Optimally Parsing a Sequence into Different Classes Based on Multiple Types of Evidence

Gary D. Stormo[1] and David Haussler[2]
[1] Dept. of Molecular, Cellular and Developmental Biology
University of Colorado, Boulder, CO 80309-0347
phone: 303-492-1476, FAX: 303-492-7744
stormo@beagle.colorado.edu
[2] Dept. of Computer and Information Sciences
University of California at Santa Cruz, Santa Cruz, CA 95064
phone: 408-459-2015, FAX: 408-459-4829
haussler@cse.ucsc.edu

Abstract

We consider the problem of parsing a sequence into different classes of subsequences. Two common examples are finding the exons and introns in genomic sequences and identifying the secondary structure domains of protein sequences. In each case there are various types of evidence that are relevant to the classification, but none are completely reliable, so we expect some weighted average of all the evidence to provide improved classifications. For example, in the problem of identifying coding regions in genomic DNA, the combined use of evidence such as codon bias and splice junction patterns can give more reliable predictions than either type of evidence alone.

We show three main results:

1. For a given weighting of the evidence a dynamic programming algorithm returns the optimal parse and any number of sub-optimal parses.
2. For a given weighting of the evidence a dynamic programming algorithm determines the probability of the optimal parse and any number of sub-optimal parses under a natural Boltzmann-Gibbs distribution over the set of possible parses.
3. Given a set of sequences with known correct parses, a dynamic programming algorithm allows one to apply gradient descent to obtain the weights that maximize the probability of the correct parses of these sequences.

Introduction

Biological macromolecules, such as DNA, RNA and protein, are long strings from a few types of monomeric units; DNA and RNA are each made from four different nucleic acid bases and proteins are made from twenty different amino acids. It is both convenient and appropriate to consider the macromolecular strings as being "texts" derived from their alphabets and methods developed for string analysis can be usefully applied (Sankoff & Kruskal 1983). The strings can often be divided into substrings, or domains, based on structural and/or functional criteria. For example, DNA sequences contain regions that code for genes, regulatory regions that affect the expression of those genes, intergenic regions between the genes and a variety of other types of regions such as replication origins, repetitive DNA segments and structural domains like centromeres and telomeres. RNA sequences can be divided into structural domains such as helices and loops, and pre-mRNA sequences can also be divided into functional domains of coding regions (exons) and intervening sequences (introns). Protein sequences can be divided into structural domains such as α-helices and β-sheets, and into functional domains described by various motifs (Henikoff & Henikoff 1991).

Dividing a sequence into its separate domains, or parsing it, can be very useful in understanding important properties of the macromolecule. Auger and Lawrence (Auger & Lawrence 1989) and Sankoff (Sankoff 1992) have published dynamic programming methods to optimally parse a sequence into domains given some types of patterns that describe the different classes of domains. These are closely related to our first result, except that we extend the method to obtain ranked suboptimal solutions. Gelfand and Rotyberg (Gelfand & Roytberg 1993) have also published a method for identifying exons and introns in genomic DNA that allows the identification of suboptimal solutions. In addition to that, we explicity develop the method to utilize mutliple types of evidence in arriving at the optimal parses. Snyder and Stormo (Snyder & Stormo 1993) have previously given a dynamic programming algorithm for optimally parsing a DNA sequence into exons and introns given the evidence from multiple statistical tests for each

type. Our first result is closely related to theirs and to further work by those authors (Snyder 1994; Batra 1993).

The use of multiple types of evidence is important in these problems. Often there are several different types of sequence features that can be used, but none are completely reliable. Using them in combination can improve classification performance. For example, in the problem of identifying the exons and introns in DNA sequences, there are ambiguous nucleotide patterns that serve as "signals" at the juntions of the two types of sequences. However, these patterns are ambiguous enough that there are many false positive occurences for every true occurence. Exons and introns also differ in certain "content" statistics; they have different overall patterns of nucleotide distributions (see (Fickett & Tung 1992) for several examples of content statisitics). However, the distributions overlap such that statistical tests are not sufficient for reliable classification. Used in combination, the "signal" and "content" methods provide increased reliability (Snyder & Stormo 1993; Stormo 1987; Brunak, Engelbrecht, & Knudsen 1991; Fields & Soderlund 1990; Guigo *et al.* 1992; Uberbacher & Mural 1991).

The main difficulty with employing multiple types of evidence is determining how to weight the various types. In GRAIL (Uberbacher & Mural 1991) a neural network is used to weight different "sensors" that each monitor a statisical measure of exon content. Snyder and Stormo (Snyder & Stormo 1993) also use a neural network to obtain weights that optimize the performance of the dynamic programming algorithm. In this paper we introduce a probabilistic method to optimize these weights, and give a more general interpretation of the objective function. Our intent is to provide a useful theoretical framework to help guide the design of these types of parsing methods for biosequences.

For simplicity this paper will use the example problem of identifying exons and introns in genomic DNA sequences. However, it should be kept in mind that the same methodology can easily be applied to the problem of protein secondary structure prediction, and many other biosequence analysis problems.

Notation

First we fix a sequence S of letters from the alphabet A,C,G,T.

The letters of S are numbered from $1 \ldots l$, where l is the length of S.

From the sequence S several T matrices are created, one for each statistic μ in a collection "exon statistics" and one for each statistic κ in a collection of "intron statistics". For each i and j, where $1 \leq i < j \leq l$, $T_\mu(i,j)$ is a real number (possibly negative) that we call the "score" for the exon statistic μ over the region $i \ldots j$ of the sequence S. If μ is a good exon statistic, then the more the region of S from $i \ldots j$ looks like an exon, the larger $T_\mu(i,j)$ will be. Statistics that measure similarity to introns are also defined in an analogous way, and these are stored in matrices $T_\kappa(i,j)$. Note that these, and other matrices, need only be stored as half-matrices for which $j > i$.

The set of T matrices are combined into two other half-matrices:[1]

$$L_E(i,j) = \sum_\mu w_\mu T_\mu(i,j)$$

$$L_I(i,j) = \sum_\kappa w_\kappa T_\kappa(i,j)$$

These represent composite exon and intron statistics, respectively, based on the underlying statistics $\{T_\mu\}$ and $\{T_\kappa\}$. The *weights* $\{w_\mu\}$ and $\{w_\kappa\}$ are adjustible real numbers, i.e. parameters of the composite exon and intron statistics.

A *parse* ϕ of the sequence S is a partition of S into consecutive nonempty regions that are alternately labeled as either exon or intron. A parse can start with either an exon or an intron region, and can also end with either an exon or an intron region. Hence there are 2^l possible parses of a sequence S of length l (each possible assignment of "exon" and "intron" to the individual letters of S corresponds to a distinct and unique parse.)[2]

The *score* of a parse ϕ, denoted $\Theta(\phi)$, is defined to be the sum of the L_E and L_I values for the regions of S that make up the parse. An example is provided in Figure 1; the parse has exons from $1..h$, $i..j$ and $k..l$, and introns in between. The score is the sum of the L_E and L_I matrix elements corresponding to those regions:

$$\begin{aligned}\Theta(\phi) &= L_E(1,h) + L_I(h+1,i-1) + L_E(i,j) \\ &\quad + L_I(j+1,k-1) + L_E(k,l) \qquad (1)\end{aligned}$$

Finding the Optimal Parse

Given the L matrices, a simple dynamic programming algorithm can find the parse of S with the highest score (called the "best parse"). The constraints that apply are that exons and introns alternate and are adjacent

[1] Without loss of generality, we can assume that one of the exon statistics is the constant function 1, i.e. $T(i,j) = 1$ for all i,j for this statistic. The weight for this statistic then plays the role of the additive constant, or bias, for exons. The analogous thing can be assumed for the intron statistics.

[2] For simplicity we assume that the entire sequence will be parsed into exon and intron sequences. In reality there are likely to be regions on both ends of the sequence that do not constitute complete exons and introns so that the correct parse may only cover a "local" region of the sequence rather than the complete sequence. The method presented here, for the "global" parse of the sequence, can easily be extended to do "local" parses, as described in (Snyder & Stormo 1993).

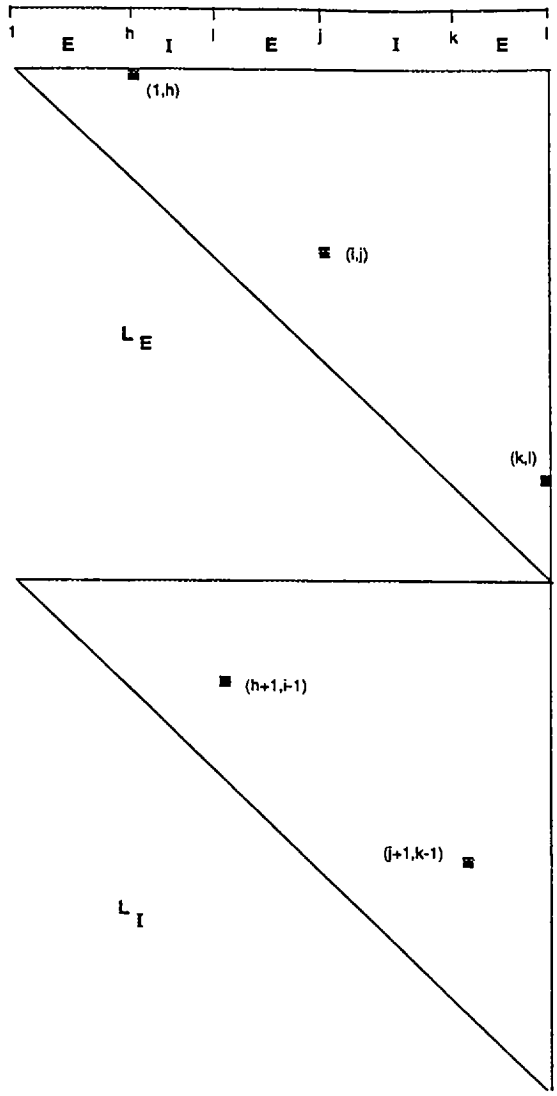

Figure 1: The horizontal line at the top represents the sequence with length l. The sequence is parsed into exons (**E**) in the regions $1..h, i..j, k..l$ and has introns (**I**) between. The two half-matrices store, for all regions, the log-likelihood values for the region being an exon or an intron in L_E and L_I, respectively. The L_E and L_I elements that correspond to the parse shown at the top are indicated.

and non-overlapping. Suppose we have already parsed the prefix of S beginning at position 1 and ending at position j. Such a parse is called a *partial parse*. Let the value $D_E(j)$ denote the score for the best partial parse that ends with an exon that terminates at position j, and the value $D_I(j)$ denote the score for the best partial parse that ends with an intron that terminates at position j. The following recursive definitions of $D_E(j)$ and $D_I(j)$ can be converted into a dynamic programming algorithm (see Auger and Lawrence (Auger & Lawrence 1989), Sankoff (Sankoff 1992) and Snyder and Stormo (Snyder & Stormo 1993)).

$$D_E(0) = D_I(0) = 0$$

$$D_E(j) = \max_{1 \le k \le j} [L_E(k, j) + D_I(k-1)]$$

$$D_I(j) = \max_{1 \le k \le j} [L_I(k, j) + D_E(k-1)]$$

The score for the best parse is $\max[D_E(l), D_I(l)]$. The actual best parse can be determined by backtracking, which is simplified if the k-values used in the maxima are kept along with the D vectors.

Although it gives the same best parse, we will see below that it is also useful to be able to do the recursions in the reverse direction:

$$D'_E(l+1) = D'_I(l+1) = 0$$

$$D'_E(j) = \max_{j \le k \le l} [L_E(j, k) + D'_I(k+1)]$$

$$D'_I(j) = \max_{j \le k \le l} [L_I(j, k) + D'_E(k+1)]$$

These D' vectors store the values of the best solutions *beginning* at position j, and the score of the best solution is $\max[D'_E(1), D'_I(1)]$. Adding the complementary vectors together gives, for every position, the score of the best solution with that position as a boundary. In particular, the score for the best solution with an exon-intron boundary where the exon ends at position j is

$$\Delta_E(j) = D_E(j) + D'_I(j+1)$$

and the analogous score for an intron-exon boundary is

$$\Delta_I(j) = D_I(j) + D'_E(j+1).$$

Note that all of the boundaries that contribute to the optimal solution will have the same Δ-values. If the parse in Figure 1 is the best parse ϕ_{best}, then:

$$\begin{aligned} \Theta(\phi_{best}) &= D_E(l) = D'_E(1) \\ &= L_E(1, h) + L_I(h+1, i-1) + L_E(i, j) \\ &\quad + L_I(j+1, k-1) + L_E(k, l) \\ &= \Delta_E(h) = \Delta_E(j) = \Delta_E(l) = \Delta_I(0) \\ &= \Delta_I(i-1) = \Delta_I(k-1) \end{aligned}$$

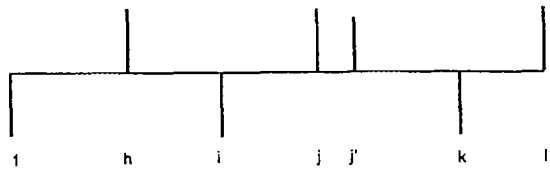

Figure 2: As in Figure 1, the horizontal line corresponds to the sequence of length l. Plotted above and below the sequence line are lines whose lengths are proportional to $\Delta_E(j)$ and $\Delta_I(j)$, respectively. An exon is a region bounded by a lower line on the left and an upper line on the right, and introns are the reverse. Lines are shown only for the optimal parse and for the highest suboptimal parse.

A simple plot of $\Delta(j)$ vs. j can be used similarly to Zuker's suboptimal plots of RNA structure and sequence alignments (Zuker 1989; 1991) to show not only the optimal solution but the best solution for all possible junctions. Figure 2 plots j versus $\Delta_E(j)$ and $-\Delta_I(j)$ for the optimal solution (from the example in Figure 1) and for the highest suboptimal solution, which moves the end of the second exon from j to j'. Additional suboptimals can be plotted on the same graph to indicate which regions are predicted consistently and which regions have multiple parses of approximately equal score (Snyder 1994).

Calculating the Probability of a Parse

We now define a probability distribution on the set of all parses of a given sequence. This can be used to measure *how* suboptimal a given suboptimal parse is, and to detect when most of the probability is concentrated on the single best parse (a good case, since suboptimal parses can be ignored here) or, in contrast, when there are a large number of parses that are all nearly as likely as the best parse (and hence suboptimal parses need to be considered, e.g. for deciding if a given region is an exon).

We define the probability of a parse by interpreting the score Θ as an unnormalized log-likelihood of the joint probability of the sequence S and the parse ϕ. Then for any parse ϕ of S, we define

$$\Pr(\phi|S) = \frac{\Pr(S,\phi)}{\sum_{\text{all parses } \phi'} \Pr(S,\phi')} = \frac{e^{\Theta(\phi)}}{Z} \qquad (2)$$

where $Z = \sum_{\text{all parses } \phi'} e^{\Theta(\phi')}$. This has a natural statistical physics interpretation as a Boltzmann-Gibbs distribution. Let $H(\phi) = -\Theta(\phi)$. The quantity $H(\phi)$ corresponds to the energy[3] of the state configuration (parse) ϕ. The normalizing constant Z is called the *partition function* in statistical physics. Calculating the probability of a given parse, including the best parse, requires calculating Z. While in most statistical physics models this calculation rapidly becomes intractable, here it is remarkably simple. It can be done with a modified version of the dynamic programming algorithm given above.

Let $V_E(j)$ be the sum of $e^{\Theta(\phi)}$ over all partial parses ϕ of the prefix of S from the beginning up to position j and ending with an exon at j. Let $V_I(j)$ be defined similarly for introns. These new V vectors can be computed according to the following recursive equations:

$$V_E(0) = V_I(0) = 1$$

$$V_E(j) = \sum_{1 \le k \le j} e^{L_E(k,j)} V_I(k-1)$$

$$V_I(j) = \sum_{1 \le k \le j} e^{L_I(k,j)} V_E(k-1)$$

It is easily verified that

$$Z = V_E(l) + V_I(l)$$

The probability of the best parse is easily calculated from this result and equation (2), as is the sum of the probabilities of any number of parses. In particular, in analogy with the function Δ defined above, we can calculate the probability that the parse of S has an exon ending at position j by summing the probability of all parses of S that have this property. In order to do this calculation efficiently, we do the reverse recursive calculations of the V values just as we did reverse calculations of the D values in the previous section. Call these reverse V-values $V'_E(j+1)$ (the sum of $e^{\Theta(\phi)}$ over all partial parses ϕ of the suffix of S from position $j+1$ to the end that start with an exon at position $j+1$) and V'_I (defined similarly for introns). Then it is easily verified that the probability that the parse of S has an exon ending at position j is given by

$$p_E(j) = \frac{V_E(j)V'_I(j+1)}{Z}.$$

This method of calculating these probabilities is closely related to the use of the *forward-backward* algorithm to compute analogous probabilities in Hidden Markov Models (Rabiner 1989).

Finally, we note that in practice all of the probabilities discussed in this section would be computed and displayed as log probabilities or negative log probabilities, to avoid problems with the large dynamic range of the raw probabilities. The negative log probability of a parse yields a positive score function that is minimized to obtain the best parse.

[3] If we want to make sure that energy is never negative, then we can define $H(\phi) = -\Theta(\phi) + c$ for a suitably large constant c. This doesn't change the theory, since subtracting a constant from $\Theta(\phi)$ for each ϕ does not change the value of $Pr(\phi)$ in equation (2).

Determining the Best Weights

Given a collection of N sequences, $S_1, \ldots, S_N$, for each of which we know the correct parse (these sequences are the "training set"), we want to find the weighting of the different types of evidence that optimizes the performance of the algorithms described above. Snyder and Stormo (Snyder & Stormo 1993) describe the use of a neural network to obtain weights that maximize the proportion of the optimal parses that are correct. They start with arbitrary weights and obtain the optimal parse. Optimal parses that differ from the correct parse are used as training examples to find new weights that give higher scores to the correct parses. Those weights are used in another round of obtaining optimal parses which generate new training examples. This procedure is iterated until there is a plateau in the performance, which is judged by the fraction of optimal parses that are correct. Our objective is somewhat different. We take a maximum likelihood approach, so we want to find the weights that maximize the probability of the correct parse so that, even in cases where the optimal parse is not correct, the correct parse has as high a probability as possible.

Assume that sequence S_i has length l_i, with the bases numbered consecutively from $1 \ldots l_i$. Let $\mathbf{w}$ denote the vector of all parameters $\{w_\mu\}$ and $\{w_\kappa\}$. For any parse ϕ of the sequence S_i, the probability of this parse is denoted $\Pr(\phi|S_i, \mathbf{w})$, and is defined as in equation (2). We call the correct parse ϕ_i^* for each training sequence S_i. We use the training set to compute maximum likelihood estimates for the parameters $\{w_\mu\}$ and $\{w_\kappa\}$, which are those that minimize

$$\sum_{i=1}^{N} -\ln \Pr(\phi_i^*|S_i, \mathbf{w})$$

This might be accomplished by many methods; we will describe the method of gradient descent. This means that we will repeatedly loop through all the training examples, and for each training example S with correct parse ϕ^*, we will modify each parameter w in $\mathbf{w}$ by the rule

$$w := w + \eta \frac{\partial \ln \Pr(\phi^*|S, \mathbf{w})}{\partial w} \tag{3}$$

where $\eta > 0$ is a suitable "learning rate". To do this, we need to calculate the above partial derivatives. For simplicity, we will drop the conditioning on S and $\mathbf{w}$ in our notation. We have

$$\begin{aligned} \frac{\partial \ln \Pr(\phi^*)}{\partial w} &= \frac{\partial}{\partial w} \ln \frac{e^{\Theta(\phi^*)}}{Z} = \frac{\partial \Theta(\phi^*)}{\partial w} - \frac{\partial \ln Z}{\partial w} \\ &= \frac{\partial \Theta(\phi^*)}{\partial w} - \frac{Z'}{Z} \end{aligned} \tag{4}$$

where

$$Z' = \sum_{\text{all parses } \phi} \frac{\partial \Theta(\phi)}{\partial w} e^{\Theta(\phi)}.$$

This equation tells us that when adapting the weights, the change in w should be proportional to the derivative of the score of the correct parse minus the derivative of the average score over all parses.

The derivative of the score of the best parse is easy to calculate, since the score function Θ is linear in each adjustible weight parameter w. To calculate the derivative of the average score, we again use dynamic programming. We state the recursive equations for this in their most general form, to show that the method also works for more complex score functions in which the weight parameters enter in a nonlinear fashion, e.g. as in multilayer neural nets. The only requirement is that the score $\Theta(\phi)$ be defined as the sum of the L_E and L_I values of the exons and introns in ϕ as above, and that the derivatives of these L_E and L_I values with respect to each of the adjustible parameters w be efficiently computed.

The derivative of the average score is Z'/Z. This can be computed by first computing Z, which we did in the previous section, and then Z', which is the unnormalized weighted average of the derivative. Before deriving the algorithm to compute Z', it is worthwhile considering a simple example. Imagine a parse ϕ that contains within it an exon from $k..j$. We will denote the partial parse that ends with that exon by $\phi(j)$ and its score by

$$\Theta(\phi(j)) = \Theta(\phi(k-1)) + L_E(k, j),$$

the score for the exon from $k..j$ and the score of the partial parse ending at $k-1$. The unnormalized likelihood of that partial parse is

$$e^{\Theta(\phi(j))} = e^{\Theta(\phi(k-1))} e^{L_E(k,j)}$$

and the derivative of the score of that partial parse is

$$\frac{\partial \Theta(\phi(j))}{\partial w} = \frac{\partial \Theta(\phi(k-1))}{\partial w} + \frac{\partial L_E(k, j)}{\partial w}.$$

The unnormalized weighted derivative is just the product of those last two equations:

$$\begin{aligned} \frac{\partial \Theta(\phi(j))}{\partial w} e^{\Theta(\phi(j))} &= e^{L_E(k,j)} \Big[\frac{\partial \Theta(\phi(k-1))}{\partial w} e^{\Theta(\phi(k-1))} \\ &\quad + \frac{\partial L_E(k, j)}{\partial w} e^{\Theta(\phi(k-1))} \Big]. \end{aligned} \tag{5}$$

Now, in order to compute Z' let us define $U_E(j)$ to be the unnormalized weighted average derivative over all partial parses with an exon ending at position j, and define $U_I(j)$ similarly for introns. We use the notation $\phi(j)_E$ for the partial parse ending with an exon at position j, and similarly $\phi(j)_I$ for the partial parse ending with an intron at j. Then

$$U_E(0) = U_I(0) = 0$$

and using equation (5)

$$U_E(j) = \sum_{\phi(j)_E} \frac{\partial \Theta(\phi(j))}{\partial w} e^{\Theta(\phi(j))}$$

$$= \sum_{1\le k\le j}\sum_{\phi(j)_E} e^{L_E(k,j)}[\frac{\partial\Theta(\phi(k-1))}{\partial w}e^{\Theta(\phi(k-1))} + \frac{\partial L_E(k,j)}{\partial w}e^{\Theta(\phi(k-1))}]$$

$$= \sum_{1\le k\le j} e^{L_E(k,j)}[\sum_{\phi(k-1)_I}\frac{\partial\Theta(\phi(k-1))}{\partial w}e^{\Theta(\phi(k-1))} + \frac{\partial L_E(k,j)}{\partial w}\sum_{\phi(k-1)_I} e^{\Theta(\phi(k-1))}]$$

$$= \sum_{1\le k\le j} e^{L_E(k,j)}[U_I(k-1) + \frac{\partial L_E(k,j)}{\partial w}V_I(k-1)]$$

where V_I is the function computed in the previous section. Similarly,

$$U_I(j) = \sum_{1\le k\le j} e^{L_I(k,j)}[U_E(k-1) + \frac{\partial L_I(k,j)}{\partial w}V_E(k-1)].$$

Finally, it is clear that

$$Z' = U_E(l) + U_I(l),$$

hence these recursive equations give an efficient method to compute Z'.

Conclusions

We have addressed the problem of parsing a sequence into a set of classes. We assume that for each class we have some statistical evidence, in the form of log-likelihoods, that can help decide if any subsequence belongs to that class. We show that, given a weighting of the evidence, we can find the optimal parse and any number of ranked suboptimal parses. We further show that the probability of any parse, or collection of parses, can also be calculated efficiently. Finally, we show that, given a set of sequences with known parses, we can use them as a training set to arrive at the weights for the different types of evidence that maximize the probablity of the correct parses. While the example we used was parsing genomic DNA into exons and introns, the method can be applied to many related problems as well, such as the problem of parsing protein sequences into α-helix, β-strand and coil regions. We can also use an analogous method to find the substitution and indel parameters that maximize the probability of the correct alignments of a set of sequences.

Apart from extensive testing of this method, this work still leaves a number of directions for further research. One would be to try to extend the training algorithm so that it can use some unparsed training sequences along with the ones for which parses are available, as these unparsed sequences will in general be much easier to obtain. One possibility would be to try an Expectation-Maximization algorithm, treating the parse as missing data. Analogous methods are used in Hidden Markov Model training algorithms (Rabiner 1989) (see also (Bourlard & Morgan 1993)). Another direction is to explore more complex score functions. We have already mentioned that the methods described here can easily be extended to score functions in which the L_E and L_I values are smooth nonlinear functions of the weights w. It is also possible to handle more complex constraints on the order in which different kinds of domains can appear in the parse. In this paper we just considered alternating exons and introns. However, in more general situations, with several kinds of domains, we can use a grammar to specify which orderings of the domains are possible, and then restrict attention to only the parses that can be derived from the grammar. If we make the grammar probabilistic, this leads to a mutual generalization of this work and the work on stochastic context-free grammars for biosequence analysis (Sakakibara *et al.* 1994). The details of this remain to be worked out.

A much more complex situation arises when distant pairs of domains in the parse can interact, such as the β-strands in a parse of a protein sequence. A good score function should take into account the evidence for such interactions. However, it appears that when one defines such a score function, then the simple dynamic programming methods developed here can no longer be used. Some possibilities for remedying this problem might be found by somehow marrying our approach with either the branch-and-bound approach taken Lathrop and Smith in their work on threading/parsing proteins (Lathrop & Smith 1994) or the Gibbs sampling approach (Geman & Geman 1984; Lawrence *et al.* 1993; Bryant & Lawrence 1993).

Acknowledgements

We thank Anders Krogh, Yoram Singer, Eric Snyder and Sajeev Batra for discussions of this work and for implementing and testing some of the ideas presented. We thank Chip Lawrence for comments on a draft. G.D.S. was supported by grants from NIH (HG00249) and DOE (ER61606). D.H. was supported by NSF grants CDA-9115268 and IRI-9123692.

References

Auger, I. E., and Lawrence, C. E. 1989. Algorithms for the optimal identification of segment neighborhoods. *Bull. Math. Biol.* 51:39–54.

Batra, S. 1993. A new algorithm for protein structure prediction: using neural nets with dynamic programming. Master's thesis, University of Colorado, Boulder.

Bourlard, H., and Morgan, N. 1993. *Connectionist Speech Recognition, a hybrid approach.* Kluwer.

Brunak, S.; Engelbrecht, J.; and Knudsen, S. 1991. Prediction of human mrna donor and acceptor sites from the dna sequence. *J. Mol. Biol.* 220:49–65.

Bryant, S. H., and Lawrence, C. E. 1993. An empirical energy function for threading protein sequence through the folding motif. *Proteins* 16:92–112.

Fickett, J. W., and Tung, C.-S. 1992. Assessment of protein coding measures. *Nucl. Acids Res.* 20:6441–6450.

Fields, C. A., and Soderlund, C. A. 1990. gm: a practical tool for automating DNA sequence analysis. *Computer Appl. Biol. Sci.* 6:263–270.

Gelfand, M. S., and Roytberg, M. A. 1993. Prediction of the exon-intron structure by a dynamic programming approach. *BioSystems* 30:173–182.

Geman, S., and Geman, D. 1984. Stochastic relaxations, Gibbs distributions and the Bayesian restoration of images. *IEEE Trans. on Pattern Analysis and Machine Intelligence* 6:721–742.

Guigo, R.; Knudsen, S.; Drake, N.; and Smith, T. 1992. Prediction of gene structure. *J. Mol. Biol.* 226:141–157.

Henikoff, S., and Henikoff, J. G. 1991. Automated asembly of protein blocks for database searching. *Nucl. Acids Res.* 23:6565–6572.

Lathrop, R. H., and Smith, T. F. 1994. A branch-and-bound algorithm for optimal protein threading with pairwise (contact potential) amino acid interactions. In *Proceedings of the 27th Hawaii International Conference on System Sciences.* Los Alamitos, CA: IEEE Computer Society Press.

Lawrence, C. E.; Altschul, S.; Boguski, M.; Liu, J.; Neuwald, A.; and Wootton, J. 1993. Detecting subtle sequence signals: A Gibbs sampling strategy for multiple alignment. *Science* 262:208–214.

Rabiner, L. R. 1989. A tutorial on hidden Markov models and selected applications in speech recognition. *Proc. IEEE* 77:257–286.

Sakakibara, Y.; Brown, M.; Mian, I. S.; Underwood, R.; and Haussler, D. 1994. Stochastic context-free grammars for modeling RNA. In *Proceedings of the Hawaii International Conference on System Sciences.* Los Alamitos, CA: IEEE Computer Society Press.

Sankoff, D., and Kruskal, J. 1983. *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison.* Addison-Wesley.

Sankoff, D. 1992. Efficient optimal decomposition of a sequence into disjoint regions, each matched to some template in an inventory. *Math. Biosci.* 111:279–293.

Snyder, E. E., and Stormo, G. D. 1993. Identification of coding regions in genomic DNA sequences: an application of dynamic programming and neural networks. *Nucl. Acids Res.* 21:607–613.

Snyder, E. E. 1994. *Identification of Protein Coding Regions in Genomic DNA.* Ph.D. Dissertation, University of Colorado, Boulder.

Stormo, G. D. 1987. Identifying coding sequences. In *Nucleic Acid and Protein Sequence Analysis, A Practical Approach*, 231–258. IRL Press.

Uberbacher, E. C., and Mural, R. J. 1991. Locating protein-coding regions in human dna sequences by a multiple sensor-neural network approach. *Proc. Natl. Acad. Sci. USA* 88:11261–11265.

Zuker, M. 1989. On finding all suboptimal foldings of an RNA molecule. *Science* 244:48–52.

Zuker, M. 1991. Suboptimal sequence alignment in molecular biology alignment with error analysis. *J. Mol. Biol.* 221:403–420.