Algorithms to be presented...

- DIALIGN2 [Morgenstern, 1999]
  - Aligns diagonals; greedy; for protein or DNA; pairwise or multiple sequence alignments
- AVID [Bray, Dubchak, Pachter, 2003]
  - Using suffix trees to identify local regions of similarity (anchors) and ultimately find the optimal alignment
- BLASTZ [Haussler & Miller groups, 2003]
  - A gapped BLAST implementation for genomes
- LAGAN/Multi-LAGAN [Batzoglou group, 2003]
  - Constrained DP

Graduate Computational Genomics
02-710 / 10-810 & MSCBIO2070

Genomic alignments and profile HMMs

Takis Benos
Lecture #18, March 6, 2008

Reading: handouts & papers
Alignments

- Assumptions
  - Two sequences are related through some evolutionary model
  - Regions of similarity are conserved in order and orientation
  - Multiple alignments: NP-complete
    - Thus heuristics (e.g., progressive alignment)

Main Idea

Genomic regions of interest contain islands of similarity, such as genes

1. Find local alignments
2. Chain an optimal subset of them
3. Refine/complete the alignment

Systems that use this idea to various degrees:
MUMmer, GLASS, DIALIGN, CHAOS, AVID, LAGAN, TBA, & others
DIALIGN2: alignment of alignments

- It is designed to:
  - Align protein or DNA sequences
  - Build multiple alignments in a greedy way, and considering consistency of the aligned segments

DIALIGN2: alignment of alignments

- Basic strategy
  - Calculate all pairwise alignments
  - Assign overlap weights to all diagonals
  - Sort diagonals by their weight
  - Include diagonals one at a time starting from the highest scoring (check for consistency!)
DIALIGN2: example

1. iteration step

weight scores:

<table>
<thead>
<tr>
<th></th>
<th>$D_1$</th>
<th>$D_2$</th>
<th>$D_3$</th>
<th>$D_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>weight</td>
<td>0.2</td>
<td>2.4</td>
<td>4.7</td>
<td>2.2</td>
</tr>
<tr>
<td>overlap weight</td>
<td>0.2</td>
<td>6.3</td>
<td>4.7</td>
<td>4.5</td>
</tr>
</tbody>
</table>

2. iteration step

AVID: a global alignment program for genomic sequences

Figure 1  The AVID algorithm structure.
Alu repeats

AVID: a global alignment program for genomic sequences

Figure 1 The AVID algorithm structure.
AVID: a global alignment program for genomic sequences

![Diagram of AVID algorithm structure]

**Figure 1** The AVID algorithm structure.

Suffix trees (refresh)

T1 = mississippi  
T2 = ississippi  
T3 = ssissippi  
T4 = ssissippi  
T5 = ssissippi  
T6 = ssissippi  
T7 = ssissippi  
T8 = ssissippi  
T9 = ssissippi  
T10 = ssissippi  
T11 = ssissippi  

T11 = i  
T11 = i  
T8 = ippi  
T5 = ippi  
T2 = ippi  
T1 = ippi  
T10 = ippi  
T9 = ippi  
T7 = ippi  
T6 = ippi  
T5 = ippi  
T4 = ippi  
T3 = ippi  
T2 = ippi  
T1 = ippi  
T10 = ippi  
T9 = ippi  
T7 = ippi  
T6 = ippi  
T5 = ippi  
T4 = ippi  
T3 = ippi  

![Suffix tree diagram]

Benos 02-710/MSCBIO2070  6-MAR-2008
**Suffix trees (refresh)**

- Allow many string-related problems to be solved in linear time (at a memory cost):
  - Pattern search: in $O(k)$
  - Longest repeated substring: in $O(n)$ (e.g., "issa" length=4)
  - Longest palindrome: in $O(n)$

---

**AVID: a global alignment program for genomic sequences**

![Diagram of AVID algorithm structure]

1. Concatenate strings (with an 'N' in-between)
2. All maximal matches are the longest repeats that cross the boundary

**Figure 1** The AVID algorithm structure.
AVID: a global alignment program for genomic sequences

1. Use heuristics to eliminate “noisy” matches.
2. Order matches placing clean matches first.
3. Use a variant of SW to select non-overlapping anchors.
4. Fix anchors and repeat for the regions in-between.

Figure 1  The AVID algorithm structure.

BLASTZ: a gapped BLAST algorithm

1. Remove lineage-specific interspersed repeats from both sequences.
2. For all pairs of spaced 12-mers (one from each sequence) that are identical except perhaps for one transition, do the following.
   2.1 Extend the induced alignment in each direction, not allowing gaps. Stop extending when the score decreases more than some threshold.
   2.2 If the gap-free alignment scores more than 3000 (say) then
       2.2.1 Repeat the extension step, but allow for gaps.
       2.2.2 Retain the alignment if it scores above 5000 (say).
3. Between each pair of adjacent alignments from step 2, repeat step 2, but using a more sensitive seeding procedure (e.g., 7-mer exact matches) and lower score thresholds both for gap-free alignments (say, 2000 instead of 3000) and for gapped alignments (say, 2000 instead of 5000).
4. Adjust sequence positions in the resulting alignments to make them refer to the original sequences (i.e., account for step 1).
5. Filter the alignments as appropriate for particular purposes. For many uses we apply axtBest, which finds a best way to align each aligned human position. For other studies, such as mapping segmental duplications, other strategies are appropriate.

Figure 1  BLASTZ in a nutshell.
LAGAN: constrained DP
[Brudno et al., 2003]

1. Find all local alignments
2. Compute a maximal-scoring ordered subset of alignments (chained alignments)
3. Limit the full NW in the restricted area

Evaluation [Brudno et al., 2003]

- Generally difficult: “correct” alignment?
- ROSETTA dataset:
  - 129 human-mouse orthologous gene pairs (with complete introns) of average length 10kb
- CFTR region:
  - 12 orthologous sequences from human, chimp, baboon, cat, dog, cow, pig, mouse, rat, chicken, fugu, zebrafish with average length 1 Mbp (range: 160 Kbp-1.8 Mbp)

<table>
<thead>
<tr>
<th>Aligner</th>
<th>100% exons</th>
<th>90% exons</th>
<th>70% exons</th>
<th>Time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIALIGN</td>
<td>89</td>
<td>96</td>
<td>98</td>
<td>388</td>
</tr>
<tr>
<td>MUMmer</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>17</td>
</tr>
<tr>
<td>GLASS</td>
<td>91</td>
<td>97</td>
<td>98</td>
<td>154</td>
</tr>
<tr>
<td>AVID</td>
<td>90</td>
<td>95</td>
<td>97</td>
<td>19</td>
</tr>
<tr>
<td>BlastZ</td>
<td>94</td>
<td>97</td>
<td>98</td>
<td>17</td>
</tr>
<tr>
<td>LAGAN</td>
<td>94</td>
<td>97</td>
<td>98</td>
<td>48</td>
</tr>
</tbody>
</table>

Columns show the percentage of exons annotated in human that are aligned to the orthologous mouse exon over at least 70%, 90%, and 100% of their length, and the time required to align the 129 sequences.
Evolution [Pollard,Bergman,et al., 2004, BMC Bioinf]

- Constructed artificial datasets based on evolutionary models learned from Drosophila genomic sequences
  - Dataset A: no indels, no constrained blocks
  - Dataset B: indels, no constrained blocks
  - Dataset C: no indels, constrained blocks
  - Dataset D: indels + constrained blocks

### Table 1: Summary of parameters used in simulations of noncoding sequence evolution.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Source</th>
<th>Refs.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequence length</td>
<td>10 Kb</td>
<td></td>
<td>this work, (Fig. 1)</td>
</tr>
<tr>
<td>AT : GC</td>
<td>60: 40</td>
<td></td>
<td>this work, [31,55]</td>
</tr>
<tr>
<td>Transition / Transversion Bias</td>
<td>2</td>
<td>D. mel</td>
<td>[25,56]</td>
</tr>
<tr>
<td>Substitution model</td>
<td>HKY85</td>
<td>D. mel</td>
<td>[54]</td>
</tr>
<tr>
<td>Point substitutions : Indels</td>
<td>10 : 1</td>
<td>D. mel</td>
<td>[22,23,25]</td>
</tr>
<tr>
<td>Indel spectrum</td>
<td>-</td>
<td>D. mel</td>
<td>[57]</td>
</tr>
<tr>
<td>Median constrained block length</td>
<td>18 bp</td>
<td>D. mel vs. D. vir</td>
<td>[23]</td>
</tr>
<tr>
<td>Mean density of constrained blocks</td>
<td>0.2</td>
<td>D. mel vs. D. vir</td>
<td>[23]</td>
</tr>
</tbody>
</table>
Evaluation
[Pollard, Bergman, et al., 2004, BMC Bioinf]

Coverage
Sensitivity

Evaluation (cntd)
[Pollard, Bergman, et al., 2004, BMC Bioinf]
For some reading


Acknowledgements

Some of the slides used in this lecture are adapted or modified slides from lectures of:

- Serafim Batzoglou, Stanford University
Protein Classification

Protein classification

- Number of protein sequences grow exponentially
- Number of solved structures grow exponentially
- Number of new folds identified very small (and close to constant)
- Protein classification can
  - Generate overview of structure types
  - Detect similarities (evolutionary relationships) between protein sequences

<table>
<thead>
<tr>
<th>SCOP release 1.67, Class</th>
<th># folds</th>
<th># superfamilies</th>
<th># families</th>
</tr>
</thead>
<tbody>
<tr>
<td>All alpha proteins</td>
<td>202</td>
<td>342</td>
<td>550</td>
</tr>
<tr>
<td>All beta proteins</td>
<td>141</td>
<td>280</td>
<td>529</td>
</tr>
<tr>
<td>Alpha and beta proteins (α/β)</td>
<td>130</td>
<td>213</td>
<td>593</td>
</tr>
<tr>
<td>Alpha and beta proteins (α+β)</td>
<td>260</td>
<td>386</td>
<td>650</td>
</tr>
<tr>
<td>Multi-domain proteins</td>
<td>40</td>
<td>40</td>
<td>55</td>
</tr>
<tr>
<td>Membrane &amp; cell surface</td>
<td>42</td>
<td>82</td>
<td>91</td>
</tr>
<tr>
<td>Small proteins</td>
<td>72</td>
<td>104</td>
<td>362</td>
</tr>
<tr>
<td>Total</td>
<td>887</td>
<td>1447</td>
<td>2630</td>
</tr>
</tbody>
</table>

Morten Nielsen, CBS, BioCentrum, DTU
Structure Classification Databases

- **SCOP**
  - Manual classification (A. Murzin)
  - scop.berkeley.edu
- **CATH**
  - Semi manual classification (C. Orengo)
  - www.biochem.ucl.ac.uk/bsm/cath
- **FSSP**
  - Automatic classification (L. Holm)
  - www.ebi.ac.uk/dali/fssp/fssp.html

Morten Nielsen, CBS, BioCentrum, DTU

Protein Classification

- Given a new protein, can we place it in its “correct” position within an existing protein hierarchy?

**Methods**

- BLAST / PsiBLAST
- Profile HMMs
- Supervised Machine Learning methods
Profile HMM

| CYB_ASCSU   | HFNGASLFFIFILHLFKGLF----FMSY--RLKK--VWVS |
| CYB6_MARPO  | HRWSASMMVLMMLHIFRVYL----TGGFKKPREI--TWVT   |
| CYB_TRYBB   | HICFTSSLYYLLYIHIFKSITLIIIFDT--IL----VWFI   |

insertion deletion

Profile HMMs

![Diagram of Profile HMMs]

length
Profile HMMs (cntd)

- Each M state has a position-specific pre-computed substitution table
- Each I state has position-specific gap penalties (and in principle can have its own emission distributions)
- Each D state also has position-specific gap penalties
  - In principle, D-D transitions can also be customized per position

Profile HMMs (cntd)

- transition between match states -
- transitions between match and insert states -
- transition within insert state -
- transition between match and delete states -
- transition within delete state -
- emission of amino acid b at a state S -
Profile HMMs (cntd)

1. **Initialization**
   - Choose the HMM length (number of “match” states)
   - Initialize parameters

2. **Training**
   - Estimate the model using Baum-Welch (or Viterbi)

3. **MSA**
   - Align all sequences to the model using Viterbi
   - Build the multiple alignment

transition probabilities ~ frequency of a transition in alignment
emission probabilities ~ frequency of an emission in alignment
pseudocounts are usually introduced

\[
a_{kl} = \frac{A_{kl}}{\sum A_{ki}}
\]

\[
e_k(a) = \frac{E_k(a)}{\sum_a E_k(a')}
\]
Profile HMM (cntd)

- Initial HMM length can be set to the average sequence length
- We need heuristics for the "insert" and "delete" states
  - e.g., "the majority rule"
- Need to avoid local maxima (in Baum-Welch/Viterbi)
  - Use "sensible" transitions (e.g., high "match" transition probabilities)
  - Run Baum-Welch/Viterbi multiple times with different initializations
- Simulated annealing
  - In Viterbi: instead of selecting maxP, you sample the path space according to their probabilities

\[
P(x) = \frac{e^{-E(x)/k_B \cdot T}}{\int_y e^{-E(y)/k_B \cdot T}}
\]

- \( T \to \infty \) All \( x \) equally probable
- \( T \to 0 \) \( P(x) = 0 \)

How to build a profile HMM

SAM-T98 Alignment Building

Start: a single sequence

Build a model from the sequence or alignment

Use the model to search for additional homologs

Reestimate the alignment with the new homologs

(Iterations 1 - 3) (Iteration 4)

End: a SAM-T98 alignment
Resources on the web

- HMMer - a free profile HMM software
  - [http://hmmer.wustl.edu/](http://hmmer.wustl.edu/)

- SAM - another free profile HMM software
  - [http://www.cse.ucsc.edu/research/compbio/sam.html](http://www.cse.ucsc.edu/research/compbio/sam.html)

- PFAM - database of alignments and HMMs for protein families and domains
  - [http://www.sanger.ac.uk/Software/Pfam/](http://www.sanger.ac.uk/Software/Pfam/)

- SCOP - a structural classification of proteins
  - [http://scop.berkeley.edu/data/scop.b.html](http://scop.berkeley.edu/data/scop.b.html)

Classification with Profile HMMs

new protein

fold

superfamily

family
Classification with Profile HMMs

- How generative models work
  - Training examples (sequences known to be members of family): positive
  - Tuning parameters with a priori knowledge
  - Model assigns a probability to any given protein sequence
  - **Idea:** The sequence from the family (hopefully) yield a higher probability than sequences outside the family

- Log-likelihood ratio as score

\[
L(X) = \log \frac{P(X | H_1) P(H_1)}{P(X | H_0) P(H_0)} = \log \frac{P(H_1 | X) P(X)}{P(H_0 | X)}
\]

Acknowledgements

Some of the slides used in this lecture are adapted or modified slides from lectures of:

- Serafim Batzoglou, Stanford University
- Morten Nielsen, CBS, BioCentrum, DTU

Theory and examples from the following: