

Class 06: Gene Alignments (2)

Held: Tuesday, 13 September 2011

Summary: Today we explore two pattern matching algorithms: BLAST and Needleman-Wunsch. We consider the BLAST algorithm in two ways: Through an analysis of the first paper on BLAST and through experiments using an implementation of that algorithm. We visit Needleman-Wunsch by considering the design of the algorithm and by running it "by hand" on sample data.

Related Pages:

- EBoard.
- Reading: Altschul et al. 1990.
- Due: Response to Altschul et al..

Notes:

- Due today: Response to Altschul *et al.*. Email to Praitis and Rebelsky.
- Some general notes on response papers.
- SandB profiles, revisited.
- Today is a very computational day.
- Remember: Project 2.6 is due Thursday.
- Programming as a video game. Level 1: It compiles (reads the source file without complaining). Level 2: It runs without crashing. Level 3: It runs and gives reasonable output on at least one input. Level 3: It runs and gives reasonable output on most inputs. Level 4: It runs and I've assured myself of its correctness with careful testing. Boss Level: I've written a formal proof that it's correct.
- EC for Kington's talk Thursday at 11 a.m.
- EC for DU and company on Robotics in Education, Thursday at 4:30 p.m. in 3821.
- EC for Friday's Biology Seminar: Bowers, "Break on Through (to the Other Side): The Assembly of Outer Membrane Proteins in *Caulobacter crescentus*."
- No home sporting events this weekend? (VB at Mac, FB at Ripon, Soccer at Principia)

Overview:

- The BLAST Paper.
- Exploring the BLAST algorithm.
- Needleman-Wunsch - The Basics.
- Using Dynamic Programming.
- An NW Example.
- Web Exporation.

Having a BLAST

Our goal is to “tease apart” the work represented by the BLAST paper.

Some Standard Questions

- What is the problem domain that the paper addresses?
- What are the authors’ primary claims about their work?
- What is the structure of the paper? (Logical, rhetorical, ...)
- Why are we reading this paper? What is important about this paper?
- What critiques do you have of the paper?

Approximate Matching and Searching

- The problem seems to be finding all “reasonable” matches of one sequence in a larger sequence or collection of sequences.
- What measurements might we use for “reasonable”?
- What is an MSP (maximal segment pair)?
- What is a locally maximal segment pair?
- Why do we care about MSPs?
- What is an obvious way to find an MSP for two sequences?
- What alternatives are there?

The BLAST Algorithm

- What are the parameters to the algorithm? (Letters, meanings, ...)
- What is the overall structure of the BLAST algorithm? (The authors claim that the algorithm has three stages.)
 - Stage 1:
 - Stage 2:
 - Stage 3:
- How do they accomplish each stage?
- What is the running time of the algorithm?

Exploring the Algorithm

- Helpful tool: PAM Matrix Calculator
 - PAM 250 is available in our Examples directory
- Input sequence: AMANAPLANPANAMA
- Database: ZZZZZZZZMANNAFLANNANANAXXXXXXX
- How do we generate the word list from the input sequence?
- Suppose our word list only contains PLAN and FLAN (everything else is too high frequency).
 - How do we search the database?

- Suppose we've matched the FLAN (a variant of the middle PLAN with high score)
 - How do we expand the word match to an approximate MSP?

Analyzing the BLAST Algorithm

- What kinds of analyses are they doing?
- Why are they doing these analyses?
- What do the analyses suggest?
- What are the potential drawbacks of using BLAST?

The Needleman-Wunsch Algorithm

- Problem domain: Finding the best match between two strings by aligning individual components.
 - With particular costs for insertion, deletion, and mutation
 - Basic value function:
 - Insertion has a value of -1
 - Deletion has a value of -1
 - Mutation has a value of 0
 - Exact match has a value of 1
 - We can use others
- Input to algorithm
 - Sequence 1 (which we might call the *database sequence*)
 - Sequence 2 (which we might call the *search sequence*)
 - The value function (which we'll leave implicit)
- Basic idea: Recursion
 - Given two sequences, simplify the sequences, find the best matches of the simplified sequences, and extend to the current sequences.
 - We'll work from right to left.
- Very general example (not worked all the way through)
 - Database: CACGTAT
 - Search: CGCA
- Three ways to align
 - We can pair the last value in each sequence and then align the rest of each sequence.
 - Pair T with A
 - Align CACGTA to CGC.
 - We can drop the last value in the database (corresponding to a deletion).
 - Accumulate a penalty for the deletion
 - Align CACGTA to CGCA.
 - We can drop the last value in the search string (corresponding to an insertion).
 - Accumulate a penalty for the insertion
 - Align CACGTAT to CGC
- Which do we choose?
 - *Whichever gives us the highest value alignment!*

- In pseudo code:

```
def valueOfBestAlignment(seq1, seq2):
    valMat = valueOfMatch(last(seq1), last(seq2)) +
              valueOfBestAlignment(dropLast(seq1), dropLast(seq2))
    valDel = deletionCost +
              valueOfBestAlignment(dropLast(seq1), seq2)
    valIns = insertionCost +
              valueOfBestAlignment(seq, dropLast(seq2))
    return max(valMat, valDel, valIns)
```

- Of course, we should handle the cases in which one of the sequences is empty.

```
if seq1.isEmpty()
    return len(seq2) * values.insertion
elif seq2.isEmpty()
    return len(seq1) * values.deletion
else
    ...
```

- But that much recursion is expensive!
 - For two strings of length n , this is somewhere between 2^n and 3^n steps.
 - For two strings of length 20, it's more than a million steps
 - For two strings of length 40, it's more than a trillion steps
 - Twenty minutes on a gigaflop machine

Dynamic Programming

- Solution: Build a table of the values
 - The value at column c , row r is the value of the best match between the first c values in the database and the first r values in the search
 - The idea of putting all the values for intermediate computations in the table is called *dynamic programming*
- Row 0 of the table shows the cost of matching the empty string against the database string.
- Column 0 of the table shows the cost of matching the search string against the empty string.

Beyond Scoring

- But we care about more than the score of an alignment.
- We also care about the way we get that alignment.
- The book suggests that we build a table that describes how to get the alignment after we've built the scoring table.
- Traditionally, we build the alignment table at the same time as we build the scoring table.
- We then read the alignment strategy off of the alignment table, starting at the lower-right corner.

An Example of NW

- We'll work on an example together.
- Note that Excel (or Gnumeric, in MathLAN) can be very helpful.
- We'll use the same example as St. Clair and Visick, partially because it makes it easier to check our work.
- Sequence 1 = CACGTAT
- Sequence 2 = CGCA
- We'll do it by hand for a bit
- But a general rule is particularly helpful, because we can just copy and paste.

Web Exploration

- Let's all have fun with the Web exploration of BLAST.

Copyright © 2009-2011 Vida Praitis and Samuel A. Rebelsky. This work is licensed under a Creative Commons Attribution-NonCommercial 2.5 License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-nc/2.5/> or send a letter to Creative Commons, 543 Howard Street, 5th Floor, San Francisco, California, 94105, USA.