Conditional Random Fields for Computational Gene Prediction

Genome Annotation

Eukaryotic Gene Structure

Gene Prediction with HMM

Limitations of HMM Approach (1)

Dependent Evidence

- HMmer protein domains predictions come from models based on known protein sequences
  - Protein sequences for the same domain are aligned
  - Conservation modelled with HMM
- But these are the same proteins searched by BLAST
- If we see a HMmer hit, we are already more likely to get a BLAST hit, and vice versa

BLAST and HMmer do not provide independent evidence - Dependence is the rule for most evidence
Dependent Evidence in HMMs

- HMMs explicitly model $P(X_i|Y_i)=P(\text{Blast, Hmmer}|Y_i)$
  - Not enough to know $P(\text{Hmmer}|Y_i)$, also need to know $P(\text{Hmmer}|Y_i, \text{Blast})$
  - Need to model these dependencies in the input data
- Every time we add new evidence (i.e. ESTs) we need to know about dependence on previous evidence
  - E.g. not just $P(\text{EST}|Y_i)$ but $P(\text{EST}|Y_i, \text{Blast}, \text{Hmmer})$
- Unpleasant and unnecessary for our task: classification
- A common strategy is to simply assume independence (Naïve Bayes assumption)

Independencies in X

HMMs make assumptions about dependencies in X

Effectively each $Y_i$ “looks” at only a contiguous subset of $X$ given the previous $Y_{i-1}$

Limitations Stem from Generative Modeling

HMMs are models of full joint probability distribution $P(X,Y)$

$P(X,Y) = P(Y|X) P(X)$

But this is all we need for gene prediction!

Generative Modeling of $P(X)$

- HMMs expend unnecessary effort to model $P(X)$ which is never needed for gene prediction
  - Must model dependencies in $X$
- During learning, we might trade accuracy in modeling $P(Y|X)$ in order to model $P(X)$ accurately
  - Less accurate gene prediction

Discriminative Models

Model conditional distribution $P(Y|X)$ directly

Discriminative models outperform generative models in several natural language processing tasks

Discriminative Model

Desirable characteristics

1. Efficient learning & inference algorithms
2. Easily incorporate diverse evidence
3. Build on best existing HMM models for gene calling
Linear Chain CRF

Hidden state labels (exon, intron, etc.)

Input data (sequence, blast hits, ESTs, etc.)

feature weights

feature functions

normalization

\[ P(Y|X) = \frac{1}{Z(X)} \exp \left( \sum_{j=1}^{J} \sum_{i} f_j(Y_i, Y_{i+1}, X) \right) \]

The Basic Idea

- Feature functions, \( f_j \), return real values on pairs of labels and input data that we think are important for determining \( P(Y|X) \)
  - e.g. If the last state \( Y_{i-1} \) was intron and we have a blast hit \( x \), we have a different probability for whether we are in an exon \( Y_i \) now.
- We may not know how this probability has changed or dependence on other evidence
- We learn this by selecting weights, \( \lambda_j \), to maximize the likelihood of training data
- \( Z(X) \) is a normalization constant that ensures that \( P(Y|X) \) sums to one over all possible \( Y \)

Using CRFs

\[ P(Y|X) = \frac{1}{Z(X)} \exp \left( \sum_{j=1}^{J} \sum_{i} f_j(Y_i, Y_{i+1}, X) \right) \]

Design
1. Select feature functions on label pairs \( (Y_i, Y_{i+1}) \) and \( X \).

Inference
2. Given weights and feature functions, find the most probable labeling \( Y \), given an input \( X \).

Learning
3. Use a training set of data to select the weights, \( \lambda \).

What Are Feature Functions?

Core issue in CRF – selecting feature functions
1. Features are arbitrary functions that return a real value for some pair of labels \( (Y_i, Y_{i+1}) \) and the input, \( X \)
   - Indicator function – 1 for certain \( (Y_i, Y_{i+1}, X) \), 0 otherwise
   - Sum, product, etc., over labels and data
   - Could return some probability over \( (Y_i, Y_{i+1}, X) \) – but this is not required
2. 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 \( P(Y|X) \)
3. Determine characteristics of the training data that must hold in our CRF model

Example Feature Function

An BLAST hit at position \( i \) impacts the probability that \( Y_i = \text{exon} \). To capture this, we can define an indicator function:

\[ f_{\text{BLAST,exon}}(Y_i, Y_{i+1}, X) = \begin{cases} 1 & \text{if } Y_i = \text{exon} \text{ and } X = \text{BLAST} \\ 0 & \text{otherwise} \end{cases} \]

Adding Evidence

An BLAST hit at position \( i \) impacts the probability that \( Y_i = \text{exon} \). To capture this, we can define an indicator function:

\[ f_{\text{BLAST,exon}}(Y_i, Y_{i+1}, X) = \begin{cases} 1 & \text{if } Y_i = \text{exon} \text{ and } X = \text{BLAST} \\ 0 & \text{otherwise} \end{cases} \]

A protein domain predicted by the tool HMMer at position \( i \) also impacts the probability that \( Y_i = \text{exon} \).

\[ f_{\text{HMMer,exon}}(Y_i, Y_{i+1}, X) = \begin{cases} 1 & \text{if } Y_i = \text{exon} \text{ and } X = \text{HMMer} \\ 0 & \text{otherwise} \end{cases} \]

But recall that these two pieces of evidence not independent
Dependent Evidence in CRFs

There is no requirement that evidence represented by feature functions be independent

• Why? CRFs do not model $P(X)$!
• All that matters is whether evidence constrains $P(Y|X)$
• The weights determine the extent to which each set of evidence contributes and interacts

A Strategy for Selecting Features

• Typical applications use thousands or millions of arbitrary indicator feature functions – brute force approach
• But we know gene prediction HMMs encode useful information

Strategy
1. Start with feature functions derived from best HMM based gene prediction algorithms
2. Use arbitrary feature functions to capture evidence hard to model probabilistically

Alternative Formulation of HMM

We can define a function $f$ over each of these pairs

$log P(Y,X) = log \left( \prod_{i=1}^{N} P(y_i | y_{i-1}) \times P(x_i | x_{i-1}) \right)$

This is the formula for a linear chain CRF with all $\lambda = 1$

Implementing HMMs as CRFs

We can implement an HMM as a CRF by choosing

$\lambda_{\text{HMM Transition}}(x_{i+1} | x_i = y_{i+1}) = \log \left( P(x_{i+1} | y_{i+1}) \right)$

$\lambda_{\text{HMM Emission}}(y_{i} | x_i) = \log \left( P(y_i | x_i) \right)$

$\lambda_{\text{HMM Transition}} = \lambda_{\text{HMM Emission}} = 1$

Either formulation creates a CRF that models that same conditional probability $P(Y|X)$ as the original HMM
Adding New Evidence

- Additional features are added with arbitrary feature functions (i.e., \( f_{\text{blast, exon}} \)).
- When features are added, learning of weights empirically determines the impact of new features relative to existing features (i.e., relative value of \( \lambda_{\text{HMM}} \) vs \( \lambda_{\text{blast, exon}} \)).

CRFs provide a framework for incorporating diverse evidence into the best existing models for gene prediction.

Conditional Independence of \( Y \)

\[
P(Y|X) = \exp \left\{ \sum_{i=1}^{J} \alpha_i \sum_{j=1}^{N} f_i(Y_j, Y_{j+1}, X) \right\} / Z(X)
\]

Both cases: \( Y_i \) conditionally independent of all other \( Y \) given \( Y_{i-1} \).

Conditional-Generative Pairs

HMMs and linear chain CRFs explore the same family of conditional distributions \( P(Y|X) \)

Can convert HMM to CRF
- Training an HMM to maximize \( P(Y|X) \) yields same decision boundary as CRF.

Can convert CRF to HMM
- Training CRF to maximize \( P(Y, X) \) yields same classification boundary as HMM.

HMMs and CRFs form a generative-discriminative pair

Ng, Jordan (2002)

Practical Benefit of Factorization

- Allows us to take a very large probability distribution and model it using much smaller distributions over "local" sets of variables.

Example: CRF with \( N \) states and 5 labels (ignore \( X \) for now)

\[
P(Y_1, Y_2, Y_3, ..., Y_N) \\
P(Y_i, Y_{i+1})
\]

\( 5^N \) \( 5^5 \times 5^N \)

(\( 5^5 \) if \( P_i = P_{i+1} \) for all \( i \))

Using CRFs

Design
1. Select feature functions on label pairs \( (Y_i, Y_{i+1}) \) and \( X \).

Inference
2. Given weights and feature functions, find the most probable labeling \( Y \) given an input \( X \).

Learning
3. Use a training set of data to select the weights \( \lambda \).
Labeling A Sequence

Given sequence & evidence X, we wish to select a labeling, Y, that is in some sense 'best' given our model.

As with HMMs, one sensible choice is the most probable labeling given the data and model:

\[
\arg \max_y P(Y|X) = \arg \max_y \exp \left( \sum_{j=1}^{N} \sum_{i} f_j(Y_i, Y_{i+1}, X) \right)
\]

But of course, we don't want to score every possible Y. This is where the chain structure of the linear chain CRF comes in handy…

Why?

Dynamic Programming

Recall From Previous Slides

Recall From Previous Slides

Combined HMM and CRF Inference

Using CRFs

Design 1. Select feature functions on label pairs \((Y_i, Y_{i+1})\) and X.

Inference 2. Given weights and feature functions, find the most probable labeling \(Y\) given an input X.

Learning 3. Use a training set of data to select the weights \(\lambda\).
Maximum Likelihood Learning

- We assume an iid training set \( \{(x(k), y(k))\} \) of K labeled sequences of length N
  - A set of manually curated genes sequences for which all nucleotides are labeled
- We then select weights, \( \lambda \), that maximize the log-likelihood, \( L(\lambda) \), of the data

\[
L(\lambda) = \sum_{k=1}^{K} \sum_{j=1}^{N} \sum_{i=1}^{M} \lambda_{ji} f_{ij}(Y^{(k)}, Y_{i-1}^{(k)}, X^{(k)}_i) \sum_{k=1}^{K} \log Z(X^{(k)})
\]

**Good news**

\( L(\lambda) \) is concave - guaranteed global max

Gradient Search

**Bad news**

No closed solution – need gradient method

**Need efficient calculation of** \( \partial L(\lambda) / \partial \lambda \) and \( Z(X) \)

**Outline**

1. Define forward/backward variables akin to HMMs
2. Calculate \( Z(X) \) using forward/backward
3. Calculate \( \partial L(\lambda) / \partial \lambda \), using \( Z(x) \) and forward/backward
4. Update each parameter with gradient search (quasi-Newton)
5. Continue until convergence to global maximum

**Very slow – many iterations of forward/backward**

Using CRFs

**Design**

1. Select feature functions on label pairs \( (Y_i, Y_{i-1}) \) and \( X \).

**Inference**

2. Given weights and feature functions, find the most probable labeling \( Y \) given an input \( X \).

**Learning**

3. Use a training set of data to select the weights, \( \lambda \).

CRF Applications to Gene Prediction

**CRF actually work for gene prediction**

- Culotta, Kulp, McCallum (2005)
- CRAIG (Bernai et al, 2007)
- PhyloCRF (Majores, http://geneprediction.org/book/)

Conrad

- A Semi-Markov CRF
  - Explicit state duration models
  - Features over intervals
- Incorporates diverse information
  - GHMM features
  - Blast
  - EST
- Comparative Data
  - Genome alignments with model of evolution
  - Alignment gaps
- Alternative Objective Function
  - Maximize expected accuracy instead of likelihood during learning
Comparison to Twinscan

- Prediction on *C. neoformans* chr 9
- Compare EST evidence
- Twinscan best previous predictor on this genome

<table>
<thead>
<tr>
<th>Model</th>
<th>Training set</th>
<th>No. of genes</th>
<th>No. of ESTs</th>
<th>Accuracy</th>
<th>Sensitivity</th>
<th>Specificity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Twinscan</td>
<td>200</td>
<td>500</td>
<td>1000</td>
<td>0.85</td>
<td>0.90</td>
<td>0.75</td>
</tr>
<tr>
<td>Twinscan</td>
<td>100</td>
<td>500</td>
<td>1000</td>
<td>0.80</td>
<td>0.85</td>
<td>0.70</td>
</tr>
<tr>
<td>Twinscan</td>
<td>50</td>
<td>500</td>
<td>1000</td>
<td>0.75</td>
<td>0.80</td>
<td>0.65</td>
</tr>
<tr>
<td>Twinscan</td>
<td>25</td>
<td>500</td>
<td>1000</td>
<td>0.70</td>
<td>0.85</td>
<td>0.60</td>
</tr>
</tbody>
</table>

**Conrad Aspergillus Predictions**

Importance of Features Determined *Empirically*

- Branch length
- Gene ontology
- Nucleotide
- CDS
- Introns
- Exons
- Start codons
- Stop codons
- Conserved regions
- Gene expression
- Homology

- Weight 1:
  - Variation
  - Length
  - Resistance

- Weight 2:
  - Gene ontology
  - Nucleotide
  - CDS
  - Introns
  - Exons

- Weight 3:
  - Start codons
  - Stop codons
  - Conserved regions
  - Gene expression
  - Homology