CSE 584A Class 16

Jeremy Buhler

March 26, 2018

1 Space Costs Matter

- Space costs of alignment are quite serious!
- Naively, aligning seqs of lengths m, n takes $\Theta(mn)$ space.
- If $n = 10000$, $m = 10000$, DP matrix takes 100 million cells!
- Hopefully, we can do better.

If we are just computing M_{ij} values (not alignments), we can use only $O(\min(m, n))$ space.

- WLOG, suppose smaller dimension of DP matrix is its rows.
- When filling in row i, need contents of row $i-1$ plus anything already computed in row i.
- Earlier rows can be discarded after use!
- (For local alignment, we must keep running max over all cells ever computed, but that's easy.)

Unfortunately, we often want the alignment as well as its score. If we do linear-space alignment, we cannot simply trace a path of back pointers! What to do?

2 Linear-Space Alignment Reconstruction

Following algorithm for reconstructing *one global alignment* is due to Hirschberg.

- **Idea**: divide-and-conquer strategy to reconstruct optimal alignment path.
- Consider an optimal global alignment A of sequences with lengths n, m.
- Path for A passes through some cell $(n/2, k^*)$ in middle row $n/2$ of DP matrix.
- **Defn**: Let $M^r_{i,j}$ be the score of an optimal alignment connecting cell i, j to cell m, n of DP matrix.
- (Equivalently, best alignment of suffixes $S[i+1..n]$ and $T[j+1..m]$.)

- **Picture**:

- **Claim**: $M_{n,m} = M_{n/2,k^*} + M^r_{n/2,k^*}$.
- **Proof**: can divide optimal alignment path from $0,0$ to n,m into part $0,0$ to $n/2,k^*$ and part from $n/2,k^*$ to n,m.
- If either part were suboptimal, we could replace it with a better part, and hence improve the overall alignment score.
- Conclude that both parts are optimal, which proves the claim. QED

How can we use linear-space DP algorithm to find k^*?

- Using standard (forward) algo, compute $M_{n/2,k}$ for all k.
- Using *backward* version of standard algo, compute $M^r_{n/2,k}$ for all k.
- Compute
$$k^* = \text{argmax}_k \left(M_{n/2,k} + M^r_{n/2,k} \right).$$
- **Picture**:

- Now we know that alignment passes through cell $n/2,k^*$. Now what?
- By construction, optimal alignment A connects consists of two parts:
  1. A_1, an optimal alignment from $0,0$ to $n/2,k^*$.
  2. A_2, an optimal alignment from $n/2,k^*$ to n,m.

- To get entire alignment, recursively reconstruct A_1 and A_2, then return path $A_1 \cdot A_2$.

To summarize in pseudocode...

DCAlign(S, T) ▷ sizes n, m
  if $n \leq 1$
    return explicit alignment of S, T
  else
    Compute $M_{n/2,k}$ for $1 \leq k \leq m$
    Compute $M^r_{n/2,k}$ for $1 \leq k \leq m$
    Compute $k^* = \text{argmax}_k \left(M_{n/2,k} + M^r_{n/2,k} \right)$

    $A_1 \leftarrow$ DCAlign($S[1..n/2]$, $T[1..k^*]$)
    $A_2 \leftarrow$ DCAlign($S[n/2+1..n]$, $T[k^*+1..m]$)
    return $A_1 \cdot A_2$

Note that we need never keep more than three DP matrix rows in memory at any time (two for backward algo, plus one with result of forward algo)!

3 Running Time

Will show that cost is asymptotically no worse than just computing the optimal alignment score.

- Let $W(n, m)$ be the cost of running algo on sequences of sizes n and m.
- For sequences of these lengths, suppose the total cost of finding the best center crossing k^* be at most $c \cdot nm$.
- Observe that we can compute the score of an optimal alignment of the input sequences directly with cost at most $c \cdot nm$, since finding k^* fills in at least as many DP cells as the direct computation.
- **Claim**: $\forall n, m, W(n, m) \leq 2c \cdot nm$
- **Proof**: by induction on n.
- **Bas**: If $n = 1$, we compute an optimal alignment of the sequences directly in time at most $cm \leq 2cm$ (by assumption). An optimal alignment path can be found trivially during alignment.
- **Ind**: After we find k^*, we are left with two recursive calls on non-overlapping parts of the sequences.
- Each call gets a chunk of S of size $n/2$.
- One call gets a chunk of T of size k^*; the other gets a chunk of size $m - k^*$.
- Hence, we have that

$$T(n, m) \leq c \cdot nm + \max_k \left[T\left(\frac{n}{2}, k\right) + T\left(\frac{n}{2}, m - k\right) \right].$$

- Plugging in the inductive hypothesis,

$$\begin{aligned} T(n,m) &\leq c \cdot nm + \max_k \left[2c\frac{n}{2}k + 2c\frac{n}{2}(m-k) \right] \\ &= cnm + \max_k \left[2c\frac{n}{2}(k+m-k) \right] \\ &= cnm + cnm \\ &= 2cnm. \end{aligned}$$

- Hence, the IH is proved. QED

Hence, an optimal global alignment can be reconstructed in linear space and time $\Theta(nm)$.

4 Extension to Local Alignment

Does Hirschberg's algorithm extend easily to local case? Yes!

- Obviously, compute L_{ij} and L^r_{ij}.
- Best local alignment might not cross row $n/2$.
- Keep track of highest alignment score entirely in top half or entirely in bottom half of matrix.
- If (say) best top half score is better than best score crossing middle row, recur only on top half. (Ditto bottom half).

Alternatively, it's easy to extend Smith-Waterman to compute the start as well as the end of an alignment in linear space, then use these two endpoints to "pin" the alignment and use Hirschberg's global algorithm to do traceback between the endpoints.