CSCE 471/871 Lecture 5: Multiple Sequence Alignments

Stephen D. Scott

Introduction: Multiple Alignments

- Start with a set of sequences
- In each column, residues are homolgous
  - Residues occupy similar positions in 3D structure
  - Residues diverge from a common ancestral residue
  - Figure 6.1, p. 137
- Can be done manually, but requires expertise and is very tedious
- Often there is no single, unequivocally "correct" alignment
  - Problems from low sequence identity & structural evolution

Outline

- Scoring a multiple alignment
  - Minimum entropy scoring
  - Sum of pairs (SP) scoring
- Multidimenisonal dynamic programming
- Progressive alignment methods
- Multiple alignment via profile HMMs

Scoring a Multiple Alignment

- Ideally, is based in evolution, as in e.g. PAM and BLOSUM matrices
- Contrasts with pairwise alignments:
  1. Position-specific scoring (some positions more conserved than others)
  2. Ideally, need to consider entire phylogenetic tree to explain evolution of entire family
- I.e. build complete probabilistic model of evolution
  - Not enough data to parameterize such a model
    $\Rightarrow$ use approximations
- Assume columns statistically independent:

$$S(m) = G + \sum_i S(m_i)$$

m_i is column i of MA m, G is (affine) score of gaps in m

Minimum Entropy Scoring

- m_i^j = symbol in column i in sequence j, c_{ia} = observed count of residue a in column i
- Assume sequences are statistically independent, i.e. residues independent within columns
- Then probability of column m_i is $P(m_i) = \prod_a p_{ia}^{c_{ia}}$, where p_{ia} = prob. of a in column i

Minimum Entropy Scoring
(cont'd)

- Set score to be $S(m_i) = -\log P(m_i) = -\sum_a c_{ia} \log p_{ia}$
  - Propotional to Shannon entropy
  - Define optimal alignment as

$$m^* = \underset{m}{\operatorname{argmin}} \left\{ \sum_{m_i \in m} S(m_i) \right\}$$

- Independence assumption valid only if all evolutionary subfamilies are represented equally; otherwise bias skews results

Sum of Pairs (SP) Scores

- Treat multiple alignment as $\binom{N}{2}$ pairwise alignments
- If $s(a,b)$ is substitution score from e.g. PAM or BLOSUM:

$$S(m_i) = \sum_{k<\ell} s(m_i^k, m_i^\ell)$$

- Caveat: $s(a,b)$ was derived for pairwise comparisons, not N-way comparisons

$$\overbrace{\log \frac{p_{abc}}{q_a q_b q_c}}^{\text{correct}} \quad \text{vs.} \quad \overbrace{\log \frac{p_{ab}}{q_a q_b} + \log \frac{p_{bc}}{q_b q_c} + \log \frac{p_{ac}}{q_a q_c} = \log \frac{p_{ab} p_{bc} p_{ac}}{q_a^2 q_b^2 q_c^2}}^{\text{SP}}$$

Sum of Pairs (SP) Scores
Example of a Problem

- Given an alignment with only "L" in column i, using BLOSUM50 yields an SP score of $S_1 = 5\binom{N}{2} = 5N(N-1)/2$
- If one "L" is replaced with "G", then SP score is $S_2 = S_1 - 9(N-1)$
- Problem:

$$\frac{S_2}{S_1} = 1 - \frac{9(N-1)}{5N(N-1)/2} = 1 - \frac{18}{5N} \ ,$$

  i.e. as N increases, $S_2/S_1 \to 1$
  - But large N should give more support for "L" in m_i relative to S_2, not less (i.e. should have S_2/S_1 decreasing)

Outline

- Scoring a multiple alignment
- Multidimenisonal dynamic programming
  - Standard MDP algorithm
  - MSA
- Progressive alignment methods
- Multiple alignment via profile HMMs

Multidimensional Dynamic Programming

- Generalization of DP for pairwise alignments
- Assume statistical independence of columns and linear gap penalty (can also handle affine gap penalties)
- $S(m) = \sum_i S(m_i)$, and $\alpha_{i_1,i_2,\ldots,i_N}$ = max score of alignment of subsequences $x^1_{1\ldots i_1}, x^2_{1\ldots i_2}, \ldots, x^N_{1\ldots i_N}$

$$\alpha_{i_1,i_2,\ldots,i_N} = \max \begin{cases} \alpha_{i_1-1,i_2-1,i_3-1,\ldots,i_N-1} + S\left(x^1_{1\ldots i_1}, x^2_{1\ldots i_2}, x^3_{1\ldots i_3}, \ldots, x^N_{1\ldots i_N}\right), \\ \alpha_{i_1,i_2-1,i_3-1,\ldots,i_N-1} + S\left(-, x^2_{1\ldots i_2}, x^3_{1\ldots i_3}, \ldots, x^N_{1\ldots i_N}\right), \\ \alpha_{i_1-1,i_2,i_3-1,\ldots,i_N-1} + S\left(x^1_{1\ldots i_1}, -, x^3_{1\ldots i_3}, \ldots, x^N_{1\ldots i_N}\right), \\ \quad \vdots \\ \alpha_{i_1-1,i_2-1,i_3-1,\ldots,i_N} + S\left(x^1_{1\ldots i_1}, x^2_{1\ldots i_2}, x^3_{1\ldots i_3}, \ldots, -\right), \\ \alpha_{i_1,i_2,i_3-1,\ldots,i_N-1} + S\left(-, -, x^3_{1\ldots i_3}, \ldots, x^N_{1\ldots i_N}\right), \\ \quad \vdots \end{cases}$$

- In each column, take all gap-residue combinations except 100% gaps

Multidimensional Dynamic Programming
(cont'd)

- Assume all N sequences are of length L
- Space complexity = $\Theta($ $)$
- Time complexity = $\Theta($ $)$
- Is it practical?

MSA [Carrillo & Lipman 88; Lipman et al. 89]

- Uses MDP, but eliminates many entries from consideration to save time
- Can optimally solve problems with $L = 300$ and $N = 7$ (old numbers)
- Uses SP scoring: $S(a) = \sum_{k<\ell} S(a^{k\ell})$, where a is MA and $a^{k\ell}$ is PA between x^k and x^ℓ induced by a
- If $\hat{a}^{k\ell}$ is optimal PA between x^k and x^ℓ (easily computed), then $S(a^{k\ell}) \leq S(\hat{a}^{k\ell})$ for all k and ℓ

MSA
(cont'd)

- Assume we have lower bound $\sigma(a^*)$ on score of optimal alignment a^*:

$$\sigma(a^*) \leq S(a^*) = \sum_{k<\ell} S(a^{*\,k\ell}) = S(a^{*\,k\ell}) + \sum_{\substack{k' < \ell' \\ (k',\ell') \neq (k,\ell)}} S(a^{*\,k'\ell'}) \leq S(a^{*\,k\ell}) + \sum_{\substack{k' < \ell' \\ (k',\ell') \neq (k,\ell)}} S(\hat{a}^{*\,k'\ell'})$$

- Thus $S(a^{*\,k\ell}) \geq \beta^{k\ell} = \sigma(a^*) - \sum_{\substack{k'<\ell' \\ (k',\ell') \neq (k,\ell)}} S(\hat{a}^{*\,k'\ell'})$

- When filling in matrix, only need to consider PAs that score at least $\beta^{k\ell}$ (Figure 6.3, p. 144)

- Can get $\sigma(a^*)$ from other (heuristic) alignment methods

Outline

- Scoring a multiple alignment
- Multidimenisonal dynamic programming
- Progressive alignment methods
  - Feng-Doolittle
  - Profile alignment
  - CLUSTALW
  - Iterative refinement
- Multiple alignment via profile HMMs

Progressive Alignment Methods

- Repeatedly perform pairwise alignments until all sequences are aligned
- Start by aligning the most similar pairs of sequences (most reliable)
  - Often start with a "guide tree"
- Heuristic method (suboptimal), though generally pretty good
- Differences in the methods:
  1. Choosing the order to do the alignments
  2. Are sequences aligned to alignments or are sequences aligned to sequences and then alignments aligned to alignments?
  3. Methods used to score and build alignments

Feng-Doolittle

1. Compute a distance matrix by aligning all pairs of sequences
   - Convert each pairwise alignment score to distance: $D = -\log \frac{S_{obs} - S_{rand}}{S_{max} - S_{rand}}$
   - S_{obs} = observed alignment score between the two sequences, S_{max} = average score of aligning each sequence to itself, S_{rand} = expected score of aligning two random sequences of same composition and length
2. Use a hierarchical clustering algorithm [Fitch & Margoliash 67] to build guide tree based on distance matrix

Feng-Doolittle
(cont'd)

3. Build multiple alignment in the order that nodes were added to the guide tree in Step 2
   - Goes from most similar to least similar pairs
   - Aligning two sequences is done with DP
   - Aligning sequence x with existing alignment a done by pairwise aligning x to each sequence in a
     - Highest-scoring PA determines how to align x with a
   - Aligning existing alignment a with existing alignment a' is done by pairwise aligning each sequence in a to each sequence in a'
     - Highest-scoring PA determines how to align a with a'
   - After each alignment formed, replace gaps with "X" character that scores 0 with other symbols
     - "Once a gap, always a gap"
     - Ensures consistency between PAs and corresponding MAs

Profile Alignment

- Allows for position-specific scoring, e.g.:
  - Penalize gaps more in a non-gap column than in a gap-heavy column
  - Penalize mismatches more in a highly-conserved column than a heterogeneous column
- If gap penalty is linear, can use SP score with $s(-, a) = s(a, -) = -g$ and $s(-, -) = 0$
- Given two MAs (profiles) a_1 (over $x^1, \ldots, x^n$) and a_2 (over $x^{n+1}, \ldots, x^N$), align a_1 with a_2 by not altering the fundamental structure of either
  - Insert gaps into entire columns of a_1 and a_2
  - $s(-, -) = 0$ implies that this doesn't affect score of a_1 or a_2

Profile Alignment
(cont'd)

- Score:

$$\begin{aligned}\sum_i S(m_i) &= \sum_i \sum_{k<\ell\le N} s(m_i^k, m_i^\ell) \\ &= \sum_i \sum_{k_1,\ell_1\in a_1} s(m_i^{k_1}, m_i^{\ell_1}) + \sum_i \sum_{k_2,\ell_2\in a_2} s(m_i^{k_2}, m_i^{\ell_2}) + \overbrace{\sum_i \sum_{k\in a_1,\ell\in a_2} s(m_i^k, m_i^\ell)}\end{aligned}$$

- Only the last term is affected by the alignment procedure, so it's the only one that needs to be optimized
- Thus alignment of profiles is similar to pairwise alignment, solved optimally via DP
- One profile can be single sequence

CLUSTALW

- Similar to Feng-Doolittle, but tuned to use profile alignment methods

1. Compute distance matrix via pairwise DP and convert to distances via [Kimura 83]
   - Score with substitution matrix based on expected similarity of final alignment
2. Use hierarchical clustering algorithm [Saitou & Nei 87] to build guide tree

CLUSTALW
(cont'd)

3. Build multiple alignment in the order that nodes were added to the guide tree in Step 2
   - Use sequence-sequence, sequence-profile, or profile-profile as necessary
   - Weight sequences to compensate for bias in SP scoring
   - Use position-specific gap-open profile penalties; e.g. more likely to allow new gap in hydrophilic regions
   - Adjusts gap penalties to concentrate gaps in a few regions
   - Dynamically adjusts guide tree to defer low-scoring alignments until later

Iterative Refinement Methods

- Start with MA, then iteratively remove one sequence (or subset) x at a time and realign to profile of remaining sequences
  $\Rightarrow$ will increase score or not change it
- Repeat with other sequences until alignment remains unchanged
- Guaranteed to reach local max if all sequences tried

Iterative Refinement Methods
[Barton & Sternberg 87]

1. Pairwise align the two most similar sequences
2. Sequence-profile align the profile of current MA to most similar sequence; repeat until all sequences aligned
3. Remove sequence x^1 and sequence-profile realign it to profile of rest; repeat for $x^2, \ldots, x^N$
4. Repeat above step until convergence

Outline

- Scoring a multiple alignment
- Multidimenisonal dynamic programming
- Progressive alignment methods
- Multiple alignment via profile HMMs
  - Multiple alignment with known profile HMM
  - Profile HMM training from unaligned sequences
    - Initial model
    - Baum-Welch
    - Avoiding local maxima
    - Model surgery

MA via Profile HMMs

- Replace SP scoring with more statistically valid HMM scheme
- But don't we need a multiple alignment to build the profile HMM?
  - Use heuristics to set architecture, Baum-Welch to find parameters

Multiple Alignment with Known Profile HMM

- Find most likely (Viterbi) path and line up residues from same match states
- Insert state emissions are not aligned (Figs. 6.4–6.6, pp. 151–152)
  - OK so long as residues are true insertions (not conserved or meaningfully alignable)
  - Other MA algorithms align entire sequences

Profile HMM Training from Unaligned Sequences

- Used by SAM

1. Choose length of model (number of match states) and initialize parameters
2. Set parameters via Baum-Welch
   - Use heuristics to avoid local optima
3. Check length of model from Step 1 and update if necessary
   - Repeat Step 2 if model length changed
4. Align all sequences to final model using Viterbi algorithm and build MA

Choosing Initial Model

- Architecture completely set once we choose number match states M
- When we started with MA, we applied heuristics to set M
- But we don't have MA!
  - Heuristic: Let M = average sequence length
  - If prior information known, use that instead
- For initial parameters, complexity of B-W search makes us want to start near good local optimum
  - Start with reasonable initial values of parameters (e.g. transitions into match states relatively large):
    - Sample from Dirichlet prior
    - Start with guess of MA

Baum-Welch
Forward Equations

$$f_{M_0}(0) = 1$$

$$\begin{aligned} f_{M_k}(i) &= e_{M_k}(x_i)\left(f_{M_{k-1}}(i-1)a_{M_{k-1}M_k} + f_{I_{k-1}}(i-1)a_{I_{k-1}M_k} + f_{D_{k-1}}(i-1)a_{D_{k-1}M_k}\right) \\ f_{I_k}(i) &= e_{I_k}(x_i)\left(f_{M_k}(i-1)a_{M_kI_k} + f_{I_k}(i-1)a_{I_kI_k} + f_{D_k}(i-1)a_{D_kI_k}\right) \\ f_{D_k}(i) &= f_{M_{k-1}}(i)a_{M_{k-1}D_k} + f_{I_{k-1}}(i)a_{I_{k-1}D_k} + f_{D_{k-1}}(i)a_{D_{k-1}D_k} \end{aligned}$$

$$f_{M_{M+1}}(L+1) = f_{M_M}(L)a_{M_MM_{M+1}} + f_{I_M}(L)a_{I_MM_{M+1}} + f_{D_M}(L)a_{D_MM_{M+1}}$$

Baum-Welch
Backward Equations

$$b_{M_{M+1}}(L+1) = 1 \;;\; b_{M_M}(L) = a_{M_MM_{M+1}} \;;\; b_{I_M}(L) = a_{I_MM_{M+1}} \;;\; b_{D_M}(L) = a_{D_MM_{M+1}}$$

$$\begin{aligned} b_{M_k}(i) &= b_{M_{k+1}}(i+1)a_{M_kM_{k+1}}e_{M_{k+1}}(x_{i+1}) + b_{I_k}(i+1)a_{M_kI_k}e_{I_k}(x_{i+1}) + b_{D_{k+1}}(i)a_{M_kD_{k+1}} \\ b_{I_k}(i) &= b_{M_{k+1}}(i+1)a_{I_kM_{k+1}}e_{M_{k+1}}(x_{i+1}) + b_{I_k}(i+1)a_{I_kI_k}e_{I_k}(x_{i+1}) + b_{D_{k+1}}(i)a_{I_kD_{k+1}} \\ b_{D_k}(i) &= b_{M_{k+1}}(i+1)a_{D_kM_{k+1}}e_{M_{k+1}}(x_{i+1}) + b_{I_k}(i+1)a_{D_kI_k}e_{I_k}(x_{i+1}) + b_{D_{k+1}}(i)a_{D_kD_{k+1}} \end{aligned}$$

Baum-Welch
Re-estimation Equations

$$
\begin{aligned}
E_{M_k}(a) &= \frac{1}{P(x)} \sum_{i:x_i=a} f_{M_k}(i) b_{M_k}(i) \\
E_{I_k}(a) &= \frac{1}{P(x)} \sum_{i:x_i=a} f_{I_k}(i) b_{I_k}(i) \\
A_{X_k M_{k+1}} &= \frac{1}{P(x)} \sum_i f_{X_k}(i) a_{X_k M_{k+1}} e_{M_{k+1}}(x_{i+1}) b_{M_{k+1}}(i+1) \\
A_{X_k I_k} &= \frac{1}{P(x)} \sum_i f_{X_k}(i) a_{X_k I_k} e_{I_k}(x_{i+1}) b_{I_k}(i+1) \\
A_{X_k D_{k+1}} &= \frac{1}{P(x)} \sum_i f_{X_k}(i) a_{X_k D_{k+1}} b_{D_{k+1}}(i)
\end{aligned}
$$

Avoiding Local Maxima

- B-W will converge to local maximum likelihood model, but how good is that globally?
- Long sequences ⇒ many parameters to optimize ⇒ increased risk of getting stuck
- Methods to avoid this:
  - Multiple runs from random start points (sometimes done in training artificial neural networks)
  - Use random pertubations of current solution to nudge it into different parts of the search space, e.g. simulated annealing

Simulated Annealing

- Based on the annealing process to crystallize certain compounds
- In optimization, this involves occasionally selecting worse solutions to allow movement to a region of the search space where a better local optimum exists
- Movement is done probabilistically (so optimization can be thought of as a Markov process), and probability of worse choice decreases as optimization progresses
- Probability of a particular solution x is

$$P(x) = \frac{1}{Z} \exp\left(-\frac{1}{T} E(x)\right),$$

  $Z = \int \exp\left(-\frac{1}{T} E(x)\right)$ is normalizer, $E(x)$ is energy (objective) function to be minimized, and T is temperature parameter that is reduced based on annealing schedule
- $T \to \infty \Rightarrow P(x) \to$ uniform, $T \to 0 \Rightarrow P(x) \to$ peaks at minimum values of $E(x)$

Simulated Annealing
(cont'd)

- For HMM, use as $E(x)$ the negative log of likelihood: $-\log P(X \mid \theta)$, so

$$P(x) = \frac{\exp\left(-\frac{1}{T}\left(-\log P(X \mid \theta)\right)\right)}{Z} = \frac{P(X \mid \theta)^{1/T}}{\int P(X \mid \theta')^{1/T} d\theta'}$$

- To sample from this distribution, can use noise injection or Viterbi estimation
  - Noise injection: Add noise to counts estimated in forward-backward procedure, decreasing noise rate slowly

Simulated Annealing
Viterbi Estimation

- Based on Viterbi alternative to B-W, in which counts come from most likely paths rather than forward-backward expectation estimates
  - In SA approximation, rather than choosing most likely path, choose a path probabilistically:

$$P(\pi) = \frac{P(\pi, x \mid \theta)^{1/T}}{\sum_{\pi'} P(\pi', x \mid \theta)^{1/T}}$$

  - Denominator comes from modified forward algorithm with exponentiated parameters
  - Use stochastic traceback to return π: For $i = L + 1$ down to 1,

$$P(\pi_{i-1} \mid \pi_i) = \frac{f_{i-1,\pi_{i-1}} \hat{a}_{\pi_{i-1},\pi_i}}{\sum_k f_{i-1,k} \hat{a}_{i,\pi_i}},$$

  $\hat{a}_{ij} = a_{ij}^{1/T}$

Model Surgery

- B-W should give reasonably good parameters to fit architecture to data
- But was the architecture accurate in the first place?
  - Too few match states ⇒ overuse of insertion states, incorrectly labeling some parts as non-matches
  - Too many match states ⇒ overuse of deletion states
- Model surgery (heuristically) identifies such problems and updates model
  - Use f-b or Viterbi to compute usage of all the model's transitions
  - If a match state M_i is used too infrequently, delete it and collapse the model
  - If an insert state I_j is used too frequently, expand it to a sequence of match states (number = average length of insertions)
- Have to recompute parameters via B-W after surgery!

Topic summary due in 1 week!