Base Calling: More Detailed Model

- Various sources of uncertainty in the measured signal
  - phasing, cross-talk, signal decay, residual effects, noise

\[ y_i = \lambda_i K(\Sigma E^T)_{ij} + \alpha_i y_{i-1} + v_i \]

- Note: S is a 4xL matrix, unit base vectors represent A, C, G, T; also, \( E^T = Q \)
- Remark: the problem similar to signal detection in communication systems
  - MIMO channel (cross-talk), intersymbol interference
• Base-calling: given parameters and noisy measurements, find $S$

\[
y_i \sim \mathcal{N}(\lambda_i K(SE^T)_i, \Sigma_i), \quad i = 1
\]

\[
y_i | y_{i-1} \sim \mathcal{N}(\lambda_i K(SE^T)_i + \alpha_i y_{i-1}, \Sigma_i), \quad i = 2, \ldots, N
\]

where $\Sigma_i$ denotes the variance of the multiplicative noise

• Note: for the time being, we are assuming the model parameters are known
EM Algorithm for Parameter Estimation

• Base-calling: given parameters and noisy measurements, find $S$...

\[
y_i \sim \mathcal{N} \left( \lambda_i K(SE^T)_i, \Sigma_i \right), \quad i = 1
\]

\[
y_i \mid y_{i-1} \sim \mathcal{N} \left( \lambda_i K(SE^T)_i + \alpha_i y_{i-1}, \Sigma_i \right), \quad i = 2, \ldots, N
\]

...but how do we find the parameters?

• EM algorithm solves the following optimization problem:

\[
\max_{\Theta} \mathbb{E}_{(\lambda, S)\mid \hat{\Theta}^{(n-1)}} \mathcal{L}(S, \lambda, \Theta)
\]

• Remark: the parameters may vary over cycles
  – reasonable to assume they remain constant over a window of length $W$
Experimental Results

- Error rates based on 75,000 reads of phi X 174 sequenced on GA II

- Bustard: zero-forcing; SMC: sequential Monte Carlo; VA: Viterbi algorithm
Experimental Results Cont’d

• Base-calling error rates and run times (per tile):

<table>
<thead>
<tr>
<th></th>
<th>error rate</th>
<th>run time (min)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bustard</td>
<td>0.0152</td>
<td>2</td>
</tr>
<tr>
<td>naiveBayesCall</td>
<td>0.0132</td>
<td>21</td>
</tr>
<tr>
<td>BayesCall</td>
<td>0.0124</td>
<td>231</td>
</tr>
<tr>
<td>SMC</td>
<td>0.0124</td>
<td>88</td>
</tr>
<tr>
<td>Viterbi algorithm</td>
<td>0.0128</td>
<td>12</td>
</tr>
</tbody>
</table>

• Parameter estimation times:

<table>
<thead>
<tr>
<th></th>
<th>run time (min)</th>
</tr>
</thead>
<tbody>
<tr>
<td>naiveBayesCall</td>
<td>1139</td>
</tr>
<tr>
<td>BayesCall</td>
<td>1139</td>
</tr>
<tr>
<td>SMC</td>
<td>30</td>
</tr>
<tr>
<td>Viterbi algorithm</td>
<td>21</td>
</tr>
</tbody>
</table>
Recall the data processing pipeline:

- Base calling
- Read mapping (alignment)
- Quality score recalibration
- SNP and genotype calling

Read mapping essentially involves alignment of a read onto a reference:
- in reference-free assembly, pairwise alignment of reads may be needed
- sequence alignment is a task encountered in various scenarios (not just in sequence reconstruction)
A large variety of problems in computational genomics involves comparison of sequences (DNA, RNA, protein):

- Reconstructing long sequences of DNA from overlapping sequence fragments (e.g., post-shotgun sequencing)

- Locating similar subsequences on a genome
  - e.g., search for regulatory elements

- Comparing protein sequences in order to determine their functions
  - 25% sequence identity often suffice that two proteins will have very similar 3-D structure / function
• Alignment: A procedure for comparing two or more sequences by searching for a series of individual characters or character patterns that are *in the same order* in the sequences
  
  • Pair-wise alignment: compare two sequences
  
  • Multiple sequence alignment: compare more than two sequences

• Example of a *pair-wise alignment*:
  
  • Input: GCGCATGGATTGAGCGA and TGCGCCATTGATGACCA
  
  • Output: an alignment, say

  -GCGC-ATGGATTGAGCGA
  TGCGCCATTGAT-GACC-A

• However, different alignments for the same sequences are possible:

  -GCGC-ATGGATTGAGCGA           ------GCGCATGGATTGAGCGA
  TGCGCCATTGAT-GACC-A           TGCGCC----ATTGATGACCA--
Sequence Similarity Metrics

Clearly, we need a measure of similarity and difference between sequences to make the task quantitative.

To come up with a “scoring mechanism”, it helps to think in evolutionary terms:

• similar sequences evolved from a common ancestor
• evolution changed the sequences from this common ancestral sequence by mutation
  • **replacement**: one letter is replaced by another
  • **deletion**: deletion of a letter
  • **insertion**: insertion of a letter

• Scoring the sequence similarity: how many and which operations took place
  • fewer operations means more similar sequences
Scoring the Quality of an Alignment

• Going back to our example:
  
  \[
  \begin{align*}
  &\text{AGCTGA} \\
  \rightarrow &\text{ATGGATTTGAGCGA} \\
  \rightarrow &\text{TGCGCCATTGAT-GACC-A}
  \end{align*}
  \]

• Three elements: perfect matches, mismatches, insertions & deletions (indels)

• The simplest model: edit distance
  
  • the smallest number of edit operations (insertions, deletions, substitutions) needed to transform one sequence to another

• Example: Consider sequences **ACCTGA** and **AGCTA**.
  
  0) **ACCTGA**
  1) **AGCTGA** (after substitution)  
  2) **AGCTA** (after deletion)

  So, the edit distance is 2:
  
  \[
  \begin{align*}
  &\text{ACCTGA} \\
  \rightarrow &\text{AGCTA} \\
  \rightarrow &\text{AGCT-A}
  \end{align*}
  \]
Scoring the Quality of an Alignment

- We may associate a score with each “operation” on a letter. For instance:
  - match: +1
  - mismatch: -1
  - indel: -2

- Formalizing the scoring operation:

\[ c : (\Sigma \cup \{-\}) \times (\Sigma \cup \{-\}) \rightarrow \mathcal{R} \]

  - \( c(x,y) \): the score of replacing \( x \) by \( y \)
  - \( c(x,-) \): the score of deleting \( x \)
  - \( c(-,x) \): the score of inserting \( x \)

- Score of an alignment is the sum of position scores

\[ \text{ACCTGA} \]

\[ \text{AGCT-A} \]

- The optimal score: maximal score over all possible alignments
Scoring the Quality of an Alignment

- The optimal score: maximal score over all possible alignments

\[
d(s_1, s_2) = \max_{\text{alignment of } s_1 \text{ and } s_2} \text{score(alignment)}
\]

- Computing the maximal score and finding the best alignment: closely related

- Going back to our first example:

```
-GCGC-ATGGATTGAGCGA
TGCGCCATTGAT-GACC-A
```

Score: \((+1\times 13)+(-1\times 2)+(-2\times 4) = 3\)

```
--------GCGC-ATGGATTGAGCGA
TGCGCC----ATTGATGACCA--
```

Score: \((+1\times 5)+(-1\times 6)+(-2\times 11) = -23\)
There are four basic (pair-wise) alignment problems:

1. **Global alignment**
   - Inputs: Two sequences $s$ and $t$ of roughly the same length
   - Question: What is the maximum similarity between them? Find the best alignment.
     
     \[
     \text{CTCTAGGGATAT} \\
     \text{C--TAG-GA-AT}
     \]

2. **Local alignment**
   - Inputs: Two sequences $s$ and $t$.
   - Question: What is the maximum similarity between a subsequence of $s$ and a subsequence of $t$? Find the most similar subsequences.
     
     \[
     \text{CTCTAGGGATAT} \\
     \text{--CTAGG-A-AT}
     \]
There are four basic (pair-wise) alignment problems:

3. **Ends-free (overlap) alignment**
   - Inputs: Two sequences s and t, possibly of different length.
   - Question: Find the best alignment between subsequences of s and t where at least one of these subsequences is a prefix of the original sequence, and one is a suffix. This alignment favors overlaps.

   \[
   \text{--CAC--CTTGC}
   \]

   \[
   \text{ATCACTCT----}
   \]

   indel operations at the beginning/end: weight 0

4. **Alignment with a gap penalty function**
   - Inputs: Two sequences s and t (possibly of different length).
   - Question: Find the best alignment between the two sequences using the gap penalty function.
   - A **gap penalty function** is a function which measures the cost of a gap as a (possibly nonlinear) function of its length.
There are four basic alignment problems:

1. **Global alignment**
   - Basic global similarity. Essential when designing certain biosensors.

2. **Local alignment**
   - Biologically often far more meaningful than global similarity – especially when long stretches of non-coding DNA are compared, since only small regions within those strings may be related. Likewise for proteins.

3. **Ends-free (overlap) alignment**
   - Useful in the shotgun sequence assembly procedure. Here, a large set of partially overlapping substrings that come from many copies of one original but unknown DNA sequences. Suffix/prefix detection essential.

4. **Alignment with a gap penalty function**
   - Useful when, e.g., comparing mRNA against the genome to reveal exons (since this alignment allows special treatment of long gaps)