CSE 584A Class 19

Jeremy Buhler

April 4, 2016

1 Filtering by Dynamic Programming Alignment

Each candidate indicates that we should search the DP matrix around it for high-scoring local alignments. What are issues in designing this search algorithm?

- Where do we start the alignment?
- Where/how do we stop the alignment? (Do we let it go to the end of the matrix?)
- **Shadowing**: an alignment unrelated to the candidate may score higher than the best one passing through the candidate.
- We say that the higher-scoring alignment *shadows* the alignment we really want. We might never see the feature represented by the candidate.

I will first describe an approach to local DP extension implemented in NCBI BLAST.

- Suppose word match is centered on bases $S[i], T[j]$ in two sequences S and T.
- will find optimal local alignment passing through cell (i, j) in matrix.

- Addresses shadowing: unrelated alignments probably do not pass through this candidate.

Idea: will form optimal "pinned" local alignment by joining two *singly pinned alignments*.

1. Compute optimal *forward* alignment A_f from cell (i, j) to some cell (x, y), for $x \geq i$, $y \geq j$.
2. Compute optimal *backward* alignment A_b from some cell (z, w) to cell (i, j), for $z \leq i$, $w \leq j$.

3. Final optimal alignment is $A_b \cdot A_f$.

How do we find the two singly pinned alignments?

- "singly-pinned" means starting point fixed, same initial row
- However, alignment can terminate at any cell in matrix (remember the best one)
- for backward problem, can either reverse sequences and use forward algorithm, or write specialized backward algorithm (reverse the DP recurrence)

2 X-Drops: How to Stop Aligning

- Joining two singly-pinned alignments lets us start from anywhere in the DP matrix, but it's still expensive.
- On average (over uniform choice of seed location), fills in fully *half* as many cells as full DP.

- This work is redone **for every candidate!!!**
- How can we reduce the amount of work per candidate?

Main idea: stop extending "unpromising alignment paths"

- An "unpromising path" has a score so low that it is unlikely to extend into an optimal alignment
- X-drop defines "unpromising:" if for some x, best alignment path to cell (a, b) has score at least x less than optimal alignment found so far, do not extend paths from (a, b) **(picture)**

- x is tunable constant, say 20-40 times average penalty per cell.
- Smaller x causes more aggressive pruning of alignment but is more likely to miss alignments that are locally bad but globally good.

Justifications for x-drop alignment

1. path would have to continue w/ extension of score at least $+x$ to compete w/opt so far
2. we judge this to be unlikely / not worth effort to find
3. if such a segment exists, it would be significant all by itself (and so should not be connected to low-scoring prefix)
4. don't want to link unrelated features in one alignment. Allowing a big drop in middle of alignment can cause two features to be joined into one.

How to cut off alignments using x-drops?

- keep track of optimal alignment score σ^* seen so far (always at least zero)
- as each matrix cell (a, b) is computed, if score$(a, b) \leq \sigma^* - x$, set (a, b)'s value to $-\infty$
- a cell with score $-\infty$ is never used to extend an alignment
- if every cell in a row becomes $-\infty$, stop after that row – all further rows will also be $-\infty$.

Idea is a good one – if every alignment path is rotten, extension should stop after just a few rows. However, each row may be very long. Can we do better? YES!

- suppose all cells $(i-1, j)$ for $j > j^*$ are $-\infty$
- suppose we find that score$(i, j') = -\infty$ for some $j' > j^*$
- **Claim**: all cells to right of (i, j') in row i will also be $-\infty$

- **Conclusion**: can stop computing row i at cell (i, j')
- remember last finite cell in row $i-1$ when computing row i

Same idea works at beginning of each row.

- suppose all cells $(i-1, k)$ for $k < k^*$ are $-\infty$
- what is first cell in row i that could have finite score?
- **Claim**: first such cell is (i, k^*)

- **Conclusion**: can start computing row i at cell (i, k^*)
- **hack**: remember first finite cell in row $i-1$ when computing row i

Alignment stops when we reach a row with no finite cells (or run out of rows.

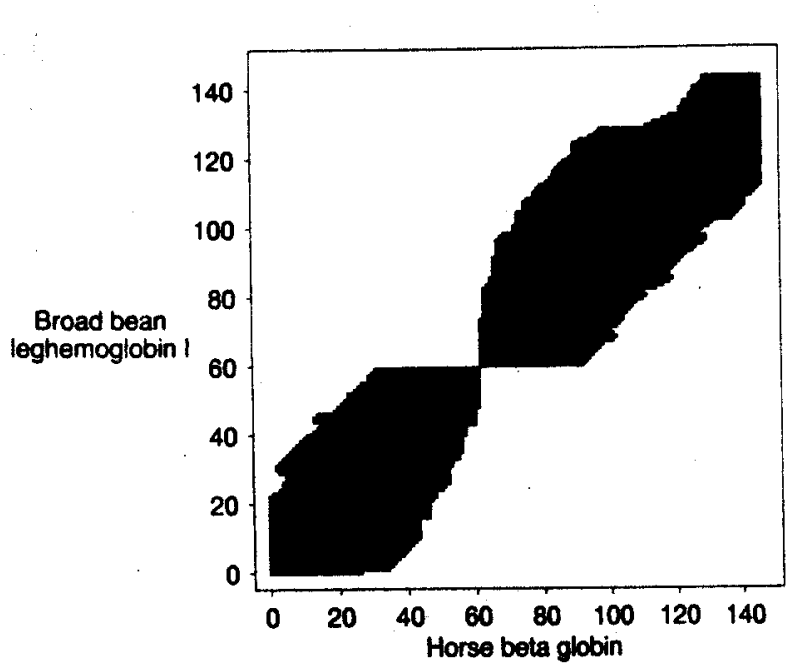
Tying all these hacks together, we get the following modified Smith-Waterman algorithm with X-dropping.

```
ALIGN(i_0, j_0, score fcn, x)
    init matrix row i_0 while score(i_0, j) > −x
    σ* ← 0
    k* ← j_0
    j* ← last finite cell on row i_0

    for each row i from i_0 + 1 to end do
        for each column j from k* to end of row do
            compute score(i, j)
            update σ* if needed
            if score(i, j) ≤ σ* − x
                score(i, j) ← −∞
                if j > j*
                    break                                   ▷ end of row i
        if no finite cells on row i
            break
        else
            set k*, j* to first, last finite cols of row i

    return best alignment score, endpoint
```

(Example from NCBI BLAST): below is a typical region explored by the alignment procedure. (Matrix is inverted top-to-bottom versus what we usually draw). Note that k^* increases monotonically with row, while j^* can vary in either direction.



3 Good and Bad Points about BLAST-style Extension

Good:

- no restriction on shape of alignment paths
- in particular, handles arbitrarily large gaps
- provides *both* endpoints of alignment in linear space (because of semi-global extension)

Bad:

- forces alignment to go through one particular base pair
- difficult to say what is complete set of explored paths in presence of x-drops
- "it's a hack" – some people who should know are suspicious of it

Is there an alternate approach?

4 Banded Smith-Waterman

A more traditional approach to speeding up gapped alignment is *banded Smith-Waterman.*

- a **band** is a contiguous set of diagonals

- **bandwidth**: # of diagonals in band
- **bandlength**: number of antidiagonals (alt. rows) in the band
- *idea*: only find alignments that are confined to the band

Why band?

- alignment paths only cross diagonals because of gaps
- no path can cross more than b diagonals without turning back
- in practice, **no large gaps** allowed
- In practice:
  - DNA: alignments do not cross long introns, do not link features separated by variable-length junk
  - Protein: alignments may not cross spacer regions inside protein, or (for six-frame translation) intergenic regions between proteins

How can we confine Smith-Waterman to a given band?

- **Rule 1**: alignments may not leave the band
- Implementation: in each row, only fill in cells inside the band
- **Rule 2**: alignments may not enter the band from outside
- Implementation: if row i runs from cell j_s to cell j_e, set columns $j_s - 1$ and $j_e + 1$ on that row to $-\infty$

What does banded alignment cost?

- max # of cells filled proportional to bandwidth times bandlength
- If bandwidth is $\Theta(1)$, cost is $\Theta(m+n)$
- typical bandwidth: 15-101 (usually odd)
- can speed up by restricting the bandlength explored

How can we apply this algorithm locally (restricted bandlength?)

- **BLAST Option**: fixed center point and X-drop cutoff
- (Don't really need to restrict cells filled per row for small bands, so just cut off if a whole row becomes $-\infty$.)
- **CrossMatch**: find optimal local alignment *anywhere* in band
- advantage: no fixed center, no cutoffs
- problem: if bandlength very long, can have shadowing problem
- Crossmatch recursively subdivides band to find suboptimal alignments – very expensive if bandlength is large

Here's another way to restrict bandlength.

- find optimal alignment for progressively longer bandlengths
- if increasing bandlength fails to improve alignment, STOP
- inexpensive option: double bandlength each time **(picture)**

- costs at most twice as much as final alignment

How to avoid *shadowing problem* w/o CrossMatch approach?

- banding keeps alignment close to diagonal of center point
- *Idea*: force alignment to pass through center's *row*

- implementation: fill band as usual, but no alignment may start below the center row (drop 0 case from recurrence)
- one tricky bit: initialization (if needed) below center