Cite as: Majoros WH (2007) Conditional Random Fields. Online supplement to: Methods for Computational Gene Prediction, Cambridge University Press.

supplement S1

Conditional Random Fields

Of the methods which we have considered so far in this volume, most have been based on purely probabilistic, generative models of sequence composition and syntax—e.g., HMM's, GHMM's, PhyloHMM's, etc. An alternative framework for sequence parsing which has become quite popular in the field of natural language processing, and which is now just beginning to see some use for gene prediction, is the *conditional random field* (CRF). CRF's differ from Markov models in a number of ways:

1. They are innately *discriminative* rather than *generative* (as mentioned in section 12.4).
2. They are based on *undirected* graphical models, rather than directed models such as Bayesian networks (section 10.4) or HMM's.
3. Although they provide a principled means of evaluating probabilities based on conditional independence relations, they are capable of utilizing external evidence and submodels which are not inherently probabilistic.

The third point is especially attractive, considering the various types of extrinsic information which either are or soon will be available for informing the gene prediction process, but which in some cases are not easily represented via probabilities.

In this chapter we describe the theoretical basis for this new modeling framework and summarize the few results reported to date for CRF-based gene prediction.

S1.1 Background: Markov Random Fields

In order to formally define CRF's we will first need to describe *Markov random fields* (MRF's), of which CRF's are a special case. Markov random fields in some ways generalize Markov chains, which we considered in some detail in Chapter 7. Recall that the (1st-order) Markov assumption for Markov chains states that the probability of a nucleotide at position i in a sequence is *conditionally independent* of all previous positions in the sequence, given the identity of the nucleotide at position $i-1$:

$$P(x_i \mid x_0 \ldots x_{i-1}) = P(x_i \mid x_{i-1}) \qquad \text{(S1.1)}$$

That is, the probability of x_i given all preceding nucleotides is the same as the probability of x_i given only the single preceding nucleotide; x_i is thus *conditionally independent* of nucleotides $x_0, \ldots, x_{i-2}$, given x_{i-1}. As we saw in section 10.4, conditional independence relations can be conveniently represented using a Bayesian network, which consists of a directed graph in which an edge $A \rightarrow B$ denotes a dependence of B on A. Figure S1.1 illustrates the Bayesian network equivalent of a hidden Markov model with 0th-order emissions, in which states and their emissions both appear as variables in the network.

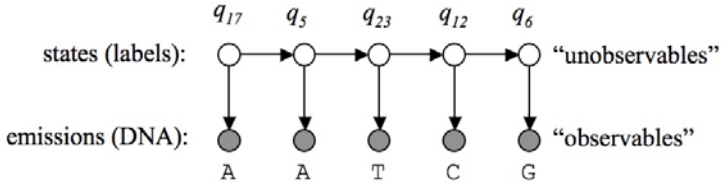

Figure S1.1 *An example Bayesian network representing the operation of an HMM. Shaded nodes represent the observable emission sequence; unshaded nodes represent the unobservable state sequence which the HMM followed during emission of the sequence.*

In the figure we have shaded the *observable* variables (i.e., the DNA sequence) and left unshaded the *unobservable* variables (the states of the model). It should be clear from the figure that the network is not simply a mirror of the state-transition diagram of the HMM, but rather represents the dynamic operation of the HMM over time, via an "unrolling" of the path ϕ through the model's states. We will see when we consider "generalized" or "semi-Markov" CRF's that a somewhat different (but hopefully more familiar) interpretation is available to us for those models—namely, the ORF graph formalism introduced in Chapter 3.

Because an understanding of graphical models is so important to our

exposition of CRF's, we provide another example of a Bayesian network for an HMM—this time a 2nd-order one—in Figure S1.2.

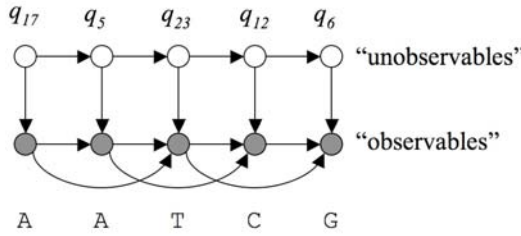

Figure S1.2 *A Bayesian network for the operation of an HMM with 2nd-order emissions.*

In this case each emission is dependent on the two previous emissions, which we have denoted via extra edges along the bottom row of the graph. Recall that we opted in Chapters 6 and 7 to formulate higher-order HMM's via the use of dependencies among emissions rather than among the states themselves. It should be clear that "higher-order" dependencies among states may also be represented in a Bayesian network via a similar proliferation of edges along the state sequence (i.e., the top row of vertices in Figures S1.1 and S1.2).

In the case of Markov chains we say that these higher-order emission dependencies result from a "higher-order" Markov assumption. In particular, for an n^{th}-order Markov chain we have:

$$P(x_i \mid x_0 \ldots x_{i-1}) = P(x_i \mid x_{i-n}, x_{i-n+1}, \ldots, x_{i-1}) \tag{S1.2}$$

so that x_i is dependent only on the previous n bases in the emission sequence. The primary difference between a Markov chain and a Markov random field is that the dependencies in the former follow a linear "chain," whereas in the latter they may assume any conceivable topology. That is, in a Markov random field the dependency structure can be represented by an arbitrary (undirected) graph, in which edges may conceivably connect any two variables represented by vertices in the graph. We will see later that there is an interesting physical interpretation of such a graph, based on interaction potentials among particles in a physical system such as a crystal lattice.

The undirected nature of the graph in an MRF simply reflects the fact that dependence relations between two variables may be inverted via *Bayes' rule* (section 2.6):

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)} \tag{S1.3}$$

where B may of course represent any joint event (e.g., $B=C \wedge D \wedge E \wedge ...$), so that for any given Bayesian network (represented by a directed graph) we may derive an alternate Bayes network by simply reversing the direction of one or more edges in the graph[1]. In this way we can see that undirected graphs provide a more general representation of the dependency structure among a set of variables than do directed graphs, and that the latter in some sense constitute specific instantiations of the former.

Definition S1.1*: A discrete-valued Markov random field (MRF) is a model which can be denoted as a 4-tuple:*

$$\mathcal{M} = (\alpha, X, P_M, G)$$

where:

α *is a finite alphabet,*

X *is a set of (observable or unobservable) variables taking values from* α*,*

$P_{\mathcal{M}}$ *is a probability distribution on the assignment of values from* α *to variables in* X*, and*

$G=(X,E)$ *is an undirected graph on* X *describing a set of dependence relations among variables,*

with the key constraint that:

$$P_{\mathcal{M}}(X_i \,|\{X_{k\neq i}\}) = P_{\mathcal{M}}(X_i \,|\mathcal{N}_G(X_i)), \qquad \text{(S1.4)}$$

for $\mathcal{N}_G(X_i)$ *the neighbors of* X_i *under* G *(i.e.,* $\mathcal{N}_G(X_i) = \{X_{j\neq i} \,|(X_i,X_j)\in E\}$*). That is, all conditional probabilities under* $P_{\mathcal{M}}$ *must obey the dependence relations given by the dependency graph* G*.*

From the above definition it can be seen that the dependency graph of a Markov random field defines a sort of generalized "Markov assumption" and thus generalizes the notion of a Markov chain to include non-linear topologies for the dependencies among the variables of the model. Although we have not specifically formulated MRF's in terms of sequences, it should be clear that sequence-based MRF's may be instantiated by including both

[1] Unfortunately, reversing directionality of edges in a directed model can in some instances induce additional, implicit dependencies due to the phenomenon of *d-separation*—see, e.g., Pearl (1991).

the emission sequence (i.e., the DNA) and the state sequence of a state-based model as variables in the MRF. In such a case we would of course refer to the emissions as *observables* and the states as *unobservables*.

More generally, since we are ultimately interested in the problem of sequence parsing, we may replace the state sequence with a *label* sequence, where different states may share the same label (i.e., `exon`, `intron`, etc.). In such a case the unobservables will consist of a series of labels to be assigned to the (parallel) series of nucleotides in the DNA sequence to be parsed. Taking a parse ϕ to denote this series of labels, we can see that an MRF is useful for parsing inasmuch as it provides a means of evaluating the probability $P(\phi|S)$ of a prospective parse ϕ for a given DNA sequence S. We will describe shortly how CRF's—a specific type of MRF—can be used to evaluate this conditional probability.

S1.2 The Hammersley-Clifford Theorem

Returning for the moment to the more general class of undirected probability models—MRF's—it may not be apparent to the reader that a potentially very significant problem regarding the consistency of the model may arise when actually inducing such a model in practice. In particular, we cannot simply set the conditional probabilities $P_{\mathcal{M}}(X_i|\mathcal{N}_G(X_i))$ arbitrarily and expect the joint probability $P_{\mathcal{M}}(X)$ to be well-defined, since the resulting joint distribution may not be unique (Besag, 1974). Thus, the problem of estimating parameters locally for neighboring variables in the network is confounded by the global constraint that we require the joint distribution to be well-defined. The solution to this problem is given by the *Hammersley-Clifford theorem*, which we reproduce without proof. The energetic reader may find a proof of this result in a number of places in the statistics literature (e.g., Besag, 1974; Clifford, 1990).

Theorem S1.1. *(Hammersley and Clifford, 1971). Let X be a set of variables and suppose P(***x***)>0 for all (joint) value assignments* **x** *to the variables in X. Then the likelihood of* **x** *under model M is given by:*

$$P_{\mathcal{M}}(\mathbf{x}) = \frac{1}{Z} e^{Q(\mathbf{x})}, \tag{S1.5}$$

for normalization factor Z:

$$Z = \sum_{\mathbf{x}'} e^{Q(\mathbf{x}')}, \tag{S1.6}$$

*where Q(***x***) has a unique expansion given by:*

$$Q(x_0, x_1, ..., x_{n-1}) = \sum_{0 \le i < n} x_i \Phi_i(x_i) + \sum_{0 \le i < j < n} x_i x_j \Phi_{i,j}(x_i, x_j) + ... \\ ... + x_0 x_1 ... x_{n-1} \Phi_{0,1,...,n-1}(x_0, x_1, ..., x_{n-1}) \quad \text{(S1.7)}$$

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

The Φ functions in the expansion of $Q(\mathbf{x})$ are called *potential functions*, for reasons that will soon become clear. The notion of a *clique* as referred to by the theorem should be familiar to most (though possibly not all) readers:

***Definition S1.2**: Let $G=(V,E)$ be an undirected graph, and let $C \subseteq V$. Then C is a* clique *iff $\forall_{(a,b) \in C,\, a \neq b}\, (a,b) \in E$. That is, a clique is any subset of vertices all of whom are direct neighbors. As a special case, we also consider any singleton set $\{v\}$ for $v \in V$ to be a clique.*

An example clique is shown in Figure S1.3. Note that, by the above definition of a clique, any subset of vertices drawn from a clique (including singleton sets) is itself a clique, so that any clique which one identifies in a graph *G* may be a subgraph of yet a larger clique. This leads naturally to the notion of a *maximal clique*, which we define next.

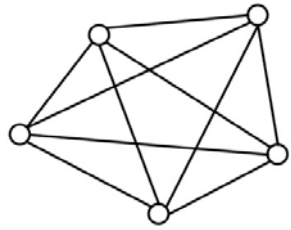

***Figure S1.3** A maximal clique comprising 5 vertices. An additional 30 (non-maximal) cliques of sizes 1 through 4 can be found within this maximal clique.*

***Definition S1.3**: Let $G=(V,E)$ be an undirected graph, and let $C \subseteq V$ be a clique. Then C is a* maximal clique *if $\neg \exists_{D \subseteq V}\, C \subset D$ for clique D of G. That is, a maximal clique C of a graph G is a clique which is not wholely contained within a larger clique of G.*

Several things should be noted about the cliques referenced in the above theorem. First, they need not be maximal cliques. Thus, since every subset of a clique is also a clique (and indeed every vertex *v* by itself forms a singleton clique $\{v\}$), for graphs containing very large cliques the evaluation of *Q* may

become very computationally expensive. Secondly, the cliques over which the Φ functions are evaluated in the expansion of Q given by Equation (S1.7) may overlap. Unlike Bayesian networks, in which care must be taken in decomposing the joint probability into terms for disjoint subsets of variables in the network, no special effort is required in MRF's to avoid the overlapping of cliques in the decomposition of the Q term.

The reason Theorem S1.1 is useful for MRF modeling is that it provides a way to evaluate probabilities (whether joint or conditional) based on the "local" functions Φ_c. Thus, we can train an MRF by learning individual Φ_c functions—one for each clique c—where the Φ_c functions may potentially evaluate to any real value and are therefore not strictly probabilistic. This ability to utilize non-probabilistic Φ_c functions lends significant flexibility to MRF's as a modeling framework.

It will be of some use to us later to keep in mind that while Theorem S1.1 stipulates that only those Φ_c for cliques c may evaluate to nonzero values, it does not state that all such Φ_c must be nonzero; in particular, there may conceivably be cliques for which Φ_c is always zero (or nearly so) and which may therefore be ignored when implementing the model in software. We will return to this possibility later in the chapter.

S1.3 The Boltzmann Analogy

Many readers will no doubt have noticed that the MRF likelihood function given by Equation (S1.5) closely resembles the *Boltzmann distribution* from statistical thermodynamics:

$$P_{boltzmann}(\mathbf{x}) = \frac{1}{Z} e^{-E(\mathbf{x})/kT}, \tag{S1.8}$$

which we saw in section 10.13 in modified form in the context of simulated annealing, and again in section 12.2 in the context of noncoding RNA prediction. The Boltzmann distribution gives the probability of a particular molecular configuration (or "microstate") $\mathbf{x}$ occurring in an ideal gas at temperature T, where $k=1.38\times10^{-23}$ is the *Boltzmann constant*. The normalizing term Z is known as the *partition function* (since it can be used to compute conditional probabilities by restricting attention to a particular partition of the sample space). The exponent E is the *Gibbs free energy* of the configuration.

The MRF probability function may be conceptualized somewhat analogously, in which the variables of the MRF can be likened to particles in a dynamical system and the summed "*potential functions*" Φ_c reflect the "interaction potentials" between neighboring variables:

$$P_{MRF}(\mathbf{x}) = \frac{1}{Z} e^{\sum_{c \in C} \Phi_c(\mathbf{x})} \tag{S1.9}$$

(where C is the set of cliques in the graph). 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—much like the undirected graph representing variable dependencies in an MRF. In this way, we may interpret an MRF's potential functions as measures of the "compatibility," "consistency," or "co-occurrence patterns" of the variable assignments in **x** at the level of local "interactions" within a neighborhood (clique) of the graph.

It should be noted, however, that the utility of MRF's is not based on this loose analogy with particle dynamics in a physical system, but is instead grounded on the formal result due to Hammersley and Clifford for undirected dependency graphs on sets of random variables. The metaphor of "interaction potential" or "compatibility" between neighboring variables may nonetheless prove useful as a conceptual guide when crafting potential functions for an MRF.

We now consider the special class of MRF's useful for evaluating the conditional probability term $P(\phi|S)$ induced by the sequence parsing problem.

S1.4 Conditional Random Fields

A *conditional random field* (*CRF*) is a Markov random field in which the unobservables are globally conditioned on the observables (Lafferty *et al*., 2001). This idea is formalized in following definition.

Definition S1.4*: A* CRF *is model which can be denoted as a 6-tuple:*

$$\mathcal{M}=(L,\alpha,Y,X,\Omega,G)$$

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 <u>undirected</u> graph describing a set of dependence relations E among variables $V=X\cup Y$, where $E\cap(X\times X)=\varnothing$,

such that $(L,Y,e^{\Sigma\Phi(c,\mathbf{x})}/Z,G-X)$ is a Markov random field, for $Z=\Sigma_{\mathbf{y}'}e^{\Sigma\Phi(c,\mathbf{x})}$, where the latter summation is over all $\mathbf{y}'\in L^{|Y|}$.

Thus, a CRF may be described as an MRF plus a set of "external" (observable) variables X, which are not considered variables of the MRF but are globally visible (as fixed constants) to the MRF's potential functions Φ_c. This idea is illustrated schematically in Figure S1.4.

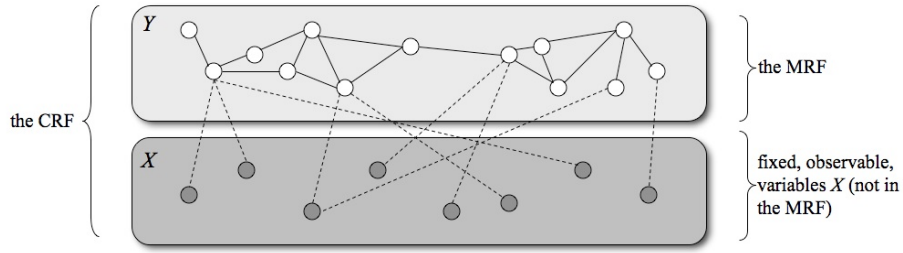

Figure S1.4 *A conditional random field conceptualized as an MRF plus a set of fixed, observable variables X, which are not included in the MRF, but are globally visible to the MRF's potential functions as fixed constants. The variables of the MRF include only the unobservables Y.*

Because the MRF component of a CRF includes only unobservables, when applying the Hammersley-Clifford theorem we will be interested only in the cliques of the unobservable subgraph G_Y of the CRF, rather than all the cliques in the entire CRF graph $G_{X\cup Y}$. In order to avoid confusion, we explicitly refer to these unobservable cliques as *u-cliques*:

Definition S1.5*: Let $\mathcal{M}=(L,\alpha,Y,X,\Omega,G)$ be a conditional random field, and let $G_Y=(Y,E\cap(Y\times Y))$ be the subgraph of G excluding all vertices from X. Then we define the u-cliques of G (and of $\mathcal{M}$) to be the set of all cliques in G_Y.*

In the remainder of this chapter, whenever we refer to the cliques of a CRF we will implicitly mean the *u*-cliques only, unless explicitly stated otherwise.

Note that in defining a CRF we have not specified a joint probability function $P_{\mathcal{M}}(\mathbf{y},\mathbf{x})$, but have instead given "local" clique-specific functions Φ_c which together define a coherent probability distribution via Hammersley-Clifford. The "conditional" nature of a CRF arises by assuming the X are fixed, so that we do not sum over them in the partition function when

computing $P(\mathbf{y}|\mathbf{x})$. That is, since the observables X are *fixed*, the *conditional probability* $P(\mathbf{y}|\mathbf{x})$ of the unobservables given the observables is:

$$P_{\mathcal{M}}(\mathbf{y} \mid \mathbf{x}) = \frac{1}{Z(\mathbf{x})} e^{Q(\mathbf{y},\mathbf{x})} = \frac{1}{\sum_{\mathbf{y}'} e^{Q(\mathbf{y}',\mathbf{x})}} e^{Q(\mathbf{y},\mathbf{x})}, \tag{S1.10}$$

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

$$Q(\mathbf{y},\mathbf{x}) = \sum_{c \in C} \Phi_c(\mathbf{y},\mathbf{x}), \tag{S1.11}$$

Recall that (by Hammersley-Clifford) $\Phi_c(\mathbf{y},\mathbf{x})$ may depend only on those unobservable variables in clique $c \subseteq Y$, which might not include all of the variables represented in vector $\mathbf{y}$. In order to account for this, whenever we write $\Phi_c(\mathbf{y},\mathbf{x})$ it should be agreed that we mean $\Phi_c(\mathbf{y}_c,\mathbf{x})$, where $\mathbf{y}_c$ denotes the vector "slice" containing only elements of $\mathbf{y}$ indexed by the vertices in c. When it is convenient we may also write $\Phi_c(c,\mathbf{x})$ to mean the same thing. Thus, Φ_c is a function specific to clique c in the graph, and $\Phi_c(c,\mathbf{x})$ denotes its application to the variables associated with those clique vertices, which we may equivalently (but with some abuse of notation) write $\Phi_c(\mathbf{y},\mathbf{x})$. The presence of the observable vector $\mathbf{x}$ in these expressions is due specifically to their inclusion in the Φ_c functions in the definition of a CRF, and should not be misconstrued as implying that any of the X are members of the u-cliques (since by definition they are not). The reason we are permitted to include the $\mathbf{x}$ as arguments to $\Phi_c(\mathbf{y},\mathbf{x})$ is that in a CRF the X are deemed constants, and may just as well have been embedded directly in the definitions of the Φ_c functions. Thus, the explicit listing of $\mathbf{x}$ in the arguments to Φ_c is merely a reminder that all the $\mathbf{x}$ are available to us when we craft our Φ_c functions.

Since the observables are embedded within the CRF, we may consider that a CRF comes into existence only after an input sequence $\mathbf{x}$ is read. This is somewhat in contrast to the notion of a generative model such an HMM, in which the model exists prior to the generation of the emitted sequence $\mathbf{x}$. In practice one generally specifies a CRF *template* to be used in constructing a new CRF for a given input sequence $\mathbf{x}$ (Sutton and McCallum, 2006), as illustrated in Figure S1.5. We will elaborate on the notion of CRF templates (and in particular, their use in parameter tying) in section S1.11.

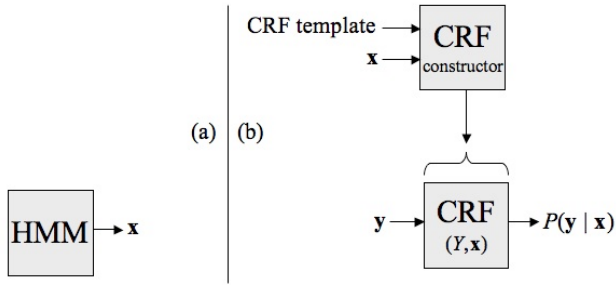

Figure S1.5 *An HMM (a) generates observables* **x**, *whereas a CRF (b) generates probabilities of putative labels* **y**. *Because the observables are considered part of the CRF, an individual CRF (bottom) may be constructed from a CRF template and a particular input sequence* **x** *(top).*

A central notion in the definition of CRF's is that dependencies among the observables X are not explicitly modeled—i.e., $E \cap (X \times X) = \varnothing$, for edges E of the dependency graph. This should not be construed to imply that the observables are considered *independent*. Rather, in CRF modeling we seek to directly model (to the extent possible) the posterior probability distribution $P(Y \mid X)$ without including any information regarding the direct interactions between individual variables in X. We will see later that indirect relations between the X may still be modeled via special unobservables called *latent variables*, or more directly through the use of *hybrid* models in which an external function is utilized for measuring interactions (or more specifically, non-independence) between observables. Thus, while the prevailing philosophy of CRF modeling favors the omission of explicit relations between the X in formulating a model, the framework is flexible enough to permit, indirectly, the modeling of dependences between the observables when necessary.

S1.5 Simplifying Assumptions

Now that we have formally defined CRF's in their most general form, it is necessary to distinguish from these the more constrained class of models which arise due to the simplifying assumptions which are almost universally encountered in practical CRF applications.

The first set of assumptions pertain to the form of the potential functions which are adopted in practice. Recalling the form of the Hammersley-Clifford theorem given above due to Besag (1974), it will be seen that practical implementations universally ignore the $x_i x_j \ldots x_k$ coefficients in the $x_i x_j \ldots x_k \Phi_{i,j,\ldots,k}(x_i x_j \ldots x_k)$ terms. These can in principle be absorbed by the potential functions—i.e.,

$$\Phi_{i,j,\ldots,k}(x_i x_j \ldots x_k) = x_i x_j \ldots x_k f_{i,j,\ldots,k}(x_i x_j \ldots x_k), \quad \text{(S1.12)}$$

so that the expansion of Q simplifies to:

$$Q(\mathbf{x}) = \sum_{0 \le i < n} \Phi_i(x_i) + \sum_{0 \le i < j < n} \Phi_{i,j}(x_i, x_j) + \ldots + \Phi_{0,1,\ldots,n-1}(x_0, x_1, \ldots, x_{n-1}) \quad \text{(S1.13)}$$

In practice the potential functions Φ_c are very often decomposed into a weighted sum of "*feature sensors*" f_k, producing:

$$P(\mathbf{y} \mid \mathbf{x}) = \frac{1}{Z} e^{\sum_{c \in C} \sum_{i \in F} \lambda_i f_i(c, \mathbf{x})}, \quad \text{(S1.14)}$$

where F is a family of feature sensors, which are specific to individual cliques. We will see that the notion of "rich features" evaluated by these feature sensors affords much flexibility to CRF's as a modeling framework.

The assumption of a "feature sensor" decomposition of the potential functions results in yet another common assumption in practical CRF applications—namely, that the feature sensors f_i may be individually trained outside the context of the CRF, and then simply "plugged into" the composite model as-is. That is, while the λ_i's are typically trained jointly for the CRF as a whole, estimation of parameters used internally by any feature sensor f_i is typically performed as an independent process, within the context of f_i alone. The later (joint) estimation of the λ_i's then leaves these other feature sensor parameters fixed. We will return to the issue of CRF training later in this chapter.

We emphasize that the simplifications described in this section are arbitrary modeling decisions and are not mandated by the formal CRF framework as we have defined it. Some of these assumptions do have a historical precedent, however, in that they have been successfully employed in the field of *natural language processing* (NLP) and shown to suffice for various prediction tasks in that field. Whether these same modeling decisions are ideal in the case of gene prediction is a matter deserving some investigation.

In terms of choosing the optimal potential functions for the task of *ab initio* gene prediction, aside from the Boltzmann analogy (i.e., "compatibility" of variable assignments) there is little concrete advice that can be given at this time; these issues therefore await further developments in the field.

S1.6 CRF's for *ab initio* Gene Finding

For gene finding, the "unobservables" in a CRF would obviously be the labels (`exon`, `intron`, etc.) to be attached to each nucleotide in the DNA sequence. The assignment of these labels constitutes a parse of the sequence. An example CRF for *ab initio* gene finding is shown in Figure S1.6.

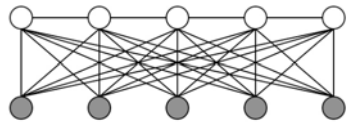

Figure S1.6 *A simple CRF for sequence parsing. Shaded vertices represent individual nucleotides in the input sequence (the observables); unshaded vertices represent labels such as* `exon` *or* `intron` *to be attached to individual nucleotides, and are thus unobservables.*

In this example CRF we allow that any unobservable (unshaded vertex) may depend on any number of observables (shaded vertices), as well as either (or both) of the unobservables for the immediately adjacent DNA positions. It can be seen from the figure that long-range dependencies between a label and any nucleotide in the sequence may thus be captured by this model; by comparison, long-range dependencies are generally not captured by HMM's. Additional dependencies between the labels themselves may also be included, though, as of this writing, current CRF-based gene finders do not yet model these types of longer-range dependencies.

Returning to the example CRF of Figure S1.6, it should be readily apparent that the *u*-cliques of this graph are of two types: those containing a single unobservable (*singleton-cliques*) and those containing a pair of adjacent unobservables (*pair-cliques*); these are illustrated in Figure S1.7. In the latter figure we have taken the liberty of including the observables in the illustrated cliques, simply as a reminder that they are globally available to the Φ_c functions. Since any given Φ_c might in practice utilize only a subset of the full observation sequence $\mathbf{x}$, we may consider that the Φ_c functions naturally induce a set of "virtual" cliques containing both unobservables and (some subset of the) observables. In this way, our decision to formally define the MRF portion of a CRF on the unobservable subgraph G_Y may be seen as somewhat arbitrary, and indeed, alternative definitions of CRF's may be found in the literature in which the induced MRF is defined over the entire graph $G_{X \cup Y}$ (e.g., McCallum *et al.*, 2003). We reiterate that the ability of the CRF formalism in accommodating various alternate modeling scenarios such as these renders them both flexible and practical.

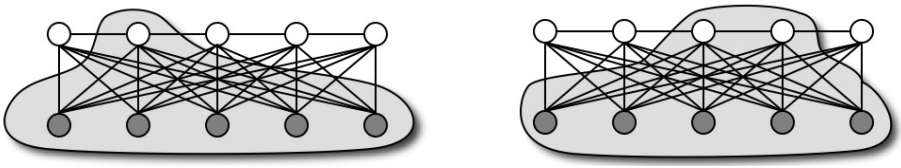

Figure S1.7 *Two types of u-cliques in a CRF for gene finding: singleton-cliques (left figure) contain only one unobservable, while pair-cliques (right figure) contain two unobservables. Observables may also be considered informally as part of these cliques, since they are globally available to the* Φ_c *functions.*

For our example CRF we can see that we require only two Φ_c functions—$\Phi_{singleton}$ for "singleton cliques" (left portion of Figure S1.7) and Φ_{pair} for "pair cliques" (right portion of same figure). These may be thought of informally as analogues of the emission and transition functions in an HMM, and indeed, several authors have pointed out that an HMM may be interpreted as a rudimentary CRF in which the Φ functions are implemented directly using the analogous HMM functions—i.e., $\Phi_{singleton}=P_e$ and $\Phi_{pair}=P_t$ (e.g., Sutton and McCallum 2006; Vinson *et al.*, 2007).

S1.7 CRF Decoding for Gene Finding

In section 6.2.1 we defined the decoding problem for HMM's as that of finding the most probable parse ϕ (the *maximum a posteriori*, or *MAP*, parse) given the DNA sequence *S*:

$$\begin{aligned} \arg\max_{\phi} P(\phi \mid S) &= \arg\max_{\phi} P(\phi)P(S \mid \phi) \\ &= \arg\max_{\phi} \sum_{y_i \in \phi} \log\left(P_{trans}(y_i \mid y_{i-1})P_{emit}(s_i \mid y_i)\right) \end{aligned} \tag{S1.15}$$

Since our goal in CRF-based gene finding is still to find the most probable parse of the input sequence, the decoding problem for CRF's may be defined similarly:

$$\arg\max_{\mathbf{y}} P(\mathbf{y}|\mathbf{x}) = \arg\max_{\mathbf{y}} \frac{1}{Z} e^{\sum_{c \in C} \Phi_c(\mathbf{y},\mathbf{x})} = \arg\max_{\mathbf{y}} \sum_{c \in C} \Phi_c(\mathbf{y},\mathbf{x}) \tag{S1.16}$$

where the parse (or "labeling") **y** encapsulates all the unobservables and the DNA sequence **x** comprises all of the observables. In derivation (S1.16) we have made use of the fact that the "partition function" *Z* is constant over the

argmax and can therefore be ignored during standard decoding. Under the common assumption that the potential functions decompose into a linear combination of feature sensors, we arrive at the following:

$$\mathbf{y}^* = \arg\max_{\mathbf{y}} \sum_{c,i} \lambda_i f_i(c, \mathbf{x}, \mathbf{y}) \qquad \text{(S1.17)}$$

where the summation is over u-cliques c and sensor types i.

Comparison of Equations (S1.15) and (S1.17) reveals that the decoding problem for both CRF's and HMM's may be formulated as a sum of local feature sensors, which in the case of the example CRF given in the previous section corresponds to successive singleton and pair cliques along the length of the sequence. It takes no stretch of the imagination to see that a dynamic programming solution for decoding of linearly-structured CRF's for *ab initio* gene finding may be derived by simply modifying the scoring scheme of the Viterbi algorithm for HMM decoding—in particular, we merely need to replace the transition probability function P_t with Φ_{pair} and the emission function P_e with $\Phi_{singleton}$. We will see in the next section that a generalized CRF for gene finding with duration modeling may similarly utilize existing frameworks for GHMM decoding. The computational complexity of these CRF decoders will be similar to that of their HMM analogues when long-range dependencies are not used in the model, or when they can be bounded by a constant distance. Thus, as the gene-finding field moves increasingly toward the use of (linear-structured) CRF's, knowledge of (and software for) efficient decoding strategies for Markovian models retains significant utility.

It is up to the modeler to determine the actual form of the feature sensor functions f_i. As we alluded to earlier, an obvious choice is to use any of the existing (probabilistic) sensors available for (G)HMM-based gene finders—i.e., any of the models described in Chapter 7. The use of non-probabilistic sensors is feasible also, including such possibilities as raw n-mer counts or any function computed from external evidence such as BLAST hits; we will see some examples of the latter in section S1.12.

We now give a detailed algorithm for *maximum a posteriori* (MAP) decoding of a linearly-structured CRF like the one shown in Figure S1.6. We will consider the related problem of posterior decoding shortly. First, we introduce some definitions. To simplify the specification of syntax rules among labels, we utilize an explicit *syntax model* encoded via the function $\pi: L \rightarrow 2^L$ mapping each label $v \in L$ to a set of labels which may immediately precede v in a legal parse. Without an explicit syntax model, syntax rules would need to be enforced by setting forbidden combinations of adjacent labelings to zero in Φ_{pair}; utilizing π allows a more efficient enumeration over possible predecessors, just as did the the λ_{trans} array in section 6.1.2 for HMM's.

For MAP decoding we need a dynamic programming matrix M of size $N \times |L|$ (for N the number of unobservables in vector $\mathbf{y}$) which we will index as $M(c,r)$ for column c and row r. Column c will correspond to unobservable u_c, and row r will correspond to the assignment of label ℓ_r to that unobservable. A separate matrix R, indexed in the same manner as M, will store the *traceback pointers* (similarly to Viterbi decoding of HMM's—section 6.2) which will be used to reconstruct the optimal parse after both matrices have been fully computed (note that by "pointer" we mean an integer index specifying a cell in the previous column of the matrix).

Algorithm S1.1:

Initialization:

$$\forall_{v \in L} \; M(0,v) = \Phi_{sng}(0,v,\mathbf{x})$$

Recurrence:

$$\forall_{\substack{v \in L \\ 1 \le k < N}} \; M(k,v) = \max_{w \in \pi(v)} \left(M(k-1,w) + \Phi_{pair}(k-1,w,v,\mathbf{x}) \right) + \Phi_{sng}(k,v,\mathbf{x})$$

$$\forall_{\substack{v \in L \\ 1 \le k < N}} \; R(k,v) = \arg\max_{w \in \pi(v)} \left(M(k-1,w) + \Phi_{pair}(k-1,w,v,\mathbf{x}) \right)$$

Termination:

$$\mathbf{y}^*_{N-1} = \arg\max_{w \in L} M(N-1,w)$$

$$\forall_{0 \le i < N-1} \; \mathbf{y}^*_i = R(i+1, \mathbf{y}^*_{i+1})$$

$$\mathbf{y}^* = (\mathbf{y}^*_0, \ldots, \mathbf{y}^*_{N-1})$$

This algorithm has the same time and space complexity as the Viterbi algorithm for HMM's, and indeed it can be seen to be virtually identical to the log-space version of that latter algorithm. Note that syntactic constraints on the permissible labelings of the very first and very last unobservables in the linear-structured CRF (which we enforced in HMM's by introducing a special start/stop state, q^0) can be easily encoded within Φ_{sng}.

The above algorithm can be used only for linearly-structured CRF's, such as the one shown in Figure S1.6. For CRF's with cycles, the above procedure is obviously inapplicable without first imposing an ordering on the unobservables, and even then the algorithm will exhibit greedy behavior which for many Φ functions will not be optimal. Generalizing this procedure so as to avoid greedy behavior would require delaying assignments to the link-back matrix for key vertices in a cycle until the entire cycle has been evaluated; because the resulting algorithm would incur a space complexity that grows with the size of the largest cycle, this places practical limits on the degree of long-range dependences which can be efficiently modeled between the unobservables (though not between an unobservable and an observable).

The problem of *posterior decoding* (which was introduced in section 6.10 for HMM's) is computationally more difficult that that of MAP decoding. Whereas MAP decoding finds the most probable vector of labels **y** to assign to the unobservables **Y**, posterior decoding instead provides a way of evaluating the probability of any *given* labeling **y**—i.e., $P(\mathbf{Y}=\mathbf{y}|\mathbf{X}=\mathbf{x})$. This is especially useful when **y** is *degenerate*—i.e., when the elements of **y** are permitted to be "don't care" symbols (which we will denote "*") instead of valid labels; the don't-care symbol prescribes a summation over the possible values for a given unobservable. In the case of gene-finding, this permits us to compute, for example, the probability that a particular interval of DNA contains a coding exon, allowing for the rest of the sequence to be labeled in any arbitrary manner.

Although we were able to ignore the partition function, $Z(\mathbf{x})$, in MAP decoding, for posterior decoding it must be explicitly computed, so that the resulting probabilities will be properly normalized. Consider the form of $Z(\mathbf{x})$:

$$Z(\mathbf{x}) = \sum_{\mathbf{y}\in L^N} e^{\sum_{c\in C}\Phi_c(\mathbf{y},\mathbf{x})} = \sum_{\mathbf{y}\in L^N}\prod_{c\in C} e^{\Phi_c(\mathbf{y},\mathbf{x})} \tag{S1.18}$$

Observe that, since $Z(\mathbf{x})$ is a sum of products, additional e^{Φ} terms multiplied by $Z(\mathbf{x})$ will distribute over all the terms in the sum. For a linearly-structured CRF, we can use this fact to derive an efficient recursion for $Z(\mathbf{x})$:

Algorithm S1.2:

Initialization:

$$\forall_{v\in L}\ Z(0,v) = e^{\Phi_{sng}(0,v,\mathbf{x})}$$

Recurrence:

$$\forall_{\substack{w\in L\\ 1\le k<N}}\ Z(k,w) = \sum_{v\in\pi(w)} Z(k-1,v)e^{\Phi_{sng}(k,w,\mathbf{x})+\Phi_{pair}(k-1,v,w,\mathbf{x})}$$

Termination:

$$Z(\mathbf{x}) = \sum_{v\in L} Z(N-1,v)$$

Since the recursive step utilizes only entries in the previous column of the dynamic programming matrix, this algorithm can be implemented using only two columns; thus, the space requirements are only $\mathbb{O}(|L|)$, though the time complexity is $\mathbb{O}(N|L|^2)$.

What remains is to compute the numerator of Eq. (S1.10)—the $e^{\Sigma\Phi(\mathbf{y},\mathbf{x})}$ term. In order to accommodate degeneracy in **y**, we first define a variable ψ as follows:

$$\psi(k) = \begin{cases} L & \text{if } \mathbf{y}_k = * \\ \{\mathbf{y}_k\} & \text{otherwise} \end{cases} \tag{S1.19}$$

If **y** is indeed degenerate, then the numerator of Eq. (S1.10) must be modified so as to sum over all labelings $\mathbf{y}'$ that are "consistent" with **y**:

$$P(\mathbf{y}|\mathbf{x}) = \frac{\sum_{\mathbf{y}' \lhd \mathbf{y}} e^{\sum_{c \in C} \Phi_c(\mathbf{y}'_c, \mathbf{x})}}{Z(\mathbf{x})}, \tag{S1.20}$$

where $\mathbf{y}' \lhd \mathbf{y}$ denotes the needed consistency relation: $\forall_{0 \le i < N} \mathbf{y}'_i \in \psi(i)$. We can compute the above numerator using the following dynamic programming recursion:

Algorithm S1.3:

Initialization:

$$\forall_{v \in \psi(0)} S(0, v) = e^{\Phi_{sng}(0, v, \mathbf{x})}$$

Recurrence:

$$\forall_{\substack{w \in \psi(k) \\ 1 \le k < N}} S(k, w) = \sum_{v \in \pi(w) \cap \psi(k-1)} S(k-1, v) e^{\Phi_{sng}(k, w, \mathbf{x}) + \Phi_{pair}(k-1, v, w, \mathbf{x})}$$

Termination:

$$S(\mathbf{y}, \mathbf{x}) = \sum_{v \in \psi(N-1)} S(N-1, v)$$

Given the above recursions, we can compute $P(\mathbf{y}|\mathbf{x})$ very simply:

$$P(\mathbf{y}|\mathbf{x}) = \frac{S(\mathbf{y}, \mathbf{x})}{Z(\mathbf{x})} \tag{S1.21}$$

The term $P(\mathbf{y}|\mathbf{x})$ will prove useful during training (section S1.10)—i.e., by maximizing $P(\mathbf{y}|\mathbf{x}, \theta)$ over all possible parameterizations θ (for each **y** and **x** fixed as specified in the training set), we will be training the model according to the objective of *conditional maximum likelihood*. For any labeling **y** in the training set involving degeneracy (i.e., "don't care" elements), the term $P(\mathbf{y}|\mathbf{x})$ will account for that degeneracy automatically, given the algorithms above.

For any degenerate labeling **y**, the above procedure will compute the probability of **y** in time $\mathbb{O}(N|L|^2)$. For large numbers of queries, this can be too slow. For example, computing the posterior probability of a particular label residing at a particular sequence index, for all possible indices and all

possible labels, would incur a time complexity of $\mathbb{O}(N^2|L|^3)$. A more efficient alternative in that case would be to pre-compute a pair of variables representing prefix sums and suffix sums of the partition function, as follows:

Algorithm S1.4:

Initialization:

$$\forall_{v \in L} \; Z_{prefix}(0, v) = e^{\Phi_{sng}(0, v, \mathbf{x})}$$

$$\forall_{v \in L} \; Z_{suffix}(N-1, v) = 1$$

Recurrence:

$$\forall_{\substack{w \in L \\ 1 \le k < N}} \; Z_{prefix}(k, w) = \sum_{v \in \pi(w)} Z_{prefix}(k-1, v) e^{\Phi_{sng}(k, w, \mathbf{x}) + \Phi_{pair}(k-1, v, w, \mathbf{x})}$$

$$\forall_{\substack{v \in L \\ 0 \le k < N-1}} \; Z_{suffix}(k, v) = \sum_{\substack{w \ni \\ v \in \pi(w)}} Z_{suffix}(k+1, w) e^{\Phi_{sng}(k+1, w, \mathbf{x}) + \Phi_{pair}(k, v, w, \mathbf{x})}$$

Given these two matrices, and the value of $Z(\mathbf{x})$, we can compute $P(\mathbf{Y}_k = \mathbf{y}_k | \mathbf{x})$ in constant time via:

$$P(\mathbf{y}_k | \mathbf{x}) = \frac{Z_{prefix}(k, \mathbf{y}_k) Z_{suffix}(k, \mathbf{y}_k)}{Z(\mathbf{x})} \tag{S1.22}$$

Posterior probabilities for all label×position pairs will now cost only $\mathbb{O}(N|L|^2 + N|L|) = \mathbb{O}(N|L|^2)$ time, though additional space is obviously required for storing the full matrices (recall that $Z(\mathbf{x})$ required only two columns of the dynamic programming matrix). Solutions to more complex queries can be constructed via appropriate modifications to Eq. (S1.22)—for example:

$$P(\mathbf{y}_k \ldots \mathbf{y}_{k+n} | \mathbf{x}) = \frac{Z_{prefix}(k, \mathbf{y}_k) e^{\sum_{i=k}^{k+n-1} \Phi_{pair}(i, \mathbf{y}_i, \mathbf{y}_{i+1}, \mathbf{x}) + \Phi_{sng}(i+1, \mathbf{y}_{i+1}, \mathbf{x})} Z_{suffix}(k+n, \mathbf{y}_{k+n})}{Z(\mathbf{x})} \tag{S1.23}$$

Note that Z_{prefix} and Z_{suffix} serve a similar role to the *forward* and *backward* variables defined in chapter 6, though in this case additional normalization (via $Z(\mathbf{x})$) is required to obtain true probabilities.

As with HMMs, the CRF decoding algorithms described here are potentially subject to numerical overflow or underflow, depending on the particular Φ functions employed. The MAP decoding procedure (Algorithm S1.1) is the least likely to cause difficulty for reasonable Φ functions, since it involves simple sums. The Z and S variables (including Z_{prefix} and Z_{suffix}),

however, utilize multiplication and exponentiation, which could potentially lead to either overflow or underflow, depending on the magnitudes (and signs) of the values returned by the Φ functions. Applying the *log* transformation to these recursions, via the *Kingsbury-Raynor* relation given in chapter 6 (Eq. 6.43), reduces these algorithms to simple sums as well. For Φ functions which often return large values, such sums may still result in overflow for long sequences; some care in crafting the Φ functions is thus warranted.

S1.8 Generalized CRF's

CRF's can be very easily generalized to model feature lengths in DNA by utilizing an *ORF graph* (section 3.4) as utilized by a standard GHMM decoder. We refer to such a CRF as a *generalized CRF* (*GCRF*). In a GCRF the set of observables X consists of all putative signals (`ATG`, `GT`, etc.) discovered during construction of the ORF graph. From the ORF graph we can obtain the corresponding dependency graph of the CRF by replacing each signal X_i with a binary variable Y_i indicating whether the signal X_i is (Y_i=1) or is not (Y_i=0) present in a putative parse ϕ. Thus, we distinguish between a labeling **y** and its corresponding parse ϕ, since the former is a vector of binary digits and the latter is a series of signals or features making up a valid gene parse. Figure S1.8 shows an example dependency graph for a GCRF.

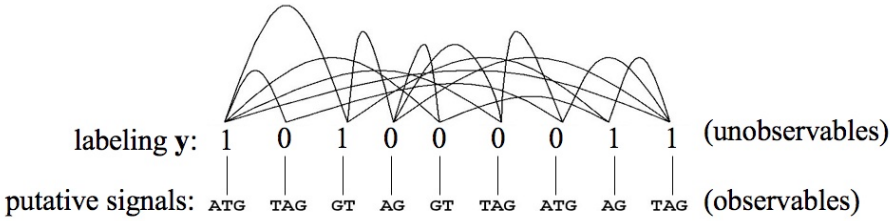

Figure S1.8 *Dependency graph for a generalized CRF (GCRF). Unobservables are binary indicators on the presence of putative signals (observables) in a putative parse. Edges in the graph represent syntax constraints, and are constructed in the usual way for an ORF graph.*

Note that in our formulation of a GCRF we assume that the original DNA sequence has (in principle) been thoroughly preprocessed to produce both the ORF graph and a set of associated scores for individual features in the graph—i.e., for each putative start/stop codon, splice site, exon, intron, etc. Thus, the GCRF construction procedure would utilize a *weighted* ORF graph, and therefore need not embed the actual DNA sequence into the CRF. Since the weights on the ORF graph are considered observables, they are available

as implicit arguments to the Φ functions of the CRF. Note that the entire ORF graph need not be explicitly represented in memory; as we will see, an appropriately modified DSP procedure from section 8.3.2 may be employed to build just the portion of the trellis needed for optimal traceback.

The cliques of the ORF graph are again *singletons* (individual signals) and *pairs of signals* (i.e., an intron bounded by `GT-AG`; an exon bounded by `ATG-GT`; etc.). The Φ_{pair} potential function may 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:

$$P(\phi|S) = \frac{1}{Z} e^{\sum_{(s,t)\in\phi} \sum_{i\in F} \gamma_i f_{emit}(s,t,i) + \mu_i f_{trans}(s,t,i) + \beta_i f_{length}(s,t,i)}, \qquad \text{(S1.27)}$$

where $(s,t)\in\phi$ are pairs of signals in a parse ϕ. A separate Φ_{sng} function for singleton cliques may be incorporated, if desired, to reflect signal scores, or the latter may be directly incorporated into Φ_{pair}. For MAP decoding we may again ignore the partition function, resulting in an argmax over a sum:

$$\arg\max_{\phi} \sum_{(s,t)\in\phi} \sum_{i\in F} \gamma_i f_{emit}(s,t,i) + \mu_i f_{trans}(s,t,i) + \beta_i f_{length}(s,t,i), \qquad \text{(S1.28)}$$

which may be evaluated efficiently using any of the existing GHMM decoders (i.e., *DSP*—section 8.3.2; *PSA*—section 8.3.1), but with the GHMM scoring system replaced by the corresponding GCRF one.

The alert reader will no doubt have realized, based on our description above, that it is possible to construct a labeling **y** which is not syntactically valid, in that the signals may not form a consistent path across the entire ORF graph. This can be rectified 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', as illustrated in Figure S1.9.

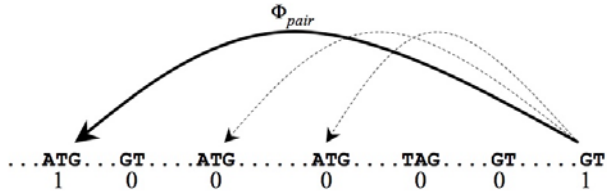

Figure S1.9 *Ensuring syntactic consistency of a labeling* **y**. *For any two vertices adjacent in the ORF graph and having label 1, all vertices between them must have label 0.*

We may thus consider that all nodes in the dependency graph (*not* the ORF graph) are directly connected via an edge, resulting in a very densely connected graph and therefore an enormous number of cliques—with some of those cliques being extremely large. However, the vast majority of those cliques do not require any explicit computation, since most serve only to enforce syntax constraints, and those constraints can in most cases be enforced via the smaller cliques embedded within them. We thus consider cliques according to the following hierarchy:

- *for cliques of size 1* : $\Phi_{sng}(\mathbf{y}_i)$ evaluates to the "signal score" for signal i if $\mathbf{y}_i=1$; otherwise $\Phi_{sng}(\mathbf{y}_i)$ evaluates to zero

- *for cliques of size 2* : if signal i may precede signal j in a syntactically valid parse, then $\Phi_{pair}(\mathbf{y}_i,\mathbf{y}_j)$ evaluates to some combination of "duration" and "emission" scores for the genomic feature spanning signals i and j if $\mathbf{y}_i=\mathbf{y}_j=1$; otherwise $\Phi_{pair}(\mathbf{y}_i,\mathbf{y}_j)$ evaluates to zero

- *for cliques of size* $n>2$: $\Phi_n(\mathbf{y}_i,\ldots,\mathbf{y}_{i+n-1})$ evaluates to $-\infty$ if any syntactic constraints are violated among the $\mathbf{y}_i,\ldots,\mathbf{y}_{i+n-1}$; otherwise $\Phi_n(\mathbf{y}_i,\ldots,\mathbf{y}_{i+n-1})$ evaluates to zero

Given these rules, cliques of size $n>2$ need not be explicitly enumerated or evaluated by the decoder, since all syntactic constraints can be enforced by Φ_{pair} (we assume that at least one syntactically valid default labeling is available—e.g., "all intergenic"). The Φ_n function described above is reflected only in the behavior of the traceback mechanism of the decoder, and thus serves only as a theoretical construct justifying the traceback algorithm's practice of ignoring all predecessors other than those explicitly defined by the ORF graph.

Finally, in order to enforce phase constraints we must utilize three copies of the ORF graph, with links between the three graphs enforcing phase constraints based on lengths of putative features. Modifications to Φ_{pair} to observe these constraints are conceptually simple but tedious to write down; we leave them as an exercise for the enthusiastic reader.

S1.9 CRF's for Comparative Gene Finding

In section 9.6 we described the use of PhyloHMM's for informing the gene prediction process via homologous sequences from related organisms. Such an informant-based approach is possible within the CRF framework as well; we refer to these as *PhyloCRF's*. An example is illustrated in Figure S1.10.

In this figure, the white vertices in the "informant" trees denote ancestral

genomes, which are neither observed nor explicitly sought—they are used merely to control for the non-independence of the informants, just as in PhyloHMM's. We assume the informants are presented to the system as a precomputed multi-alignment between the informants and the target sequence. Note that while the ancestral genome vertices are being used to denote dependencies between observables, since no two observables are connected directly by an edge in the dependency graph the resulting model still fits our formal definition of a CRF.

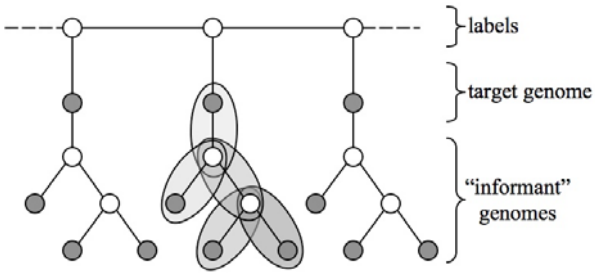

Figure S1.10 *Dependency graph for a (0^{th}-order) PhyloCRF. White nodes in the informant trees denote ancestral genomes, which are unobservable. Cliques in the trees consist of singletons (not shown) and pairs of species, which correspond to branches in the phylogeny.*

Nevertheless, the ancestral genomes introduce an added complexity not yet addressed by our CRF framework, since they are unobservables for which we do not desire a prediction. The latter class of CRF variables we call *latent variables*, and we denote this set L, so that our model now consists of three disjoint sets of variables: X (observables), Y (labels), and L (latent variables). Decoding in the presence of latent variables requires an extra summation in order to marginalize over all possible assignments of values to the latent variables:

$$P_{\mathcal{M}}(\mathbf{y} \mid \mathbf{x}) = \sum_{\mathbf{h}} P_{\mathcal{M}}(\mathbf{y}, \mathbf{h} \mid \mathbf{x}) = \sum_{\mathbf{h}} \frac{1}{Z(\mathbf{x})} e^{\mathcal{Q}(\mathbf{y},\mathbf{h},\mathbf{x})} = \frac{\sum_{\mathbf{h}} e^{\mathcal{Q}(\mathbf{y},\mathbf{h},\mathbf{x})}}{\sum_{\mathbf{y}',\mathbf{h}'} e^{\mathcal{Q}(\mathbf{y}',\mathbf{h}',\mathbf{x})}} \qquad \text{(S1.29)}$$

(Quattoni *et al*., 2006), where **h** is the vector of latent variable assignments, and **x** and **y** once again denote the input DNA sequence and its labeling, respectively (we assume for now that our PhyloCRF's are *non-generalized*, so that the labels would denote actual gene features rather than binary indicators as in a GCRF). For standard Viterbi-like decoding we can again ignore the denominator:

$$\arg\max_{\mathbf{y}} P_{\mathcal{M}}(\mathbf{y} \mid \mathbf{x}) = \arg\max_{\mathbf{y}} \sum_{\mathbf{h}} e^{Q(\mathbf{y},\mathbf{h},\mathbf{x})} \tag{S1.30}$$

Unfortunately, performing this sum over the latent variables outside of the potential function Q can be very time-consuming, compared to the dynamic programming approach of Felsenstein's "pruning" algorithm (section 9.6.1) for probabilistic phylogenetic models. However, evaluating Q on the u-cliques C as usual and shuffling terms gives us:

$$\begin{aligned} \sum_{\mathbf{h}} e^{\sum_{c \in C} \Phi(c,\mathbf{h},\mathbf{x})} &= \sum_{\mathbf{h}} e^{\sum_{c \in C} \log(e^{\Phi(c,\mathbf{h},\mathbf{x})})} \\ &= \sum_{\mathbf{h}} e^{\log \prod_{c \in C} e^{\Phi(c,\mathbf{h},\mathbf{x})}} = \sum_{\mathbf{h}} \prod_{c \in C} e^{\Phi(c,\mathbf{h},\mathbf{x})} \end{aligned} \tag{S1.31}$$

In the case of our tree-structured PhyloCRF, the final term in this formula admits a factorization which exactly mirrors the recursion in Felsenstein's algorithm (leaving out the singleton latent cliques, since they do not affect decoding). We will illustrate this with an example.

In Figure S1.11 we show a single copy of the phylogeny-like tree structure from an example PhyloCRF having four informants (d, e, f, and g). The remaining vertices are the target species (a) and the latent variables for the ancestral species (b, c).

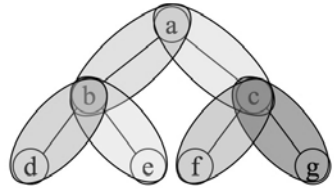

Figure S1.11 *Clique decomposition for a phylogeny. Singleton cliques are omitted for clarity.*

To simplify notation, let $\xi(\cdot)$ denote $e^{\Phi(\cdot)}$. Then the $\sum\prod e^{\Phi(c,\mathbf{y},\mathbf{x})}$ term from Equation (S1.22) expands along the cliques of the tree (i.e., the vertices and edges) as follows:

$$\begin{aligned} \sum_{a}\sum_{b}\sum_{c}\sum_{d}\sum_{e}\sum_{f}\sum_{g} & \xi(a,b)\xi(a,c)\xi(b,d)\xi(b,e)\xi(c,f) \\ & \cdot \xi(c,g)\xi(a)\xi(b)\xi(c)\xi(d)\xi(e)\xi(f)\xi(g) \end{aligned} \tag{S1.32}$$

Note that $\xi(b)$ and $\xi(c)$ may effectively be dropped, since we have no prior

expectations on the latent variables; *b* and *c* in this example are useful only in controlling for non-independence of their observable neighbors, and this function is achieved via the pair cliques.

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

$$\sum_a \xi(a)\left[\sum_b \xi(a,b)\left(\sum_d \xi(b,d)\xi(d)\right)\left(\sum_e \xi(b,e)\xi(e)\right)\right] \cdot\left[\sum_c \xi(a,c)\left(\sum_f \xi(c,f)\xi(f)\right)\left(\sum_g \xi(c,g)\xi(g)\right)\right] \tag{S1.33}$$

Now let us compare the CRF formulation (S1.24) to the Bayesian network formulation under Felsenstein's recursion (S1.25, below), where $P_{a\to b}$ is the lineage-specific substitution probability, $\delta(d,x_d)$ is *Kronecker's delta function* (section 2.1), and $P_{HMM}(a)$ is the probability of *a* under a standard HMM (i.e., the "HMM" part of a "PhyloHMM").

$$\sum_a P_{HMM}(a)\left[\sum_b P_{a\to b}\left(\sum_d P_{b\to d}\delta(d,x_d)\right)\left(\sum_e P_{b\to e}\delta(e,x_e)\right)\right] \cdot\left[\sum_c P_{a\to c}\left(\sum_f P_{c\to f}\delta(f,x_f)\right)\left(\sum_g P_{c\to g}\delta(g,x_g)\right)\right] \tag{S1.34}$$

Thus, the likelihood computation under a PhyloCRF exactly mirrors that of a PhyloHMM, and may therefore utilize an efficient dynamic programming solution based on a postorder evaluation of Felsenstein's recursion (section 9.6.1).

In the CRF formulation we may also opt to introduce λ terms as in the common "linear combination" expansion of *Q*:

$$\sum_a \xi(a)\left[\sum_b \lambda_{a,b}\xi(a,b)\left(\sum_d \lambda_{b,d}\xi(b,d)\xi(d)\right)\left(\sum_e \lambda_{b,e}\xi(b,e)\xi(e)\right)\right] \cdot\left[\sum_c \lambda_{a,c}\xi(a,c)\left(\sum_f \lambda_{c,f}\xi(c,f)\xi(f)\right)\left(\sum_g \lambda_{c,g}\xi(c,g)\xi(g)\right)\right] \tag{S1.35}$$

which may allow the CRF trainer to learn more discriminative "branch lengths." Unfortunately, to do so would be to model the effect of branch lengths on substitution probabilities as being linear, which is not ideal. A more principled solution is to simply include the branch-length parameters

from within the individual ξ functions in the system-level optimization when the CRF itself is trained (rather than training these submodels generatively and then leaving them fixed during later CRF training). Whether doing so would result in more accurate predictions in practice is a question for future research.

In the above description we have glossed over a fine point regarding the clique decomposition of the phylogeny networks. In particular, we considered cliques over the entire CRF graph $G_{X \cup Y}$, rather than over just the observable subgraph G_Y. As we mentioned earlier, however, the global availability of the observables to the Φ_c functions allows us to define a set of "virtual" cliques which may contain observables in addition to unobservables.

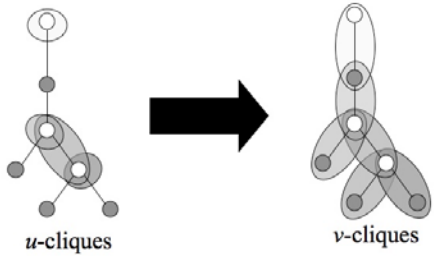

Figure S1.12 *Relation between u-cliques (left) and "virtual" cliques (v-cliques; right). The u-cliques may be expanded as needed within individual potential functions so as to contain observables, resulting in v-cliques.*

In Figure S1.12 we illustrate the relation between the *u*-cliques (shown on the left) and the "virtual" cliques (or *v-cliques*; shown on the right). In particular, the singleton *u*-clique $\{u\}$ for any unobservable vertex u having one or more observable neighbors in the full CRF graph $G_{X \cup Y}$ may be expanded into one or more virtual cliques so as to cover all possible edges in the phylogeny. License to perform this expansion is granted by the assumption that the Φ_c functions have access to the full vector $\mathbf{x}$ of observables, and that these functions may be crafted individually for each *u*-clique to perform any necessary computation in modeling the dependences between unobservables in the *u*-clique and any subset of observables.

In the above development we have ignored one other important issue. It may be recalled that in our formulation of PhyloHMM's in Chapter 9 we assumed the existence of separate evolution models $\psi_q \in \Psi$ for the different states q in the HMM part of the PhyloHMM. It was in fact the observed difference in evolutionary rates between coding and noncoding regions of genes which originally motivated the use of phylogenetic and other homology-aware models for comparative gene prediction. In the PhyloHMM these distinct evolution models exerted their influence via the term:

$$P(I_i^{(1)},...,I_i^{(n)} \mid S_i, y_i) \tag{S1.36}$$

for the i^{th} state (y_i) in a prospective parse. This term was then decomposed via Felsenstein's recursion into terms for individual branches in the phylogeny. In the PhyloCRF we can achieve a similar effect by introducing additional edges in the dependency graph so as to directly relate the label at each position i with all of the informants and their ancestors, as illustrated in Figure S1.13.

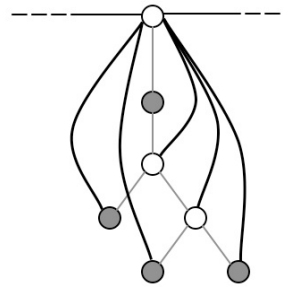

Figure S1.13 *Modeling the influence of position label (top node) on conservation patterns in the phylogeny portion of a CRF (tree structure at bottom).*

The effect of this change will be to enlarge the cliques in the phylogeny portion of the graph, so that each clique includes the putative label at each position in the target sequence. The label (i.e., `exon`, `intron`, etc.) would then be available as an argument to the corresponding Φ_c functions, which may then take the coding/noncoding status of the putative label into consideration when evaluating the conservation patterns among the informants, just as in the PhyloHMM (i.e., analogous to Equation S1.27).

Although we have limited our treatment of Phylo-CRF's to 0^{th}-order models—i.e., omitting any context dependence in the modeling of substitution patterns across any branch of the phylogeny—it is certainly possible to consider higher-order models. Unfortunately, the mere task of drawing the dependency graph for these higher-order models becomes somewhat arduous, especially when one endeavors then to determine precisely which set of additional dependency edges will result in a useful set of cliques. A simpler (though perhaps less rigorous) alternative is to turn this methodology around, by first selecting (from the complete graph containing all possible edges) the set of cliques which may be expected to best reflect any context-dependence in substitution rates, and then stipulating that all other cliques evaluate to zero under their respective potential functions.

That is, rather than crafting a dependency graph and then laboriously finding all of the cliques in this graph (of which there may be many more

than we would like to include in our model), we may instead choose a (manageably sized) set of cliques to represent the context dependence in substitution rates, and then ignore all other cliques. Although the latter strategy may seem rather *ad hoc*, an alternative view is that the CRF framework, rather than constraining the modeler or making his or her job unnecessarily laborious, should instead enable the modeler to directly encode within the model whatever prior knowledge s/he possesses concerning the system being modeled. Under this philosophy, the decision to use a "cliques first" or an an "edges first" strategy in specifying the model should be left open to the modeler as per the specific requirements of the particular system under study.

S1.9 External Evidence in Linear-chain CRF's

An alternative "heuristic" approach to the incorporation of external evidence in CRF's for sequence parsing involves the use of a rudimentary, linearly-structured CRF into which submodels of any type may be incorporated via the feature sensors f_i introduced in section S1.5. These feature sensors may encapsulate quite sophisticated submodels, such as the (probabilistic) phylogenetic models described in section 9.6. Such submodels are referred to as sensors for "rich features", since the internal dependency structure among the elements of the submodel are entirely abstracted away from the CRF proper and evaluated opaquely by the sensor. Because the internal dependencies among the observables (i.e., the input DNA sequence and any other evidence read in by the system) are abstracted into "rich features", the entire collection of observables may be represented by a single, atomic entity in the CRF, as illustrated in Figure S1.14.

Such a model is called a *linear-chain CRF* (*LC-CRF*) (Sutton and McCallum, 2006), since it contains only a single observable in addition to the linear chain of unobservables (the sequence labels). Because of the simplified structure, the *u*-cliques are now trivially identifiable as *singletons* (i.e., a single unobservable and the observable) and *pairs* (a pair of adjacent unobservables and the single observable). We denote the corresponding potential functions for these cliques f_{emit} and f_{trans}, respectively:

$$P(\mathbf{y} \mid \mathbf{x}) = \frac{1}{Z} e^{\sum_{c \in C} \sum_{i \in F} \lambda_i f_i(c, \mathbf{x})} = \frac{1}{Z} e^{\sum_{t=0}^{|S|-1} \pi f_{emit}(\mathbf{x}, \mathbf{y}_t) + \mu f_{trans}(\mathbf{x}, \mathbf{y}_{t-1}, \mathbf{y}_t)} \quad \text{(S1.37)}$$

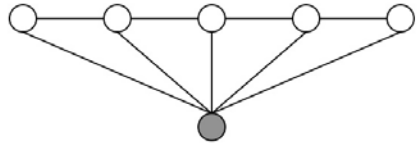

Figure S1.14 *A linear-chain CRF (LC-CRF). Unobservables are linked into a linear chain. Higher-order relations among unobservables are possible (not shown). Observables are denoted using a single shaded vertex, for notational simplicity.*

Additional "informant" evidence is now modeled not with additional vertices, but with additional "rich feature" functions in the decomposition of *Q*:

$$P(\mathbf{y} \mid \mathbf{x}) = \frac{1}{Z} e^{\sum_{c \in C} \sum_{i \in F} \lambda_i f_i(c, \mathbf{x})} = \frac{1}{Z} e^{\sum_{t=0}^{|S|-1} \pi f_{emit}(\mathbf{x}, \mathbf{y}_t) + \mu f_{trans}(\mathbf{x}, \mathbf{y}_{t-1}, \mathbf{y}_t) + \sum_{i \in F} \beta_i f_{rich(i)}(\mathbf{x}, \mathbf{y}_t)} \quad \text{(S1.38)}$$

where the "informants" and other external evidence are now encapsulated in **x**. The resulting decoder may be described as:

$$\mathbf{y}^* = \underset{\mathbf{y}}{\arg\max} \sum_{t=0}^{|S|-1} \pi f_{emit}(\mathbf{x}, \mathbf{y}_t) + \mu f_{trans}(\mathbf{x}, \mathbf{y}_{t-1}, \mathbf{y}_t) + \sum_{i \in F} \beta_i f_{rich(i)}(\mathbf{x}, \mathbf{y}_t) \quad \text{(S1.39)}$$

which it may be noted is reminiscent of the approach taken by a number of combiner-type programs (section 9.2). In this way, it may be argued that an LC-CRF with rich feature sensors constitutes a combiner program with a more theoretical—rather than *ad hoc*—formulation based on Markov random fields.

A *Phylo-LC-CRF* may now be formulated by simply incorporating a standard phylogeny module from a PhyloHMM directly into an LC-CRF via a rich feature sensor (Vinson *et al.*, 2007), as illustrated in Figure S1.15. Note that the resulting model is a hybrid between an *undirected* model (the CRF) and a *directed* model (the phylogeny). The advantages of such a formulation include the ability to re-use existing software from a PhyloHMM implementation, and possibly a reduced tendency toward overtraining, due to the opacity of the submodels during CRF (system-level) training. A possible disadvantage is that the resulting model may not be maximally discriminative—ie., greater predictive accuracy may conceivably be achieved with less separation between the CRF and its submodels, so that a discriminative training regime may be applied to jointly optimize all the parameters of the composite system. We address the issue of CRF training

next.

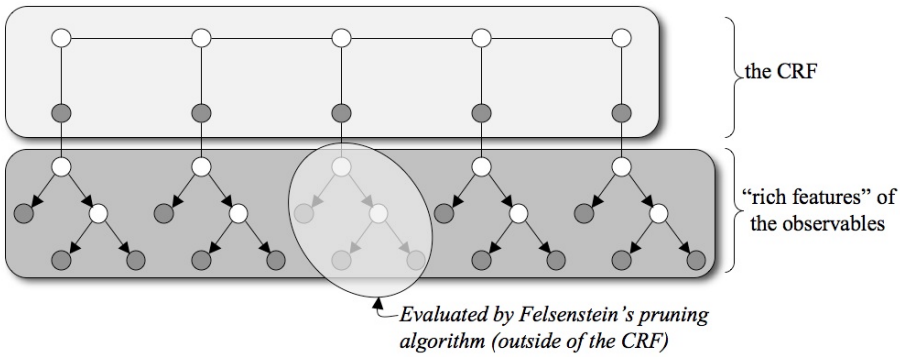

Figure S1.15 *A Phylo-LC-CRF. Informants and their phylogenetic relations are abstracted away from the CRF proper via "rich feature" functions, resulting in a hybrid between an undirected model (the CRF) and a directed model (the phylogeny).*

S1.10 Training CRF's

As explained in earlier chapters, probabilistic gene finders based on HMM's and their several variants are most commonly trained via *maximum likelihood* (ML):

$$\begin{aligned}\theta_{MLE} &= \arg\max_{\theta}\left(\prod_{(S,\phi)\in T} P_\theta(S,\phi)\right) \\ &= \arg\max_{\theta}\left(\prod_{(S,\phi)\in T}\prod_{y_i\in\phi} P_e(S_i \mid y_i, d_i) P_t(y_i \mid y_{i-1}) P_d(d_i \mid y_i)\right)\end{aligned} \tag{S1.40}$$

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).

As we saw in section 6.9, an alternative, "*discriminative training*" objective function for (G)HMM's is *conditional maximum likelihood* (*CML*), which is generally optimized via gradient ascent or some EM-like approach:

$$\theta_{CML} = \arg\max_{\theta}\left(\prod_{(S,\phi)\in T} P_\theta(\phi|S)\right), \tag{S1.41}$$

Although CML is rarely used for training gene-finding HMM's, it is a very natural objective function for CRF's, and indeed is considered the "default" training method for CRF's in the same way that ML serves as a default method for HMM's. Unfortunately, CML training is generally much slower than the methods used for training GHMM-based models, since the latter typically consist of submodels which are individually trained using closed-form expressions involving simple *n*-mer counts (recall that EM is virtually never applied in GHMM training, and is unnecessary for many practical HMM topologies).

Another disadvantage of CML training is that the resulting models may have a greater tendency toward overfitting. One common method for avoiding overfitting during CRF training is the use of *regularization* (Tikhonov, 1963), in which extreme values of parameters are penalized via an additional term in the objective function:

$$f_{objective}(\theta) = P_\theta(\mathbf{y} \mid \mathbf{x}) - \frac{\| \theta \|^2}{2\sigma^2}, \tag{S1.42}$$

where $\|\theta\|$ is the Euclidean 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 optimally (Sutton and McCallum, 2006).

In the regularization method, the above function $f_{objective}$ (or one similar to this one) serves as the objective function during training, in place of the usual $P(\mathbf{y}|\mathbf{x})$ objective function of 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 and McCallum, 2006).

Actual parameter estimation in CRF's is typically performed using a gradient-based method such as those described in supplementary chapter S2. We will briefly consider the computation of gradients in the CML framework. By combining the **x**'s and **y**'s from different training cases into one large graph, we can dispense with the multiplication in Eq. (S1.41):

$$\theta_{CML} = \arg\max_\theta P_\theta(\mathbf{y}|\mathbf{x}) = \arg\max_\theta \frac{1}{Z(\mathbf{x},\theta)} e^{\sum_{c \in C} \Phi_c(\mathbf{y},\mathbf{x},\theta)} \tag{S1.43}$$

The $P(\mathbf{y}|\mathbf{x})$ term can be computed via posterior decoding, as described earlier. However, since the **y**'s in the training set will typically be non-degenerate, it may be slightly more efficient in practice to compute the terms $1/Z(\mathbf{x})$ and $e^{\Sigma\Phi(y,c)}$ directly; an efficient algorithm for computing $Z(\mathbf{x})$ was given in Algorithm S1.2 (section S1.7), while the exponential term can be

computed via simple summation over the cliques of the graph (no dynamic programming is required for this latter part).

The actual gradients are given by:

$$\frac{\partial P(\mathbf{y}|\mathbf{x},\theta)}{\partial \theta_i} = \frac{e^{\sum_{c\in C}\Phi_c(c,\theta)}}{Z(\mathbf{x})} \sum_{c\in C} \frac{\partial \Phi_c(c,\theta)}{\partial \theta_i} - \frac{e^{\sum_{c\in C}\Phi_c(c,\theta)}}{Z(\mathbf{x})^2} \frac{\partial Z(\mathbf{x},\theta)}{\partial \theta_i} \tag{S1.44}$$

where $\partial\Phi_c(c,\theta)/\partial\theta_i$ may be computable analytically if possible, or can otherwise be estimated via 2-point symmetric differencing:

$$\frac{\partial \Phi_c(c,\theta)}{\partial \theta_i} \approx \frac{\Phi_c(c,\theta_{<i,\Delta>}) - \Phi_c(c,\theta_{<i,-\Delta>})}{2\Delta} \tag{S1.45}$$

where $\theta_{<i,\Delta>}$ denotes the vector obtained from θ when Δ is added to the i^{th} element. The partial derivatives of the partition function are given by:

$$\frac{\partial Z(\mathbf{x},\theta)}{\partial \theta_i} = \left(\sum_{\mathbf{y}\in L^N} e^{\sum_{c\in C}\Phi_c(c,\theta)} \right) \left(\sum_{c\in C} \frac{\partial \Phi_c(c,\theta)}{\partial \theta_i} \right) \tag{S1.46}$$

which it can be seen is simply a sum over $\partial\Phi_c(c,\theta)/\partial\theta_i$ terms given above.

An alternative to CML—*maximum expected boundary accuracy*—was recently introduced by Gross *et al.* (2007). The latter work also reported successful use of an alternative optimization technique called *RPROP* (Riedmiller and Braun, 1992), which utilizes only the sign of each partial derivative, while ignoring the actual magnitude of the gradient. The latter algorithm has been reported to be very fast when applied to the training of neural networks, and its use for training gene-finding models deserves further study.

S1.11 Templates and Parameter Tying

As we noted in section S1.4, a CRF does not technically come into existence until an input sequence **x** is read into memory and used to instantiate a CRF template along the length of the sequence. We will consider very briefly how this may be formalized.

Figure S1.16 illustrates the basic scenario for sequence-based CRF's. We begin with a template (shown at left in the figure) which contains internal references to its own nodes (dashed line in the figure). As the template is instantiated along the length of the observation sequence, these internal references may be redirected so as to point to corresponding vertices in prior

invocations of the template (middle portion of the figure). Finally, the directionality of these temporal edges may be abolished to arrive at a fully undirected graph (right figure), which constitutes the final CRF.

A key advantage to formulating CRF's in terms of templates is the ability to *tie* parameters between successive invocations of a template (Sutton and McCallum, 2006). In other words, training of the model consists only of training of the *template*—i.e., estimation of parameters for the template rather than for an entire CRF comprising numerous instantiations of a single template. When the trained template is instantiated multiple times into a CRF, the resulting model will in general contain far more parameters than the template itself, yet the parameters of the full CRF deriving from a single template parameter will all be *tied* so as to have the same value. This strategy has the advantage of reducing the number of parameters which need to be estimated, thereby reducing the tendency toward overtraining of the model.

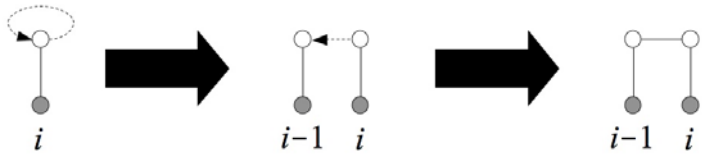

Figure S1.16 *Template instantiation for a simple sequence-based CRF template. The template (left) contains internal references to its own nodes (dashed line) which are redirected to corresponding nodes at the previous position, i-1 (middle figure). Directionality of the templated edges is abolished to arrive at the final undirected model (right).*

Formally, we define a *CRF sequence template* as a 7-tuple $\mathcal{T}=(L,\alpha,Y,X,\Omega,G,R)$, where L, α, Y, X, Ω, and G are defined as in a CRF, and R is a set of directed edges in $V\times V$, for $V=Y\cup X$, $X=\{x\}$. Note that the Φ functions in Ω must be defined over pairs of variable sets from adjacent template instantiations—e.g., $\Phi_c(\mathbf{y},\mathbf{x},\mathbf{y}',\mathbf{x}')$. We call the edges in R *references*. Instantiation of $\mathcal{T}$ into a full CRF $\mathcal{M}=(L,\alpha,Y',X',\Omega,G')$ is accomplished as follows.

Suppose an input sequence $\mathbf{x}$ of length n is provided. Define $Y'=\{y_{i,j}|0\le i<n, 0\le j<k\}$, for $Y=\{y_0,...,y_{k-1}\}$, and let $f(y_{i,j})=y_j$. Define $X'=\{x_i|0\le i<n\}$, where $x_i=\mathbf{x}_i$, and let $f(x_i)=x$. Let $p(x_i)=i$ and $p(y_{i,j})=i$. Define $G'=(V',E')$, for $G=(V,E)$, $V'=Y'\cup X'$, $E'=\{(w,z)|(f(w),f(z))\in E\}\cup\{(w,z)|(f(w),f(z))\in R \wedge p(w)=p(z)+1\}$. Finally, in order for $\mathcal{M}$ to be a CRF, any Φ_c in Ω for non-clique c in G' must evaluate to zero.

We leave the generalization of this scheme to more diverse types of CRF's as an exercise for the diligent reader.

S1.12 Case Studies

We now very briefly summarize the few results which have been reported to date in the gene-finding literature regarding the application of CRF's to the problem of eukaryotic gene prediction.

The first is due to Cullotta *et al.* (2005), who utilize a non-generalized, linear-chain CRF with linear-combination potential functions to predict human genes. Syntactic constraints (enforced by the potential functions) are encoded via a finite-state grammar including elements for canonical splice sites, start and stop codons, phase-specific introns, intergenic regions, and a cyclical (i.e., three-periodic) codon model.

Features of the linear potential functions included the following:

- binary indicators on all *n*-mers of size 1, 2, 3, 5, and 6 in the immediate vicinity of the current position
- counts of all *n*-mers of sizes 1-3 within the previous 40 bases
- counts of all *n*-mers of sizes 1-3 within the following 10 bases
- number of BLASTX hits covering the current position
- maximum score of any BLASTX hit covering the current position
- sum of scores of all BLASTX hits covering the current position

The resulting CRF was compared to the GHMM-based system *GENIE* (Kulp *et al.*, 1996) on a set of 450 human genes containing at least one intron. Although a version of the CRF lacking BLASTX features did not significantly outperform the GHMM-based system, incorporation of homology information into both the CRF and the GHMM showed that the CRF was indeed able to outperform the generative model by an appreciable margin (Culotta *et al.*, 2005: Table 2). The authors of this particular study point out that the CRF was somewhat handicapped in this comparison, in that the GHMM system was generalized (i.e., included non-geometric length modeling) whereas the CRF was not.

A second case study is provided by the recent work due to Vinson *et al.* (2007), who incorporated into a generalized CRF framework standard Phylo/GHMM feature sensors (29 in total), as well as a small number of non-probabilistic features comprising binary indicators on the existence of EST hits covering the current position or gaps in the current column of the informant alignment. The resulting model was tested on multiple gene sets from the fungal genome *Crytococcus neoformans*. Ten-fold cross-validation showed that the CRF model outperformed the dual-genome gene-finder *TWINSCAN* (described in section 9.1.1) as well as a PhyloHMM implementation provided by the same authors.

A key contribution of the Vinson study was their demonstration of the value of hybrid CRF models, since the resulting system contained both

undirected models (the CRF) and directed models (the phylogenetic sensor and the Markov chain sensors). Indeed, Vinson *et al.* explicitly recommend using probabilistic feature functions whenever possible, and resorting to non-probabilistic features only when necessary.

A third case study is provided by Gross *et al.* (2006), who constructed a (non-generalized) CRF for the prediction of genes in *Drosophila melanogastor* and provided results showing yet again that the CRF framework can produce greater predictive accuracy than a comparable generative model. The primary feature utilized by this CRF consisted of *n*-mer frequencies of size 3. The latter authors also compare several objective functions for training, and show that one in particular based on the goal of *empirical risk minimization* (section 10.14) provides significant accuracy gains over the others. Another implementation was described more recently by this group (Gross *et al.* 2007), in which SVM's (see section 10.14) and pre-computed alignments are used as feature sensors.

A final case study is provided by the report due to Bernal *et al.* (2007), who compare a generalized linear-chain CRF to several GHMM-based systems for prediction of human genes, showing that the CRF is capable of significantly outperforming the generative models for *ab initio* gene parsing.

Features utilized in this latter study include:

- binary indicators on individual *n*-mers
- putative feature lengths
- IMM's (section 7.2.5)—8^{th} order for coding, 4^{th} for noncoding
- Percent {G,C} within a 10kb window
- codon frequencies
- phase bias
- number of tandem repeats (section 11.4)
- WMM scores (section 7.3.1), WWAM scores (section 7.3.3)

These features are combined via the typical linear-combination formulation of Q (section S1.5).

It is perhaps interesting to note that very early gene-finding work by Stormo and Haussler (1994), which we mentioned in section 12.4, illustrated a method for non-generative modeling of gene structure which can now be seen to be very similar to the CRF approach.

S1.13 Practical Considerations

It is worthwhile at this point to take stock of the potential advantages and disadvantages associated with the use of CRF's in their varied forms versus Pair/Phylo/G/HMM-based models (collectively, "Markovian" or "generative" systems) for gene prediction as presented in the foregoing chapters of this book.

The two primary advantages of CRF's over HMM's, in our opinion, are (1) the inherently discriminative nature of CRF's (when trained via CML), and (2) their flexibility in incorporating external evidence such as sequence similarity to homologous and/or expressed genes. We elaborate on these two points below.

In the case of the first point, it should be noted that HMM's and LC-CRF's have been formally proven to constitute what is known as a *generative/discriminative pair* (Sutton and McCallum 2006), in the sense that both models utilize the same underlying family of parametric probability distributions but fit that model using different optimization criteria, with the generative model optimizing the fit to the joint distribution $P(X,Y)$ and the discriminative model optimizing the fit either to the posterior distribution $P(Y|X)$ or to the expected predictive accuracy of the deployed classifier (Ng and Jordan, 2002). Since the expected predictive accuracy of the resulting model is what we would ultimately like to maximize, it would seem that CRF's would be preferable, at least in principle, to HMM's in virtually all cases.

In the more general comparison between CRF's and the various HMM extensions such as GHMM's, PHMM's, and PhyloHMM's, we first point out that for each of the latter models there exists an equivalent CRF—we have already demonstrated this in the case of GHMM's and PhyloHMM's, and leave it as an exercise for the reader to show that other HMM variants can be reformulated as CRF's. Each of these CRF's may be extended via appropriate λ terms, which provide a sort of "hook" for discriminative training of the overall model in a way which is not possible with the corresponding generative model.

Indeed, as has already been noted several times in this volume, the use of so-called "fudge factors" in generative parsing models is deemed by some a "necessary evil" in the pursuit of greater predictive accuracy by practical gene-finding systems (e.g., sections 7.3.4, 9.6.7, 12.4). By simply calling the resulting model a CRF rather than a Pair/Phylo/G/HMM, the use of such "fudge factors" in many cases need not be viewed as entirely unprincipled.

This leads us into the second point made above, regarding the incorporation of arbitrary external evidence—a capability traditionally attributed only to the *ad hoc* class of gene finders known as "combiners" (section 9.2). As noted earlier in this chapter, the similarity with several combiner systems would seem to place CRF's in the upper echelon of integrative gene-finding systems, since combiner methods have traditionally performed exceptionally well (e.g., Allen and Salzberg, 2005) in comparison to *ab initio* systems. Unlike traditional combiners, however, which have tended to be somewhat *ad hoc* in nature, CRF's are based on a (largely) solid theoretical foundation.

Other virtues of the CRF framework include results regarding the convexity of various objective functions for training (e.g., Sutton and

McCallum, 2006) and the capability—as yet still highly under-utilized in the field of gene finding—to model long-range dependencies, either between unobservables and observables, or between the unobservables themselves.

While it should be clear from our exposition of CRF's that they generally possess greater modeling flexibility than HMM's, it is not so clear that this increased flexibility is uniformly conducive to greater gene-finding accuracy. In particular, one might wonder whether the greater number of modeling decisions with which a modeler is faced in the CRF framework might lead, at least in some cases, to greater freedom to make the *wrong* decisions. By contrast, the purely probabilistic nature of HMM's and their variants may leave the modeler fewer possibilities for making poor modeling decisions.

The latter considerations might be interpreted as support for the discipline endorsed by Vinson and colleagues, who (as we noted earlier) recommend that the CRF modeler "use probabilistic models for feature functions when possible and add non-probabilistic features only when necessary" (Vinson *et al.*, 2007). Such an approach is exemplified by the hybrid model described in section S1.9, and suggests a practical method for extending an existing Markovian system into a CRF—namely, by progressively incorporating λ terms and other CRF-specific features (e.g., opening up the submodels to system-level parameter estimation for greater discrimination, etc.) into the model while ensuring via controlled tests that the predictive accuracy of the model has not suffered as a result. In this way the current generation of Pair/Phylo/G/HMM-based systems might be progressively evolved into CRF-based systems through a careful refactoring of systems already known to produce relatively accurate predictions into more discriminative models with possibly higher accuracy. That this is indeed a practical option for maintainers of existing, HMM-based systems should be readily apparent given the similarity of the decoding problem for CRF's and G/HMM's; as we have previously noted, a G/HMM decoder can be almost trivially adapted for use in a G/CRF-based gene-finding system, with the time complexity of the resulting parser remaining unchanged so long as any long-range dependencies included in the model can be bounded by a small constant.

Additional References

See the print edition for the full list of references.

Allen JE, Salzberg SL (2005) JIGSAW: integration of multiple sources of evidence for gene prediction. *Bioinformatics* 21:3596-3603.

Bernal A, Crammer K, Hatzigeorgiou A, Pereira F (2007) Global discriminative learning for higher-accuracy computational gene prediction. *PLoS Comput Biol* 3:e54.

Besag J (1974) Spatial interaction and the statistical analysis of lattice systems.

Journal of the Royal Statistical Society B 36, pp192-236.
Culotta A, Kulp D, McCallum A (2005) Gene prediction with conditional random fields. *Technical Report UM-CS-2005-028*. University of Massachusetts, Amherst.
Fletcher R (1980) *Practical Methods of Optimization. Volume 1: Unconstrained Optimization*. Wiley, New York.
Felsenstein J (1981) Evolutionary trees from DNA sequences. *Journal of Molecular Evolution* 17:368-376.
Galassi M, Davies J, Theiler J, Gough B, Jungman G, Booth M, Rossia F (2005) *GNU Scientific Library Reference Manual*, 2nd ed. Network Theory Ltd, Bristol, UK.
Gross SS, Chuong BD, Sirota M, Batzoglou S (2007) CONTRAST: a discriminative, phylogeny-free approach to multiple informant *de novo* gene prediction. *Genome Biol.* 8:R269.
Gross SS, Russakovsky O, Do CB, Batzoglou S (2006) Training conditional random fields for maximum labelwise accuracy. In: *Advances in Neural Processing Systems 19*.
Kulp D, Haussler D, Reese M, Eeckman F (1996) A generalized hidden Markov model for the recognition of human genes in DNA. *ISMB '96*.
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*.
McCallum A, Rohanimanesh K, Sutton C (2003) Dynamic conditional random fields for jointly labeling multiple sequences. *Proceedings of the 2003 Conference on Neural Information Processing*.
Quattoni A, Wang S, Morency L-P, Collins M, Darrell T (2006) Hidden-state conditional random fields. *MIT CSAIL Technical Report.*
Reidmiller M, Braun H (1993) A direct adaptive method for faster backpropagation learning: the RPROP algorithm. *Proc IEEE Intl Conf Neural Networks.* San Francisco. pp586-591.
Stormo GD, Haussler D (1994) Optimally parsing a sequence into different classes based on multiple types of evidence. In: *Proceedings of the International Conference on Intelligent Systems for Molecular Biology* (ISMB'94) 2:369-375.
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.
Tikhonov AN (1963) Solution of incorrectly formulated problems and the regularization method. *Soviet Math Dokl* 4:1035-1038.
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

Elizabeth Rach provided useful comments and suggestions which have improved the text and technical content of this chapter.