Notes on Sequence Patterns

1 Patterns and Pattern Classes

By *sequence pattern* (sometimes called a "motif") we mean a set of short sequences that is precisely specified by some formalism; a long sequence is said to *match* such a pattern if it contains a segment from that set. A *pattern class* is the set of sequence patterns specified by some formalism. For instance, the set containing only the 6-mer `ACGTAC` is a pattern, which is matched by all sequences containing that 6-mer. The set of all 4096 6-mer patterns is a pattern class.

For another example of a pattern class, consider all 6-mers of nucleotide classes ("ambiguous nucleotides"). For each of the 15 possible non-empty subsets of {`A`,`C`,`G`,`T`} there is a standard symbol. For instance, {`A`,`C`,`G`,`T`} is denoted `N` (perhaps for aNything) and {`A`,`G`} is denoted `R` (for puRine). Thus, one pattern in this pattern class consists of the set of 6-mers specified by `ACRTAC`, i.e., the set containing `ACGTAG` and `ACATAG`.

A more general pattern class is the *regular expressions*, which allows ambiguous nucleotides and variable-sized spacers. (Biologists tend to use a more limited class of regular expressions than considered by computer scientists.) The pattern element `N*` or `.*` denotes an arbitrary string of symbols (possibly of length 0), `N[3-5]` might denote any string of between 3 and 5 letters, and $\mathtt{N}^4$ might denote any string of length 4. For instance, the regular expression $\mathtt{CGGN}^4\mathtt{SWN}^5\mathtt{CCG}$ is a pattern that might be used for GAL4 binding sites.

A fourth pattern class consists of 6-mers allowing one insertion or deletion. For instance, one pattern in this class is specified by "`ACGTAG` allowing one indel", which consists of `ACGTAC` itself, plus the six 5-mers that can be obtained from it by deleting one letter, plus all the 7-mers that can be obtained by inserting one letter, such as `ACGATAC`.

2 Pattern Matching and Pattern Discovery

For any pattern class, several questions arise.

1. How well does the pattern class represent biological features of interest? That is, given a set of short DNA sequences defined by some biological criterion, say, the ones bound by a certain transcription factor, how similar is that set to the most similar set in that pattern class? For instance, 6-mers of ambiguous nucleotides do a reasonable job of representing the binding sites of some transcription factors (in the sense that one pattern in that class provides a useful approximation to the binding sites), but they work poorly for e.g., GAL4. In particular, if the set of sequences that we wish to approximate aren't all of the same length, this may strongly influence the choice of pattern class.

2. How efficient is the process of pattern matching? Given a pattern and a long sequence, we want to determine if the sequence matches the pattern. Usually, pattern matching is not a serious problem in practice.

3. How efficient is the process of pattern discovery? In the simpler version of this problem, we're given a set of short sequences of biological interest, e.g., determined by some experiment to be bound be a certain protein, and we want to represent them by a pattern from the given class. In the more difficult version, we're given a set of long sequences and want to find the pattern or patterns from the given class whose matches are over-represented in the given sequences,

relative to some background population of long sequences. For pattern classes that contain relatively few patterns, such as 6-mers, it may be feasible to perform pattern discovery by applying pattern matching with every pattern in the class. For pattern classes that contain an infinite number of pattern specifications, such as weight matrices, that is clearly impossible.

One pattern class is *more expressive* than another if it includes every pattern in the other class. For instance, 6-mers with ambiguous nucleotides are more expressive than 6-mers of nucleotides. In general, a more expressive pattern class will be better for representing biologically important sets of short sequences, but at increased cost of pattern matching and especially pattern discovery.

For certain pattern classes, such as regular expressions or sets of k-mers, computer scientists have invented very clever methods for pattern matching. These methods might helpful for implementing pattern discovery by exhaustive pattern matching. One such method is called *position trees* or *suffix trees*. The long sequence being searched is "pre-processed" by placing it into a certain data structure. This can be done in time proportional to the length of the sequence. Computer space is somewhat of a problem, but the yeast genome can be processed this way on a workstation. Given that data structure, all positions that match a given 6-mer (or any short sequence) can be found in time proportional to the length of the short sequence plus the number of matches (i.e., proportional to the time necessary to read the pattern and print the output). The Vilo *et al.* paper uses an improvement on the original methods that allows searches for a class of patterns that is somewhat more general than just nucleotide k-mers.

For regular expressions (even the more general class considered by computer scientists), efficient pattern matching begins by pre-processing *the pattern specification*. After that, pattern matching runs in time proportional to just the length of the sequence being searched (i.e., independent of the size of the pattern). It is somewhat surprising that you can search for the matches to a set of 100 6-mers about as efficiently as you can search for matches to a single 6-mer. Some work has been done on pattern *discovery* for regular expressions, but other than the approach used by Vilo *et al.* (which treats a very limited class of regular expressions), I don't believe it has found much application in bioinformatics.

3 More General Pattern Classes

Two of the most popular pattern classes in bioinformatics are "position weight matrices" (PWMs) and "hidden Markov models" (HMMs). To a first approximation, HMMs are an extension of PWMs that permits gaps. PWMs are particularly popular for representing protein binding sites in genomic DNA sequences, whereas an HMM are typically used for representing a protein family or domain, consisting of highly conserved segments separated by less conserved, variable-length linkers (hence the need to handle gaps). Here we describe PWMs in detail; HMMs are covered in Bioinformatics I. Before digging into PWMs, we mention a few research questions in this important and active field.

- The favorite way to do pattern discovery for PWMs, called the Gibbs Sampler (described below), is highly probabilistic, i.e., there is no guarantee that the theoretically optimal solution will be found in some pre-specified amount of time. Some current projects aim to develop "deterministic" pattern discovery methods, e.g., for a pattern class that is more expressive than the one handled by the position-tree method of Vilo *et al.*.

- Sometimes, other information is available that may permit a more useful problem to be solved. For instance, suppose that the long DNA sequences being searched are from the corresponding (i.e., orthologous) regions of several species (as opposed to coming from the same genome). If a phylogenetic tree for those species is known, it can provide additional data for identifying matching segments. In particular, the Gibbs Sampler treats sequences in a symmetric manner (all sequences are equal), whereas it may be appropriate to treat evolutionarily distant sequences in a manner different than that for similar sequences.

4 The Gibbs Sampler

Currently, the "Gibbs Sampler" is perhaps the most popular computational approach for identifying potential binding sites in a collection of upstream regions. In essence, it looks for a PWM with k columns and a score-threshold such that the set of k-mers matching the matrix is highly unlikely to occur by chance. (After finding such a PWM, it can cross out matching k-mers, then repeat the computation to find other conserved regions.) Websites for this kind of program and for MEME, which uses a related "expectation maximization" algorithm, are given at the class website.

Here we sketch the Gibbs Sampler algorithm. Recall that a set of similar k-mers can be summarized by a *frequency matrix* with 4 rows and k columns, such as the following. (There is an extra column at the start to indicated the relevant nucleotide.)

A	0.1	0.6	0.5	0.1	0.4
C	0.2	0.1	0.1	0.2	0.1
G	0.2	0.1	0.3	0.3	0.4
T	0.5	0.2	0.1	0.4	0.1

Intuitively, the frequency matrix represents a DNA pattern, and it can the thought of in several ways. First, it provides a statistical model for generating random k-mers that look like the pattern (the sum of entries in each column is 1). For instance, with the above matrix, the second entry of the k-mer is `A` with probability 0.6. Alternatively, we can compute the probability that a particular k-mer is generated by the matrix. Note that we then *multiply* the matrix entries. For instance, the probability that `TAAGA` is generated equals the probability that the first entry is `T` times the probability that the second entry is `A`, and so on, i.e., $0.5 \times 0.6 \times 0.5 \times 0.3 \times 0.4 = 0.018$. Thus, the matrix assigns a number to each k-mer, and the sum of those numbers over all 4^k k-mers is 1.

Gibbs Sampler Algorithm:

1. Each iteration starts with one k-mer from each of the m sequences.
2. Pick one of the k-mers at random, say the one from the r-th sequence.
3. Create a frequency matrix Q from the remaining $m-1$ k-mers..
4. For each k-mer w in the r-th sequence, determine the probability p_w that it is generated by Q, as in the preceding paragraph.
5. Choose a k-mer from the r-th sequence randomly, with probably proportional to p_w, and add it to the $m-1$ other k-mers.
6. Go to step 2.

As you might expect, the above algorithm is just an outline of what the programs really do. First, conditions for terminating the loop are needed; the algorithm is guaranteed to converge to a local (but not always global) maximum, but the iteration needs to be stopped at some point. Other important issues that need to be addressed include how to pick k (the pattern length) and how to pick m, since we frequently don't know how many upstream regions contain matches to the pattern and since we may want to allow for several matches within one upstream region. Also, there needs to be a reasonable measure of statistical significance determined for each matrix (or rather, for the set of k-mers that match the matrix to within the specified threshold).