Gene Finding
CMSC 423
Finding Signals in DNA

• We just have a long string of A, C, G, Ts. How can we find the “signals” encoded in it?

• Suppose you encountered a language you didn’t know. How would you decipher it?

• **Idea #1:** Based on some external information, build a model (like an HMM) for how particular features are encoded.

• **Idea #2:** Find patterns that appear more often than you expect by chance. (“the” occurs a lot in English, so it may be a word.)

• Today: we explore methods based mostly on idea #1. Next time, we will explore idea #2.
“Central Dogma” of Biology

DNA =

- double-stranded, linear molecule
- each strand is string over \{A,C,G,T\}
- strands are complements of each other (A \leftrightarrow T; C \leftrightarrow G)
- substrings encode for genes
- most of which encode for proteins
The Genetic Code

- There are 20 different amino acids & 64 different codons.
- Lots of different ways to encode for each amino acid.
- The 3rd base is typically less important for determining the amino acid.
- Three different “stop” codons that signal the end of the gene.
- Start codons differ depending on the organisms, but AUG is often used.
The Gene Finding Problem

• Genes are subsequences of DNA that (generally) tell the cell how to make specific proteins.

• How can we find which subsequences of DNA are genes?

Start Codon: ATG
Stop Codons: TGA, TAG, TAA

Challenges:
The start codon can occur in the middle of a gene.
The stop codon can occur in nonsense DNA between genes.
The stop codon can occur “out of frame” inside a gene.
Don’t know what “phase” the gene starts in.
A Simple Gene Finder

1. Find all stop codons in genome

2. For each stop codon, find the in-frame start codon farthest upstream of the stop codon, without crossing another in-frame stop codon.

   GGC  **TAG**  ATG  AGG  GCT  CTA  ACT  **ATG**  GGC  GCG  **TAA**

   Each substring between the start and stop codons is called an ORF “open reading frame”

3. Return the “long” ORF as predicted genes.

   3 out of the 64 possible codons are stop codons ⇒ in random DNA, every 22nd codon is expected to be a stop.
Gene Finding as a Machine Learning Problem

• Given training examples of some known genes, can we distinguish ORFs that are genes from those that are not?

• **Idea:** can use distribution of codons to find genes.
  • every codon should be about equally likely in non-gene DNA.
  • every organism has a slightly different bias about how often certain codons are preferred.
  • could also use frequencies of longer strings (k-mers).
## Bacillus anthracis (anthrax) codon usage

<table>
<thead>
<tr>
<th>Codon</th>
<th>Functional Amino Acid</th>
<th>Frequency</th>
<th>Codon</th>
<th>Functional Amino Acid</th>
<th>Frequency</th>
<th>Codon</th>
<th>Functional Amino Acid</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>UUU F</td>
<td>0.76</td>
<td>UCU S</td>
<td>0.27</td>
<td>UAU Y</td>
<td>0.77</td>
<td>UGU C</td>
<td>0.73</td>
<td></td>
</tr>
<tr>
<td>UUC F</td>
<td>0.24</td>
<td>UCC S</td>
<td>0.08</td>
<td>UAC Y</td>
<td>0.23</td>
<td>UGC C</td>
<td>0.27</td>
<td></td>
</tr>
<tr>
<td>UUA L</td>
<td>0.49</td>
<td>UCA S</td>
<td>0.23</td>
<td>UAA *</td>
<td>0.66</td>
<td>UGA *</td>
<td>0.14</td>
<td></td>
</tr>
<tr>
<td>UUG L</td>
<td>0.13</td>
<td>UCG S</td>
<td>0.06</td>
<td>UAG *</td>
<td>0.20</td>
<td>UGG W</td>
<td>1.00</td>
<td></td>
</tr>
<tr>
<td>CUU L</td>
<td>0.16</td>
<td>CCU P</td>
<td>0.28</td>
<td>CAU H</td>
<td>0.79</td>
<td>CGU R</td>
<td>0.26</td>
<td></td>
</tr>
<tr>
<td>CUC L</td>
<td>0.04</td>
<td>CCC P</td>
<td>0.07</td>
<td>CAC H</td>
<td>0.21</td>
<td>CGC R</td>
<td>0.06</td>
<td></td>
</tr>
<tr>
<td>CUA L</td>
<td>0.14</td>
<td>CCA P</td>
<td>0.49</td>
<td>CAA Q</td>
<td>0.78</td>
<td>CGA R</td>
<td>0.16</td>
<td></td>
</tr>
<tr>
<td>CUG L</td>
<td>0.05</td>
<td>CCG P</td>
<td>0.16</td>
<td>CAG Q</td>
<td>0.22</td>
<td>CGG R</td>
<td>0.05</td>
<td></td>
</tr>
<tr>
<td>AUU I</td>
<td>0.57</td>
<td>ACU T</td>
<td>0.36</td>
<td>AAU N</td>
<td>0.76</td>
<td>AGU S</td>
<td>0.28</td>
<td></td>
</tr>
<tr>
<td>AUC I</td>
<td>0.15</td>
<td>ACC T</td>
<td>0.08</td>
<td>AAC N</td>
<td>0.24</td>
<td>AGC S</td>
<td>0.08</td>
<td></td>
</tr>
<tr>
<td>AUA I</td>
<td>0.28</td>
<td>ACA T</td>
<td>0.42</td>
<td>AAA K</td>
<td>0.74</td>
<td>AGA R</td>
<td>0.36</td>
<td></td>
</tr>
<tr>
<td>AUG M</td>
<td>1.00</td>
<td>ACG T</td>
<td>0.15</td>
<td>AAG K</td>
<td>0.26</td>
<td>AGG R</td>
<td>0.11</td>
<td></td>
</tr>
<tr>
<td>GUU V</td>
<td>0.32</td>
<td>GCU A</td>
<td>0.34</td>
<td>GAU D</td>
<td>0.81</td>
<td>GGU G</td>
<td>0.30</td>
<td></td>
</tr>
<tr>
<td>GUC V</td>
<td>0.07</td>
<td>GCC A</td>
<td>0.07</td>
<td>GAC D</td>
<td>0.19</td>
<td>GGC G</td>
<td>0.09</td>
<td></td>
</tr>
<tr>
<td>GUA V</td>
<td>0.43</td>
<td>GCA A</td>
<td>0.44</td>
<td>GAA E</td>
<td>0.75</td>
<td>GGA G</td>
<td>0.41</td>
<td></td>
</tr>
<tr>
<td>GUG V</td>
<td>0.18</td>
<td>GCG A</td>
<td>0.15</td>
<td>GAG E</td>
<td>0.25</td>
<td>GGG G</td>
<td>0.20</td>
<td></td>
</tr>
</tbody>
</table>
An Improved Simple Gene Finder

- Score each ORF using the product of the probability of each codon:

\[ \text{GFScore}(g) = \Pr(\text{codon}_1) \times \Pr(\text{codon}_2) \times \Pr(\text{codon}_3) \times \cdots \times \Pr(\text{codon}_n) \]

But: as genes get longer, GFScore\((g)\) will decrease.

So: we should calculate GFScore\((g[i...i+k])\) for some window size \(k\).

The final GFSCORE\((g)\) is the average of the Scores of the windows in it.
Eukaryotic Genes & Exon Splicing

Prokaryotic (bacterial) genes look like this:

Eukaryotic genes usually look like this:

Introns are thrown away

Exons are concatenated together

This spliced RNA is what is translated into a protein.
A (Bad) HMM Eukaryotic Gene Finder

Arrows show transitions with non-zero probabilities

What are some reasons this HMM gene finder is likely to do poorly?
Bad Eukaryotic Gene Finder

- The positions in the codons are treated independently: the probability of emitting a base can’t depend on which previous base was emitted.

- Only one strand of the DNA is considered at once.

- Length distributions of introns and exons are not considered.
An Generalized HMM-based Gene Finder

GlimmerHMM model
An Generalized HMM-based Gene Finder

GlimmerHMM model
## GlimmerHMM Performance

<table>
<thead>
<tr>
<th>Species</th>
<th>Nuc Sens</th>
<th>Nuc Prec</th>
<th>Nuc Accur</th>
<th>Exon Sens</th>
<th>Exon Prec</th>
<th>Exact Genes</th>
<th>Size of test set</th>
</tr>
</thead>
<tbody>
<tr>
<td>D. rerio</td>
<td>93%</td>
<td>78%</td>
<td>86%</td>
<td>77%</td>
<td>69%</td>
<td>24%</td>
<td>549 genes</td>
</tr>
<tr>
<td>C. elegans</td>
<td>96%</td>
<td>95%</td>
<td>96%</td>
<td>82%</td>
<td>81%</td>
<td>42%</td>
<td>1886 genes</td>
</tr>
<tr>
<td>Arabidopsis</td>
<td>97%</td>
<td>99%</td>
<td>98%</td>
<td>84%</td>
<td>89%</td>
<td>60%</td>
<td>809 genes</td>
</tr>
<tr>
<td>Cryptococcus</td>
<td>96%</td>
<td>99%</td>
<td>98%</td>
<td>84%</td>
<td>88%</td>
<td>53%</td>
<td>350 genes</td>
</tr>
<tr>
<td>Coccidioides</td>
<td>99%</td>
<td>99%</td>
<td>99%</td>
<td>84%</td>
<td>86%</td>
<td>60%</td>
<td>503 genes</td>
</tr>
<tr>
<td>Brugia</td>
<td>93%</td>
<td>98%</td>
<td>95%</td>
<td>78%</td>
<td>83%</td>
<td>25%</td>
<td>477 genes</td>
</tr>
</tbody>
</table>

- % of true gene nucleotides that GlimmerHMM predicts as part of genes.
- % of predicted in-gene nucleotides that are correct.
- % of predicted exons that are true exons.
- % of true exons that GlimmerHMM found.
- % of genes perfectly found.
Compare with GENSCAN

- On 963 human genes:

<table>
<thead>
<tr>
<th></th>
<th>Nuc Sens</th>
<th>Nuc Prec</th>
<th>Nuc Acc</th>
<th>Exon Sens</th>
<th>Exon Prec</th>
<th>Exon Acc</th>
<th>Exact Genes</th>
</tr>
</thead>
<tbody>
<tr>
<td>GlimmerHMM</td>
<td>86%</td>
<td>72%</td>
<td>79%</td>
<td>72%</td>
<td>62%</td>
<td>67%</td>
<td>17%</td>
</tr>
<tr>
<td>Genscan</td>
<td>86%</td>
<td>68%</td>
<td>77%</td>
<td>69%</td>
<td>60%</td>
<td>65%</td>
<td>13%</td>
</tr>
</tbody>
</table>

- Note that overall accuracy is pretty low.
Recap

• Simple gene finding approaches use codon bias and long ORFs to identify genes.

• Many top gene finding programs are based on generalizations of Hidden Markov Models.

• Basic HMMs must be generalized to emit variable sized strings.