Genomics Mid2

tyang27

September 2019

1 Global Alignment

Edit distance (Special case)

Edit distance $\leq$ hamming distance, since edit distance includes hamming distance, but can do better if shifts.

Weight function

	A	G	T	C	X
A	0	1	1	1	1
G	1	0	1	1	1
T	1	1	0	1	1
C	1	1	1	0	1
X	1	1	1	1	

Global Alignment

Weight function

Penalties are positive, or positive for matches and negative for everything else (but need to change from min to max).

Minimize:

	A	G	T	C	X
A	0	+	+	+	+
G	+	0	+	+	+
T	+	+	0	+	+
C	+	+	+	0	+
X	+	+	+	+	

Maximize:

	A	G	T	C	X
A	+	−	−	−	−
G	−	+	−	−	−
T	−	−	+	−	−
C	−	−	−	+	−
X	−	−	−	−	

Algorithm

```
global(s1, s2):
    dp = [0] * (|s1| * |s2|)
    dp[0,j] = j
    dp[i,0] = i
    dp[i,j]=min/max(
        dp[i-1,j] + delta(s1[i-1], X),
        dp[i,j-1] + delta(X, s2[j-1]),
```

```
        dp[i-1,j-1] + delta(s1[i-1], s2[j-1])
        )
    return min/max(dp[|s1|,:])
```

For traceback, stop when you get to top right.

2 Approximate matching

Weight function

	A	G	T	C	X
A	0	1	1	1	1
G	1	0	1	1	1
T	1	1	0	1	1
C	1	1	1	0	1
X	1	1	1	1	

Algorithm

```
approximate(s1, s2):
    dp = [0] * (|s1| * |s2|)
    dp[i,0] = i
    dp[i,j] = min(
        dp[i-1,j] + delta(s1[i-1], X),
        dp[i,j-1] + delta(X, s2[j-1]),
        dp[i-1,j-1] + delta(s1[i-1], s2[j-1])
        )
    return min(dp[-1,:])
```

For traceback, get to the top.

3 Smith Waterman for local alignment

Greatest global alignment between substrings.

Weight function

Needs to be positive matches and negative non-matches, since othwerise our cap of 0 won't work. Non-matches can't be 0, bc then we would have more solutions to trace back from when really there are fewer.

	A	G	T	C	X
A	+	−	−	−	−
G	−	+	−	−	−
T	−	−	+	−	−
C	−	−	−	+	−
X	−	−	−	−	

Algorithm

```
local(s1, s2):
    dp = [0] * (|s1| * |s2|)
    dp[i,0] = dp[i-1,0] + delta(s1[i-1], X)
    dp[0,j] = dp[0,j-1] + delta(X, s2[j-1])
    dp[i,j] = max(
        dp[i-1,j] + delta(s1[i-1], X),
        dp[i,j-1] + delta(X, s2[j-1]),
        dp[i-1,j-1] + delta(s1[i-1], s2[j-1]),
```

```
        0
        )
    return max(dp[:,:])
```

For traceback, stop when you get to 0.

4 Misc for DP

Edit distance

$O((m+1)(n+1)) = O(mn)$ table filling
$O(m+n)$ traceback

Tips for finding weight function

Transitions: AG, CT

- Determine how much gaps cost.
- Look at the diagonals. If the characters are the same, you can get the match cost. If not, it might be a transversion or transition, but you need to first see if it matches the gap cost. If it does not match the gap cost, it must have come from the diagonal.

5 Laws of assembly

1. If suffix of A is a prefix of B, might be an overlap in the genome. Might be approximate due to ploidy and sequencing error.
2. More coverage leads to more and longer overlaps.
3. Repeats make assembly difficult, but being able to faithfully recreate genome depends on repetitive patterns and read length.

6 Overlap graph

Nodes are all k-mer reads. Edges are labelled with length of overlaps greater than a value. An overlap is when prefix of one matches suffix of another.

1. Suffix tree, $A\$_0B\$_1$. $O(n+d^2)$.
2. Local alignment, with infinity in the top row, and 0's in the first column. This makes it so that you can start from any place in the suffix string, but you must start from the start of the prefix string. Solution is in last row, find above threshold within certain edit. $O(nm)$.

7 Shortest common superstring

1. NP-hard, will need to rely on heuristics (greedy) that might not be optimal.
2. Collapses repeats.

8 De Brujin Graph

Algorithm

1. Split k-mers into left and right k-1-mers.
2. Draw arrows between k-1-mers that come from the same k-mer. One node per distinct k-1-mer.

Improving space bound

One edge per k-mer O(N), or alternatively, one edge per distinct k-mer with labeled count O(min(N,G)). Good for when high coverage.

Eulerian

Directed, connected graph is eulerian if

1. Exists eulerian walk
2. At most two semi-balanced nodes and rest are balanced. If all balanced, eulerian walk is cycle. Semi-balanced if indegree and outdegree differ by one. Balanced if indegree and outdegree equal.

Perfect de brujin graph is eulerian because each k-mer split into left and right k-1-mers. A k-1-mer will occur in a left and right k-1-mer for different k-mers, so these will be balanced. The ends will be semibalanced, since the start will not appear in a right k-1-mer, and the end will not appear in a left k-1-mer.

Disadvantages

- Uses coverage to solve some repeat issues, but still some ambiguity issues. Can still mess up some ordering/cycle/repeats e.g. xAxBx, xBxAx.
- Imperfect sequencing:
  - Coverage gaps - disconnected graph
  - Coverage difference - semi-balanced nodes
  - Hereditary differences or sequence errors will also create semi-balanced nodes or disconnected graphs.
- Short, exact overlaps, so lose ability to resolve some repeats.
- Errors have more profound impact on shape of graph.
- Loss of read coherence, since might make some new paths that do not exist.

9 Overlap-Layout-Consensus

1. Overlap: Use generalized suffix tree ($X\$_0Y\$_1$) or dynamic programming (A suffix first column is 0, B prefix first row is infinity) find overlaps.
2. Layout: With approximate matching, remove transitively inferrable matches until nonresolveable repeats left. Output the nonbranching regions. Prune out some dead ends.
3. Consensus: Take a majority vote for what the nucleotide should be.

10 De Brujin Graph

1. Error correction: Idea is that errors will make a frequent read into an infrequent read. For each k-mer in each read, if it occurs less than a threshold, check within the neighborhood (within hamming or edit distance) of that string, and replace it with a more frequent one. An issue with this is cold start – at first, no reads will pass the threshold. In addition, need to tune hyperparamaters.
2. Create De Brujin Graph.
3. Refine - remove tips and islands, summarize bubbles (alleles).

11 Scaffolding

Use pair-end reads to order contigs. Paired-end reads also tell you how far apart contigs are.