Optimal DNA signal recognition models with a fixed amount of intrasignal dependency

Broňa Brejová, Daniel G. Brown, and Tomáš Vinař

School of Computer Science, University of Waterloo Waterloo ON N2L 3G1 Canada
{bbrejova, browndg, tvinar}@math.uwaterloo.ca

Abstract. We study new probabilistic models for signals in DNA. Our models allow dependencies between multiple non-adjacent positions, in a generative model we call a higher-order tree. Computing the model of maximum likelihood is equivalent in our context to computing a minimum directed spanning hypergraph, a problem we show is NP-complete. We instead compute good models using simple greedy heuristics. In practice, the advantage of using our models over more standard models based on adjacent positions is modest. However, there is a notable improvement in the estimation of the probability that a given position is a signal, which is useful in the context of probabilistic gene finding. We also show that there is little improvement by incorporating multiple signals involved in gene structure into a composite signal model in our framework, though again this gives better estimation of the probability that a site is an acceptor site signal.

1 Introduction

Accurate detection of DNA signals is essential for the proper identification of important features found in DNA. Here, we study new probabilistic models for detecting such signals based on optimizing directed spanning hypertrees in hypergraphs that represent all possible dependencies of bounded cardinality among positions of the signal. We evaluate performance of our new models on human splice site recognition. Our new models offer modest improvement over existing techniques, most notably in improving the accuracy of the probabilistic model, rather than its usefulness as a classifier.

Our study is motivated by gene finding, where we desire signal detectors based on generative probabilistic models that use only small window around the functional site for their prediction. Altough it is often possible to increase signal detector prediction accuracy by considering properties of the wider sequence (such as the coding potential of the region upstream from donor splice site), such information is already considered in other parts of a gene finder.

Generative probabilistic models commonly used for signal detection include position weight matrices [20, 21], maximum decomposition trees [5], and Chow-Liu trees [8, 7]. Here, we extend these techniques to a wider class of models. Our new models, described in Section 2, allow for more complicated dependencies among the positions of a signal than were previously modeled. In particular, we model dependencies within a signal as a directed hypertree whose nodes are the positions of the signal. The root of the hypertree has no dependencies on other positions, while all other positions are dependent on the positions represented by the tail of the hyperedge that is incident on them. When the hyperedges are just normal directed edges, this is equivalent to the Chow-Liu tree models studied in this context by Cai et al. [7], but allowing multiple dependencies at a position allows more richness in the set of possible signal models being considered.

Unfortunately, computing the optimal signal model for a training set from the set of hypertrees we study is NP-hard, as we show in Section 3. However, one can either use integer programming (which is practical as long as the number of dependencies a position has is small) or simple greedy heuristics in practice. For the problems we have considered, the difference between the optimal model and the one found by greedy heuristics was small.

We have tested our models by using human chromosome 22, which is extremely well annotated. Our experimental results are found in Section 4. In practice, the improvement of our hypertree-based models over more standard signal detection algorithms, such as second order position weight matrices (PWMs), is modest. In this sense, our results are similar to the results of Cai *et al.* [7], who show that tree dependency structures do not improve much over first order PWMs.

However, we are able to offer a notable improvement in the prediction of the probability that a given position is a signal. This us useful for gene finding, given that gene finders typically integrate this predicted probability into their other inference engines. If one can improve reliability of the scores returned by signal detectors, perhaps one can improve *de novo* gene finding as well.

In particular, when tested on a long testing sequence, if our model for donor sites predicts that a possible sequence is a donor with probability 6%, then with probability 6.7% it actually *is* such a signal. In contrast, when the more standard second order PWM predicts a donor signal with the same probability, it is actually a donor signal with probability 7.5%. This pattern consists across several different families of signals, and at many probability values.

Our models are general enough to be adapted to other probabilistic inference scenarios beyond gene finding. For example, they are appopriate for modeling dependencies among positions in protein domains, for purpose of quickly screeing a database of proteins for a match to a non-consecutively dependent motif. We are currently investigating their integration into a gene finder.

2 Models of dependencies in signals

A useful way to represent biological signals is by probabilistic generative models. For our purposes, signals are short windows of fixed length in the sequence located near the biologically significant boundary or functional site. For example, donor and acceptor splice signals are located at exon-intron and intron-exon boundaries respectively.

The simplest generative model for signals is the *position weight matrix* (PWM) [20, 21]. A PWM gives the probability that each position in the signal is a particular character, and allows all positions to be independent. If $r(i, b)$ is the probability that the ith position in the signal is the base b, then the probability that the model generates a a given sequence S of length n is $\prod_{i=1}^{n} r(i, S_i)$.

The basic assumption of the PWM is that the probability of generating the character depends only on its position in the signal. Many researchers (*e.g.*, [6, 22]) demonstrated that this assumption is false. In fact, there exist dependencies in signals between positions which are several bases apart in the sequence. Higher order PWMs [22] allow the incorporation of dependencies between adjacent positions in the sequence. For example, in a first order PWM, each position is dependent on its immediate predecessor in the sequence; to generate the signal, one starts with the first position; then each subsequent character is picked using a probability distribution based on its predecessor.

Here, we investigate an extension of the PWMs to allow for multiple dependencies among positions in the signal which are not adjacent, and also to allow for dependencies between pairs of related signals. Such signals, especially corresponding donor and acceptor sites in introns, may be mutually dependent, and we wish to capture this information.

In our model, each position in the signal is dependent on a fixed set of other positions and there are no cyclic dependencies. We can view such a model as a directed acyclic graph (DAG), where nodes represent positions in the signal, and edges represent dependencies between the positions. Each node is assigned a probability distribution of bases depending only on the bases on positions which are its immediate ancestors in the DAG (see examples in Figure 1). We call the underlying DAG the *signal model topology*, and the maximum in-degree of a node in such a graph is the model's *order*. Such models are also called *Bayesian networks* and are extensively used in machine learning [13].

Models with order zero are exactly PWMs. If the order is 1, then the underlying topology is a directed tree. Such model was used for classification of splice sites before by Cai et al. [7] and Agarwal and Bafna [1]. The special case where the topology is restricted to paths was used for identifying transcription factor binding sites by Ellrott et al. [12]. PWMs of k-th order can be expressed as k-th order signal models (see Figure 1 for an example of a second order PWM). These network models are actually quite general: for example, the maximum dependency decomposition (MDD) model used by Burge [6] to model donor splice sites can be expressed as a fourth-order model. However, this is not very practical; such a model would contain many parameters not used by the MDD model. In general, we call all order k models the *higher order trees of k-th order* (or HOT-k).

To generate a signal in one of these models, one iteratively finds positions whose antecedents have been fixed, and then chooses the sequence value at that position dependent on the predecessors. This can of course be done in $O(kn)$ time, if the signal is of length n and k is the order of the model.

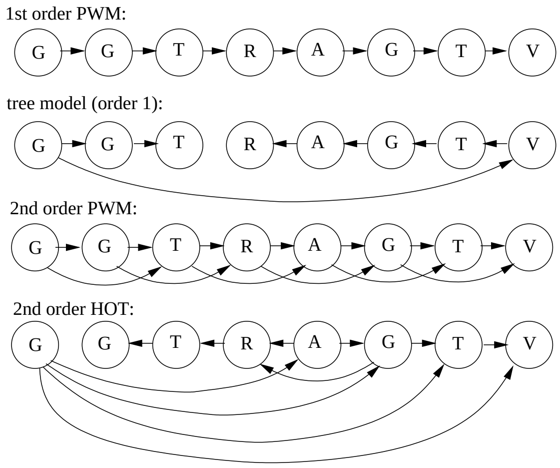

Fig. 1. Examples of different model topologies for donor signal positions $(-1, +7)$.

Similarly, one can compute the probability that the model generates a given sequence S by multiplying the dependent probabilities. If the ith position in the sequence is dependent on positions $d_{i,1} \dots d_{i,k_i}$, and we denote $r(i, b, x_1, \dots x_{k_i}) = \Pr[S_i = b | d_{i,1} = x_1, d_{i,2} = x_2, \dots, d_{i,k_i} = x_{k_i}]$, the probability of generating the sequence S by the model is

$$\Pr(S|+) = \prod_{i=1}^{n} r(i, s_i, s_{d_{i,1}}, s_{d_{i,2}}, \dots, s_{d_{i,k_i}}).$$

When using such model as a classifier, one also wants to compute the probability that the seen sequence S constitutes the signal. This can be done using Bayes rule:

$$\Pr(+|S) = \frac{\Pr(S|+) \cdot \Pr(+)}{\Pr(S|+) \cdot \Pr(+) + \Pr(S|-) \cdot (1 - \Pr(+))}.$$

Here, $\Pr(+)$ is a frequency of the signal occuring in the sequence, and $\Pr(S|-)$ is the probability of the sequence S occuring in a background model (in our work, we use a fifth-order position-independent Markov chain as the background model).

3 Choosing the best model

Here, we investigate methods for estimation of parameters of HOT models to maximize likelihood of the training data set. Once the topology of the model is fixed, we only need to count the frequencies from the training data. One needs to pick a good choice for the model's topology itself, before computing the parameters at each position. Here, we describe how to choose the optimal topology, given a training data set, and prove that the problem is NP-hard. Given its hardness, we considered integer programming techniques to find the optimal hypertree topology. In fact, the improvement relative to simpler greedy algorithms we will describe was minor. Therefore, in our experiments, we use the greedy algorithm to determine the topology of the model.

Note on the number of model parameters. If the topology of the model is fixed, the number of parameters that need to be trained depends on two variables: it is linear in the length of the signal, and exponential in the order of the model. However, the amount of data available for training the probability distribution of any given position does not change with the order of other positions or length of the signal. Therefore the model's order is the most relevant parameter in the training, and the number of data needed for training a k-th order HOT model is comparable to the amount of data needed for training a PWM of the same order. However, it is still possible for HOT to overfit during training because we are searching for model over a wider family (namely, all possible topologies).

Connection to hypergraphs. To formalize the problem of finding the best topology, we formulate the problem in terms of hypergraphs. A hypergraph is a pair $\mathcal{H} = (V, \mathcal{E})$, where V is a set of vertices, and $\mathcal{E} = \{E_1, E_2, \ldots, E_m\}$ is a set of directed hyperedges. Each *directed hyperedge* $E = (T, h)$ has a tail T, which is a subset of V, and a head h, which is a single vertex.[1] Let the *order of a directed hyperedge* be the cardinality of its tail.

A *directed hypertree* is a hypergraph $\mathcal{H}$, where all nodes are the tail of at most one hyperdge, and the directed graph which can be obtained by replacing every hyperedge $(\{v_1, \ldots, v_k\}, v)$ with k edges (v_1, v), (v_2, v), …, (v_k, v) is acyclic. A *spanning directed hypertree*, for us, is a directed hypertree where each vertex is a head of a hyperedge (note, that a hyperedge can have an empty tail). Spanning directed hypertrees are exactly equivalent to spanning outtrees in ordinary directed graphs.

Let us define a directed hypergraph analogous to the complete graph. The complete hypergraph is a hypergraph that contains all hyperedges (T, h) of order at most k.

There is an easy correspondence between the spanning directed hypertrees and HOT models: for a given vertex, all incoming edges in the HOT model can be represented as a single hyperedge. Such hyperedges form a spanning directed hypertree. Now we can make the following statement.

Theorem 1. *Let $\mathcal{H}$ be a complete hypergraph of order k on a set of vertices representing positions in the signal. Let weight of every hyperedge $(T, h) \in \mathcal{H}$ be $H(T \cup \{h\}) - H(T)$, where $H(P)$ is the entropy of the signal positions from set P in the training set of signals $S^{(1)}, \ldots, S^{(m)}$.*

Then the directed acyclic graph M^* corresponding to the minimum spanning hypertree $\mathcal{M}^*$ of graph $\mathcal{H}$ yields a topology of the HOT model of order k with the maximum likelihood of the training set $S^{(1)}, \ldots, S^{(m)}$.

[1] Sometimes, directed hyperedges are defined as a pair (T, H), where both tail T and head H are sets of vertices, rather than head being always a single vertex [14]. However, we will use the simpler definition here.

Proof. For any set of positions P, let $f_P(x_P)$ be the number of occurences of string x_P at positions in P in the training set $S^{(1)}, \ldots, S^{(m)}$. In this notation, the entropy $H(P)$ can be expressed as

$$H(P) = -\sum_{x_P} \frac{f_P(x_P)}{m} \cdot \log \frac{f_P(x_P)}{m}.$$

We have noted earlier that in the maximum likelihood model with fixed topology M, the string S has probability

$$\Pr(S \mid M) = \prod_{(T,h)\in\mathcal{E}(M)} \frac{f_{T\cup\{h\}}(S_{T\cup\{h\}})}{f_T(S_T)},$$

where $\mathcal{E}(M)$ is the set of hyperedges in the corresponding spanning directed hypertree. To maximize the likelihood of generating $S^{(1)}, \ldots, S^{(m)}$, we have to maximize over all possible model topologies M:

$$\Pr(S^{(1)}, \ldots, S^{(m)} \mid M) = \prod_{i=1}^{m} \Pr(S^{(i)} \mid M) = \prod_{i=1}^{m} \prod_{(T,h)\in\mathcal{E}(M)} \frac{f_{T\cup\{h\}}(S^{(i)}_{T\cup\{h\}})}{f_T(S^{(i)}_T)}$$

$$= \prod_{(T,h)\in\mathcal{E}(M)} \frac{\prod_{x_{T\cup\{h\}}} f_{T\cup\{h\}}(x_{T\cup\{h\}})^{f_{T\cup\{h\}}(x_{T\cup\{h\}})}}{\prod_{x_T} f_T(x_T)^{f_T(x_T)}}$$

We want to maximize $\Pr(S^{(1)}, \ldots, S^{(m)} \mid M)$ which is equivalent to minimizing $-(1/m)\log \Pr(S^{(1)}, \ldots, S^{(m)} \mid M)$:

$$-\frac{1}{m}\log \Pr(S^{(1)}, \ldots, S^{(m)} \mid M) =$$

$$\sum_{(T,h)\in\mathcal{E}(M)} \left[-\sum_{x_{T\cup\{h\}}} \frac{f_{T\cup\{h\}}(x_{T\cup\{h\}})}{m} \log f_{T\cup\{h\}}(x_{T\cup\{h\}}) \right] + \left[\sum_{x_T} \frac{f_T(x_T)}{m} \log f_T(x_T) \right]$$

$$= \sum_{(T,h)\in\mathcal{E}(M)} H(T,h) - H(T),$$

which is exactly what we wanted to prove. □

Solving the minimum spanning directed hypertree problem. Theorem 1 shows, how to reformulate the problem of finding the graphical model maximizing the likelihood of our training data to the problem of finding the minimum spanning directed hypertree problem.

To solve this problem, first consider its special case, where $k = 1$. In this case, we are looking for the minimum spanning directed tree of a complete graph. Chow and Liu [8] showed, that the problem is equivalent to finding the minimum spanning undirected tree, where weight of the edge (u, v) is $H(u, v) - H(u) - H(v)$. This can be easily done by Prim's algorithm (see, e.g. [10]) in $O(n^2 \log n)$ time.

Unfortunately, in the general case, this method cannot be extended. The following theorem shows that the problem is hard, even for $k = 2$.

Theorem 2. *Finding the minimum spanning directed hypertree in the hypergraph is NP-hard, even if all the edges are of degree at most 2.*

Proof. We prove NP-hardness by reduction from a special case of the problem of minimum feedback arc set restricted to directed graphs with indegree at most 2. Minimum feedback arc set problem is to find a minimum edge set whose removal makes the graph acyclic. It was proven NP-complete by a reduction from vertex cover [17] and this reduction can be easily modified to produce graphs with indegree at most 2, thus proving the special case of minimum feedback arc is also NP-hard.

To prove that the minimum spanning directed hypertree is NP-hard we take a directed graph G with indegree at most 2 and create a hypergraph $\mathcal{H}$ on the same set of vertices. For every vertex v we find the set X of tails of edges incoming to v in G. By our assumption X has at most 2 elements. For each subset A of X we create a hyperedge (A, v) with cost $|X| - |A|$. To complete the graph, we add all other possible hyperedges with tails of size at most 2, but with a large cost C, so that they will not be chose.

As established earlier, each spanning hypertree of $\mathcal{H}$ corresponds to a directed acyclic graph M and clearly M is a subgraph of G, if it does not use one of the very high-cost edges. Conversely, if we remove a feedback arc set from G we get a directed acyclic graph that correspond to some spanning hypertree of $\mathcal{H}$. Moreover for every edge deleted from G the cost of the corresponding hypertree increases by one. Therefore graph G has a feedback arc set of size at most k if and only if $\mathcal{H}$ has a spanning hypertree of cost at most k. □

As a result of Theorem 2, we must either consider slow solution algorithms, or use heuristic methods to find the optimal hypertree topology. We present each of these ideas in what follows.

Exact algorithms for finding the optimal hypertree. We first identified optimal hypertrees by using an integer linear programming algorithm. Integer linear programming is guaranteed to give us the optimal hypertree, yet it uses potentially exponential runtime [19].

Our IP model of the problem includes two kinds of variables. One family of variables models the acyclicity of the spanning hypergraph, by requiring that we order the nodes in the hypergraph, and require the heads of hyperedges to be later in the ordering than their tails. The second family of variables model the hyperedges of the graph. We require that each chosen hyperedge be properly ordered, and that each node have only one incoming hyperedge, to again ensure the hypertree property.

In particular, we assign a decision variable $b_{i,j}$ to each pair of distinct positions, i and j. This variable is set to 1 exactly when i comes before j in the ordering of nodes, and 0 otherwise. There is a variable $a_{T,h}$ for each possible directed hyperedge $E = (T, h)$, where $|T| \leq k$, and where $h \notin T$. When $a_{T,h} = 1$, the hyperedge E is in the chosen spanning hypertree, while when it is 0, the hyperedge is not in the chosen hypertree.

We model the ordering constraint on the nodes by requiring that the order relationship is antisymmetric and that there are no 3-cycles in it. In particular, we require that $b_{i,j} + b_{j,i} = 1$ for all pairs i and j and that $b_{i,j} + b_{j,k} + b_{k,i} \leq 2$ for all triplets i, j and k of distinct nodes.

The requirement that chosen hyperedges are properly ordered is modeled by the constraint $a_{T,h} \leq b_{x,h}$ for all nodes x in T. This requires that all nodes in the head of the hyperedge are before the tail node, for chosen hyperedges.

Finally, we require that every node has an incoming edge (in the case of the tree's root, this will have no tail nodes). This is the constraint $\sum_{E:E=(T,h)} a_{T,h} = 1$ for all nodes h.

The cost of a chosen hypertree is $\sum_{E=(T,h)} a_{T,h} w_{T,h}$, where $w_{T,h}$ is the cost of including the hyperedge $E = (T, h)$ in the tree.

This gives the integer linear program:

$$\min \sum_{E=(T,h)} w_{T,h} a_{T,h}, \qquad \text{subject to:}$$

$$\begin{aligned}
b_{i,j} + b_{j,i} &= 1, \text{ for all pairs } i \text{ and } j,\\
b_{i,j} + b_{j,k} + b_{k,i} &\leq 2, \text{ for all triplets } i,\ j \text{ and } k,\\
a_{T,h} &\leq b_{x,h}, \text{ for all hyper edges } E = (T,h) \text{ and nodes } x \text{ in } T,\\
\sum_{E:E=(T,h)} a_{T,h} &= 1, \text{ for all nodes } h,\\
a_{T,h} &\in \{0,1\}, \text{ for all hyperedges } E = (T,h),\\
b_{i,j} &\in \{0,1\}, \text{ for pairs of nodes } i \text{ and } j.
\end{aligned}$$

We used the integer programming solver CPLEX [15] to solve moderate-sized instances of these problems, where $k = 2$ (so all hyperedges have at most two head nodes). The runtime of the optimization procedure was only a few hours, and we were able to compute optimal hypertrees for many of the signals we discuss in Section 4. We note that computing these optimal signal models is something one does not have to do often, so spending a reasonable amount of computational effort on finding the right model is appropriate, assuming runtime is at all reasonable.

However, the improvement over the simple greedy algorithm we discuss in the next subsection was not large enough in our actual examples to justify its much larger runtime. In some applications, however, the added quality given by integer programming might be appropriate.

Simple heuristic. In practice, we have used a simple greedy heuristic to compute good HOT models. We start with a single node in the spanning hypertree $\mathcal{T}$. In each iteration, we add one more vertex v into the hypertree $\mathcal{T}$, where v is the head of the shortest hyperedge (T, h) such that $T \in \mathcal{T}$ and $h \notin \mathcal{T}$. This can be implemented in $O(kn^{k+1})$ time. Since the hypertrees generated in this way can differ depending on the starting vertex, we run the algorithm for every vertex as a starting point and choose the shortest resulting hypertree.

Note, that this is just a simple variation of Prim's algorithm for maximum spanning trees [10], however, unlike in the case of undirected spanning trees, in the case of directed spanning hypertrees this process does not guarantee the optimal result. This simple heuristic gave hypertrees with good performance often enough that we did not use the integer programming model in practice. It is certainly reasonable to consider more complicated heuristics for the problem.

Again, once we had the chosen spanning directed hypertree topology, we then computed the positional probabilities by using the empirical values from the training data set.

Note on undirected graphical models. Inference of the optimal topology of fixed order has been studied for undirected graphical models in the context of machine learning. Srebro and Karger showed that the problem is NP-hard and presented approximation algorithms for it [16]. Independently, Bach and Jordan [4] described the problem and used practical heuristic algorithms to solve it. They also applied their solution to a simple biological signal data set. Finally, Andersen and Fleischner stuied the problem of finding a minimum spanning undirected hypertree [3], who showed that this problem is also NP-hard. However, the result is not related, because of significant differences in properties of undirected and directed hypertrees.

4 Experiments

We tested the usefulness of higher order models for identifying splicing signals in human DNA. In general, our experiments show that there are some cases where including multiple dependencies and not requiring them to be adjacent can help with sensitivity of models, but in general, the improvement is quite slight; this is similar to the results of Cai *et al.* [7], who found this with tree models.

However, one useful finding is that higher order tree models can be better in predicting the probability that a given position actually *is* a signal. This probability can be used in other programs, such as probabilistic gene finders. Perhaps our most interesting result is that using a higher order tree model allows substantially better prediction of the probability that a position is a donor site than is available with other measures.

We also show that including branch site models does not aid much in detecting acceptor sites, but that this inclusion, again, helps with estimating the probability that sites are acceptor sites.

Data sets. We used two sets of annotated human sequences in our experiments. The first is the Genscan data set of Burge and Karlin [5]. This data set is divided into two parts: a training set (B97T) and a smaller testing set (B97). Our second data set is the Sanger Centre annotation of Human chromosome 22, release 3.1b [11]. We divided the chromosome, masked for repeats, into 73 pieces, each of roughly 500KB in length, and then randomly divided these into

Name	Abbreviation	Length	Donor sites True	Donor sites False	Acceptor sites True	Acceptor sites False
Genscan (training)	B97T	2.9 MB	1,237	146,204	1,236	212,144
Genscan (testing)	B97	0.6 MB	339	30,319	339	45,512
Chromosome 22 (training)	C22T	19.0 MB	1,952	1,000,646	1,965	1,546,120
Chromosome 22 (testing)	C22	15.8 MB	1348	814,665	1366	1,229,695

Table 1. Characteristics of used data sets. Only canonical donor (GT) and acceptor (AG) sites were considered as true sites. We list any two-base sequence of GT or AG in the sequences as a false donor or false acceptor site, respectively.

training (C22T) and testing (C22) sets. We present basic characteristics of the data sets in Table 1. These characteristics were used to compute $\Pr(+)$ in our experiments.

The C22/C22T data set contains significantly more false donor and acceptor sites per true site than does the Genscan set. This is because the B97/B97T set contains mostly genic sequence, with very short intergenic regions before and after the genes. On the other hand, the C22/C22T data set is an entire chromosome, where genes are a small fraction of all DNA. As we will see, this significantly affects the specificity of our models and prevents us from directly comparing the results from the two data sets.

To avoid overfitting, we only trained first-order models on the training set B97T, while we used both first-order and second-order models with C22T. Note, that the test set B97 is curated and contains only sequences with high-quality annotation. On the other hand, the test data set C22 is an annotation of a whole chromosome, and may contain several errors.

Accuracy measures. We used two measures of accuracy in our experiments. The first is the sensitivity and specificity of classification. Our models can be used as classifiers when we set a threshold on $\Pr(+|S)$. Sequences with score below the threshold are classified as not being signals, while sequences with score above the threshold are classified as signals.

Let us denote the number of true positives as TP, false positives FP, true negatives TN, and false negatives FN. We measure classification accuracy with the sensitivity $SN = TP/(TP + FN)$ and specificity $SP = TP/(TP + FP)$. If we lower the threshold, sensitivity increases and specificity decreases. Thus for a given model, we can plot a curve depicting the tradeoff, and compare different models.

Our interest in gene finding and generative probabilistic models has inspired another measure of accuracy. In these applications, signal models are not only used as classifiers, but the estimation of $\Pr(+|S)$, the probability that the evaluated sequence is a signal, is also used. Therefore, this estimate should be as close to the actual probability that the sequence is a signal as possible.

To evaluate accuracy of the predicted signal probability, we first divide the range of scores into buckets (in logarithmic scale). In each bucket, we compute the number of positive and negative examples among the sequences whose estimated probability causes them to be placed in the bucket. Then we compare the

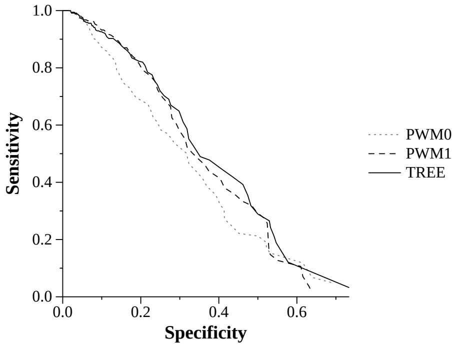

Fig. 2. Sensitivity vs. specificity for donor splice site on data set B97.

predicted signal probability corresponding to the bucket with the actual fraction of positive examples belonging to the bucket (actual signal probability). We quantify this measure using the standard Pearson correlation weighted by number of samples in each bucket.

In what follows, we abbreviate the various chosen signal topology models as follows: PWM-k is the chosen position weight matrix of kth order, TREE is the model obtained by optimizing the first order HOT model, and HOT-2 is the second order HOT model selected by our heuristics.

4.1 Donor site experiments

For our experiments with donor sites, we represented donor sites by a window of length 12, from the position -4 to the position $+7$ (consensus `VMAG|GTRAGTRN`). We developed first-order models, using the smaller B97T data set, and higher-order models, using the larger C22T set.

First order models. We trained the PWM-0 and first-order models (PWM-1, TREE) on the training set B97T and evaluated their performance on testing set B97. We found two non-adjacent dependencies in the TREE model: positions $+3$ and $+5$ both depend on the position -1, rather than on one of their direct neighbors.

Figure 2 shows the classification power of these models. The tree model improves on both models that do not consider non-adjacent dependencies (PWM-0, PWM-1); the improvement is more apparent in the regions around 40% and 70% sensitivity (see Table 2). We note that the improvement obtained at the 35% level of sensitivity by the TREE model is comparable to the improvement reported for the maximum dependency decomposition model at 50% level by [6].

Higher order models. To evaluate second order models, we used the data set C22T to train models of order at most 2 (PWM-0, PWM-1, TREE, PWM-2, and HOT-2). We then evaluated their prediction accuracy on the testing set C22. We found several non-adjacent dependencies (*i.e.*, dependencies at distance greater than 2) in model HOT-2. Position -4 depended on $+7$, positions $-3 \ldots -1$ on $+3$, and positions $+4 \ldots +5$ on -1.

Model	Sensitivity level 20%	35%	50%	70%	90%
PWM-0	50.0%	39.2%	31.7%	17.7%	7.9%
PWM-1	52.1%	44.2%	33.3%	24.3%	13.0%
tree	54.1%	47.4%	33.6%	26.0%	12.9%
MDD (*)	54.3%	–	36.0%	–	13.4%

Table 2. Specificity of models for donor splice sites on data set B97. We include the results for the maximum dependency decomposition model on the same set, from [6], for comparison.

Model	Sensitivity level 5%	20%	35%	50%	70%	90%
PWM-0	12.8%	9.2%	7.4%	5.7%	3.2%	1.4%
PWM-1	16.0%	11.6%	9.5%	6.9%	4.2%	1.8%
TREE	14.0%	11.3%	9.0%	7.1%	4.4%	1.7%
PWM-2	16.0%	11.2%	9.4%	7.0%	4.2%	1.7%
HOT-2	17.9%	11.0%	9.3%	6.6%	4.5%	1.9%

Table 3. Specificity for donor splice site models at given levels of sensitivity, on data set C22.

Table 3 shows the models' classification performance. The performance of all models except PWM-0 is roughly equivalent. This equivalence is caused mostly by the much higher presence of false positives in this data set. HOT-2 has a slightly higher specificity for low and high values of sensitivity, and a slightly lower specificity in the middle range.

We notice real improvement in our second evaluation measure: the comparison of the predicted signal probability with the real signal probability (Figure 3). The figure shows that in general, most models slightly underpredict the probability for low-scoring possible signals, while greatly overpredict the probability for high-scoring possible signals. This is the case for all tested models except the HOT-2 model, which consistently slightly underpredicts at all scores. The advantage of the HOT-2 model can also be seen in the correlation coefficient between the predicted and actual signal probability (see Figure 3): the HOT-2 model has a strongly higher correlation (0.955) in our measure than does the second best model (PWM-2, with correlation coefficient 0.911).

The HOT-2 model has approximately the same classification power as other models, while showing significant improvement in predicting accurately the signal probability. We thus suggest that it is a better model for incorporating into gene finding programs and other probabilistic frameworks than the other models.

We also compared the HOT-2 model to other models on a data set that had fewer false positives. We trained all models on the C22T data set (to avoid overfitting) and then tested them on the B97 data set. We show the results in Figure 4. They show that HOT-2 has highest specificity at high sensitivity settings (for sensitivities more than 50%), while at lower sensitivities (less than

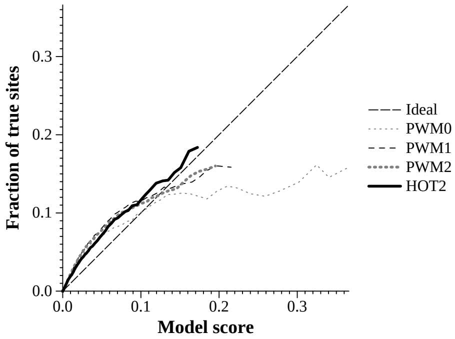

Fig. 3. Predicted signal probability versus actual signal probability for donor splice site signal on data set C22. For example, the bucket with average score 0.066 in the HOT2 model contains 258 samples, out of which 20 are true donor sites. This corresponds to the y-coordinate $0.078 = 20/258$. The correlation coefficients between the predicted probability and the actual probability are 0.955 for HOT2, 0.911 for PWM2, 0.890 for PWM1 and 0.827 for PWM0.

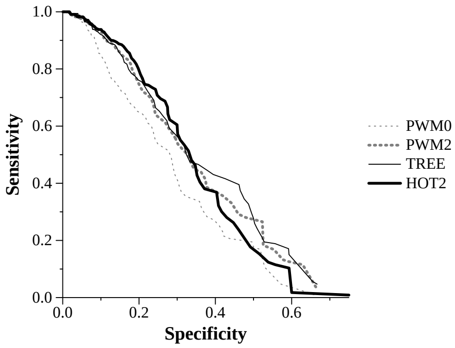

Fig. 4. Accuracy statistics for donor splice site, training on data set C22, testing on data set B97.

50%), the TREE model has the best specificity. The advantage of the HOT-2 model at the high levels of sensitivity is small, however.

4.2 Acceptor sites and branch sites

Next, we studied acceptor sites, using a window of size 13 from position -10 to position $+2$ (consensus `YYYYYYVCAG|GNN`). Burge [6] pointed out that the acceptor site does not exhibit strong non-adjacent dependencies in the signal. In this case, our experiments confirmed that the possibility of non-adjacent dependencies in the models TREE and HOT-2 does not gain any advantage, and the performance of all models except PWM-0 is roughly equivalent on both data sets.

Model	Data set B97 without branch	Data set B97 with branch	Data set C22 without branch	Data set C22 with branch
PWM0	0.825	0.930	0.760	0.920
PWM1	0.882	0.906	0.920	0.961
TREE	0.872	0.912	0.918	0.959
PWM2	–	–	0.920	0.952
HOT-2	–	–	0.925	0.955

Table 4. Correlation coefficient between predicted signal probability and actual signal probability, for models of acceptor sites on the data sets B97 and C22 that do and do not include branch site enhancement.

We also tried to enhance prediction of acceptor sites by trying to locate branch sites in the sequences, upstream of the acceptor site. Since branch sites are not generally annotated, we used an iterative refining procedure described by Zhang [22] to train a model for recognizing branch sites. We started with a simple model based on the consensus sequence NNNYTVAYYYYYYYYY (in a window of size 16, from positions -6 to $+9$). In each iteration, we located the best-scoring sequence in the sequence window from 50 positions before each acceptor site to 10 positions before the acceptor site, and used this sequence to train the new model. We performed three iterations to train branch site models. Lastly, we paired the possible branch sites we found with their corresponding acceptor sites and built new models for both signals together that allow inter-signal dependencies.

To estimate the signal probability of each possible acceptor site A, we first located the most likely branch site B in the window from 10 to 50 bases upstream from the possible acceptor A and then estimated $\Pr(+|B, A)$ of both signals B and A together.

Several interesting dependencies appeared in the optimal HOT-2 model. There are dependencies of the acceptor positions -10, -6, -5, -3, and $+2$ on the branch site position -1, of the acceptor position -9 on the branch site position $+9$, of the acceptor position -7 on the branch site positions $+6$ and $+9$, and of the branch site position -2 on the acceptor site position -1.

However, considering these dependencies do not improve classification power significantly, according to our experiments. In general, using branch site models improved specificity very slightly (usually less than 1%). Interestingly, however, the correlation between predicted and actual probability of the signal is much improved in all models by using the branch site; we show these data in Table 4.

4.3 Dependencies among three signals

Previous studies have shown a relationship between the strengths of corresponding donor and acceptor sites [22, 6, 9]. Inspired by this, we investigated whether dependencies between positions of these signals can be found. We trained our models on a composite signal of donor, branch, and acceptor sites extracted from

Signal	Intra-signal adjacent	Intra-signal adjacent	Inter-signal
Donor	0.0597	0.0535	branch: 0.0066
			acceptor: 0.0094
Branch	0.1044	0.0303	acceptor: 0.0136
Acceptor	0.0835	0.0210	

Table 5. The strongest dependencies found among signals. Shown are the strongest dependencies between adjacent positions, non-adjacent intrasignal positions and intersignal positions. The values shown in the table are differential entropy $H(i)+H(j)-H(i,j)$; this is the value used in determining the best tree structure. Higher table entries correspond to stronger dependencies.

the same introns. The models were used to classify whether given pair of donor and acceptor come from the same intron. However, the dependencies between the signals were so weak that they do not facilitate such classification with any reasonable degree of accuracy. Table 5 shows that the dependencies between the signals are much weaker than the dependencies between both non-adjacent and adjacent positions within the same signal.

5 Conclusions

We have presented a new model for biological signals found in DNA. Our new approach allows for positions in signals to be dependent on a small number of other positions, which need not be their direct predecessors. This model is an extension of both higher order position weight matrices, which are typically used for this problem, and of the Chow-Liu trees studied by Cai et al. [7] for this purpose.

Our models are based on directed hypertrees of dependence structure. For a given training data set, we have shown that it is NP-hard to compute the optimal dependency model from the family we consider, as long as positions are allowed to depend on multiple sites. However, in practice, we are able to compute good models, either by using integer programming or much simpler greedy heuristics.

In practice, we find that the addition of flexibility in characterizing dependencies in a model is modestly helpful, but not profoundly so. There is some advantage over the simply ordered model of position weight matrices, especially in the case of donor site prediction. We were also able to demonstrate that the signal probability estimates given by our new models are much more reliable predictors of actual probability of the sequence being a signal than the scores given by PWMs or tree models. This is useful in the context of probabilistic gene finding, where this estimation is one of many that are integrated into the gene finding process.

Future work. Several questions remain from our work. The first has to do with overtraining. Is it possible to estimate the size of data set needed to avoid overfitting these more complicated models? Clearly, they depend on more parameters than the corresponding higher order PWMs, since one must also infer the hypertree topology. On a related note, can overfitting be prevented by combining parameters from several orders, as is done in interpolated Markov chains [18]?

Second, in this work we chose a model with maximum likelihood. This resulted in models that were good predictors of a probability that a given site is a signal, but their discriminative power was not improved compared to other methods. Perhaps, the discriminative power could be improved by estimating parameters using different optimization criterion, as is suggested by Akutsu et al. [2] for PWMs.

Third, inter-signal dependencies between donor, branch, and acceptor sites do not help us to improve prediction. Another context in which intersignal dependencies might be discovered is in the core promoter, which is a region in DNA sequence, composed of several smaller signals located close to each other, that work together in transcription initiation. Our methods may help with recognition of core promoters.

Fourth, can we apply hypertree PWMs to other classification or prediction problems? Chow-Liu trees have been used successfully in numerous applications, which suggests that the more rich hypertree PWMs may, as well.

Lastly, we are currently investigating the effect of integrating these models together with probabilistic gene finders.

Acknowledgments. This work has been supported by a grant from the Human Science Frontier Program, a grant from the Natural Sciences and Engineering Research Council of Canada, and by an Ontario Graduate Scholarship.

References

1. P. Agarwal and V. Bafna. Detecting non-adjoining correlations within signals in DNA. In *Proceedings of the Second Annual International Conference on Research in Computational Molecular Biology (RECOMB 1998)*, pages 2–8. ACM Press, 1998.
2. Tatsuya Akutsu, Hideo Bannai, Satoru Miyano, and Sascha Ott. On the complexity of deriving position specific score matrices from examples. In A. Apostolico and M. Takeda, editors, *Combinatorial Pattern Matching, 13th Annual Symposium (CPM 2002)*, volume 2373 of *Lecture Notes in Computer Science*, pages 168–177. Springer, 2002.
3. L. D. Andersen and H. Fleischner. The NP-completeness of finding A-trails in Eulerian graphs and of finding spanning trees in hypergraphs. *Discrete Applied Mathematics*, 59:203–214, 1995.
4. F. R. Bach and M. I. Jordan. Thin junction trees. In Thomas G. Dietterich, Suzanna Becker, and Zoubin Ghahramani, editors, *Proceedings of NIPS 2001*, pages 569–576. MIT Press, 2001.
5. C. Burge and S. Karlin. Prediction of complete gene structures in human genomic DNA. *Journal of Molecular Biology*, 268:78–94, 1997.
6. C. B. Burge. Modeling dependencies in pre-mRNA splicing signals. In S. L Salzberg, D. B. Searls, and S. Kasif, editors, *Computational Methods in Molecular Biology*, pages 129–164. Elsevier, Amsterdam, 1998.

7. D. Cai, A. Delcher, B. Kao, and S. Kasif. Modeling splice sites with Bayes networks. *Bioinformatics*, 16(2):152–158, 2000.
8. C. K. Chow and C. N. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, IT-14(3):462–467, 1968.
9. F. Clark and T. A. Thanaraj. Categorization and characterization of transcript-confirmed constitutively and alternatively spliced introns and exons from human. *Human Molecular Genetics*, 11(4):451–454, 2002.
10. T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, Cambridge, Mass., 2nd edition, 2001.
11. I. Dunham et al. The DNA sequence of human chromosome 22. *Nature*, 402:489–495, 1999.
12. K. Ellrott, C. Yang, F. M. Sladek, and T. Jiang. Identifying transcription factor binding sites through Markov chain optimization. In *Proceedings of the European Conference on Computational Biology (ECCB 2002)*, pages 100–109, 2002.
13. N. Friedman, D. Geiger, and M Goldszmidt. Bayesian network classifiers. *Machine Learning*, 29:131–163, 1997.
14. G. Gallo, G. Longo, S. Pallottino, and S. Nguyen. Directed hypergraphs and applications. *Discrete Applied Mathematics*, 42:177–201, 1993.
15. ILOG Inc. CPLEX optimizer, 2000. Computer software.
16. D. Karger and N. Srebro. Learning Markov networks: Maximum bounded tree-width graphs. In *Proceedings of the Twelfth Annual Symposium on Discrete Algorithms (SODA 2001)*, pages 392–401. SIAM, 2001.
17. R. M. Karp. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103, New York, 1972. Plenum Press.
18. S. L. Salzberg, A. L. Delcher, S. Kasif, and O. White. Microbial gene identification using interpolated Markov models. *Nucleic Acids Research*, 26(2):544–548, 1998.
19. A. Schrijver. *Theory of Linear and Integer Programming*. Wiley and sons, 1986.
20. R. Staden. Computer methods to aid the determination and analysis of DNA sequences. *Biochemical Society Transactions*, 12(6):1005–1008, 1984.
21. G. D. Stormo, T. D. Schneider, L. E. Gold, and A. Ehrenfeucht. Use of the 'Perceptron' algorithm to distinguish translational initiation sites in *E. coli*. *Nucleic Acids Research*, 10(9):2997–3011, 1982.
22. M. Q. Zhang. Statistical features of human exons and their flanking regions. *Human Molecular Genetics*, 7(5):919–932, 1998.