Lecture 11: Gene Prediction

Study Chapter 6.11-6.14

Problem Set #1: Study Session
2/19 from 5pm-7pm in SN011
Gene Prediction: Computational Challenge

• Gene: A sequence of nucleotides coding for protein

• Gene Prediction Problem: Determine the beginning and end positions of genes in a genome
Gene Prediction: Computational Challenge

2/18/15
Comp 555 Spring 2015
Exons and Introns

- In eukaryotes, the gene is a combination of coding segments (exons) that are interrupted by non-coding segments (introns)

- This makes computational gene prediction in eukaryotes even more difficult

- Prokaryotes, one-cell animals without a cell nucleus (i.e. bacteria, archaea) don’t have introns - Genes in prokaryotes are continuous

- What are the hints that a gene is nearby?
Central Dogma and Splicing

exon1 intron1 exon2 intron2 exon3

transcription

splicing

translation

exon = coding
intron = non-coding
• The genome provides subtle hints of where exon/intron boundaries might occur
• The dinucleotides GT and AG on the left- and right-hand sides of an intron are highly conserved. (immediately adjacent to the exons)
Two Approaches to Gene Prediction

- **Statistical**: coding segments (exons) have typical sequences on either end and use different subwords than non-coding segments (introns).

- **Similarity-based**: many human genes are similar to genes in mice, chicken, or even bacteria. Therefore, already known mouse, chicken, and bacterial genes may help to find human genes.
TAA, TAG and TGA correspond to 3 Stop codons that (together with Start codon ATG) delineate Open Reading Frames.
Six Frames in a DNA Sequence

- start codons - ATG
- stop codons – TAA, TAG, TGA
Open Reading Frames (ORFs)

- Detect potential coding regions by looking at **ORFs**
  - A genome of length $n$ is comprised of $(n/3)$ codons
  - Stop codons break genome into segments between consecutive Stop codons
  - The subsegments of these that start from the Start codon (ATG) are ORFs
    - ORFs in different frames may overlap

---

Genomic Sequence

Open reading frame
Long open reading frames may be a gene
- At random, we should expect one stop codon every \((64/3) \approx 21\) codons
- However, genes are usually much longer than this

A basic approach is to scan for ORFs whose length exceeds certain threshold
- This is naïve because some genes (e.g. some neural and immune system genes) are relatively short
Testing ORFs: Codon Usage

• Create a 64-element hash table and count the frequencies of codons in an ORF
• Amino acids typically have more than one codon, but in nature certain codons are used more often
• Uneven use of the codons may characterize a real gene
• This compensates for pitfalls of the ORF length test
## Codon Usage in the Human Genome

<table>
<thead>
<tr>
<th></th>
<th><strong>T</strong></th>
<th><strong>C</strong></th>
<th><strong>A</strong></th>
<th><strong>G</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>T</strong></td>
<td>TTT</td>
<td>Phe</td>
<td>46</td>
<td>TTA</td>
</tr>
<tr>
<td></td>
<td>TTC</td>
<td>Phe</td>
<td>54</td>
<td>TCG</td>
</tr>
<tr>
<td></td>
<td>TTA</td>
<td>Leu</td>
<td>8</td>
<td>TCA</td>
</tr>
<tr>
<td></td>
<td>TTG</td>
<td>Leu</td>
<td>13</td>
<td>TCG</td>
</tr>
<tr>
<td><strong>C</strong></td>
<td>CTT</td>
<td>Leu</td>
<td>13</td>
<td>CCT</td>
</tr>
<tr>
<td></td>
<td>CTC</td>
<td>Leu</td>
<td>20</td>
<td>CCC</td>
</tr>
<tr>
<td></td>
<td>CTA</td>
<td>Leu</td>
<td>7</td>
<td>CCA</td>
</tr>
<tr>
<td></td>
<td>CTG</td>
<td>Leu</td>
<td>40</td>
<td>CCG</td>
</tr>
<tr>
<td><strong>A</strong></td>
<td>ATT</td>
<td>Ile</td>
<td>36</td>
<td>ACT</td>
</tr>
<tr>
<td></td>
<td>ATC</td>
<td>Ile</td>
<td>47</td>
<td>ACC</td>
</tr>
<tr>
<td></td>
<td>ATA</td>
<td>Ile</td>
<td>17</td>
<td>ACA</td>
</tr>
<tr>
<td></td>
<td>ATG</td>
<td>Met</td>
<td>100</td>
<td>ACG</td>
</tr>
<tr>
<td><strong>G</strong></td>
<td>GTT</td>
<td>Val</td>
<td>18</td>
<td>GCT</td>
</tr>
<tr>
<td></td>
<td>GTC</td>
<td>Val</td>
<td>24</td>
<td>GCC</td>
</tr>
<tr>
<td></td>
<td>GTA</td>
<td>Val</td>
<td>12</td>
<td>GCA</td>
</tr>
<tr>
<td></td>
<td>GTG</td>
<td>Val</td>
<td>46</td>
<td>GCG</td>
</tr>
</tbody>
</table>
## Codon Usage in the Mouse Genome

<table>
<thead>
<tr>
<th></th>
<th>T</th>
<th>C</th>
<th>A</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>TTT</td>
<td>TCT</td>
<td>TAT</td>
<td>TGT</td>
</tr>
<tr>
<td></td>
<td>TTC</td>
<td>TCC</td>
<td>TAC</td>
<td>TGC</td>
</tr>
<tr>
<td></td>
<td>TTA</td>
<td>TCA</td>
<td>TAA</td>
<td>TGA</td>
</tr>
<tr>
<td></td>
<td>TTG</td>
<td>TCG</td>
<td>TAG</td>
<td>TGG</td>
</tr>
<tr>
<td>T</td>
<td>CTT</td>
<td>CCT</td>
<td>CAT</td>
<td>CGT</td>
</tr>
<tr>
<td></td>
<td>CTC</td>
<td>CCC</td>
<td>CAC</td>
<td>CGC</td>
</tr>
<tr>
<td></td>
<td>CTA</td>
<td>CCA</td>
<td>CAA</td>
<td>CGA</td>
</tr>
<tr>
<td></td>
<td>CTG</td>
<td>CCG</td>
<td>CAG</td>
<td>CGG</td>
</tr>
<tr>
<td>C</td>
<td>ATT</td>
<td>ACT</td>
<td>AAT</td>
<td>AGT</td>
</tr>
<tr>
<td></td>
<td>ATC</td>
<td>ACC</td>
<td>AAC</td>
<td>AGC</td>
</tr>
<tr>
<td></td>
<td>ATA</td>
<td>ACA</td>
<td>AAA</td>
<td>AGA</td>
</tr>
<tr>
<td></td>
<td>ATG</td>
<td>ACG</td>
<td>AAG</td>
<td>AGG</td>
</tr>
<tr>
<td>A</td>
<td>GTT</td>
<td>GCT</td>
<td>GAT</td>
<td>GGT</td>
</tr>
<tr>
<td></td>
<td>GTC</td>
<td>GCC</td>
<td>GAC</td>
<td>GGC</td>
</tr>
<tr>
<td></td>
<td>GTA</td>
<td>GCA</td>
<td>GAA</td>
<td>GGA</td>
</tr>
<tr>
<td></td>
<td>GTG</td>
<td>GCG</td>
<td>GAG</td>
<td>GGG</td>
</tr>
</tbody>
</table>

- **Phe**: 39, 61
- **Ser**: 16, 21, 21, 6
- **Tyr**: 41, 59, 27
- **Cys**: 45, 55
- **Leu**: 7, 10, 13, 21, 35
- **Val**: 14, 24, 19, 43
- **Ala**: 23, 36, 30, 11
- **Thr**: 24, 28, 40, 8
- **Ile**: 27, 43, 29, 100
- **Pro**: 30, 34, 10, 6
- **Gln**: 76, 49, 29, 15
- **Asn**: 41, 59, 58, 42
- **Glu**: 39, 61, 49, 51
- **Gly**: 17, 32, 29, 15
- **Stop**: 27, 27, 45, 25
An ORF is more “believable” than another if it has more “likely” codons

Do sliding window calculations to find ORFs that have the “likely” codon usage

Allows for higher precision in identifying true ORFs; much better than merely testing for length.

However, average vertebrate exon length is 130 nucleotides, which is often too small to produce reliable peaks in the likelihood ratio

Further improvement: in-frame hexamer count (frequencies of pairs of consecutive codons)
Similarity-Based Approach

- Genes in different organisms are similar
- The similarity-based approach uses known genes in one genome to predict (unknown) genes in another genome
- **Problem:** Given a known gene and an unannotated genome sequence, find a set of substrings of the genomic sequence whose concatenation best fits the gene
  - An alignment problem!
Chaining Local Alignments

• Locate codon substrings that match a given protein subsequence, putative (candidate) exons
• Define a putative exons as 3-tuples \((l, r, w)\)
  \(l = \text{left starting position}\)
  \(r = \text{right ending position}\)
  \(w = \text{weight} \) based on some scoring function
  \(w(\# \text{ of amino acid’s matched, codon freq, ...})\)
• Look for a maximum chain of substrings
  – Chain: a set of non-overlapping nonadjacent intervals.
Exon Chaining Problem

- Locate the beginning and end of each interval ($2n$ points)
- Find the “best” path
Exon Chaining Problem: Formulation

• **Exon Chaining Problem**: Given a set of putative exons, find a maximum set of non-overlapping putative exons

• **Input**: a set of weighted intervals (putative exons)

• **Output**: A maximum chain of intervals from this set

Would a greedy algorithm solve this problem?
Exon Chaining Problem: Graph Representation

- A greedy solution takes the highest scoring chain first, followed by largest one left in the uncovered range, and so on until none can be taken, the score is the sum of all taken chains

- This problem can be solved with dynamic programming in $O(n)$ time, by constructing a graph
Exon Chaining Algorithm

ExonChaining (G, n)  //Graph, number of intervals
1  for i ← to 2n
2     s_i ← 0
3  for i ← 1 to 2n
4     if vertex v_i in G corresponds to right end of the interval I
5         j ← index of vertex for left end of the interval I
6         w ← weight of the interval I
7         s_j ← max {s_j + w, s_{i-1}}
8     else
9         s_i ← s_{i-1}
10    return s_{2n}
Exon Chaining: Deficiencies

- Poor definition of the putative exon endpoints
- Optimal interval chain may not correspond to a valid alignment
  - We enforce genomic order but not protein order
  - First interval may correspond to a suffix, whereas second interval may correspond to a prefix
  - Combination of such intervals is not a valid alignment
Spliced Alignment

- Mikhail Gelfand and colleagues proposed a **spliced alignment** approach of using a protein *within one genome to reconstruct* the exon-intron structure of a *(related) gene in another genome.*
  - Begins by selecting either all putative exons between potential acceptor and donor sites or by finding all substrings similar to the target protein (as in the Exon Chaining Problem).
  - This set is further filtered in a such a way that attempt to retain all true exons, but allow some false ones. *(Many false positives, but no false negatives)*
Spliced Alignment Problem: Formulation

- **Goal**: Find a chain of blocks in a genomic sequence that best fits a target sequence
- **Input**: Genomic sequences $G$, target sequence $T$, and a set of candidate exons $B$.
- **Output**: A chain of exons $\Gamma$ such that the global alignment score between $\Gamma^*$ and $T$ is maximum among all chains of blocks from $B$.

$\Gamma^*$ - concatenation of all exons from chain $\Gamma$
Lewis Carroll Example


IT WAS BRILLIANT THRILLING MORNING AND THE SLIMY HELLISH LITHE DOVES GYRATED AND GAMBED NIMBLY IN THE WAVES
Lewis Carroll Example

Lewis Carroll Example

'Twas brillig, and the slithy toves did gyre and gimble in the wabe.

Twas brillig, and the slithy toves did gyrate and gamble in the wabe.

T'was brillig, and the slithy toves did gyrate nimbly in the wabe.

T'was brillig, and the slithy toves did gyrate nimbly in the wabe.

T'was brillig, and the slithy toves did gyrate nimbly in the wabe.
Exon Chaining vs Spliced Alignment

- In Spliced Alignment, every path spells out a string obtained by concatenating the labels of its vertices.
- The overall path gives an optimal alignment score between concatenated labels (blocks) and target sequence.
- Defines weight as the sum of vertex weights rather than as the sum of edge weights as in Spliced Alignment.
- Exon Chaining assumes the positions and weights of exons are pre-defined.

*How to solve using Dynamic Programming?*
Spliced Alignment: Idea

- Compute the best alignment between $i$-prefix of genomic sequence $G$ and $j$-prefix of target $T$:
  \[ S(i,j) \]

- But what is “$i$-prefix” of $G$?
- There may be a few $i$-prefixes of $G$ depending on which block $B$ we are in.
- Compute the best alignment between $i$-prefix of genomic sequence $G$ and $j$-prefix of target $T$ under the assumption that the alignment uses the block $B$ at position $i$,
  \[ S(i,j,B) \]

- Two cases to consider, block $B$ starts at $i$, or it does not
Spliced Alignment Recurrence

If \( i \) is not the starting vertex of block \( B \):

\[
S(i, j, B) = \max \left\{ \begin{array}{ll}
S(i-1, j, B) - \sigma \\
S(i, j-1, B) - \sigma \\
S(i-1, j-1, B) + \delta(g_i, t_j)
\end{array} \right.
\]

If \( i \) is the starting vertex of block \( B \):

\[
S(i, j, B) = \max \left\{ \begin{array}{ll}
S(i-1, j, B) - \sigma \\
\max_{\text{all blocks } B'} S(\text{end}(B'), j-1, B') - \delta(g_i, t_j) \\
\max_{\text{all blocks } B'} S(\text{end}(B'), j, B') - \sigma
\end{array} \right.
\]

Recall \( \sigma \) was the cost of an indel.
After computing the three-dimensional table $S(i, j, B)$, the score of the optimal spliced alignment is:

$$\max_{\text{all blocks } B} S(\text{end}(B), \text{length}(T), B)$$
• Considering multiple $i$-prefixes leads to slow down. running time:

$$O(mn^2 |B|)$$

where $m$ is the target length, $n$ is the genomic sequence length and $|B|$ is the number of blocks.

• A *mosaic effect*: short exons are easily combined to fit any target protein
Spliced Alignment: Speedup
Spliced Alignment: Speedup

\[ P(i,j) = \max_{\text{all blocks } B \text{ preceding position } i} S(\text{end}(B), j, B) \]