DNA Feature Sensors

B. Majoros

Duke
UNIVERSITY

What is Feature Sensing?

A *feature* is *any DNA subsequence of biological significance*.

For practical reasons, we recognize two broad classes of features:

signals — short, fixed-length features such as start codons, stop codons, or splice sites

content regions — variable-length features such as exons or introns

We thus disinguish between two broad classes of *feature sensors*:

signal sensors — these typically scan a DNA sequence, using a sliding-window approach, to identify specific locations in the DNA where a signal is likely to reside

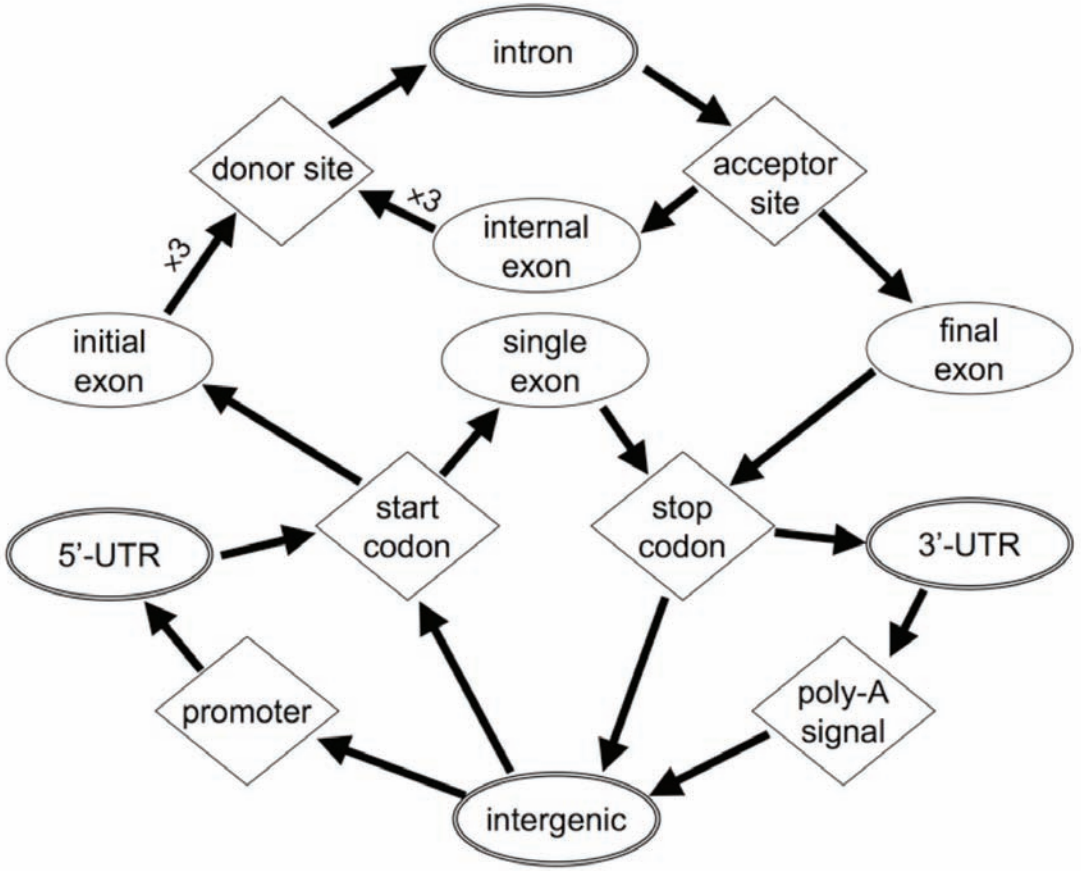
content sensors — these are typically used to assign a score to the region lying between two signals; e.g., a putative exon lying between a putative start codon and a putative donor splice site

Feature Sensing in a GHMM

We will see later that an ensemble of *signal sensors* and *content sensors* for gene features can be most elegantly utilized for the task of gene prediction by incorporating them into a *Generalized Hidden Markov Model* (*GHMM*).

Within each state of a GHMM resides a signal sensor (diamonds) or a content sensor (ovals).

The sensor belonging to a given state is used to evaluate the probability of that state emitting a given DNA interval as a putative feature of the appropriate type.



Feature Probabilities

Within a GHMM-based gene finder, *feature evaluation* consists of the computation of $P(S \mid q)$, the probability of a putative feature S, conditional on the type of feature begin modeled (i.e., state q):

$$P(S \mid \theta_q) = P_e(S \mid q)$$

The expression $P(S \mid \theta_q)$ makes explicit that the probability is conditional on the model parameters θ_q of the feature sensor, which is specific to the state q in the GHMM where the sensor resides.

In the terminology of HMM's, this is the *emission probability* for state q.

Conditioning on Feature Length

One advantage of the GHMM framework is that it alows explicit modeling of feature length. We thus refine our formulation of the feature sensing problem by conditioning emission probabilities on the *feature length* (or *duration*) d:

$$P(S \mid \theta_q, d) = \frac{P(S \mid \theta_q)}{\sum_{S' \in \alpha^d} P(S' \mid \theta_q)}$$

That is, we normalize the probability of a putative feature by dividing by the sum of probabilities of all possible features of the same length (d=|S|).

Since signals are by definition fixed-length features, for a signal state q we have $P(S \mid \theta_q, d) = P(S \mid \theta_q)$ whenever d matches the length prescribed by the signal sensor, and $P(S \mid \theta_q, d) = 0$ otherwise.

Markov Chains

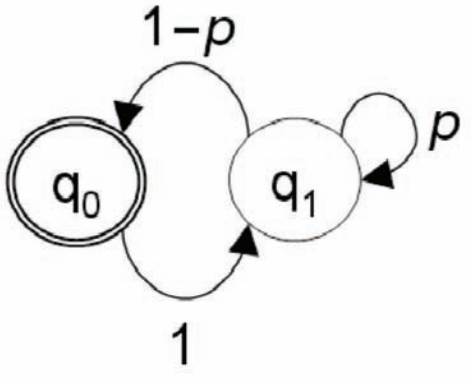
In a previous lecture we learned about Hidden Markov Models (HMM's). In the gene-finding literature the term *Markov chain* (*MC*) is reserved for HMM's in which all states are *observable* — i.e., a Markov model in which no states are *hidden*.

One formulation of a Markov-chain-based content sensor is as a 2-state Markov chain with higher-order emissions:

$M=(Q, \alpha, P_t, P_e)$,

where:

$Q = \{q_0, q_1\}$,
$P_t = \{(q_0, q_1, 1), (q_1, q_1, p), (q_1, q_0, 1-p)\}$.



State q_0 is the silent start/stop state. Because all emissions come from state q_1, there are no hidden states; thus, decoding is trivial: there is only one path for any given emission sequence.

Conditioning on Durations

Conditioning on *feature length* (=*state duration*) is trivial in the case of a Markov chain:

$$P(S \mid \theta, d) = \frac{\left(\prod_{i=0}^{d-1} P_e(x_i)\right) p^{d-1}(1-p)}{\sum_{S' \in \alpha^d} \left(\prod_{i=0}^{d-1} P_e(x'_i)\right) p^{d-1}(1-p)}$$

Fortunately, it turns out that:

$$\sum_{S' \in \alpha^d} \left(\prod_{i=0}^{d-1} P_e(x'_i)\right) = 1$$

so we can simplify the conditional probability to:

$$P(S \mid \theta, d) = \prod_{i=0}^{d-1} P_e(x_i)$$

Thus, we can evaluate a putative feature S under a Markov chain θ by simply multiplying emission terms $P_e(x_i)$ along the sequence.

Higher-order Chains

Higher-order chains are easily accommodated by observing the preceding n-gram $x_{i-n}...x_{i-1}$ when computing the emission score for x_i:

$$P(S \mid \theta, d) = \prod_{i=0}^{d-1} P_e(x_i \mid x_{i-n}...x_{i-1})$$

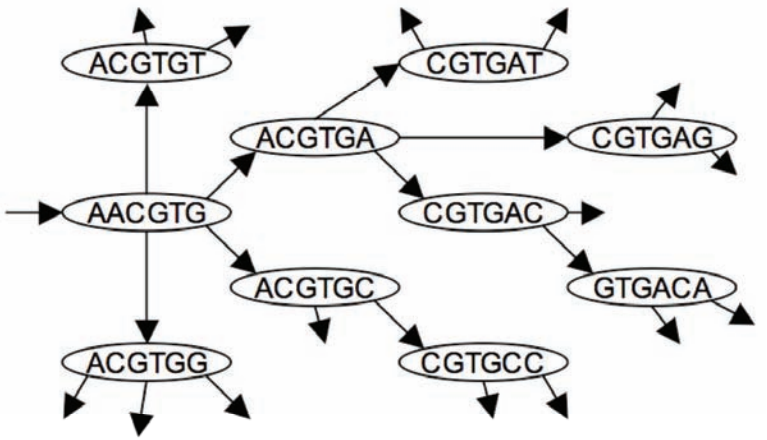
This is called an *n^{th}-order Markov chain*.

At the beginning of a putative feature we may wish to condition on fewer preceding bases:

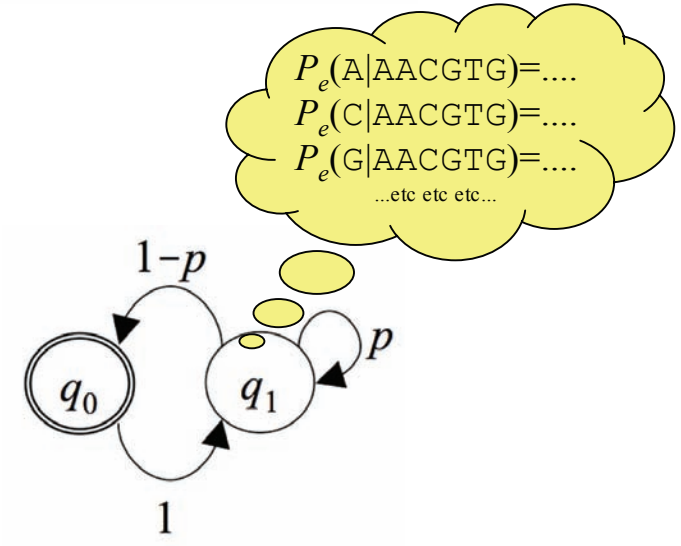
$$P(S \mid \theta, d) = \prod_{i=0}^{n-1} P_e(x_i \mid x_0...x_{i-1}) \prod_{i=n}^{d-1} P_e(x_i \mid x_{i-n}...x_{i-1})$$

Recall that products of large numbers of probabilities tend to cause *underflow* in the computer — working in *log-space* is thus recommended (regardless of the order of the model).

Two Definitions of Markov Chains



- "6th-order Markov chain"
- 4^6 states
- 0th-order emissions, $P_e(x)$=100%
- transitions incorporate the emission and transition probabilities from the 2-state model, so lengths are still geometrically distributed



- "6th-order Markov chain"
- 2 states
- 6th-order emissions
- explicit self-transitions (p) impose geometric distribution

both are HMM's, and both are entirely equivalent

Training a Markov Chain

For training, we need only count the (n+1)-mer occurrences and convert these to conditional probabilities via simple normalization:

$$P_e(g_n \mid g_0 \cdots g_{n-1}) \approx \frac{C(g_0 \cdots g_n)}{\sum_{s \in \alpha} C(g_0 \cdots g_{n-1} s)}$$

where $C(g_0 \cdots g_n)$ is the number of times the (n+1)-mer $G = g_0 \cdots g_n$ was seen in the set of training features for this sensor.

For large n, the sample sizes for these estimations can be small, leading to *sampling error*. We will see one method for mitigating this effect (*interpolation*).

Three-periodic Markov Chains

When modeling coding exons, we can take advantage of the *periodicity of base frequencies* induced by the codon triplets making up the CDS.

Suppose we are given a forward-strand training exon $E=x_0x_1x_2\ldots x_{m-1}$ and are told that exon E begins in phase ω. Then we can train a set of three Markov chains M_0, M_1, and M_2 on E by modifying the normal training procedure to observe the rule that only n-grams ending at a position $y = 3x+(i-\omega) \bmod 3$ relative to the beginning of the exon can be used in the training of chain M_i, for all x such that $0 \leq 3x+(i-\omega) \bmod 3 < m$.

The collection $M=(M_0,M_1,M_2)$ constitutes a *three-periodic Markov chain*. Applying a *3PMC* M to compute the conditional probability of a sequence S in phase ω on the forward strand can be achieved via:

$$P(S \mid M,d,\omega) = \prod_{i=0}^{d-1} P_e(x_i \mid M_{(\omega+i) \bmod 3})$$

Duke UNIVERSITY

Interpolated Markov Chains

For higher-order chains we can *interpolate* between orders to mitigate the effects of sampling error:

$$P_e^{IMM}(s \mid g_0 \ldots g_{k-1}) =$$

$$\begin{cases} \lambda_k^G P_e(s \mid g_0 \ldots g_{k-1}) + (1 - \lambda_k^G) P_e^{IMM}(s \mid g_1 \ldots g_{k-1}) & \text{if } k > 0 \\ P_e(s) & \text{if } k = 0 \end{cases}$$

$$\lambda_k^G = \begin{cases} 1 & \text{if } m \geq 400 \\ 0 & \text{if } m < 400 \text{ and } c < 0.5 \\ \dfrac{c}{400} \displaystyle\sum_{x \in \alpha} C(g_0 \ldots g_{k-1} x) & \text{otherwise} \end{cases}$$

resulting in an *Interpolated Markov Model* (IMM) or *Interpolated Markov Chain* (IMC). m is the sample size, k is the order of the model, and c is derived from a χ^2 statistic (see Salzberg *et al.*, 1998).

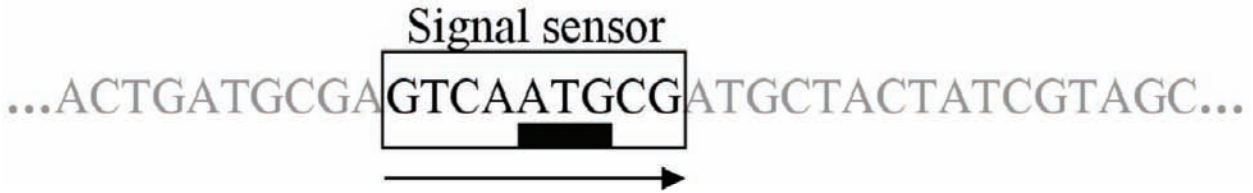
Duke UNIVERSITY

Signal Sensors

The types of signals sought by signal sensors can usually be characterized by a fairly consistent *consensus sequence*. Some canonical signal consensus sequences in typical eukaryotes are:

- `ATG` for start codons
- `TAG`, `TGA`, or `TAA` for stop codons
- `GT` for donor splice sites
- `AG` for acceptor splice sites
- `AATAAA` or `ATTAAA` for polyadenylation signals
- `TATA` for the TATA-box portion of a human promoter signal

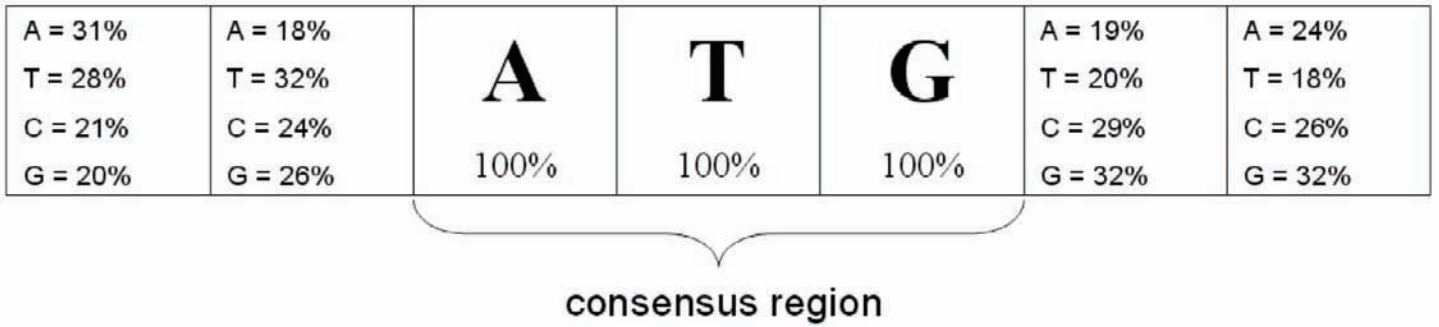
The consensus region of a signal is useful for limiting the number of positions in the DNA sequence at which the "sliding window" needs to be evaluated:



Duke UNIVERSITY

Weight Matrices

One of the simplest signal sensors is the *weight matrix*, or *WMM* (also called a *Position-specific Weight Matrix*, or *PWM*). A weight matrix is simply a fixed window in which each cell of the window has a nucleotide distribution:



The *consensus region* of the WMM is the span of cells in which a valid consensus for the signal of interest is expected to occur. For start codons, for example, we can scan the input sequence for the motif `ATG`, position the sensor around the `ATG` so that these three bases fall within the consensus region of the sensor, and then evaluate the sensor on the entire subregion covered by the sensor window.

Evaluating a Weight Matrix

Evaluating a WMM at a given position in a DNA sequence is as simple as looking up each nucleotide falling within the window in the position-specific base distribution associated with the corresponding cell of the sensor's window, and multiplying these together:

A = 31% T = 28% C = 21% G = 20%	A = 18% T = 32% C = 24% G = 26%	**A** 100%	**T** 100%	**G** 100%	A = 19% T = 20% C = 29% G = 32%	A = 24% T = 18% C = 26% G = 32%
C	T	**A**	**T**	**G**	A	C
0.21	0.32	**1.0**	**1.0**	**1.0**	0.19	0.26

$$P(\text{CT}\underline{\textbf{ATG}}\text{AC}|q_{start\_codon})$$

$$= 0.21 \times 0.32 \times 1.0 \times 1.0 \times 1.0 \times 0.19 \times 0.26$$

$$= 0.00331968$$

Duke UNIVERSITY

Weight Array Matrices

Just as we generalized HMM's and Markov chains to higher orders, we can utilize higher-order distributions in a WMM.

Given a model *M* of length L_M, we can condition each cell in the sensor's window on some number *n* of previous bases observed in the sequence:

$$P(x_i \ldots x_{i+L_M-1} \mid M, L_M) = \prod_{j=0}^{L_M-1} P_M(j, x_{i+j}, x_{i+j-n} \ldots x_{i+j-1})$$

This is called a *weight array matrix* (*WAM*). A variant, called the *windowed WAM*, or *WWAM* (Burge, 1998), pools counts over several positions when training the model, in order to reduce the incidence of sampling error:

$$P_e(g_n \mid j, g_0 \ldots g_{n-1}) = \frac{\sum_{k=-2}^{2} C(j+k, g_0 \ldots g_n)}{\sum_{k=-2}^{2} \sum_{s \in \alpha,} C(j+k, g_0 \ldots g_{n-1} s)}$$

Duke UNIVERSITY

Local Optimality Criterion

When identifying putative signals in DNA, we may choose to completely ignore low-scoring candidates in the vicinity of higher-scoring candidates. The purpose of the *local optimality criterion* is to apply such a weighting in cases where two putative signals are very close together, with the chosen weight being 0 for the lower-scoring signal and 1 for the higher-scoring one (Pertea *et al.*, 2001).

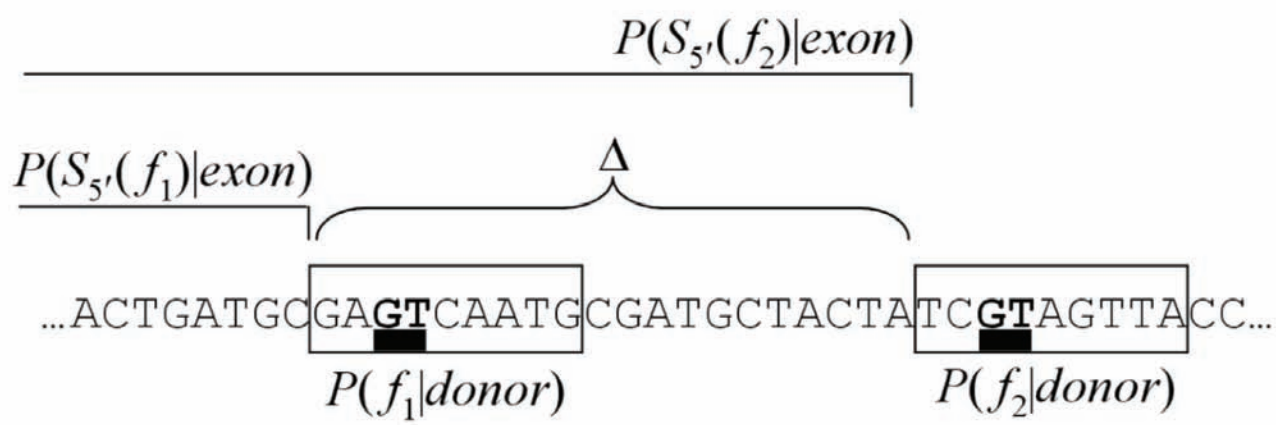


***Figure 7.8**: The rationale for imposing a local optimality criterion for signal sensing. The probability $P(S_{5'}(f_2)|exon)$ can be seen to correspond to a region only slightly larger than that of $P(S_{5'}(f_1)|exon)$. If the contribution Δ to the difference in these two terms varies, due to statistical fluctuation, more than the difference between $P(f_1|donor)$ and $P(f_2|donor)$, then the lower-scoring donor site may be inadvertently favored during gene prediction.*

Duke UNIVERSITY

Coding-noncoding Boundaries

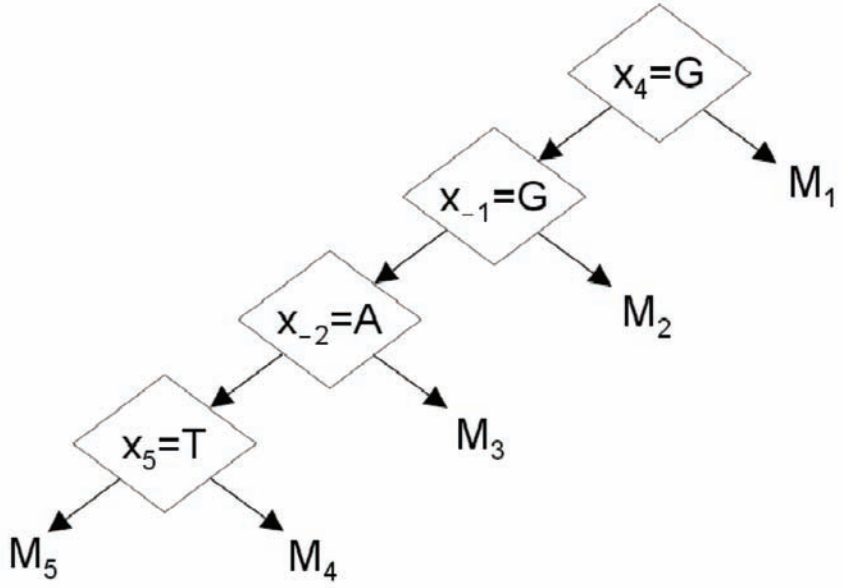
A key observation regarding splice sites and start and stop codons is that all of these signals delimit the boundaries between coding and noncoding regions within genes (although the situation becomes more complex in the case of *alternative splicing*). One might therefore consider weighting a signal score by some function of the scores produced by the coding and noncoding content sensors applied to the regions immediately 5′ and 3′ of the putative signal:

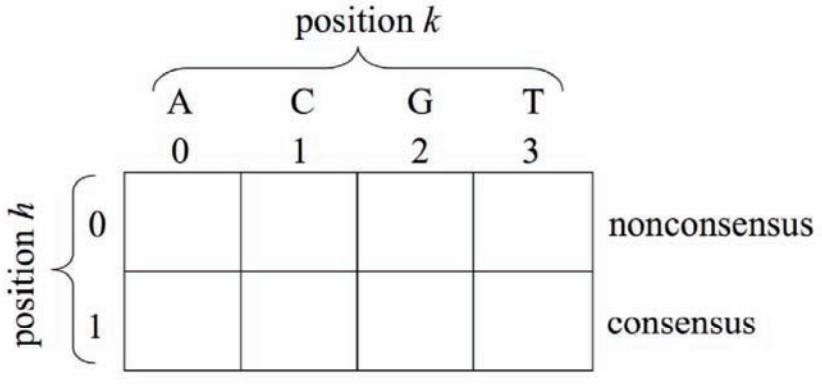
$$\frac{P(S_{5'}(f) \mid coding)}{P(S_{5'}(f) \mid noncoding)} P(f \mid donor) \frac{P(S_{3'}(f) \mid noncoding)}{P(S_{3'}(f) \mid coding)}$$

This approach is used in the popular program *GeneSplicer* (Pertea *et al.*, 2001).

Maximal Dependence Decomposition

A special type of signal sensor based on decision trees was introduced by the program GENSCAN (Burge, 1998). Starting at the root of the tree, we apply predicates over base identities at positions in the sensor window, which determine the path followed as we descend the tree. At the leaves of the tree are weight matrices specific to signal variants as characterized by the predicates in the tree.





Training an *MDD* tree can be performed by performing a number of χ^2 *tests of independence* between cells in the sensor window. At each bifurcation in the tree we select the predicate of *maximal dependence*.

Duke UNIVERSITY

Probabilistic Tree Models

Another tree-based signal sensor model is the *Probabilistic Tree Model* (*PTM*—Delcher *et al.*, 1999):

$$P(x_0 x_1 \ldots x_{L-1}) = \left(\prod_{\substack{\{i \mid \neg\exists \\ (i \to j) \in E\}}} P(x_i) \right) \left(\prod_{\substack{\{i \mid \exists \\ (i \to j) \in E\}}} P(x_i \mid succ(x_i)) \right)$$

for dependency graph $G=(V,E)$ with vertices $V=\{x_i \mid 0 \le i < L\}$ and edges $E \subseteq V \times V$, where $succ(x_i)=\{x_j \mid i \to j \in E\}$. If the dependency graph is a tree, this simplifies to:

$$P(x_0 x_1 \ldots x_{L-1}) = \left(\prod_{\substack{\{i \mid \neg\exists \\ (i \to j) \in E\}}} P(x_i) \right) \left(\prod_{(i \to j) \in E} P(x_i \mid x_j) \right)$$

In the PTM approach to dependence decomposition, dependencies in training data are modeled using *mutual information*:

$$M(X,Y) = \sum_{\substack{x \in dom(X), \\ y \in dom(Y), \\ P(x,y)>0}} P(x,y) \log_2 \frac{P(x,y)}{P(x)P(y)}$$

Summary

- *Feature sensing* is used by states of a GHMM or other gene-finding framework to evaluate parts of putative genes
- *Signal sensors* model fixed-length features such as start and stop codons, splice sites, and polyadenylation signals
- *Content sensors* model variable-length features such as exons, introns, and intergenic regions separating genes
- Some common signal sensor models are *WMM*'s, *WAM*'s, *WWAM*'s, *MDD* trees, and *PTM*'s
- The most common content sensor is the simple *Markov chain*
- *Interpolation* can be used for higher-order Markov chains, to avoid the effects of sampling error