Conditional Random Fields
for eukaryotic gene prediction

B. Majoros

Duke
UNIVERSITY

Recall: Discrete-time Markov Chains

A hidden *Markov model* for discrete sequences is a *generative* model denoted by:

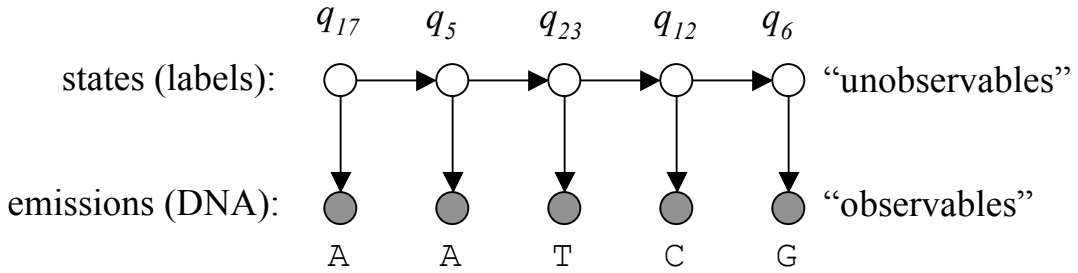
$$M = (Q, \alpha, P_t, P_e)$$

where:

- $Q = \{q_0, q_1, \ldots, q_n\}$ is a finite set of discrete states,
- α is a finite alphabet such as {A, C, G, T},
- $P_t(q_i \mid q_j)$ is a set of transition probabilities between states,
- $P_e(s_i \mid q_j)$ is set of emission probabilities within states.

During operation of the machine, emissions are *observable*, but states are not.

The (0^{th}-order) *Markov assumption* indicates that each state is dependent only on the immediately preceding state, and each emission is dependent only on the current state:

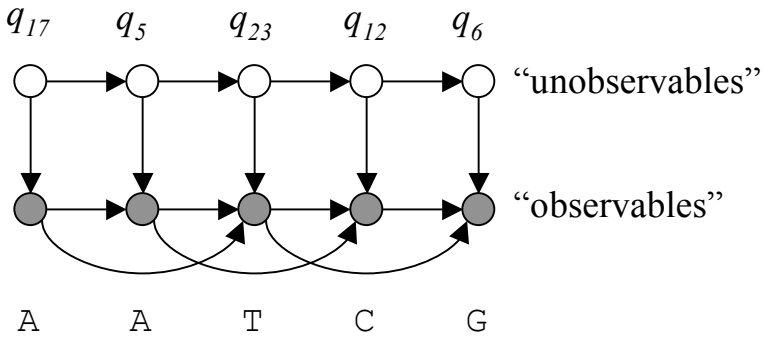


Decoding is the task of *finding the most probable values for the unobservables*.

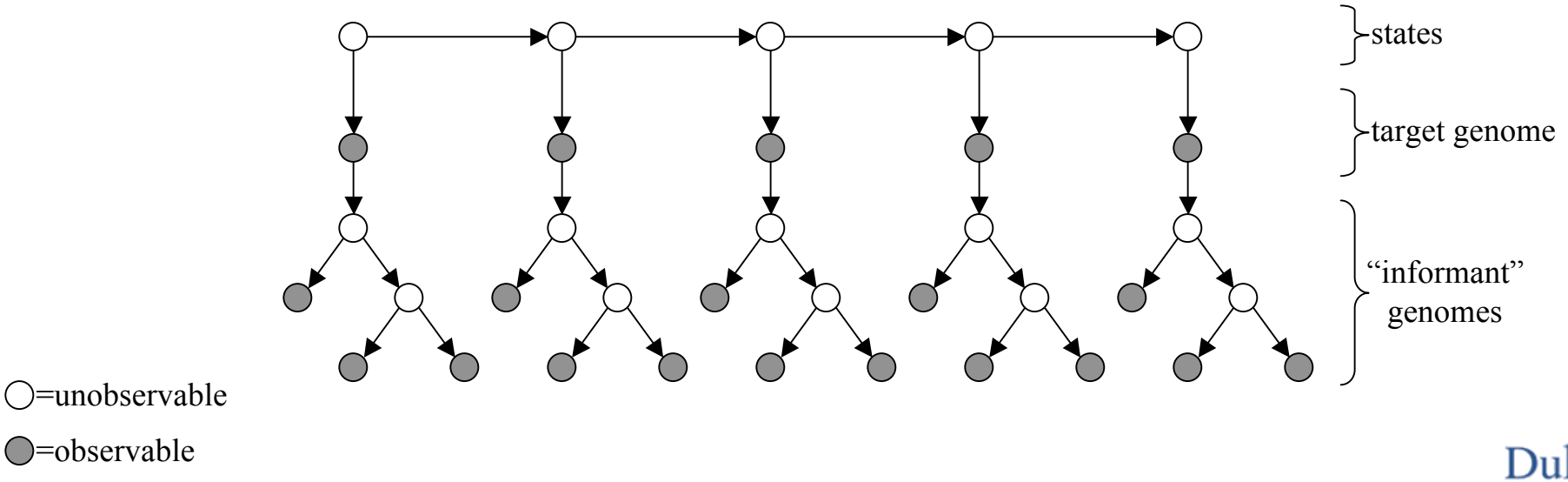
Duke UNIVERSITY

More General Bayesian Networks

Other topologies of the underlying *Bayesian network* can be used to model additional dependencies, such as higher-order emissions from individual states of a Markov chain:



Incorporating *evolutionary conservation* from an alignment results in a *PhyloHMM* (also a Bayesian network), for which efficient decoding methods exist:



Duke UNIVERSITY

Markov Random Fields

A (discrete-valued) *Markov random field* (*MRF*) is a 4-tuple $M=(\alpha, X, P_M, G)$ where:

- α is a finite *alphabet*,
- X is a set of (observable or unobservable) *variables* taking values from α,
- P_M is a *probability distribution* on variables in X,
- $G=(X, E)$ is an undirected graph on X describing a set of *dependence relations* among variables,

such that $P_M(X_i|\{X_{k\neq i}\}) = P_M(X_i|\mathcal{N}_G(X_i))$, for $\mathcal{N}_G(X_i)$ the neighbors of X_i under G.

That is, the conditional probabilities as given by P_M must obey the dependence relations (a generalized "Markov assumption") given by the undirected graph G.

A problem arises when actually inducing such a model in practice—namely, that we can't just set the conditional probabilities $P_M(X_i \mid \mathcal{N}_G(X_i))$ arbitrarily and expect the joint probability $P_M(X)$ to be well-defined (Besag, 1974).

Thus, the problem of estimating parameters locally for each neighborhood is confounded by constraints at the global level...

Duke UNIVERSITY

The Hammersley-Clifford Theorem

Suppose $P(\mathbf{x})>0$ for all (joint) value assignments $\mathbf{x}$ to the variables in X. Then by the Hammersley-Clifford theorem, the likelihood of $\mathbf{x}$ under model M is given by:

$$P_M(\mathbf{x}) = \frac{1}{Z} e^{Q(\mathbf{x})}$$

for normalization term Z:

$$Z = \sum_{\mathbf{x}'} e^{Q(\mathbf{x}')}$$

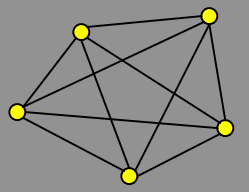
where $Q(\mathbf{x})$ has a unique expansion given by:

$$Q(x_0, x_1, \ldots, x_{n-1}) = \sum_{0 \le i < n} x_i \Phi_i(x_i) + \sum_{0 \le i < j < n} x_i x_j \Phi_{i,j}(x_i, x_j) + \ldots$$

$$\ldots + x_0 x_1 \ldots x_{n-1} \Phi_{0,1,\ldots,n-1}(x_0, x_1, \ldots, x_{n-1})$$

and where any Φ_i term not corresponding to a *clique* must be zero. (Besag, 1974)

What is a clique?



A clique is any subgraph in which all vertices are neighbors.

The reason this is useful is that it provides a way to evaluate probabilities (whether joint or conditional) based on the "local" functions Φ.

Thus, we can train an MRF by learning individual Φ functions—one for each clique.

Conditional Random Fields

A *Conditional random field* (*CRF*) is a *Markov random field* of *unobservables* which are globally conditioned on a set of *observables* (Lafferty *et al.*, 2001):

Formally, a CRF is a 6-tuple $M=(L,\alpha,Y,X,\Omega,G)$ where:

- L is a finite *output alphabet* of *labels*; e.g., {`exon`, `intron`},
- α is a finite *input alphabet* e.g., {`A`, `C`, `G`, `T`},
- Y is a set of *unobserved variables* taking values from L,
- X is a set of (fixed) *observed variables* taking values from α,
- $\Omega = \{\Phi_c : L^{|Y|} \times \alpha^{|X|} \rightarrow \mathbb{R}\}$ is a set of *potential functions*, $\Phi_c(\mathbf{y},\mathbf{x})$,
- $G=(V, E)$ is an undirected graph describing a set of *dependence relations* E among variables $V = X \cup Y$, where $E \cap (X \times X) = \varnothing$,

such that $(\alpha, Y, e^{\Sigma\Phi(c,\mathbf{x})}/Z, G-X)$ is a Markov random field.

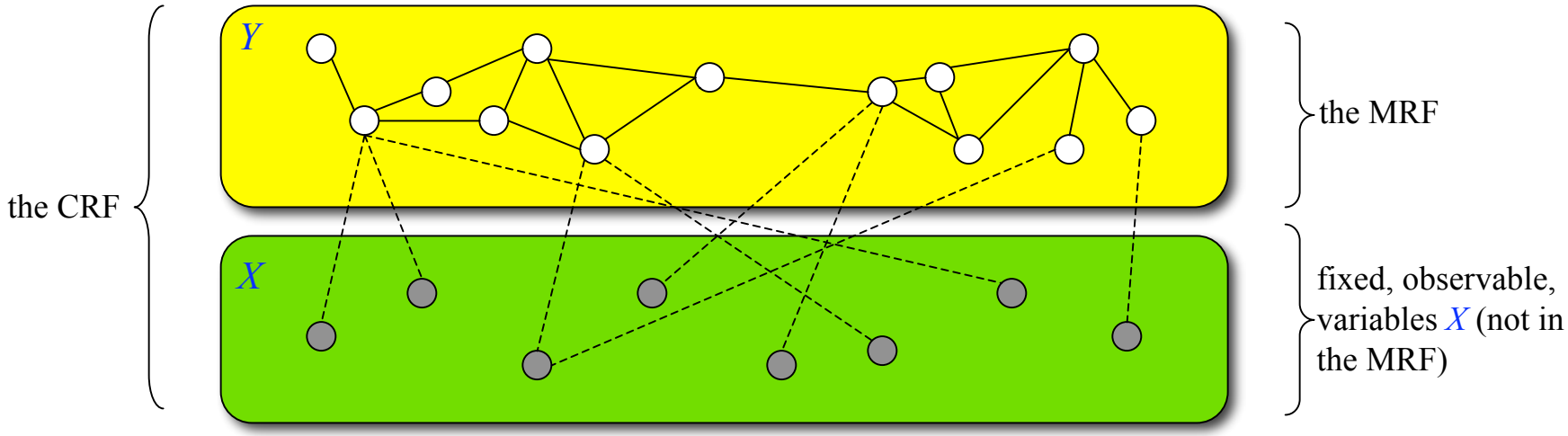
Note that:

1. The observables X are not included in the MRF part of the CRF, which is only over the subgraph $G-X$. However, the X are deemed *constants*, and are *globally visible* to the Φ functions.

2. We have not specified a probability function P_M, but have instead given "local" *clique-specific* functions Φ_c which together define a coherent probability distribution via Hammersley-Clifford.

CRF's versus MRF's

A Conditional random field is effectively an MRF plus a set of "external" variables X, where the "internal" variables Y of the MRF are the unobservables (○) and the "external" variables X are the observables (●):



Thus, we could denote a CRF informally as:

$$C=(M, X)$$

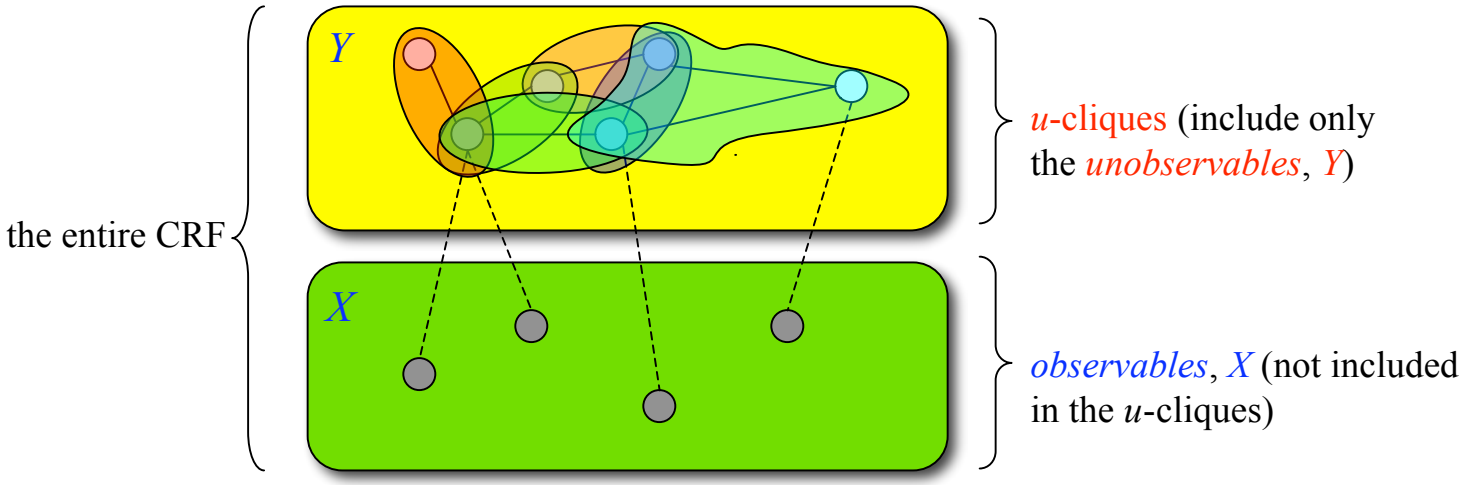
for MRF M and external variables X, with the understanding that the graph $G_{X \cup Y}$ of the CRF is simply the graph G_Y of the underlying MRF M plus the vertices X and any edges connecting these to the elements of G_Y.

Note that in a CRF *we do not explicitly model any direct relationships between the observables (i.e., among the X)* (Lafferty *et al.*, 2001).

Duke UNIVERSITY

U-Cliques

Because the observables X in a CRF are not included in the CRF's underlying MRF, Hammersley-Clifford applies only to the cliques in the MRF part of the CRF, which we refer to as the *u-cliques*:



Thus, we define the *u-cliques* of a CRF to be the cliques of the unobservable subgraph $G_Y=(Y, E_Y)$ of the full CRF graph $G_{X\cup Y}=(X\cup Y, E_{X\cup Y})$; $E_Y\subseteq Y\times Y$.

Whenever we refer to the "cliques" C of a CRF we will implicitly mean the *u*-cliques only. Note that we are permitted by Hammersley-Clifford to do this, since only the unobservable subgraph G_Y of the CRF will be treated as an MRF.

(NOTE: we will see later, however, that we may selectively include observables in the u-cliques)

Conditional Probabilities in a CRF

Since the observables X are *fixed*, the *conditional probability* $P(Y \mid X)$ of the unobservables given the observables is:

$$P_M(\mathbf{y} \mid \mathbf{x}) = \frac{1}{Z(\mathbf{x})} e^{Q(\mathbf{y},\mathbf{x})} = \frac{1}{\sum_{\mathbf{y}'} e^{Q(\mathbf{y}',\mathbf{x})}} e^{Q(\mathbf{y},\mathbf{x})}$$

Note that we are not summing over **x** in the denominator

where $Q(\mathbf{y},\mathbf{x})$ is evaluated via the *potential functions*—one per u-clique in the (MRF) dependency graph G_Y:

$$Q(\mathbf{y},\mathbf{x}) = \sum_{c \in C} \Phi_c(\mathbf{y}_c, \mathbf{x})$$

where $\mathbf{y}_c$ denotes the "slice" of vector $\mathbf{y}$ consisting of only those elements indexed by the set c (recall that, by Hammersley-Clifford, Φ_c may only depend on those variables in clique c).

Several important points:

1. The u-cliques C need not be *maximal cliques*, and they may *overlap*
2. The u-cliques contain only unobservables ($\mathbf{y}$); nevertheless, $\mathbf{x}$ is an argument to Φ_c
3. The probability $P_M(\mathbf{y}|\mathbf{x})$ is a *joint distribution* over the unobservables Y

The first point is one advantage of MRF's—the modeler need not worry about decomposing the computation of the probability into non-overlapping conditional terms. By contrast, in a Bayesian network this could result in "double-counting" of probabilities, and unwanted biases.

Common Assumptions

A number of *ad hoc* modeling decisions are typically made with regard to the form of the potential functions:

1. The $x_i x_j \ldots x_k$ coefficients in the $x_i x_j \ldots x_k G_{i,j,\ldots,k}(x_i, x_j, \ldots, x_k)$ terms from Besag's formula are typically ignored (they can in theory be absorbed by the potential functions).

2. Φ_c is typically decomposed into a weighted sum of feature sensors f_i, producing:

$$P(\mathbf{y} \mid \mathbf{x}) = \frac{1}{Z} e^{\sum_{c \in C} \sum_{i \in F} \lambda_i f_i(y_c, \mathbf{x})}$$

(Lafferty *et al.*, 2001)

3. Training of the model is typically performed in two steps (Vinson *et al.*, 2007):

   (*i*) train the individual feature sensors f_i (*independently*) on known features of the appropriate type

   (*ii*) learn the λ_i's using a *gradient ascent* procedure applied to the *entire* model all at once (not separately for each λ_i).

Simplifications for Efficient Decoding

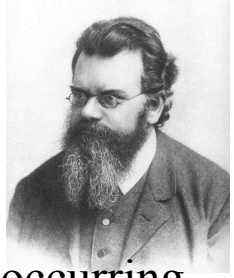
For "standard" decoding (i.e., *not* posterior decoding), in which we merely wish to find the most probable assignment $\mathbf{y}$ to the unobservables Y, *we can dispense with the partition function* (which is fortunate, since in the general case its computation may be intractable):

$$\arg\max_{\mathbf{y}} P(\mathbf{y} \mid \mathbf{x}) = \arg\max_{\mathbf{y}} \frac{1}{Z} e^{\sum_{c \in C} \Phi_c(\mathbf{y}_c, \mathbf{x})} = \arg\max_{\mathbf{y}} \sum_{c \in C} \Phi_c(\mathbf{y}_c, \mathbf{x})$$

In cases where the partition function is efficiently computable (such as for *linear-chain CRF's*, which we will describe later), posterior decoding is also feasible.

We will see later how the above optimization may be efficiently solved using dynamic programming methods originally developed for HMM's.

The Boltzmann Analogy



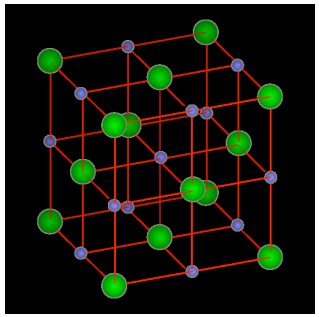
The *Boltzmann-Gibbs distribution* from statistical thermodynamics is strikingly similar to the MRF formulation:

$$P_{boltzmann}(\mathbf{x}) = \frac{1}{Z} e^{-E(\mathbf{x})/kT}$$

This gives the probability of a particular molecular configuration (or "microstate") **x** occurring in an ideal gas at temperature T, where k=1.38×10^{-23} is the *Boltzmann constant*. The normalizing term Z is known as the *partition function*. The exponent E is the *Gibbs free energy* of the configuration.

The MRF probability function may be conceptualized somewhat analogously, in which the summed "*potential functions*" Φ_c *(notice the difference in sign versus –E/kT)* reflect the "interaction potentials" between variables, and measure the "compatibility," "consistency," or "co-occurrence patterns" of the variable assignments **x**:

$$P_{MRF}(\mathbf{x}) = \frac{1}{Z} e^{\sum_{c \in C} \Phi_c(\mathbf{x}_c)}$$

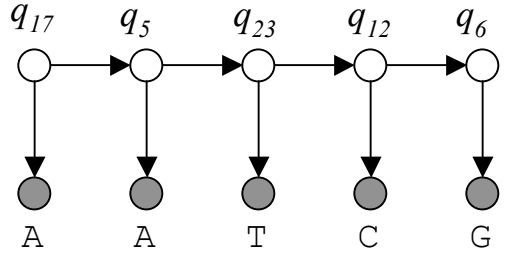


The analogy is most striking in the case of *crystal structures*, in which the molecular configuration forms a lattice described by an undirected graph of atomic-level forces.

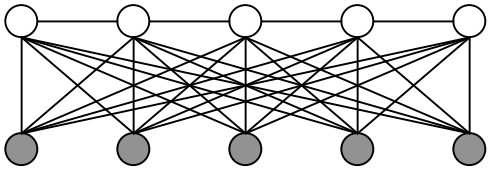
Although intuitively appealing, this analogy is not the justification for MRF's—the Hammersley-Clifford result provides a mathematically justified means of evaluating an MRF (and thus a CRF), and is not directly based on a notion of state "dynamics".

CRF's for DNA Sequence

Recall the directed dependency model for a (0^{th}-order) HMM:

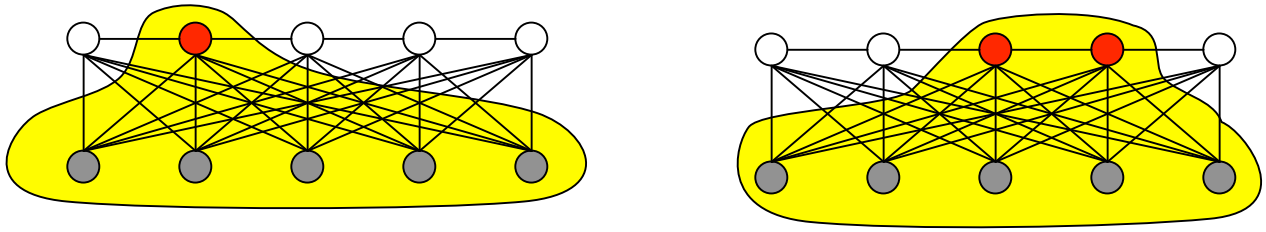


For gene finding, the unobservables in a CRF would be the labels (`exon`, `intron`) for each position in the DNA. In theory, these may depend on any number of the observables (the DNA):



Note that longer-range dependencies between labels are theoretically possible, but are not commonly used in gene finding (yet)

The *u-cliques* in such a graph can be easily identified as being either *singleton* labels or *pairs* of adjacent labels:



Such a model would need only two Φ_c functions—$\Phi_{singleton}$ for "singleton label" cliques (left figure) and Φ_{pair} for "pair label" cliques (right figure). We *could* evaluate these using the standard *emission* and *transition* distributions of an HMM (but we don't have to).

Duke UNIVERSITY

CRF's versus HMM's

Recall the decoding problem for HMM's, in which we wish to find the most probable parse ϕ of a DNA sequence S, in terms of the transition and emission probabilities of the HMM:

$$\arg\max_{\phi} P(\phi \mid S) = \arg\max_{\phi} P(\phi) P(S \mid \phi) = \arg\max_{\phi} \sum_{y_i \in \phi} \log\left(P_{trans}(y_i \mid y_{i-1}) P_{emit}(s_i \mid y_i) \right)$$

The corresponding derivation for CRF's is:

$$\arg\max_{\phi} P(\phi \mid S) = \arg\max_{\phi} \frac{1}{Z} e^{\sum \lambda f(c,S)} = \arg\max_{\phi} \sum_{c,i} \lambda_i f_i(c,S)$$

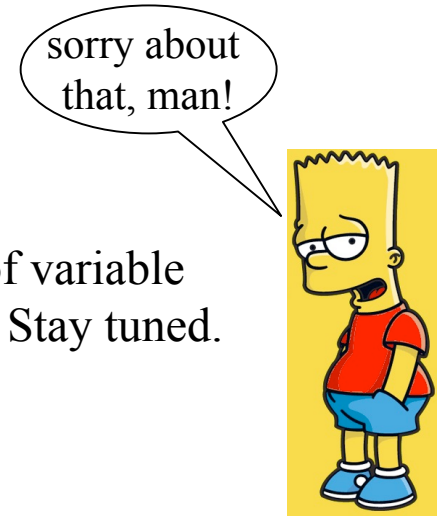
Note several things:

1. Both optimizations are over *sums*—this allows us to use any of the dynamic programming HMM/GHMM decoding algorithms for fast, memory-efficient parsing, with the CRF scoring scheme used in place of the HMM/GHMM scoring scheme.

2. The CRF functions $f_i(c,S)$ may in fact be implemented using any type of sensor, including such *probabilistic sensors* as Markov chains, interpolated Markov models (IMM's), decision trees, phylogenetic models, etc..., as well as any *non-probabilistic* sensor, such as n-mer counts or binary indicators on the existence of BLAST hits, etc...

How to Select Optimal Potential Functions

?



Aside from the Boltzmann analogy (i.e., “compatibility” of variable assignments), little concrete advice is available at this time. Stay tuned.

Training a CRF — Conditional Max Likelihood

Recall that (G)HMM's are typically trained via *maximum likelihood* (*ML*):

$$\theta_{MLE} = \arg\max_{\theta} \left(\prod_{(S,\phi)\in T} P_\theta(S,\phi) \right)$$

$$= \arg\max_{\theta} \left(\prod_{(S,\phi)\in T} \prod_{y_i \in \phi} P_e(S_i \mid y_i, d_i) P_t(y_i \mid y_{i-1}) P_d(d_i \mid y_i) \right)$$

due to the ease of computing this for fully-labeled training data—the P_e, P_t, and P_d terms can be maximized *independently* (and very *quickly* in the case of non-hidden Markov chains).

An alternative "*discriminative training*" objective function for (G)HMM's is *conditional maximum likelihood* (*CML*), which must be trained via gradient ascent or some EM-like approach:

$$\theta_{CML} = \arg\max_{\theta} \left(\prod_{(S,\phi)\in T} P_\theta(\phi \mid S) \right)$$

Although CML is rarely used for training gene-finding HMM's, it is a very natural objective function for CRF's, and is commonly used for training the latter models. Various gradient ascent approaches may be used for CML training of CRF's.

Thus, compared with Markov chains, CRF's should be more discriminative, much slower to train and possibly more susceptible to over-training.

Avoiding Overfitting with Regularization

Because CRF's are discriminatively trained, they sometimes suffer from overfitting of the model to the training data. One method for avoiding overfitting is *regularization*, which penalizes extreme values of parameters:

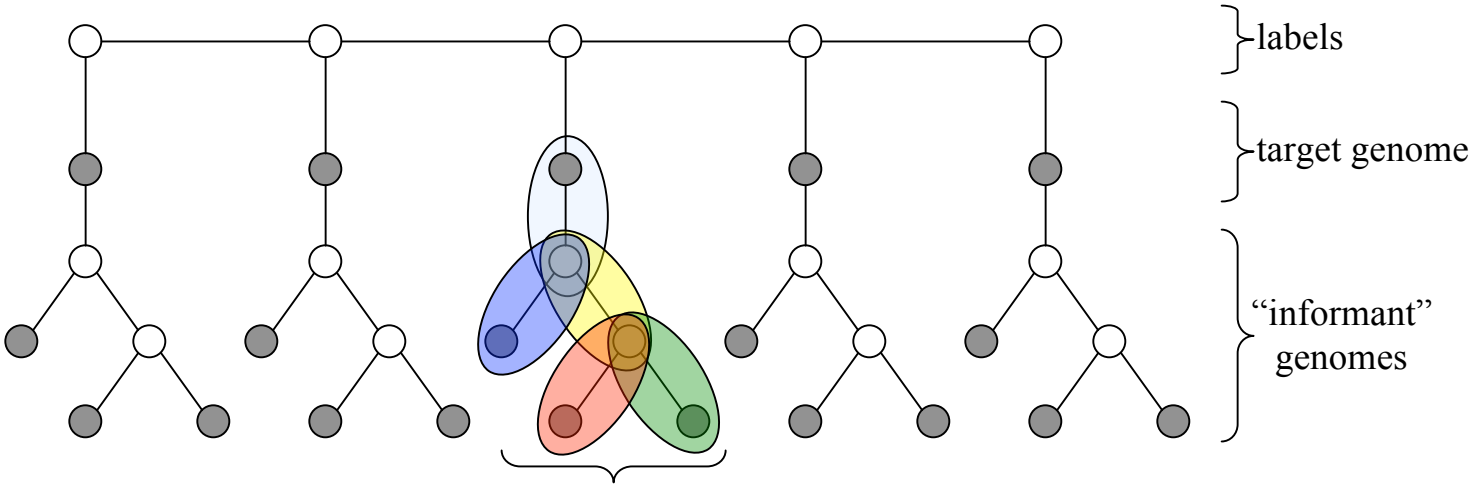
$$f_{objective}(\theta) = P_\theta(\mathbf{y} \mid \mathbf{x}) - \frac{\|\theta\|^2}{2\sigma^2}$$

where $\|\theta\|$ is the norm of the parameter vector θ, and σ is a regularization parameter (or "*metaparameter*") which is generally set in an *ad hoc* fashion but is thought to be generally benign when not set correctly (Sutton & McCallum, 2007).

The above function $f_{objective}$ serves as the objective function during training, in place of the usual $P(\mathbf{y}|\mathbf{x})$ objective function of *conditional maximum likelihood* (CML) training. Maximization of the objective function thus performs a modified conditional maximum likelihood optimization in which the parameters are simultaneously subjected to a Gaussian prior (Sutton & McCallum, 2007).

Phylo-CRF's

Analogous to the PhyloHMM's described earlier, we can formulate a "PhyloCRF" by incorporating phylogeny information into the dependency graph:



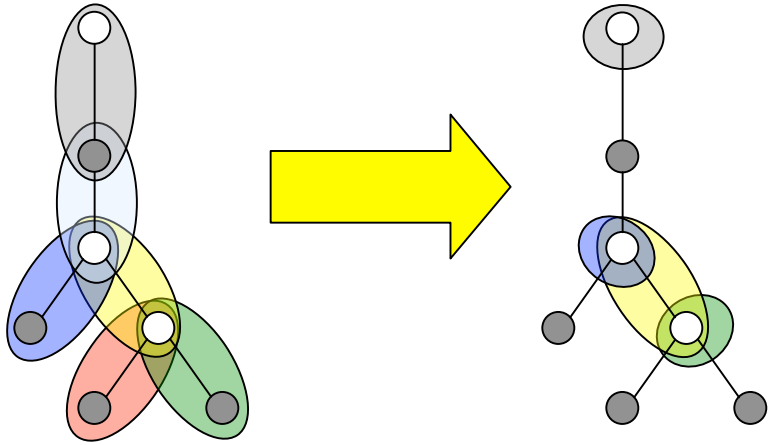
Note how the clique decomposition maps nicely into the recursive decomposition of Felsenstein's algorithm!

The white vertices in the informant trees denote *ancestral genomes*, which are not available to us, and which we are not interested in inferring; they are used merely to control for the non-independence of the informants. We call these *latent variables*, and denote this set L, so that the model now consists of three disjoint sets of variables: X (observables), Y (labels), and L (latent variables).

Note that this is still technically a CRF, since the dependencies between the observables are modeled only indirectly, through the latent variables (which are unobservable).

U-cliques in a PhyloCRF

Note that the "cliques" identified in the phylogeny component of our PhyloCRF contained observables, and therefore are not true *u*-cliques. However, we can identify *u*-cliques corresponding (roughly) to the original cliques, as follows:



Recall that the observables **x** are globally visible to all Φ functions. Thus, we are free to implement any specific Φ_c so as to utilize any subset $\mathbf{x}'$ of the observables.

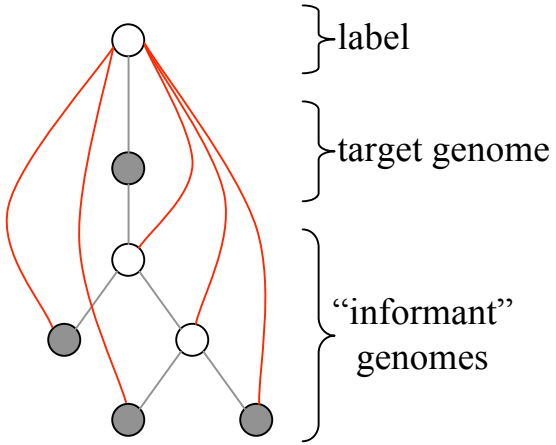
As a result, any *u*-clique c may be treated by Φ_c as a "virtual clique" (*v-clique*) $c \cup \mathbf{x}'$ which includes observables from $\mathbf{x}'$. In this way, the *u*-cliques (shown on the right above) may be effectively expanded to include observables as in the figure on the left.

Including Labels in the Potential Functions

In order for the patterns of conservation among the informants to have any effect on decoding, the Φ_c functions evaluated over the branches of the tree need to take into consideration the putative label (e.g., `coding`, `noncoding`) at the current position in the alignment. This is analogous to the use of separate *evolution models* for the different states q in a PhyloHMM:

$$P(I^{(1)},\ldots,I^{(n)} \mid S, q)$$

The same effect can be achieved in the PhyloCRF very simply by introducing edges connecting all informants and their ancestors directly to the label:



The only effect on the clique structure of the graph is to include the label in all (maximal) cliques in the phylogeny. The Φ_c functions can then evaluate the conservation patterns along the branches of the phylogeny in the specific context of a given label—i.e.,

$$\Phi_{mr}(X_{mouse}=\texttt{C}, X_{rodent}=\texttt{G}, Y=\texttt{exon}) \quad \text{vs.} \quad \Phi_{mr}(X_{mouse}=\texttt{C}, X_{rodent}=\texttt{G}, Y=\texttt{intron})$$

Duke UNIVERSITY

The Problem of Latent Variables

In order to compute $P(\mathbf{y}|\mathbf{x})$ in the presence of latent variables, we have to sum over all possible assignments $\mathbf{l}$ to the variables in L:

$$P_M(\mathbf{y} \mid \mathbf{x}) = \sum_{\mathbf{l}} P_M(\mathbf{y},\mathbf{l} \mid \mathbf{x}) = \sum_{\mathbf{l}} \frac{1}{Z(\mathbf{x})} e^{Q(\mathbf{y},\mathbf{l},\mathbf{x})} = \frac{\sum_{\mathbf{l}} e^{Q(\mathbf{y},\mathbf{l},\mathbf{x})}}{\sum_{\mathbf{y}',\mathbf{l}} e^{Q(\mathbf{y}',\mathbf{l},\mathbf{x})}}$$

(Quattoni *et al.*, 2006)

For "Viterbi" decoding we can again ignore the denominator:

$$\arg\max_{\mathbf{y}} P_M(\mathbf{y} \mid \mathbf{x}) = \arg\max_{\mathbf{y}} \sum_{\mathbf{l}} e^{Q(\mathbf{y},\mathbf{l},\mathbf{x})}$$

Unfortunately, performing this sum over the latent variables outside of the potential function Q will be much slower than Felsenstein's dynamic programming method for evaluating phylogenetic trees having "latent" (i.e., "ancestral") taxa.

However, evaluating Q on the cliques $c \in C$ as usual (but omitting singleton cliques containing only a latent variable) and shuffling terms gives us:

$$\sum_{\mathbf{l}} e^{\sum_{c \in C} \Phi(c,\mathbf{x})} = \sum_{\mathbf{l}} e^{\sum_{c \in C} \log(e^{\Phi(c,\mathbf{x})})} = \sum_{\mathbf{l}} e^{\log \prod_{c \in C} e^{\Phi(c,\mathbf{x})}} = \sum_{\mathbf{l}} \prod_{c \in C} e^{\Phi(c,\mathbf{x})}$$

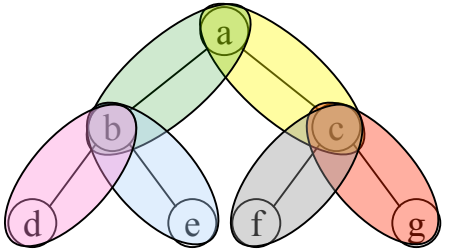
Now we can expand the summation over individual latent variables and factor individual summations *within* the evaluation of Q...

Duke UNIVERSITY

Factoring Along a Tree Structure

Consider the tree structure below. To simplify notation, let $\xi(\cdot)$ denote $e^{\Phi(\cdot)}$. Then the $\sum\prod e^{\Phi(c,\mathbf{x})}$ term from the previous slide expands along the cliques of the tree as follows:

$$\sum_a \sum_b \sum_c \sum_d \sum_e \sum_f \sum_g \xi(a,b)\xi(a,c)\xi(b,d)\xi(b,e)\xi(c,f)\xi(c,g)\xi(a)\xi(b)\xi(c)\xi(d)\xi(e)\xi(f)\xi(g)$$



Any term inside a summation which does not contain the summation index variable can be *factored out* of that summation:

CRF:
$$\sum_a \xi(a)\left[\sum_b \xi(a,b)\left(\sum_d \xi(b,d)\xi(d)\right)\left(\sum_e \xi(b,e)\xi(e)\right)\right]\left[\sum_c \xi(a,c)\left(\sum_f \xi(c,f)\xi(f)\right)\left(\sum_g \xi(c,g)\xi(g)\right)\right]$$

Felsenstein:
$$\sum_a P_{HMM}(a)\left[\sum_b P_{a\to b}\left(\sum_d P_{b\to d}\delta(d,x_d)\right)\left(\sum_e P_{b\to e}\delta(e,x_e)\right)\right]\left[\sum_c P_{a\to c}\left(\sum_f P_{c\to f}\delta(f,x_f)\right)\left(\sum_g P_{c\to g}\delta(g,x_g)\right)\right]$$

Now compare the *CRF formulation* (top) to the Bayesian network formulation under *Felsenstein's recursion* (bottom), where $P_{a\to b}$ is the lineage-specific *substitution probability*, $\delta(d,x_d)=1$ iff $d=x_d$ (otherwise 0), and $P_{HMM}(a)$ is the probability of a under a standard HMM.

We can also introduce λ terms as in the common "linear combination" expansion of Q:

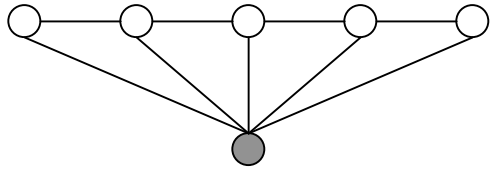
$$\sum_a \xi(a)\left[\sum_b \lambda_{a,b}\xi(a,b)\left(\sum_d \lambda_{b,d}\xi(b,d)\xi(d)\right)\left(\sum_e \lambda_{b,e}\xi(b,e)\xi(e)\right)\right]\left[\sum_c \lambda_{a,c}\xi(a,c)\left(\sum_f \lambda_{c,f}\xi(c,f)\xi(f)\right)\left(\sum_g \lambda_{c,g}\xi(c,g)\xi(g)\right)\right]$$

which may allow the CRF trainer to learn more discriminative "branch lengths".

Duke UNIVERSITY

Linear-Chain CRFs (LC-CRF's)

A common CRF topology for sequence parsing is the *linear-chain CRF* (*LC-CRF*) (Sutton & McCallum, 2007):



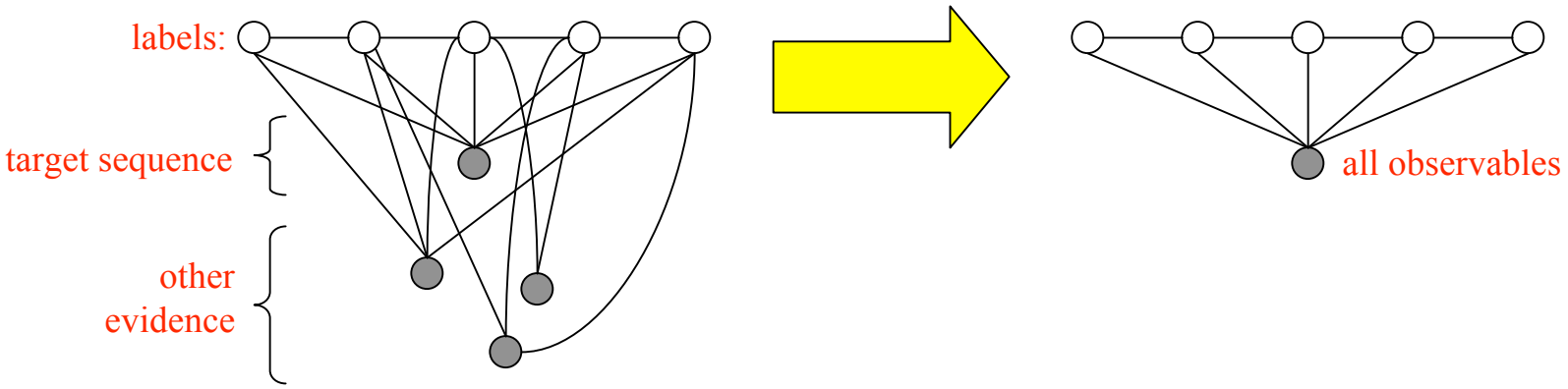
For visual simplicity, all of the observables are denoted by a single shaded node. Because of the simplified structure, the *u*-cliques are now trivially identifiable as *singleton labels* (corresponding to "emission" functions f_{emit}) and *pairs of labels* (corresponding to "transition" functions f_{trans}):

$$P(\mathbf{y} \mid \mathbf{x}) = \frac{1}{Z} e^{\sum_{c \in C} \sum_{i \in F} \lambda_i f_i(c)} = \frac{1}{Z} e^{\sum_{t=0}^{|S|-1} \pi f_{emit}(\mathbf{x}, \mathbf{y}_t) + \mu f_{trans}(\mathbf{x}, \mathbf{y}_{t-1}, \mathbf{y}_t)}$$

where we have made the common modeling assumption that the Φ functions expand as linear combinations of "feature functions" f_i.

Abstracting External Information via Feature Functions

The "feature functions" of a CRF's provide a convenient way of incorporating additional external evidence:



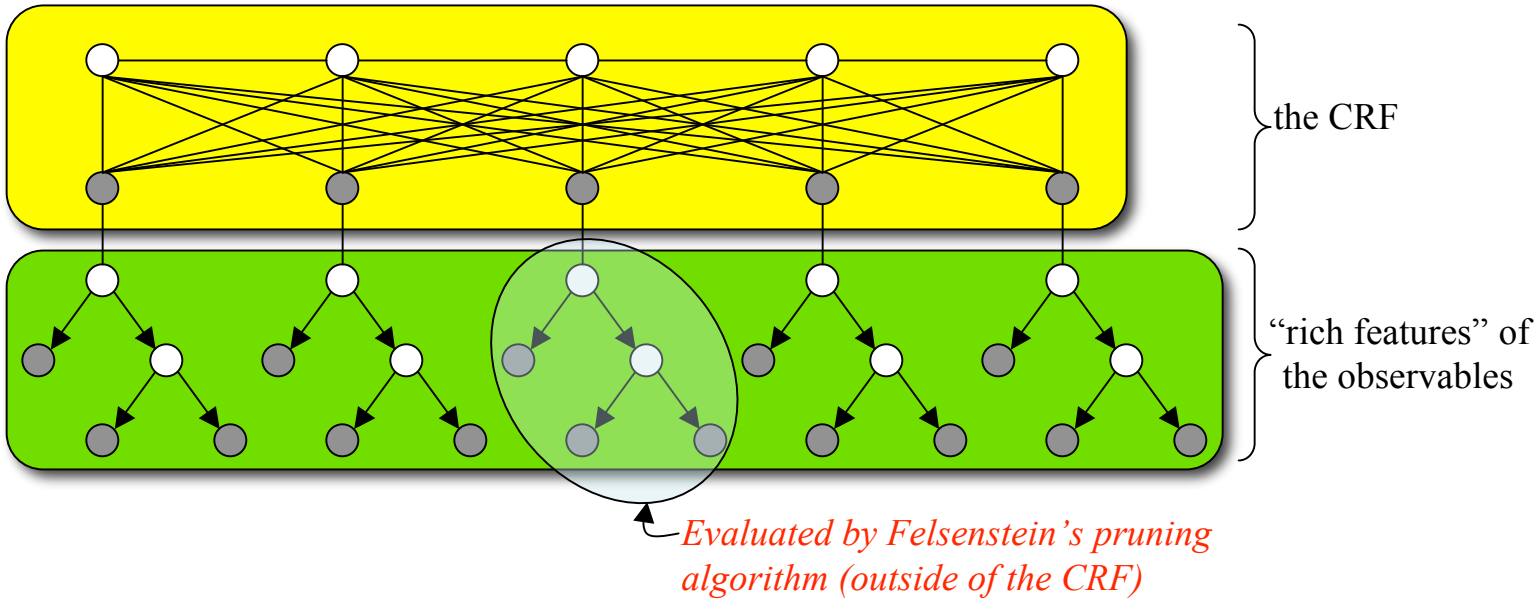
Additional "informant" evidence is now modeled not with additional vertices in the dependency graph, but with additional "rich feature" functions in the decomposition of Q (Sutton & McCallum, 2007):

$$P(\mathbf{y} \mid \mathbf{x}) = \frac{1}{Z} e^{\sum_{c \in C} \sum_{i \in F} \lambda_i f_i(c)} = \frac{1}{Z} e^{\sum_{t=0}^{|S|-1} \pi f_{emit}(\mathbf{x}, \mathbf{y}_t) + \mu f_{trans}(\mathbf{x}, \mathbf{y}_{t-1}, \mathbf{y}_t) + \sum_{i \in F} \beta_i f_{rich(i)}(\mathbf{x}, \mathbf{y}_t)}$$

where the "informants" and other external evidence are now encapsulated in **x**.

Phylo-CRF's Revisited

Now a "PhyloCRF" can be formulated more simply as a CRF with a "rich feature" function that applies Felsenstein's algorithm in each column (Vinson *et al.*, 2007):



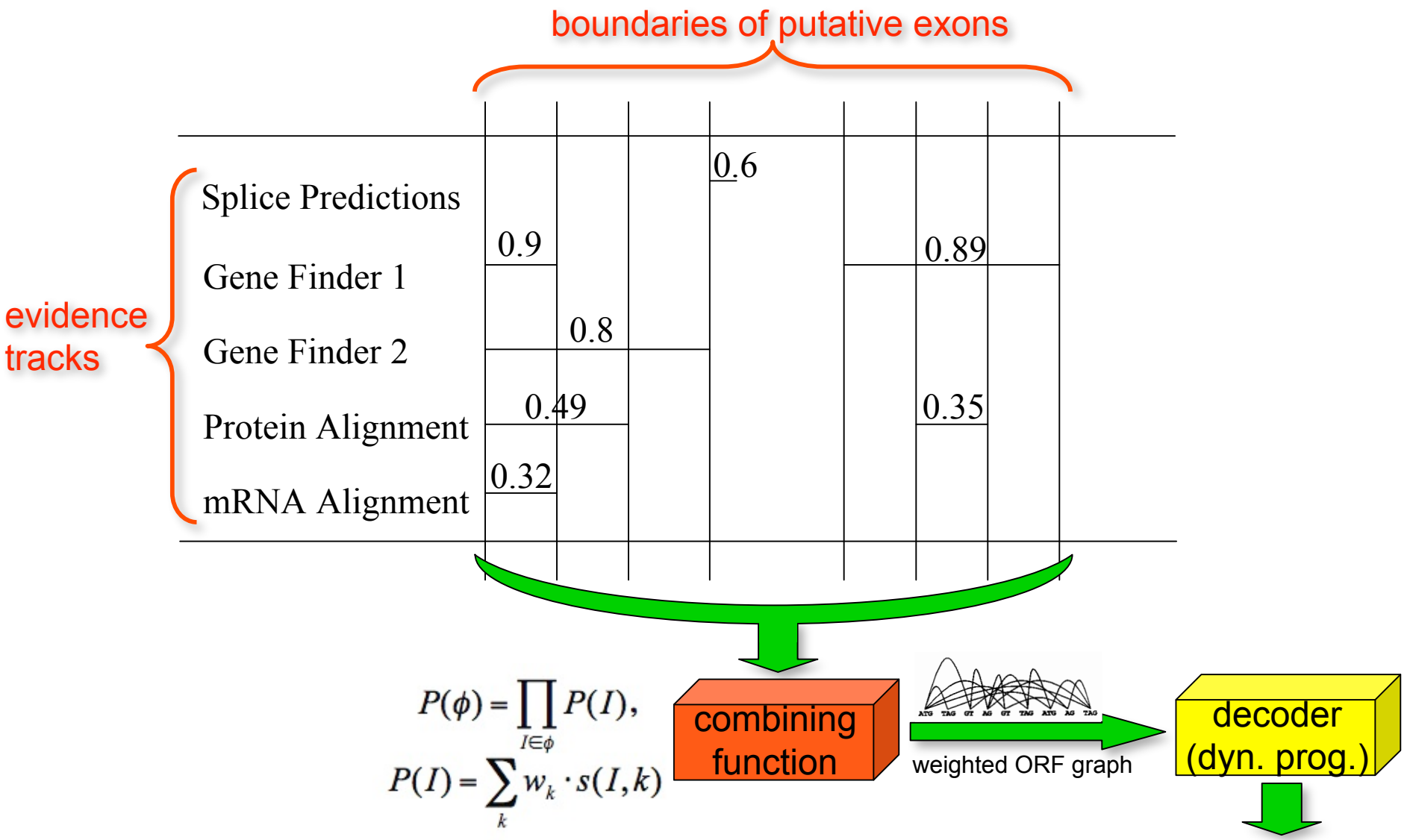
Note that the resulting model is a *hybrid* between an undirected model (the CRF) and a directed model (the phylogeny).

Is this optimal? Maybe not—the CRF training procedure cannot modify any of the parameters *inside* of the phylogeny submodel so as to improve discrimination (i.e., labeling accuracy).

Then again, this separation may help to prevent *overfitting*.

Duke UNIVERSITY

This Sounds Like a "Combiner"!



(upper figure due to J. Allen, Univ. of MD)

A CRF is in some sense just a theoretically-justified "Combiner" program

Duke UNIVERSITY

So, Why Bother with CRF's at All?

Several advantages are still derived from the use of the "hybrid" CRF (i.e., CRF's with "rich features"):

1. The λ's provide a "hook" for *discriminative training* of the overall model (though they do not attend to the optimality, at the global level, of the parameterizations of the submodels).

2. For certain training regimes (e.g., CML), the objective function is provably *convex*, ensuring convergence to a global optimum (Sutton & McCallum, 2007).

3. *Long-range dependencies* between the unobservables may still be modeled (though this hasn't so far been used for gene prediction).

4. Use of a linear chain CRF (LC-CRF) usually renders the partition function efficiently computable, so that *posterior decoding* is feasible.

3. Using a system-level CRF provides a theoretical justification for the use of so-called *fudge-factors* (i.e., the λ's) for weighting the contribution of submodels...

The Ubiquity of Fudge Factors

Many "probabilistic" gene finders utilize fudge factors in their source code, despite no obvious theoretical justification for their use:

- folklore about $\sqrt{7}/3$ in the source code of a certain popular *ab initio* gene finder[1]
- fudge factor in: NSCAN ("conservation score coefficient"; Gross & Brent, 2005)
- fudge factor in: ExoniPhy ("tuning parameter"; Siepel & Haussler, 2004)
- fudge factor in TWAIN ("percent identity"; Majoros *et al.*, 2005)
- fudge factor in GlimmerHMM ("optimism"; M. Pertea, *pers. communication*)
- fudge factor in TIGRscan ("optimism"; Majoros *et al.*, 2004)
- *lack* of fudge factors in EvoGene (Pedersen & Hein, 2003)

Thus, these programs are all instances of (highly simplified) CRF's!

or, to put it another way:

We should have been using CRF's all along...

[1] *folklore also states that this programs's author made a "pact with the devil" in exchange for gene-finding accuracy; attempts to replicate this effect have so far been unsuccessful (unpub. data).*

Duke UNIVERSITY

Vinson *et al.*: PhyloCRF's

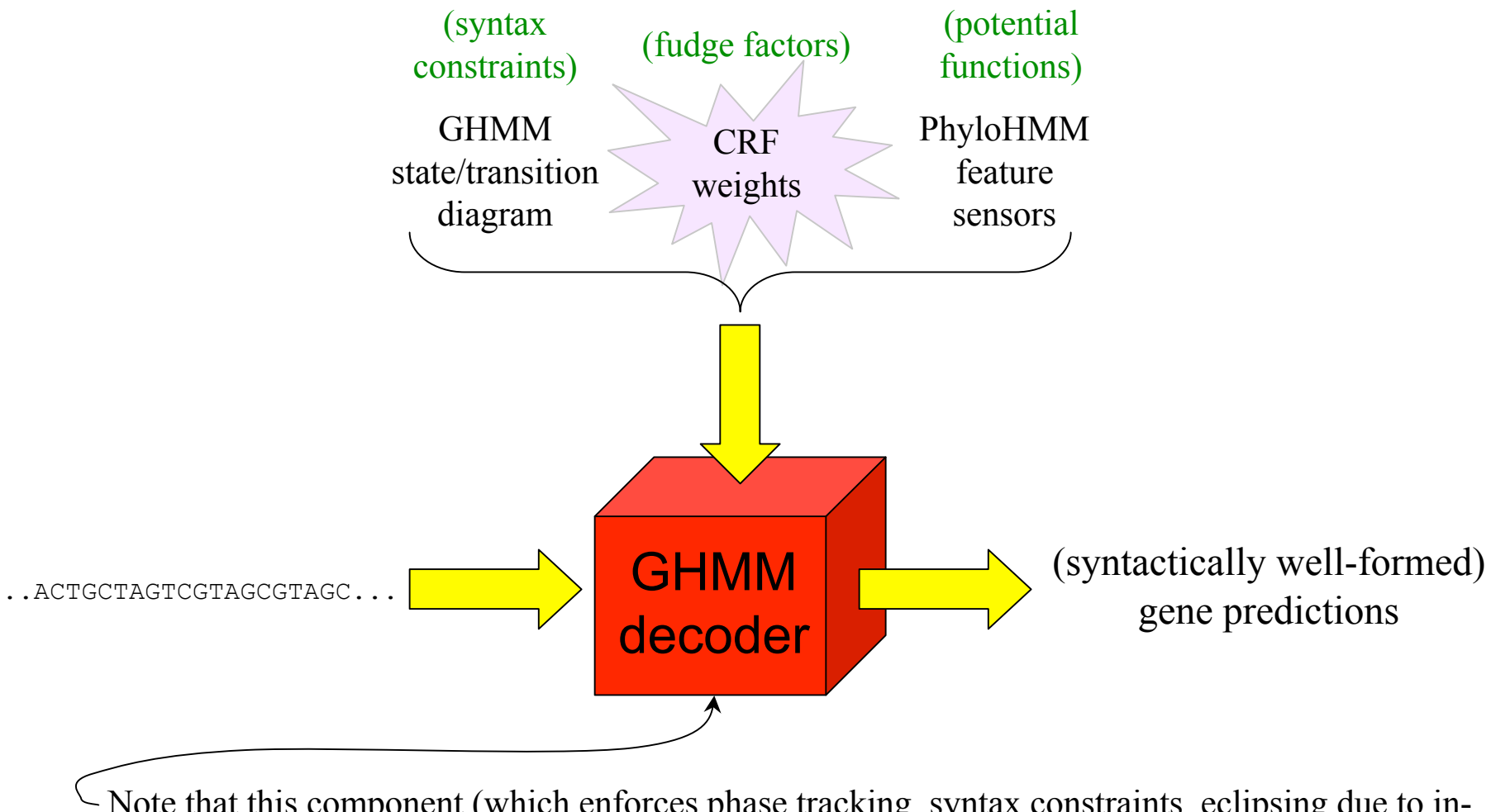
Vinson *et al*. (2007) implemented a phylogenetically-aware LC-CRF using the following features:

- standard GHMM signal/content *sensors*
- standard GHMM state topology (i.e., gene syntax)
- a standard *phylogeny* module (i.e., Felsenstein's algorithm)
- a *gap* term (for gaps in the aligned informant genome)
- an *EST* term

These authors also suggest the following principle for designing CRF's for gene prediction:

> *"...use probabilistic models for feature functions when possible and add non-probabilistic features only when necessary".* (Vinson *et al*., 2007)

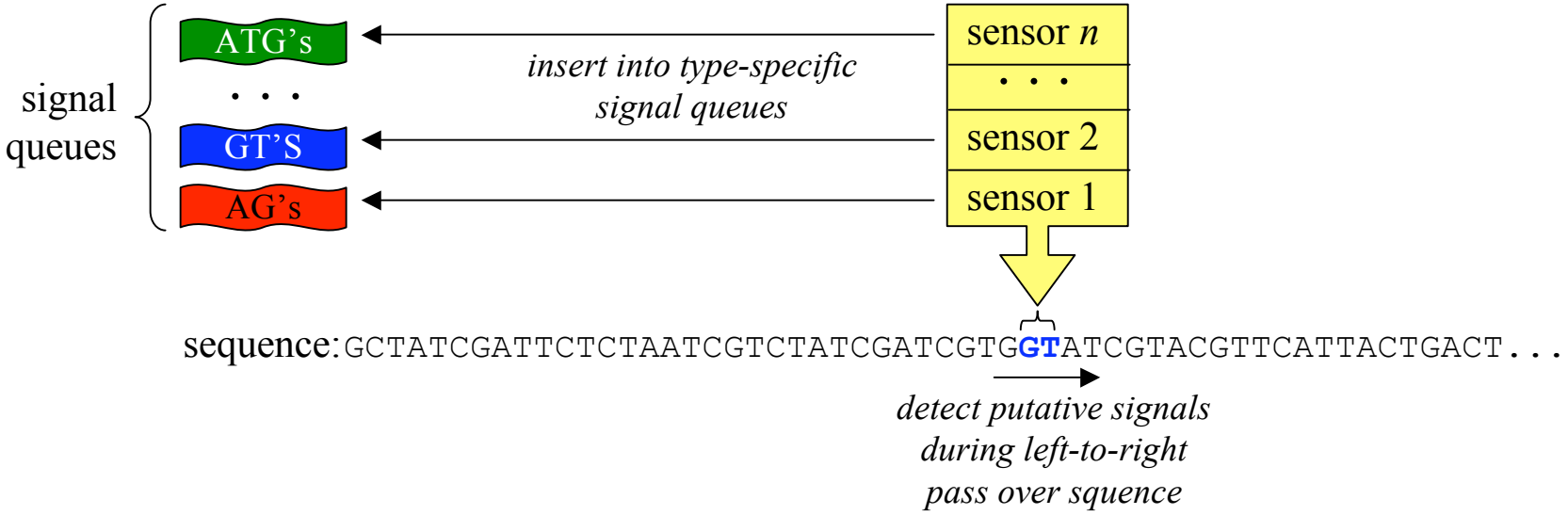
So...How Different Is This, Really?

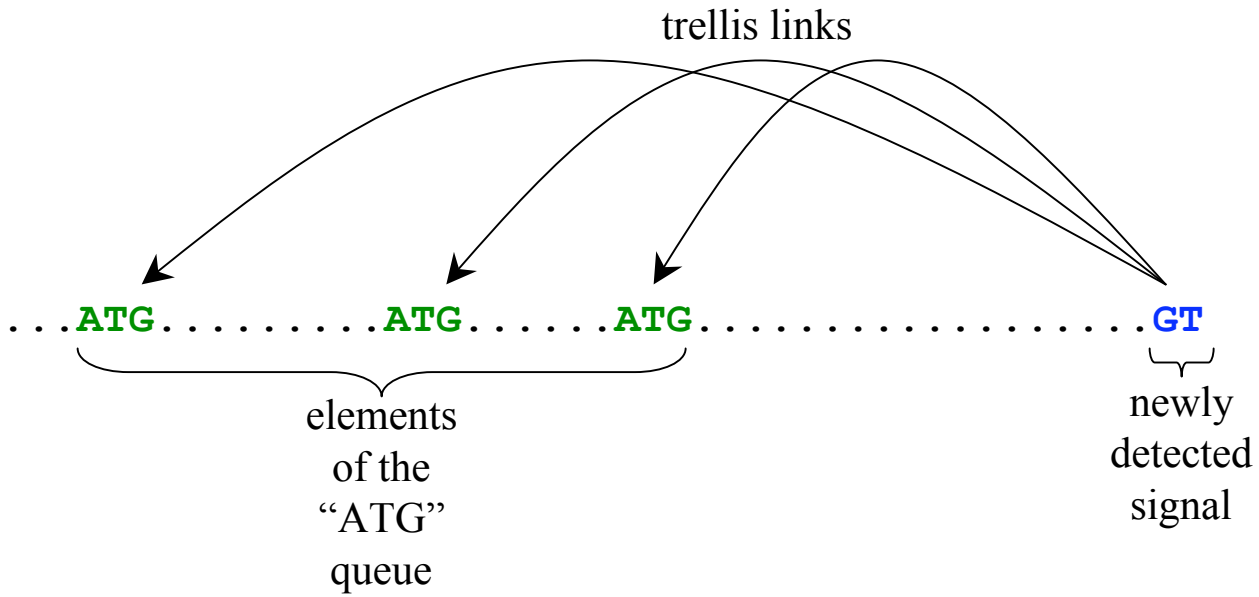


Note that this component (which enforces phase tracking, syntax constraints, eclipsing due to in-frame stop codons, etc.) is often the most difficult part of a eukaryotic gene finder to efficiently implement and debug. All of these functionalities are needed by CRF-based gene finders. Fortunately, the additive nature of the (log-space) HMM and CRF objective functions enables very similar code to be used in both cases.

Duke UNIVERSITY

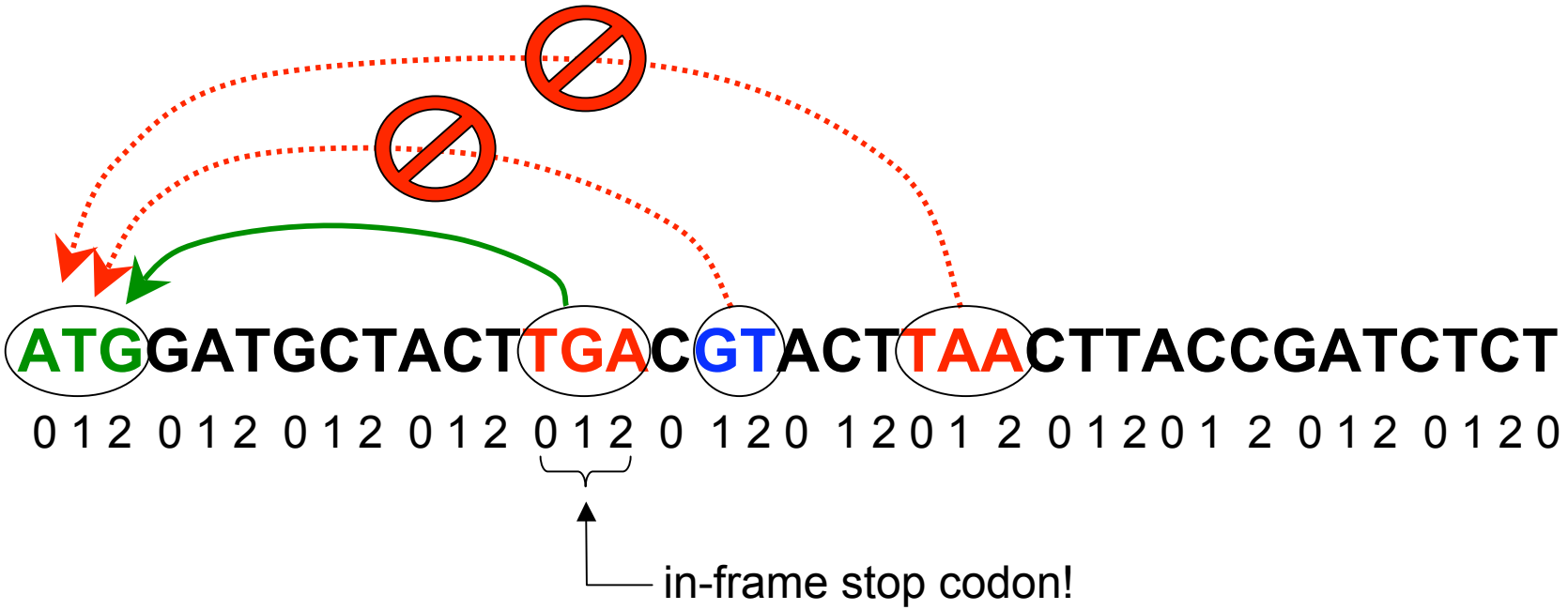
Recall: Decoding via Sensors and Trellis Links





Duke UNIVERSITY

Recall: Phase Constraints and "Eclipsing"



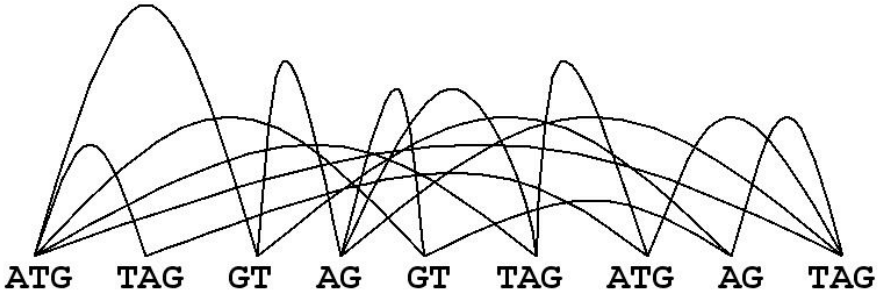
All of these syntactic constraints have to be tracked and enforced, just like in a "generative" gene finder!

In short: *gene syntax hasn't changed, even if our model has!*

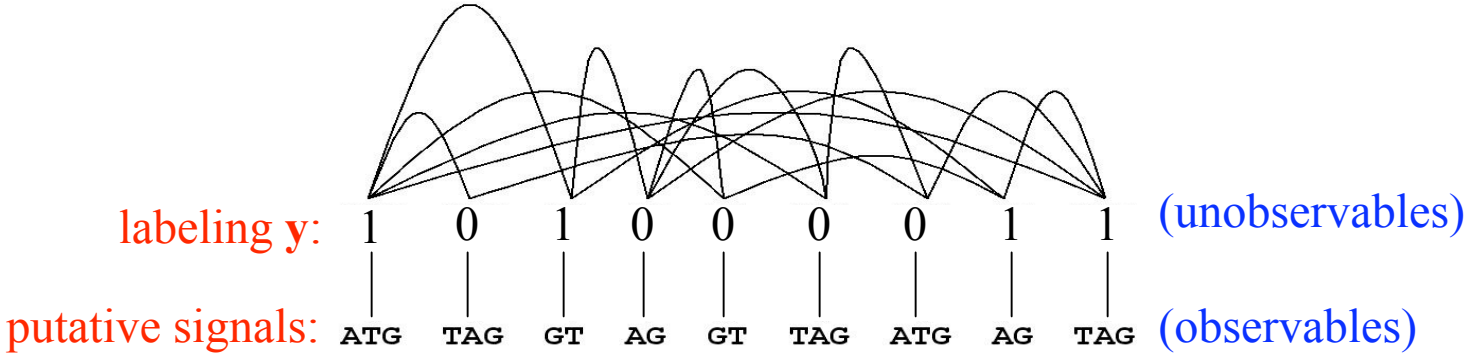
Duke UNIVERSITY

"Generalized" or "Semi-Markov" CRF's

A CRF can be very easily generalized into a "GCRF" so as to model feature lengths, by utilizing an *ORF graph* as described previously for GHMM's:



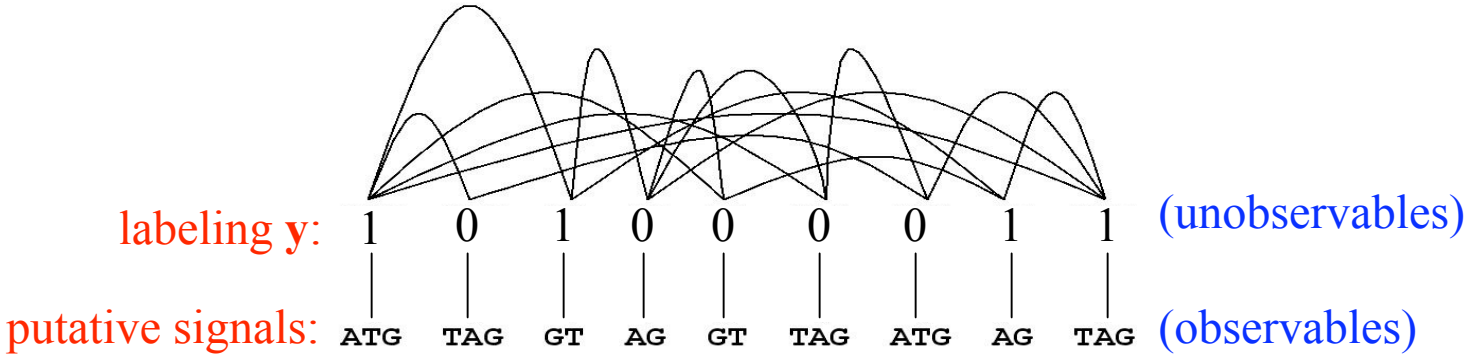
The labeling **y** of a GCRF is a *vector of indicators* from {0,1}, where a '1' indicates that the corresponding signal in the ORF graph is part of the predicted parse ϕ, and a '0' indicates that it is not. We can then use the ORF graph to relate the *labels* (unobservables) instead of the *putative signals* (the observables), to obtain a CRF:



Although this figure does not show it, each label will also have dependencies on other nearby labels in the graph, besides those adjacent via the "ORF graph" edges—i.e., there are implicit edges not shown in this representation. We will come back to this.

Cliques in a GCRF

The *u*-cliques of the GCRF are *singletons* (individual signals) and *pairs of signals* (i.e., an intron, an exon, a UTR, etc.):



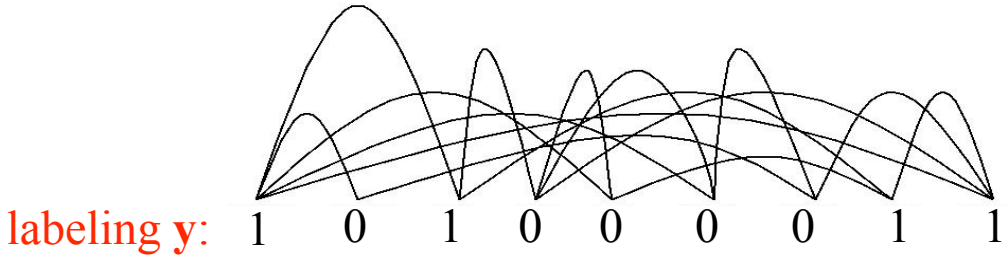
The Φ_{pair} potential function can thus be decomposed into the familiar three terms for "*emission potential*", "*transition potential*", and "*duration potential*", which may be evaluated in the usual way for a GHMM, or via non-probabilistic methods if desired:

$$P(\phi \mid S) = \frac{1}{Z} e^{\sum_{(s,t)\in\phi} \lambda_{emit} f_{emit}(t,S) + \lambda_{trans} f_{trans}(s,t) + \lambda_{length} f_{length}(s,t)}$$

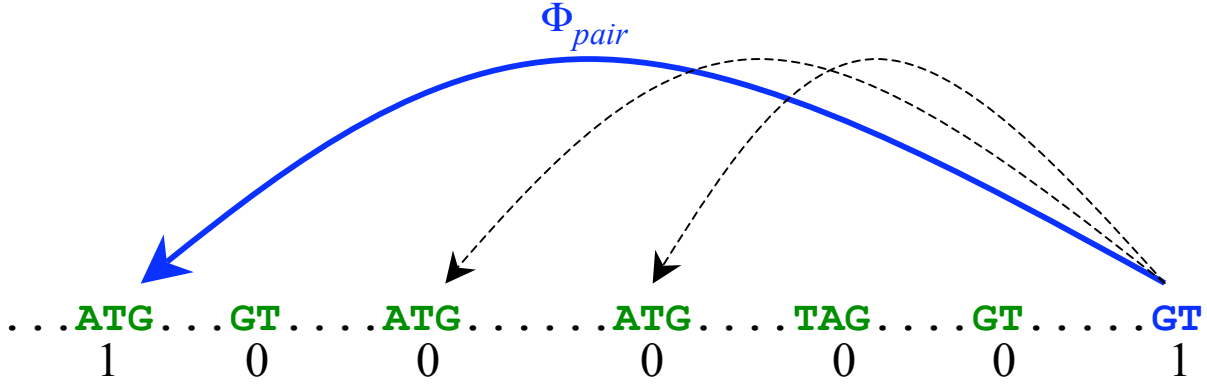
where $(s,t)\in\phi$ are pairs of signals in a parse ϕ. Under Viterbi decoding this again simplifies to a summation, and is thus efficiently computable using any GHMM decoding framework (but with the CRF scoring function in place of the GHMM one).

Enforcing Syntax Constraints

Note that it is possible to construct a labeling **y** which is not syntactically valid, because the signals do not form a *consistent path across the entire ORF graph*. We are thus interested in constraining the Φ functions so that only valid labelings have nonzero scores:



This can be handled by augmenting Φ_{pair} so as to evaluate to 0 unless the pair is well-formed: i.e., *the paired signals must be labeled '1' and all signals lying between them must be labeled '0'*:



Finally, to enforce *phase constraints* we need to use *three copies of the ORF graph*, with links between the three graphs enforcing phase constraints based on lengths of putative features (not shown).

Duke UNIVERSITY

Summary

A *CRF*, as commonly formulated for gene prediction, is essentially just a GHMM/GPHMM/PhyloGHMM, except that:

- every sensor has a *fudge factor*
- those fudge factors now have a *theoretical justification*
- the fudge factors should be optimized systematically, rather than being *tweaked by hand* (currently the norm)
- the sensors need not be *probabilistic* (i.e., n-gram counts, gap counts, binary indicators reflecting presence of genomic elements such as CpG islands or BLAST hits or ...)

CRF's may be viewed as theoretically justified *combiner-type* programs, which traditionally have produced very high prediction accuracies despite being viewed (in the pre-CRF world) as *ad hoc* in nature.

Use of *latent variables* allows more general modeling with CRF's than via the simple *"rich feature"* approach.

THE END

References

Besag J (1974) Spatial interaction and the statistical analysis of lattice systems. *Journal of the Royal Statistical Society B* 36, pp192-236.

Lafferty J, McCallum A, Pereira F (2001) Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In: *Proc. 18th International Conf. on Machine Learning*.

Quattoni A, Wang S, Morency L-P, Collins M, Darrell T (2006) Hidden-state conditional random fields. *MIT CSAIL Technical Report.*

Sutton C, McCallum A (2006) An introduction to conditional random fields for relational learning. In: Getoor L & Taskar B (eds.) *Introduction to statistical relational learning.* MIT Press.

Vinson J, DeCaprio D, Pearson M, Luoma S, Galagan J (2007) Comparative Gene Prediction using Conditional Random Fields. In: B Scholkpf, J Platt, T Hoffman (eds.), *Advances in Neural Information Processing Systems* 19, MIT Press, Cambridge, MA.

Acknowledgements

Sayan Mukherjee and Elizabeth Rach provided invaluable comments and suggestions for these slides.