be expected for the Weighbor method, which is optimized for the JC model, but is not as easily explained for the other methods. Figure 6 shows the error rate of NJ on trees of diameter 0.4 under the two models of evolution. NJ clearly does better under the JC model than under the K2P+Gamma model; other methods result in similar curves. Correlating the decrease in performance with specific features in the model is a challenge, but the results clearly indicate that experimentation with various models of evolution (beyond the simple JC model) is an important requirement in any study.

6 Conclusion

In earlier studies we presented the DCM-NJ+MP method and showed that it outperformed the NJ method for random trees drawn from the uniform distribution on tree topologies and branch lengths as well as for trees drawn from a more biologically realistic distribution, in which the trees are birth-death trees with a moderate deviation from ultrametricity. Here we have extended our result to include the Weighbor and
Performance on data

• Statistical consistency or inconsistency is an asymptotic statement, and requires a proof; it really has nothing much to say about performance on finite data.
• To evaluate performance on finite data, we use simulations.
• MP is surprisingly good for many model conditions (even sometimes better than some statistically consistent methods).
• Typically ML outperforms all other methods on simulated data in terms of topological accuracy.
What do we need?

• The best ML software cannot scale to really large datasets:
  – RAxML is good for long alignments as long as not too many sequences
  – FastTree is good for many thousands (even 1,000,000) sequences, as long as short
  – Many datasets take weeks to analyze and use terabytes of memory

• Opportunity for high impact