Gene Prediction
Introduction

- **Gene**: A sequence of nucleotides coding for protein
- **Gene Prediction Problem**: Determine the beginning and end positions of genes in a genome

```plaintext
at gc at gc gg gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc t a a g ct g gg at cc g at g a c a at gc at gc
Introduction

- **Gene**: A sequence of nucleotides coding for protein
- **Gene Prediction Problem**: Determine the beginning and end positions of genes in a genome

```
ATGCATGCGCTATGCTAATGCGGCTATGCTAAGCTGGGATCCGATGCATGC
```

```
GCAAAGCTGGGATCCGATGCTAATGCT
```

```
CGCATGCCGTATGCTAATGCT
```

```
GCGCTATGCTAATGCT
```

```
CATGC
```
Introduction

- In 1960’s it was discovered that the sequence of codons in a gene determines the sequence of amino acids in a protein
  - an incorrect assumption: the triplets encoding for amino acid sequences form contiguous strips of information.
- A paradox: genome size of many eukaryotes does not correspond to “genetic complexity”, for example, the salamander genome is 10 times the size of that of human.
- 1977 – discovery of “split” genes: experiments with mRNA of hexon, a viral protein:
  - mRNA-DNA hybrids formed three curious loop structures instead of contiguous duplex segments (seen in an electron microscope)
Central Dogma and Splicing

exon1  intron1  exon2  intron2  exon3

transcription  

splicing  

translation  

exon = coding
intron = non-coding
Gene prediction is hard

- The Genome of many eukaryotes contain only relatively few genes (Human genome 3%).
- Many false splice sites & other signals.
- Very short exons (3bp), especially initial.
- Many very long introns.
- Alternative splicing
A. **Statistical or ab initio methods**: These methods attempt to predict genes based on statistical properties of the given DNA sequence. Programs are e.g. Genscan, GeneID, GENIE and FGENEH.

B. **Comparative methods**: The given DNA string is compared with a similar DNA string from a different species at the appropriate evolutionary distance and genes are predicted in both sequences based on the assumption that exons will be well conserved, whereas introns will not. Programs are e.g. CEM (conserved exon method) and Twinscan.

C. **Homology methods**: The given DNA sequence is compared with known protein structures. Programs are e.g. TBLASTN or TBLASTX, Procrustes and GeneWise.
A. Statistical ab initio methods

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

- E.g. for the bases around the transcription start site we may have the following observed frequencies (given by this position specific weight matrix (PSWM)):

<table>
<thead>
<tr>
<th></th>
<th>-8</th>
<th>-7</th>
<th>-6</th>
<th>-5</th>
<th>-4</th>
<th>-3</th>
<th>-2</th>
<th>-1</th>
<th>+1</th>
<th>+2</th>
<th>+3</th>
<th>+4</th>
<th>+5</th>
<th>+6</th>
<th>+7</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>.16</td>
<td>.29</td>
<td>.20</td>
<td>.25</td>
<td>.22</td>
<td>.66</td>
<td>.27</td>
<td>.15</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>.28</td>
<td>.24</td>
<td>.11</td>
<td>.26</td>
</tr>
<tr>
<td>C</td>
<td>.48</td>
<td>.31</td>
<td>.21</td>
<td>.33</td>
<td>.56</td>
<td>.05</td>
<td>.50</td>
<td>.58</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>.16</td>
<td>.29</td>
<td>.24</td>
<td>.40</td>
</tr>
<tr>
<td>G</td>
<td>.18</td>
<td>.16</td>
<td>.46</td>
<td>.21</td>
<td>.17</td>
<td>.27</td>
<td>.12</td>
<td>.22</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>.48</td>
<td>.20</td>
<td>.45</td>
<td>.21</td>
</tr>
<tr>
<td>T</td>
<td>.19</td>
<td>.24</td>
<td>.14</td>
<td>.21</td>
<td>.06</td>
<td>.02</td>
<td>.11</td>
<td>.05</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>.09</td>
<td>.26</td>
<td>.21</td>
<td>.21</td>
</tr>
</tbody>
</table>

- This can then be used together in a log-likelihood scoring model in order to distinguish certain recognition sites (such as transcription start sites, or promoter regions) from non-recognition sites.
A. Gene prediction in prokaryotes

gene structure

- Most DNA is coding
- No introns
- Promoters are DNA segments upstream of transcripts that initiate transcription

- Promoter *attracts* RNA Polymerase to the transcription start site
A. Gene prediction in prokaryotes

gene structure

- Upstream transcription start site (TSS; position 0) there are promoters
Promoter Structure in Prokaryotes (E.Coli)

Transcription starts at offset 0.
- Pribnow Box (-10)
- Gilbert Box (-30)
- Ribosomal Binding Site (+10)

Promoter structure in prokaryotes

Transcription start site

-30 region
-10 region

TTGACA
AACTGT

TATAAT
ATATTA

Pribnow box

TTGACA  TATAAT

consensus sequences

mRNA

AUG
Open Reading Frames (ORFs)

- Detect potential coding regions by looking at ORFs
  - A genome of length \( n \) is comprised of \( \frac{n}{3} \) codons
  - Stop codons (TAA, TAG or TGA) 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

| ATG | | | | | | TGA |

Open reading frame
```
Six Frames in a DNA Sequence

- stop codons – TAA, TAG, TGA
- start codons – ATG
1. **Evaluation of ORF length**
   - In an “random” DNA, the average distance between stop codons is $\frac{64}{3} \approx 21$ which is much less than the average length of a protein ($\approx 300$)
   - Simple algorithm, poor performance

2. **Evaluation of codon usage**
   - Codon usage in coding regions differs from the codon usage in non-coding regions

3. **Evaluation of codon preference**
   - One amino acid is coded by several different codons, some of them are used more often than the others (see below)

4. **Markov models and HMMs**
2. ORF prediction – codon usage

I. Codon usage (see below)
   - Create a 64-element hash table and count the frequencies of codons in an ORF
   - Uneven use of the codons may characterize a real gene
   - This compensate for pitfalls of the ORF length test

II. Hexamer counts
   - Frequency of occurrences of oligonucleotides of length 6 in a reading frame
   - Usually modeled as fifth-order Hidden Markov Models
     \[ P(x_n=s \mid \cap_{j<n} x_j) = P(x_n=s \mid x_{n-1}x_{n-2}x_{n-3}x_{n-4}x_{n-5}) \]
2. ORF prediction – codon usage

- Vector with 64 components – frequencies of usage for each codon

<table>
<thead>
<tr>
<th>AA</th>
<th>Codon</th>
<th>/1000</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gly</td>
<td>GGG</td>
<td>1.89</td>
</tr>
<tr>
<td>Gly</td>
<td>GGA</td>
<td>0.44</td>
</tr>
<tr>
<td>Gly</td>
<td>GGU</td>
<td>52.99</td>
</tr>
<tr>
<td>Gly</td>
<td>GGC</td>
<td>34.55</td>
</tr>
<tr>
<td>Glu</td>
<td>GAG</td>
<td>15.68</td>
</tr>
<tr>
<td>Glu</td>
<td>GAA</td>
<td>57.20</td>
</tr>
<tr>
<td>Asp</td>
<td>GAU</td>
<td>21.63</td>
</tr>
<tr>
<td>Asp</td>
<td>GAC</td>
<td>43.26</td>
</tr>
</tbody>
</table>
Amino acids typically have more than one codon, but in nature certain codons are more in use.

<table>
<thead>
<tr>
<th></th>
<th>U</th>
<th>C</th>
<th>A</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>UUU</td>
<td>Phe 57</td>
<td>UCU</td>
<td>Ser 16</td>
<td>UAU</td>
</tr>
<tr>
<td>UUC</td>
<td>Phe 43</td>
<td>UCC</td>
<td>Ser 15</td>
<td>UAC</td>
</tr>
<tr>
<td>UUA</td>
<td>Leu 13</td>
<td>UCA</td>
<td>Ser 13</td>
<td>UAA</td>
</tr>
<tr>
<td>UUG</td>
<td>Leu 13</td>
<td>UCG</td>
<td>Ser 15</td>
<td>UAG</td>
</tr>
<tr>
<td>CUU</td>
<td>Leu 11</td>
<td>CCU</td>
<td>Pro 17</td>
<td>CAU</td>
</tr>
<tr>
<td>CUC</td>
<td>Leu 10</td>
<td>CCC</td>
<td>Pro 17</td>
<td>CAC</td>
</tr>
<tr>
<td>CUA</td>
<td>Leu 4</td>
<td>CCA</td>
<td>Pro 20</td>
<td>CAA</td>
</tr>
<tr>
<td>CUG</td>
<td>Leu 49</td>
<td>CCG</td>
<td>Pro 51</td>
<td>CAG</td>
</tr>
<tr>
<td>AUU</td>
<td>Ile 50</td>
<td>ACU</td>
<td>Thr 18</td>
<td>AAU</td>
</tr>
<tr>
<td>AUC</td>
<td>Ile 41</td>
<td>ACC</td>
<td>Thr 42</td>
<td>AAC</td>
</tr>
<tr>
<td>AUA</td>
<td>Ile 9</td>
<td>ACA</td>
<td>Thr 15</td>
<td>AAA</td>
</tr>
<tr>
<td>AUG</td>
<td>Met 100</td>
<td>ACG</td>
<td>Thr 26</td>
<td>AAG</td>
</tr>
<tr>
<td>GUU</td>
<td>Val 27</td>
<td>GCU</td>
<td>Ala 17</td>
<td>GAA</td>
</tr>
<tr>
<td>GUC</td>
<td>Val 21</td>
<td>GCC</td>
<td>Ala 27</td>
<td>GAC</td>
</tr>
<tr>
<td>GUA</td>
<td>Val 16</td>
<td>GCA</td>
<td>Ala 22</td>
<td>GAA</td>
</tr>
<tr>
<td>GUG</td>
<td>Val 36</td>
<td>GCG</td>
<td>Ala 34</td>
<td>GAG</td>
</tr>
</tbody>
</table>
## Codon Usage in Mouse Genome

<table>
<thead>
<tr>
<th>AA</th>
<th>codon</th>
<th>/1000</th>
<th>frac</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ser</td>
<td>TCG</td>
<td>4.31</td>
<td>0.05</td>
</tr>
<tr>
<td>Ser</td>
<td>TCA</td>
<td>11.44</td>
<td>0.14</td>
</tr>
<tr>
<td>Ser</td>
<td>TCT</td>
<td>15.70</td>
<td>0.19</td>
</tr>
<tr>
<td>Ser</td>
<td>TCC</td>
<td>17.92</td>
<td>0.22</td>
</tr>
<tr>
<td>Ser</td>
<td>AGT</td>
<td>12.25</td>
<td>0.15</td>
</tr>
<tr>
<td>Ser</td>
<td>AGC</td>
<td>19.54</td>
<td>0.24</td>
</tr>
<tr>
<td>Pro</td>
<td>CCG</td>
<td>6.33</td>
<td>0.11</td>
</tr>
<tr>
<td>Pro</td>
<td>CCA</td>
<td>17.10</td>
<td>0.28</td>
</tr>
<tr>
<td>Pro</td>
<td>CCT</td>
<td>18.31</td>
<td>0.30</td>
</tr>
<tr>
<td>Pro</td>
<td>CCC</td>
<td>18.42</td>
<td>0.31</td>
</tr>
<tr>
<td>Ala</td>
<td>GCG</td>
<td>6.72</td>
<td>0.10</td>
</tr>
<tr>
<td>Ala</td>
<td>GCA</td>
<td>15.80</td>
<td>0.23</td>
</tr>
<tr>
<td>Ala</td>
<td>GCT</td>
<td>20.12</td>
<td>0.29</td>
</tr>
<tr>
<td>Ala</td>
<td>GCC</td>
<td>26.51</td>
<td>0.38</td>
</tr>
<tr>
<td>Leu</td>
<td>CTG</td>
<td>39.95</td>
<td>0.49</td>
</tr>
<tr>
<td>Leu</td>
<td>CTA</td>
<td>7.89</td>
<td>0.10</td>
</tr>
<tr>
<td>Leu</td>
<td>CTT</td>
<td>12.97</td>
<td>0.16</td>
</tr>
<tr>
<td>Leu</td>
<td>CTC</td>
<td>20.04</td>
<td>0.25</td>
</tr>
<tr>
<td>Gln</td>
<td>CAG</td>
<td>34.18</td>
<td>0.75</td>
</tr>
<tr>
<td>Gln</td>
<td>CAA</td>
<td>11.51</td>
<td>0.25</td>
</tr>
</tbody>
</table>
3. ORF prediction – codon preference

- For each reading frame a codon preference statistics at each position is computed. The statistic is calculated over a window of length $l_w$ ($l_w$ is usually between 25 and 50), where the window is moved along the sequence in increments of three bases, maintaining the reading frame. The magnitude of the codon preference statistic is a measure of the likeness of a particular window of codons to a predetermined preferred usage.
- The statistic is based on the concept of **synonymous codons**. Synonymous codons are those codons specifying the same amino acid.
3. ORF prediction – codon preference

- Example:
  Leucine, Alanine and Tryptophan are coded by 6, 4 and 1 different codons respectively. Hence in a uniformly random DNA they should occur in the ratio 6:4:1. But in a protein they occur in the ratio 6.9:6.5:1.

- For a codon $c$
  
  $f_c$  codon’s frequency of occurrence in the window.
  $F_c$  the total number of occurrences of $c$’s synonymous family in the window.
  $r_c$  the calculated number of occurrences of $c$ in a random sequence of length $l_w$ with the same base composition as the sequence being analyzed.
  $R_c$  the calculated number of occurrences of $c$’s synonymous family in a random sequence of length $l_w$ with the same base composition as the sequence being analyzed.
3. ORF prediction – codon preference

- $f_c$ codon’s frequency of occurrence in the window.
- $F_c$ the total number of occurrences of c’s synonymous family in the window.
- $r_c$ the calculated number of occurrences of c in a random sequence of length $l_w$ with the same base composition as the sequence being analyzed.
- $R_c$ the calculated number of occurrences of c’s synonymous family in a random sequence of length $l_w$ with the same base composition as the sequence being analyzed.

- The codon c preference statistic:
  \[
  p_c = \frac{f_c / F_c}{r_c / R_c}
  \]

- If $p_c = 1$, c is used equally in a random sequence and in the codon frequency table.
3. ORF prediction – codon preference

- When an aminoacid is coded by codons $c_1, c_2, \ldots, c_k$, then obviously
  $$\prod_{j=1}^{k} p_{c_j} \leq 1$$

- The probability of the sequence in the window $w$ is then
  $$P(w) = \prod_{i=1}^{l_w} p_{c_j}$$

- Log-based score is used instead and the codon preference statistics for each window is
  $$P_w = e^{\sum_{i=1}^{l_w} \log p_{c_i}}$$

- A correction: 0 in the codon frequency table is replaced by $1/F_c$
4. ORF prediction – using Markov models and HMM

- There are many more ORFs than real genes. E.g., the E. coli genome contains about 6600 ORFs but only about 4400 real genes. Markov model and an HMM can be used to distinguish between non-coding ORFs and real genes.
- DNA can be modeled as 64-state Markov chain of codons:
  - Probabilities that a certain codon is followed by another one in a coding ORF is computed.
  - Probability of the chain is then computed in the form of log-odds score.
  - Non-coding ORF has log-odds distribution centered around around 0.
A. Gene prediction in eukaryotes

gene structure

- Alternating exons and introns

- Introns start usually by AG and ends by GT

- Types of exons
  1. Initial exons
  2. Internal exons
  3. Terminal exons
  4. Single-exon genes, i.e. genes without introns.
Consensus splice sites – splicing signals

- Try to recognize location of splicing signals at exon-intron junctions
  - This has yielded a weakly conserved donor splice site and acceptor splice site
- Profiles for sites are still weak, and lends the problem to the Hidden Markov Model (HMM) approaches, which capture the statistical dependencies between sites
In the exon-intron junctions there is a large similarity to the consensus sequence → algorithms based on position specific weight matrices.

However, this is far too simple, since it does not use all the information encoded in a gene. Thus more integrated approaches are sought. This naturally leads us to Hidden Markov Models.
A simple HMM $M$ for gene detection

- States are ‘in exon’ and ‘in intron’
- $p$ probability that the process stays ‘in exon’; $1-p$ probability that the process switches into ‘in intron’
- $q$ probability that the process stays ‘in intron’; $1-q$ probability that the process switches into ‘in exon’
- The probability that an exon has length $k$ is
  $$P(\text{exon of length } k \mid M) = p^k (1-p)$$

```plaintext
<table>
<thead>
<tr>
<th>Letter</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0.2</td>
</tr>
<tr>
<td>C</td>
<td>0.3</td>
</tr>
<tr>
<td>G</td>
<td>0.3</td>
</tr>
<tr>
<td>T</td>
<td>0.2</td>
</tr>
</tbody>
</table>
```

DNA Nucleotide Frequencies
A simple HMM $M$ for gene detection

- $p^k (1-p)$ implies geometric distribution which does not correspond to the real distribution of lengths of introns and exons

---

**Graphs:**
- **Introns:** Distribution of intron lengths.
- **Initial Exons:** Distribution of initial exon lengths.
A simple HMM M for gene detection

- $p^k (1-p)$ implies geometric distribution which does not correspond to the real distribution of lengths of introns and exons.
A simple HMM M for gene detection

- If an exon is too short (under 50bp), the spliceosome (enzyme that performs the splicing) has not enough room.
- Exons that are longer than 300 bp are difficult to locate.
- Typical numbers for vertebrates:
  - mean gene length $\approx 30$kb,
  - mean coding region length $\approx 1-2$kb.
- we need other models that can model biological exon lengths
Ribosomal Binding Site

1055 E. coli Ribosome binding sites listed in the Miller book
TestCode

- Statistical test described by James Fickett in 1982: tendency for nucleotides in coding regions to be repeated with periodicity of 3
  - Judges randomness instead of codon frequency.
  - Finds “putative” coding regions, not introns, exons, or splice sites.
- TestCode finds ORFs based on compositional bias with a periodicity of three.
• Define a window size no less than 200 bp, slide the window along the sequence down 3 bases. In each window:
  – Calculate for each base \{A, T, G, C\}
    • \( \text{max} (n_{3k+1}, n_{3k+2}, n_{3k}) / \text{min} (n_{3k+1}, n_{3k+2}, n_{3k}) \)
    • Use these values to obtain a probability from a lookup table (which was a previously defined and determined experimentally with known coding and noncoding sequences)

• Probabilities can be classified as indicative of "coding" or "noncoding" regions, or “no opinion” when it is unclear what level of randomization tolerance a sequence carries

• The resulting sequence of probabilities can be plotted
TestCode Sample Output

Coding
No opinion
Non-coding
Popular Gene Prediction Algorithms

• **GENSCAN**: uses modified Hidden Markov Models (HMMs) – semi-Markov model – based on statistical methods and on data from an annotated training set

• **TWINSCAN**
  – Uses both HMM and similarity (e.g., between human and mouse genomes)
B. Comparative gene finding

• **Idea**: the level of sequence conservation between two species depends on the function of the DNA, e.g. coding sequence is more conserved than intergenic sequence.

• **Program Rosetta**: 
  – first computes a global alignment of two homologous sequences
  – and then attempts to predict genes in both sequences simultaneously.

• **A conserved exon method**: that uses local conservation.

• **Orthologous Genes**: homologous genes in two species that have a common ancestor.
Using Known Genes to Predict New Genes

- Some genomes may be very well-studied, with many genes having been experimentally verified.
- Closely-related organisms may have similar genes.
- Unknown genes in one species may be compared to genes in some closely-related species.
- Most human genes have mouse orthologs:
  - 95% of coding exons are in a one-to-one correspondence between the two genomes.
  - 75% of orthologous coding exons have equal length, and
  - 95% have equal length modulo 3.
  - Intron lengths differ by an average of 50%.
  - The coding sequence similarity between the two organisms is around 85%,
  - the intron sequence similarity is around 35%,
  - 5’ UTRs and 3’ UTRs around 68%.
Similarity-Based Approach to Gene Prediction

- 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

We try to identify small islands of similarity corresponding to similarities between exons
Reverse Translation

- Given a known protein, find a gene in the genome which codes for it.
- One might infer the coding DNA of the given protein by reversing the translation process
  - Inexact: amino acids map to > 1 codon.
  - This problem is essentially reduced to an alignment problem.
- This reverse translation problem can be modeled as traveling in Manhattan grid with free horizontal jumps
  - Complexity of Manhattan is $n^3$
- Every horizontal jump models an insertion of an intron
- Problem with this approach: would match nucleotides pointwise and use horizontal jumps at every opportunity
Comparing Genomic DNA Against mRNA

Portion of genome

exon1  intron1  exon2  intron2  exon3
Using Similarities to Find the Exon Structure

- The known frog gene is aligned to different locations in the human genome
- Find the “best” path to reveal the exon structure of human gene
Bioinformatics Algorithms

Finding Local Alignments

• Use local alignments to find all islands of similarity
Chaining Local Alignments

- Find substrings that match a given gene sequence (candidate exons)
- Define a candidate exons as 
  \[(l, r, w)\]
  (left, right, weight defined as score of local alignment)
- 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 \((2n \text{ points})\)
- 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.
This problem can be solved with dynamic programming in $O(n)$ time.
Exon Chaining Algorithm

**ExonChaining** \((G, n)\)  //Graph, number of intervals

for \(i \leftarrow 1\) to \(2n\)

\[ s_i \leftarrow 0 \]

for \(i \leftarrow 1\) to \(2n\)

if vertex \(v_j\) in \(G\) corresponds to right end of the interval /

\( j \leftarrow \) index of vertex for the left end of the interval /

\( w \leftarrow \) weight of the interval /

\( s_j \leftarrow \max \{s_j + w, s_{i-1}\} \)

else

\( s_i \leftarrow s_{i-1} \)

return \(s_{2n}\)
Exon Chaining: Deficiencies

- Poor definition of the putative exon endpoints.
- Optimal chain of intervals may not correspond to any valid alignment:
  - First interval may correspond to a suffix, whereas second interval may correspond to a prefix.
  - Combination of such intervals is not a valid alignment.
Infeasible Chains

- Red local similarities form two non-overlapping intervals but do not form a valid global 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 such way that attempts to retain all true exons, with some false ones.
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 $B$.

$\Gamma^*$ – the concatenation of all exons from chain $\Gamma$.
Lewis Carroll Example

'TWAS BRILLIANT, AND THE SLITHY TOWERS DID GREY AND GIMBLE IN THE WABE
TWAS BRILLIANT, AND THE SLITHY TOWERS DID GREY AND GIMBLE IN THE WABE
TWAS BRILLIANT, AND THE SLITHY TOWERS DID GREY AND GIMBLE IN THE WABE
TWAS BRILLIANT, AND THE SLITHY TOWERS DID GREY AND GIMBLE IN THE WABE
T'HILLING AND HELISH DOVES GYRATED AND GAMMED IN THE WAVE
T'HILLING AND HELISH DOVES GYRATED AND GAMMED IN THE WAVE
T'HILLING AND HELISH DOVES GYRATED AND GAMMED IN THE WAVE
T'HILLING AND HELISH DOVES GYRATED AND GAMMED IN THE WAVE

Bioinformatics Algorithms
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.
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 $i$
  \[ S(i,j,B) \]
The score of a prefix alignment

- Given a position $i$, let $\Gamma = (B_1, \ldots, B_k, \ldots, B_t)$ be a chain such that some block $B_k$ contains $i$.
- We define

$$\Gamma^*(i) = B_1 * B_2 * \ldots * B_k(i)$$

as the concatenation of $B_1 \ldots B_{k-1}$ and the $i$-prefix of $B_k$.
- Then

$$S(i, j, k) = \max_{\text{all chains } \Gamma \text{ containing block } B_k} s(\Gamma^*(i), T(j)),$$

is the optimal score for aligning a chain of blocks up to position $i$ in $G$ to the $j$-prefix of $T$.
- The values of this matrix can be computed using dynamic programming.
Spliced Alignment Initialization

1. For $j=0$, $S(i, 0, B)$ corresponds to an alignment of blocks in front of $B$ to gaps and $g_{\text{first}(B)} \ldots g_i$ to gaps

$$S(i, 0, B) = \sum_{l=\text{first}(B)}^{i} \text{indel}$$

1. For $j>0$ and when there does not exist a block preceding $B$

$$S(i, j, B) = s(g_{\text{first}(B)} \ldots g_i, t_1 \ldots t_j)$$

- $\text{first}(B)$ = index of the first base of $B$
### Spliced Alignment Recurrence

- If $i$ is not the starting vertex of block $B$:
  \[
  S(i, j, B) = \max \{ 
  S(i - 1, j, B) - \text{indel penalty}, 
  S(i, j - 1, B) - \text{indel penalty}, 
  S(i - 1, j - 1, B) + \delta(g_i, t_j) \} 
  \]

- If $i$ is the starting vertex of block $B$:
  \[
  S(i, j, B) = \max \{ 
  S(i, j - 1, B) - \text{indel penalty}, 
  \max_{\text{all blocks } B'} \ S(\text{end}(B'), j, B') - \text{indel penalty}, 
  \max_{\text{all blocks } B'} \ S(\text{end}(B'), j - 1, B') + \delta(g_i, t_j) \} 
  \]

- After computing the three-dimensional table $S(i, j, B)$, the score of the optimal spliced alignment is:
  \[
  \max_{\text{all blocks } B} S(\text{end}(B), \text{length}(T), B) 
  \]
Spliced Alignment: Complications

• 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
Bioinformatics Algorithms

Spliced Alignment: Speedup

T WAS BRILLIANT AND THRILLING MORNING GAMES CYCLED \nSIMILARLY IN THE WAVES

AND GAMBLED

IN THE WAVES
Spliced Alignment: Speedup

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

- In Spliced Alignment, every path spells out string obtained by concatenation of labels of its edges. The weight of the path is defined as optimal alignment score between concatenated labels (blocks) and target sequence
  - Defines weight of entire path in graph, but not the weights for individual edges.
- Exon Chaining assumes the positions and weights of exons are pre-defined.
Gene Prediction: Aligning Genome vs. Genome

- Align entire human and mouse genomes.

- Predict genes in both sequences simultaneously as chains of aligned blocks (exons).

- This approach does not assume any annotation of either human or mouse genes.
Gene Prediction Tools

- GENSCAN/Genome Scan
- TwinScan
- Glimmer
- GenMark
The GENSCAN Algorithm

- Algorithm is based on probabilistic model of gene structure similar to Hidden Markov Models (HMMs).
- GENSCAN uses a training set in order to estimate the HMM parameters, then the algorithm returns the exon structure using maximum likelihood approach standard to many HMM algorithms (Viterbi algorithm).
  - Biological input: Codon bias in coding regions, gene structure (start and stop codons, typical exon and intron length, presence of promoters, presence of genes on both strands, etc)
  - Covers cases where input sequence contains no gene, partial gene, complete gene, multiple genes.

- GENSCAN limitations:
  - Does not use similarity search to predict genes.
  - Does not address alternative splicing.
  - Could combine two exons from consecutive genes together
Incorporates similarity information into GENSCAN: predicts gene structure which corresponds to maximum probability conditional on similarity information

Algorithm is a combination of two sources of information
- Probabilistic models of exons-introns
- Sequence similarity information
TwinScan

- Aligns two sequences and marks each base as gap (-), mismatch (:), match (|), resulting in a new alphabet of 12 letters: \( \Sigma = \{A-, A:, A|, C-, C:, C|, G-, G:, G|, T-, T:, T|\} \).
- Run Viterbi algorithm using emissions \( e_k(b) \) where \( b \in \Sigma, k \).
- The emission probabilities are estimated from human/mouse gene pairs.
  - Ex. \( e_I(x|) < e_E(x|) \) since matches are favored in exons, and
  \[ e_I(x-) > e_E(x-) \]
  since gaps (as well as mismatches) are favored in introns.
  - Compensates for dominant occurrence of poly-A region in introns
Glimmer

- Gene Locator and Interpolated Markov ModelER
- Finds genes in bacterial DNA
- Uses interpolated Markov Models
- Made of 2 programs
  - **BuildIMM**
    - Takes sequences as input and outputs the Interpolated Markov Models (IMMs)
  - **Glimmer**
    - Takes IMMs and outputs all candidate genes
    - Automatically resolves overlapping genes by choosing one, hence limited
    - Marks “suspected to truly overlap” genes for closer inspection by user
GenMark

- Based on non-stationary Markov chain models
- Results displayed graphically with coding vs. noncoding probability dependent on position in nucleotide sequence