6.047 / 6.878 Computational Biology: Genomes, Networks, Evolution
Fall 2008

6.047/6.878 Computational Biology Nov. 4, 2008

Lecture 18: CRFs for Computational Gene Prediction

Lecturer: James E. Galagan

1 Overview of gene prediction

One of the fundamental problems in computational biology is to identify genes in very long genome sequences. As we know, DNA is a sequence of nucleotide molecules (a.k.a. bases) which encode instructions for generation of proteins. However, not all of these bases are responsible for protein generation. As an example shown in the 4th slide on page 1 of [2], in the eukaryotic gene structure, only exons contribute to protein manufacturing. Other segments, like intron, intergenic, start, stop, are not directly responsible for protein production. Therefore our task in gene prediction (or genome annotation) is, given a DNA sequence (X) with zero or more genes and some "evidence" associated with the given DNA sequence, **determine a labeling (Y) that assigns to each base a label according to the functionality of that part of the gene.** For example, the labels can be intergenic, start, exon, acceptor, intron, donor, stop, etc.

How do we do gene prediction? It turns out that we can make use of several types of evidence. For example, some gene parts are associated with short fixed sequences, which are called signals. However, since such short sequences appear randomly everywhere in the DNA sequence, we cannot solely use these signals for gene prediction purposes. Other evidence relies on the tendency of certain genomic regions to have specific base composition (a.k.a. content measures). For example, there are usually multiple codons for each amino acid and not all codons are used equally often, which gives rise to different ratios of nucleotides in coding sequences. Apart from this evidence, which stem from DNA properties, there is also evidence like direct experimental evidence, BLAST hits, HMMer hits, etc. **The main challenge of gene prediction algorithms is how to take advantage of all these types of evidence together.**

The most popular method so far of combining evidence is to use Hidden Markov Models (HMMs). In an HMM, we model the labels as hidden states and assume that the hidden states constitute a Markov chain. We assign an emission probability to every ($x \in X, y \in Y$) to model the probability of observing base x when the hidden state is y. This model is also called a generative model, since a convenient way to think of HMMs is to imagine that there is a machine with a button. By pushing the button, one can generate a genome sequence according to some probability distribution. In gene prediction, we use maximum-likelihood training to compute state transition probabilities and emission probabilities, and then find the most likely hidden state sequence $Y = y_1 \cdots y_n$. Note that HMMs in fact model a *joint distribution* over bases and hidden states, namely $P(X, Y) = P(\text{Labels}, \text{Sequence})$. However, in gene prediction problems, we are given X and only need to predict Y. In other words, we want the conditional probability distribution, $P(Y|X)$. There are also some so-called generalized hidden Markov models (GHMMs). In these models, each state may emit more than one symbol, emission probabilities may be modeled by any arbitrary probabilistic models, and sometimes the feature lengths are explicitly modeled according to experimental data (instead of restricted to geometric distributions as required by HMMs). As an example, the 2nd slide on page 2 of [2] demonstrates Genscan, a software based on GHMMs. The 3rd slide shows a comparison of performances (in

terms of both Sensitivity and Specificity measures) of four different gene prediction softwares based on GHMMs. One can see that there is still room for improvement, and this is mostly due to the limitations of HMMs.

2 Why Conditional Random Fields (CRFs)

There are certain intrinsic limitations to the HMM approach. First, as we mentioned earlier, the most natural distribution to model is the conditional distribution instead of the joint distribution. HMMs expend unnecessary effort to model the joint distribution $P(X, Y) = P(Y|X) * P(X)$, when our real goal is just $P(Y|X)$. What's even worse is that during the learning process we might have to trade the accuracy in modeling $P(Y|X)$ in order to model $P(X)$ more accurately, and this leads to less accurate gene prediction.

Another limitation is that since HMMs are directed graphical models, each component of HMMs has strict probabilistic semantics. Therefore it is hard for HMMs to model *dependent* evidence. However as we know in gene prediction, there are many dependent evidence that need to be incorporated. For example, HMMer protein domain predictions come from models based on known protein sequences and these sequences are the same proteins searched by BLAST. Therefore evidence coming from HMMer hits and BLAST hits are by no means independent. In fact, dependence is the rule for most evidence available for gene prediction. One major drawback of HMMs is that it is very cumbersome to model arbitrary dependent evidence of the input sequences. One may try to modify HMMs by adding more hidden states to model such dependency, but this approach usually leads to training and inference that is computationally intractable and/or suffers from data sparsity problems.

One common strategy is to simply assume independence (naive Bayesian assumption) among the conditional probabilities. In other words, we would make the assumption that

$$P(X|Y_i Y_{i-1} \dots Y_1) = P(X|Y_i)$$

, but this assumption is typically a false one in practice.

All these difficulties with HMMs motivated people to propose an alternative approach known as *discriminative models* to model the conditional distribution $P(Y|X)$ directly. Contrary to the directed graphical model of HMMs, this is an undirected graphical model (i.e. Markov random field). In the undirected graphical models, edges between nodes no longer bear probabilistic semantics. Instead, an edge simply represent some "compatibility function" (the $f$eature functions) between the values of the random variables. These feature functions are then globally normalized to assign a probability to each graph configuration. Because of this change in semantics, we can simply add edges between label y_j and any set of bases in X without creating additional dependencies among nodes in Y. We can also model the conditional probability $P(y_j|X)$ without considering the dependencies among X.

A particular instance of the undirected graphical model is called the conditional random field model (CRF) [1]. In this model, a set of unobserved variables are conditioned on a set of observed variables which are called conditional random fields. Our new CRF models have the following three desirable characteristics:

[1] Like HMMs, the CRF models were also borrowed from the field of natural language processing (NPL) by computational biologists.

1. There are efficient learning algorithms to compute the parameters of the model and efficient inference algorithms to predict hidden states;

2. It is easy to incorporate diverse evidence;

3. We can build on the best existing HMMs for gene calling.

A linear chain CRF is shown in the last slide on page 4 of [2]. This is a very simple CRF model which enables us to do efficient training and inference. In this model, the input data (DNA sequence, BLAST hits, ESTs, etc) is denoted collectively by X. The hidden state labels (exon, intron, etc) are denoted by Y. We arrange all the labels in a linear chain and assign a set of feature functions to every clique $\{y_{i-1}, y_i, X\}$ and denote them by $f_j(y_{i-1}, y_i, X)$, where $j = 1, \ldots, J$. We also assign a weight λ_j to each feature function and finally define the conditional probability to be

$$P(Y|X) = \frac{1}{Z(X)} \exp\left\{ \sum_{j=1}^{J} \lambda_j \sum_{i=1}^{N} f_j(y_i, y_{i-1}, X) \right\},$$

where the normalization factor $Z(X)$ is given by

$$Z(X) = \sum_{Y} \exp\left\{ \sum_{j=1}^{J} \lambda_j \sum_{i=1}^{N} f_j(y_i, y_{i-1}, X) \right\}.$$

Doing so guarantees that

$$\sum_{Y} P(Y|X) = 1$$

as required by the semantic of probabilities.

The basic idea underling this formality is that feature functions $\{f_j\}$ return real values on pairs of labels and input data that we think are important for determining $P(Y|X)$. We may not know how the conditional probability is changed by this feature function or what is its dependence on other evidence. However, we model this dependency by weights $\{\lambda_j\}$ and learn these weights to maximize the likelihood of training data. Finally we normalize the probability to ensure that $P(Y|X)$ sums to one over all possible Y's. The advantage of this formality is that it captures the conjunction property between labels and input data directly without resort to $P(X)$.

There are three fundamental steps in applying CRF models to gene prediction: **Design, Inference and Learning.** In the following sections we will look at these three steps in details.

3 CRF Design

Selecting feature functions is the core issue in CRF applications. Technically speaking, feature functions can be arbitrary functions that return real values for some pair of labels (y_i, y_{i-1}) and the input X. However we want to select feature functions that capture constraints or conjunctions of label pairs (y_i, y_{i-1}) and the input X that we think are important for $P(Y|X)$. Commonly-used feature functions include:

- indicator function where $f_j(y_i, y_{i-1}, X) = 1$ for certain clique $\{y_i, y_{i-1}, X\}$ and 0 otherwise;
- sum, product, etc over labels and input data;
- some probability distribution over cliques $\{y_i, y_{i-1}, X\}$, and so on.

As an interesting example, we can see two feature functions in the slides 4 and 5 on page 5 of [2]. We are in the situation that both BLAST and HMMer evidence suggest labeling y_i as exon. This will be hard to handle in HMMs but is very easy in CRF model: we simply include a BLAST feature function and a HMMer feature function and assign two weights λ_{BLAST} and λ_{HMMer} for these two feature functions. Thus, we do not need to worry about the dependence between the BLAST and HMMer evidence. There is no requirement that evidence represented by feature functions be independent. The underlying reason for this is we do not model $P(X)$ in CRF models. All we care about is which evidence contributes to the conditional probability $P(Y|X)$. The weights will determine the (relative) extent to which each set of evidence contributes and interacts.

In practice, there may be thousands or even millions of arbitrary indicator feature functions to incorporate into our CRF model. A naive approach is try all of them via brute force search, which is obviously impractical. However, a much better strategy is to take advantage of all the research on HMMs during the past decade by using the best HMM as our starting point. That is, we start with feature functions derived from the best HMM-based gene prediction algorithms. To see why this strategy works, one may look at slides 3, 4, 5 and 6 on page 6 of [2]. As it is shown there, HMM is simply a special case of CRF with all the feature function weights $\lambda = 1$ and each feature function taking the special form

$$\begin{aligned} f_{\text{HMM}}(y_i, y_{i-1}, x_i) &= \log(P(y_i|y_{i-1}) \cdot P(x_i|y_i)) \\ &= \log(P(y_i|y_{i-1})) + \log(P(x_i|y_i)) \\ &= f_{\text{HMM-Transition}} + f_{\text{HMM-Emission}}, \end{aligned}$$

where we define $f_{\text{HMM-Transition}}$ as

$$f_{\text{HMM-Transition}}(y_i, y_{i-1}, x_i) = \log(P(y_i|y_{i-1}))$$

and define $f_{\text{HMM-Emission}}$ by

$$f_{\text{HMM-Emission}}(y_i, y_{i-1}, x_i) = \log(P(x_i|y_i)).$$

Thus, we see that HMMs are just a very special form of the CRF model. As a result, adding new evidence to existing HMM becomes simple because we can add arbitrary feature functions to the CRF-version of our HMM and then learn the weights of these new feature functions empirically. These weights will capture the impact of the incorporating new features into our original HMM model.

HMMs and linear chain CRF explore the same family of conditional distributions $P(Y|X)$, and we can convert between HMMs and linear chain CRFs. In fact, HMMs and CRFs form a generative-discriminative pair as demonstrated in slide 3 on page 7 of [2]. An important reason for adopting linear chain CRF is because the underlying probability distribution is a factoring distribution, reducing the computation complexity of probability distributions from exponential time to linear time. Another reason is efficient inference capabilities, as we see in the next section.

4 CRF Inference

Recall that our ultimate goal in gene prediction is, when given a sequence of DNA and some evidence (denote them collectively by X), to select the best labeling Y. As with HMMs, the most natural choice for "best labeling" is simple the most probable labeling given by

$$\arg\max_Y P(Y|X) = \arg\max_Y \left\{ \frac{1}{Z(X)} \exp\left\{ \sum_{j=1}^{J} \lambda_j \sum_{i=1}^{N} f_j(y_i, y_{i-1}, X) \right\} \right\}.$$

But of course we can not afford to score every possible Y. The chain structure of the linear chain CRF plays a crucial role in this respect since it guarantees the sub-optimality structure of $P(Y|X)$, just the same as in HMMs we have seen before. Therefore, following the notation we used in the Viterbi algorithm, let $v_k(i)$ be the probability of the most likely path passing through i and ending at state k, then we have the recurrent relation

$$v_k(i) = \max_\ell \left\{ v_\ell(i-1) \cdot \exp\left(\sum_{j=1}^{J} \lambda_j f_j(k, \ell, X) \right) \right\}.$$

This recurrent relation enables us to compute the most probable labeling by dynamic programming, as the Viterbi algorithm for HMMs.

In fact to see the relation between CRFs and HMMs more explicitly, as shown in slides 2, 3 and 4 on page 8 of [2], we can define the quantity Ψ_{HMM} for HMMs by

$$\Psi_{HMM}(y_i, y_{i-1}, X) = P(y_i|y_{i-1})P(x_i|y_i),$$

and define an analogous quantity for generic CRFs by

$$\Psi_{CRF}(y_i, y_{i-1}, X) = \exp\left\{ \sum_j \lambda_j f_j(y_i, y_{i-1}, x_i) \right\}.$$

Thus, we can rewrite all HMM equations (Viterbi, Forward, Backward, etc) in terms of Ψ_{HMM}, and then replace Ψ_{HMM} with Ψ_{CRF}, and get a set of analogous equations for CRF models.

5 CRF Learning

Suppose we are given an independent and identically distributed (i.i.d.) training set $\{(X^{(k)}, Y^{(k)})\}$, which is usually a set of manually curated genes sequences for which all nucleotides are labeled. Define

$$P_\lambda(Y|X) = \frac{1}{Z_\lambda(X)} \exp\left\{ \sum_{j=1}^{J} \lambda_j \sum_{i=1}^{N} f_j(y_i, y_{i-1}, X) \right\},$$

where

$$Z_\lambda(X) = \sum_Y \exp\left\{ \sum_{j=1}^{J} \lambda_j \sum_{i=1}^{N} f_j(y_i, y_{i-1}, X) \right\}.$$

We would like to chose $\lambda = (\lambda_1, \ldots, \lambda_J)$ to maximize $L(\lambda) = \prod_{k=1}^{K} P_\lambda(Y^{(k)}|X^{(k)})$, that is, the product of the conditional probabilities of all the training data. For numerical stability convenience (to avoid computational underflow), we instead maximize $\ell_\lambda = \log L(\lambda)$, the *log-likelihood* of the data

$$\ell(\lambda) = \sum_{k=1}^{K} \sum_{j=1}^{J} \lambda_j \sum_{i=1}^{N} f_j(y_i^{(k)}, y_{i-1}^{(k)}, X^{(k)}) - \sum_{k=1}^{K} \log Z_\lambda(X^{(k)}).$$

As one can show, $\ell(\lambda)$ is concave and therefore is guaranteed to have a global maximum. The global maximum is obtained at $\frac{\delta \ell(\lambda)}{\delta \lambda} = 0$. As we see from a homework problem, at the maximum we have

$$\sum_{k}^{K} \sum_{i=1}^{N} f_j(y_i^{(k)}, y_{i-1}^{(k)}, X^{(k)}) = \sum_{k}^{K} \sum_{i=1}^{N} \sum_{Y'} f_j(y_i^{'(k)}, y_{i-1}^{'(k)}, X^{(k)}) P_{\text{model}}(Y'|X^{(k)}).$$

Note that the lefthand side of the above equation is the actual count in the training data while the righthand side corresponds to the expected count under the CRF modeled distribution. This nice feature guarantees that, using maximum likelihood training, the characteristics of the training data must also hold in our CRF model.

However, the feature functions are usually given in "blackbox" forms and in general are not invertible. Therefore we have no way to find the global maximum assignment of weights in analytical closed forms. We can in stead rely on a gradient search algorithm outlined below.

Gradient Search Algorithm for Computing λ

1 Define forward/backward variables similar to HMMs

2 Set weights λ arbitrarily

3 Compute $Z(X)$ using forward/backward algorithms

4 Compute $\delta \ell(\lambda)/\delta \lambda_i$ using $Z(X)$ and forward/backward algorithms

5 Update each parameter by gradient search using $\delta \ell(\lambda)/\delta \lambda_i$ (quasi-Newton method)

6 Go to step 3 until convergence to global maximum

6 CRF Applications to Gene Prediction

To summarize, in order to apply CRFs in gene prediction, we first design the feature functions on label pairs $\{y_i, y_j\}$ and X. Then we use training data set to select the weights λ. Finally, by

$$P(Y|X) = \frac{1}{Z(X)} \exp\left\{ \sum_{j=1}^{J} \lambda_j \sum_{i=1}^{N} f_j(y_i, y_{i-1}, X) \right\},$$

we can use the selected feature functions and computed weights to find the most probable labeling Y for a given input sequence X.

On page 9 of [2], we find a brief introduction to Conrad, a gene predictor based on CRF. One can see that, when predicting on C. neoformans chr 9, Conrad outperforms the best previous predictor Twinscan.

References

[1] Scribe notes - Lecture 6: CRFs for Computational Gene Prediction. MIT 6.047/6.878 Computational Biology, Sept. 25, 2007.

[2] James Galagan. Conditional Random Fields for Computational Gene Prediction. Lecture note 6 of MIT 6.047/6.878 Computational Biology, Sept. 2007.

[3] Aron Culotta, David Kulp and Andrew McCallum. Gene prediction with conditional random fields. Technical report UM-CS-2005-028, University of Massachusetts, Amherst, http://www.cs.umass.edu/~culotta/pubs/culotta05gene.pdf