PSSMs and the Gibbs Sampler

Local multiple sequence alignment involves the discovery, modeling, and recognition of conserved patterns or motifs in multiple (and potentially very many) DNA or protein sequences.

- In *discovery*, we are given *unlabeled* sequences. The task is to identify one or more shared, conserved motifs in these sequences. In machine learning terms, this is equivalent to *labeling* the sequences. For example, each symbol in a sequence might be labeled "1" if it is in the conserved pattern and "0" if it is not. More complex labeling schemes representing more than one motif or different substructures within a motif are also possible.

- In *modeling*, we are given a local multiple alignment as input. In machine learning terms, we are given *labeled* sequences. The task is to construct a probabilistic model that represents the properties of each column in the alignment (i.e., the symbols we are likely to observe at that position) in an efficient manner and that can be used for searching for new instances of the pattern.

- In *recognition*, we are given a new unlabeled sequence containing zero, one, or more than one instances of the motif of interest. A probabilistic model of the motif is used to search the unlabeled sequence for instances of the motif. The location and extent of each motif identified is reported.

In the next two lectures, we will discuss Position Specific Scoring Matrices (PSSM's), a formalism for modeling local multiple alignments, and the Gibbs Sampler, a discovery method that uses the PSSM formalism. PSSM's and the Gibbs sampler are suitable for ungapped motifs only. The *Hidden Markov Model (HMM)* is a formalism that can be used for both modeling and discovery of patterns that contain gaps. We will discuss HMM's immediately following the Gibbs sampler.

In these notes PSSM's and Gibbs sampler are presented in terms of amino acid motifs. Both formalisms can be equally well applied to patterns in nucleic acid sequences. In fact, discovering and modeling transcription factor binding sites in DNA sequences is a common application of the Gibbs sampler in bioinformatics.

Position Specific Scoring Matrices

Position Specific Scoring Matrices (PSSM's) are a formalism for *modeling* ungapped local alignments. Like scoring matrices used for pairwise alignments, PSSM's are based on a log-odds formalism. Recall that both the PAM and BLOSUM matrices are defined in terms of an alternate hypothesis, H_a, that a pair of sequences are related at a given evolutionary divergence and a null

hypothesis, H_0, that the sequences are unrelated and any observed similarity is due to chance. In such matrices, $S[j,k]$ is the logarithm of the likelihood ratio:

$$S[j,k] = \log_2 \frac{P[j \text{ aligned with } k|H_a]}{P[j \text{ aligned with } k|H_0]}. \tag{1}$$

A PSSM is similarly defined in a log-likelihood framework. In this case, the scoring matrix is used to score a candidate instance of a pattern in a single sequence. The focus is on the observation of a particular amino acid at a particular position in the pattern. We derive a *propensity* matrix, P, representing the likelihood ratio

$$P[i,j] = \frac{q[i,j]}{p[i]}, \tag{2}$$

where $q[i,j]$ is the probability of observing amino acid i at position j in the pattern under the alternate hypothesis. The probability of the same event under the null hypothesis is $p[i]$, the background distribution of amino acid i.

To complete the definition of $P[i,j]$, we still need an expression for the numerator, $q[i,j]$. Given an ungapped local alignment, A, representing k instances of a motif of width w, the frequency of amino acid $i \in \Sigma$ at position j in the alignment is

$$q[i,j] = \frac{c[i,j] + b}{k + b \cdot |\Sigma|}, \tag{3}$$

where $c[i,j]$ is the number of i's at position j and b is a pseudocount. Note that A is a $k \times w$ matrix and $q[i,j]$ is a $|\Sigma| \times w$ matrix.

Pseudocounts are introduced to account for possible examples of the pattern that are not represented in the training data, i.e., the matrix A. It is possible that a particular amino acid, i, can occur at position j in this motif, but that this case does not arise in any of the sequences in A. If $P[i,j] = 0$, then the resulting PSSM will assign a score of zero to any sequence with an i at position j, preventing the future discovery of this variant of the motif. To account for this, pseudocounts are used to give every i,j combination a small, non-zero probability. The normalization in the denominator is adjusted accordingly by the term $b \cdot |\Sigma|$. In the example used in class, we assumed that $b = 1$. For those interested in exploring this further, a more general treatment of pseudocounts is given in Section 5.6 of Durbin's book. This is not required for the course.

The log odds scoring matrix is

$$S[i,j] = \log_2 P[i,j]. \tag{4}$$

Given a new, unlabeled sequence, t, of length n, we can search it for an instance of the pattern by

scoring each position in the sequence as follows:

$$\mathcal{S}[t,o] = \sum_{j=1}^{w} S[t[j+o],j], \tag{5}$$

where the offset, o, ranges from 0 to n-w. The offset with the highest score is most likely to be an instance of the motif. To be convincing, the score must also be high in an absolute sense, not just higher that the scores associated with other offsets. When there may be more than one instance of the motif in t, offsets with near optimal scores should also be considered.

Note that the score of a window of length w at offset o in t, is a log likelihood ratio

$$\mathcal{S}[t,o] = \log_2 \frac{\Pr(\text{motif at offset } o|H_a)}{\Pr(\text{motif at offset } o|H_0)}, \tag{6}$$

where H_a is the alternate hypothesis that t contains the pattern at offset o and H_0 is the null hypothesis (no pattern, background frequencies). To see this, consider that

$$\begin{aligned}
\mathcal{S}[t,o] &= \sum_{j=1}^{w} S[t[j+o],j] \\
&= \sum_{j=1}^{w} \log_2 P[t[j+o],j] \\
&= \sum_{j=1}^{w} \log_2 \frac{q[t[j+o],j]}{p(t[j+o])} \\
&= \log_2 \prod_{j=1}^{w} \frac{q[t[j+o],j]}{p(t[j+o])} \\
&= \log_2 \frac{\prod_{j=1}^{w} q[t[j+o],j]}{\prod_{j=1}^{w} p(t[j+o])}.
\end{aligned}$$

The numerator is the probability that the w residues starting at position $o+1$ in t represent an instance of the motif. The denominator is the probability of observing those residues under the null hypothesis.

Gibbs sampler for motif discovery

The *Gibbs sampler* is an algorithm for *discovery* of ungapped local alignments that uses the PSSM formalism as its basic data structure. The application of the Gibbs sampler for motif finding in biomolecular sequences was proposed first by Chip Lawrence and his colleagues in 1993 (Lawrence

et al., Science. 1993 262(5131):208-14.) The Gibbs sampler is a general method for estimating a joint probability distribution by repeated calculations of a conditional distribution, using a Markov Chain Monte Carlo (MCMC) approach. For those interested in more theoretical aspects, Ewens and Grant discuss the Gibbs sampler for biomolecular motif discovery in the MCMC framework in their book (Section 10.5 in the first edition). A general introduction to the Gibbs sampler in a statistical context can be found in *Explaining the Gibbs sampler*, G. Casella & E. I. George, The American Statistician, 46:167-174, 1992, listed under "optional readings" on the syllabus page. These readings are not required for the course.

The Gibbs sampler takes as input k sequences, $t_1 \ldots t_k$, of lengths $n_1 \ldots n_k$. The output is a set of k subsequences, one in each input sequence, that are "most similar" to each other. Here, our measure of "most similar" is a likelihood function derived from the propensity matrix, P, defined in Equation 2. Note that the Gibbs sampler assumes that the sequences share an ungapped pattern of length w and that each sequence contains exactly one instance of this pattern. The length of the pattern, w, must either be supplied by the user or determined during the discovery process.

A brute force approach to identifying such a pattern is exhaustive enumeration: Consider all possible sets of starting positions or *offsets* $\{o_1 \ldots o_k\}$, where $0 < o_z < n_z - w + 1$. Here the offset refers to the position *before* the first symbol in the motif; for a given value of o, the motif is the subsequence from position $o+1$ to $o+w$. For each set of offsets, score the local alignment consisting of the k subsequences of length w,

$$
\begin{aligned}
&t_1[(o_1+1) \ldots (o_1+w)] \\
&t_2[(o_2+1) \cdots (o_2+w)] \\
&\qquad\qquad \ldots
\end{aligned}
$$

and so on. The alignment of the highest scoring pattern is then reported. The computational cost of this approach is prohibitive for all, but the smallest problem instances. The Gibbs sampler is a more efficient approach to searching the space of all possible motifs, which does not require that all possible alignments be considered. Another probabilistic search procedure called *Expectation Maximization (EM)* can also be used to identify conserved, ungapped motifs. We will discuss EM briefly in the context of HMM's later in the course. EM is discussed in detail in 03-712.

We first introduce a "hill-climbing" algorithm for motif discovery that has the same iterative structure as the Gibbs sampler, but is not guaranteed to converge to the global optimum. The algorithm proceeds by iteratively improving an estimate of a conserved motif until no further improvement is seen. Since the motif is ungapped, it is completely defined by the set of offsets $\{o_1 \ldots o_k\}$ that specify the starting points of the motif in each of the k sequences. At each step,

one sequence, t^*, is removed from consideration and a PSSM is obtained from the subsequences of length w starting at the current best estimate of the offsets in the remaining $k-1$ sequences. This PSSM is then used to obtain a new estimate in t^*, by selecting the offset with the highest score.

Algorithm: Hill-Climbing

Input:

Sequences $t_1, \ldots, t_k$ of lengths $n_1, \ldots, n_k$.

Initialization:

$z = 1$ # index of special sequence.
$t^* = t_z, n^* = n_z$ # t1 is the special sequence.
for ($x = 2$ to k) $index[x–1] = x$ # index of non-special sequences
for ($y = 1$ to $k - 1$) {
  $x = index[y]$
  $o_x = rand(1, n_x - w)$ # Guess starting offsets
  $A'[y, 1 \cdots w] = t_x[(o_x+1) \cdots (o_x + w)]$
}
Calculate $P[i, j]$, the propensity matrix of A' with pseudocounts

Search for pattern:

Repeat
{
  $o^* = \max\{\mathcal{S}(t^*, o)\}$ for $o = 0$ to $(n^*–w)$ # Select starting offset in t^*
  $y = rand(1, k–1)$ # Select new special sequence
  $A'[y, 1 \cdots w] = t^*[(o^*+1) \cdots (o^* + w)]$ # Replace new special with $t*$ in $A\prime$
  $r = index[y]$; $index[y] = z$; $z = r$ # store ptr to t^* in $index$
  $t^* = t_z$; $n^* = n_z$ # initialize new t^*
  Calculate $P[i, j]$, the propensity matrix of A' with pseudocounts
  $S[i, j] = \log_2 P]i, j]$
} until($P[\cdot, \cdot]$ stops changing)
Obtain A by adding $t^*[(o^*+1) \cdots (o^* + w)]$ to A'
Compute the log odds scoring matrix, S, from A.

Output:

Local multiple sequence alignment A with scoring matrix S.

In the Hill-Climbing algorithm above, the matrices P and S are the propensity and log odds matrices defined in Equations 2 and 4. The notation $t[x \cdots y]$ is used as shorthand for the substring of a given string, t, starting at position x, up to and including position y. Note that A' and P are $(k–1) \times w$ matrices, whereas the output matrices A and S are $k \times w$ matrices. The use of pseudocounts when calculating P and S is recommended to ensure all symbols in the alphabet are represented.

In each iteration, a row is removed from A' and replaced with a new subsequence of length w from t^*. In order to simplify the book keeping associated with selecting a sequence at random from the $k-1$ sequences represented in the current iteration of A', we introduce an array called `index` that contains the indices of the sequences currently in A'. The row to be removed is selected by generating a random number, r, between 1 and $k-1$; the index of the sequence to be removed is `index[r]`.

The selection of a new offset, o^*, in t^* is a crucial aspect of the convergence of this algorithm. In the Hill-Climbing algorithm, the subsequence with the highest score is selected. Initially, selecting the subsequence with the highest score might seem an attractive strategy, but this could trap the algorithm in a local optimum. Instead, the algorithm selects a window in t^* at random from all windows starting at offsets ranging from 0 to $n^*–w1$. The probability of selecting a particular offset, o, is biased by the probability of the subsequence at that offset so that higher scoring windows have a greater chance of being selected. The probability of the subsequence starting at offset $o+1$ in this context is denoted $pdf(o)$ and is proportional to the score of the subsequence with respect to the current estimate of the propensity matrix.

Selecting a value of o with probability $pdf(o)$ requires a method for obtaining a random number conditioned on an arbitrary probability distribution. This random number can be obtained by calculating the cummulative distribution function

$$cdf(o) = \sum_{i=0}^{n^*-w} pdf(i),$$

a monotonically increasing function with domain $[0, n^*–w]$ and range $[0, 1]$. Its inverse, $cdf^{-1}(r)$, is a function defined on the domain $[0, 1]$, with range $[[0, n^*–w]$. The offset of the new subsequence is defined to be $o* = cdf^{-1}(r)$, where r is an uniformly distributed random number in the interval, $[0, 1]$. The index o^* generated by this procedure is a random number with distribution $pdf(o)$.

Algorithm: Gibbs
Input:

Sequences $t_1, \ldots, t_k$ of lengths $n_1, \ldots, n_k$.

Initialization:

$z = 1$ # index of special sequence.
$t^* = t_z, n^* = n_z$ # t1 is the special sequence.
for ($x = 2$ to k) $index[x–1] = x$ # index of non-special sequences
for ($y = 1$ to $k - 1$) {
  $x = index[y]$
  $o_x = rand(1, n_x - w)$ # Guess starting offsets
  $A'[y, 1 \cdots w] = t_x[(o_x+1) \cdots (o_x + w)]$
}
Calculate $P[i, j]$, the propensity matrix of A' with pseudocounts

Search for pattern:

Repeat
{
  for ($o = 0$ to $(n^*–w)$)
  {
    $$pdf(o) = \frac{\prod_{j=1}^{w} P[t^*[o+j], j]}{\sum_{l=0}^{n^*-w} \prod_{j=1}^{w} P[t^*[l+j], j]}$$
  }
  With probability $pdf[o]$, $o^* = o$ # Select starting offset in t^*
  $y = rand(1, k–1)$ # Select new special sequence
  $A'[y, 1 \cdots w] = t^*[(o^*+1) \cdots (o^* + w)]$ # Replace new special with $t*$ in A'
  $r = index[y]$; $index[y] = z$; $z = r$ # store ptr to t^* in $index$
  $t^* = t_z$; $n^* = n_z$ # initialize new t^*
  Calculate $P[i, j]$, the propensity matrix of A' with pseudocounts
} until($P[\cdot, \cdot]$ stops changing)
Obtain A by adding $t^*[(o^*+1) \cdots (o^* + w)]$ to A'
Compute the log odds scoring matrix, S, from A.

Output:

Local multiple sequence alignment A with scoring matrix S.

Potential pitfalls: There are various potential pitfalls associated with the Gibbs sampler, as with any algorithmic attempt to discover biological truth. For one thing, you could find a statistically significant or biologically meaningful pattern that is not the pattern you are looking for. In addition, problems can arise if a sequence has no copy of the pattern or has more than one copy.

Using this algorithm to obtain meaningful solutions requires a number of decisions that are not programmatically determined and require *ad hoc* solutions, possibly guided by the user's biological intuition:

- Selecting the window size, w
- Selecting the starting configuration
- Selecting values for pseudocounts
- Termination condition: how should the algorithm decide when to stop?

These issues are discussed in greater detail in Lawrence et al. (1993), which is available via the "optional readings" column of the syllabus.

Convergence: The Gibbs sampler models the search for an optimal local alignment as a Markov Chain, in which each state is a set of k subsequences of length w. It can be shown that this Markov Chain has a stationary distribution and that the state corresponding to the most likely pattern has high probability in that distribution. In theory, this process is guaranteed to converge to the optimal solution, given "enough time." In practice, the sampler can get stuck in local optima. An approach to avoiding this problem is to run the procedure several times with different starting configurations. This is discussed in greater detail in the materials listed under "optional reading."