Lecture 7:
Weighted linked lists

• Applications (via sequence graphs):
  – regions of atypical residue composition
  – motif clusters
  – read count data

• Finding multiple high-scoring paths

• “D-segments”

• Statistical significance
Weighted Linked Lists (WLLs)

- **WLL** is a linked list with weights on each edge – simplest kind of WDAG.
- Paths = ‘segments’ or ‘regions’

![Diagram of a weighted linked list with arrows and weights]

-2 \[\rightarrow\] 2 \[\rightarrow\] -1 \[\rightarrow\] 2 \[\rightarrow\] 1 \[\rightarrow\] -1 \[\rightarrow\] -1 \[\rightarrow\] 1 \[\rightarrow\] -2 \[\rightarrow\] 1 \[\rightarrow\] -2 \[\rightarrow\] 2 \[\rightarrow\]

**highest-scoring segment**
• Find highest-scoring segments by dynamic programming
  – Much better than “brute force” algorithm!

• Beginning & end of best path determine path uniquely, so
  – traceback is unnecessary
  – single pass through list suffices to find best path.
from lecture 6:

• To reconstruct best path, need “traceback” pointer to immediate predecessor of $v$ in best path:

$$T(v) = \begin{cases} v & w(v) = 0 \\ \arg \max_{u \in \text{parents}(v)} (w(u) + w((u,v))) & w(v) \neq 0 \end{cases}$$

– in preceding graph, $T(v)$ is the parent on red edge coming into $v$
  • if more than one such edge, pick one at random;
  • if no such edge, $T(v) = v$

• Sometimes useful to record beginning of best path:

$$B(v) = \begin{cases} v & w(v) = 0 \\ B(T(v)) & w(v) \neq 0 \end{cases}$$
Applications to Sequences

• A sequence graph of a sequence is linked list whose edges are labelled by sequence residues (in order):

• e.g. graph for sequence ACCGCTGCGAAG is:
Weighted Sequence Graphs

• If attach weight to each residue, sequence graph becomes a WLL.

```
A → C → C → G → C → T → G → C → A → A → G
```

Highest-scoring segment

• Useful for identifying sequence regions (‘target regions’) with atypical composition:
• In DNA:
  – GC-rich regions in AT-rich thermophile genomes
    • generally correspond to RNA genes (Rob Klein & Sean Eddy)
  – horizontally transferred regions
  – isochores (mammalian DNA)

• In proteins:
  – hydrophobic regions (transmembrane segments)
  – hydrophilic regions (loops, intrinsically disordered regions)
  – acidic or basic regions
‘Optimal’ scores

- **Assume** sequence consists of
  - _target regions_ with residue freqs $t_r$
  - _background regions_ with residue freqs $b_r$
  - _independence assumption_ applies in both

- **Then** ‘best’ scoring system to detect the target regions uses LLRs:
  $$s(r) = \log(t_r / b_r)$$

- if residue freqs are unknown, can usually estimate iteratively
Can use *non-residue-based* scores to find:

- Regions enriched in particular sequence *motifs*:
  - CpG islands in mammalian genomes
    - positive weight (e.g. +17) to the first C of each CpG, and
    - negative weight (e.g. −1) to every other base
    (This approach was used in *Nature* human genome paper).
  - Regions rich in (known) transcription-factor motifs
  - Optimal scores are LLRs, but now based on ‘symbol frequencies’ (where symbol = presence/absence of motif)
• Regions targeted by *next-gen read experiments* (symbols = *read counts*)
  – CNVs (Homework 5)
  – Hypersensitive sites
  – CHIP-seq

• Conserved regions in *sequence alignments* (symbols = *alignment columns*)
CNVs & Read Depth

• CNV = ‘copy number variant’– e.g. region that is single copy in reference sequence but duplicated in sample

• One way to detect: map reads from sample onto reference, look for regions of atypical coverage depth

‘Single-copy’ in sample and reference

multi-copy in sample
HW 5: finding CNVs using D-segments

- **data**: next-gen read alignments to genome
- observed symbols: *counts* of # *read starts* at each position (0, 1, 2, $\geq 3$)
  - frequencies from *Poisson dist’n* with appropriate mean
- target regions: *heterozygous duplications*
  - One chrom = reference allele, other is dup
  - Poisson mean $= 1.5 \times$ background mean
Finding *multiple* high-scoring segments

- In general, expect several regions of particular type in a given sequence – not just one!
- So want to find multiple high-weight paths in a WDAG
- But not interested in slight perturbations of previously found paths
- One strategy:
  - Find highest-weight path
  - ‘Mask it’ (remove its edges from graph)
  - Repeat above two steps until scores no longer ‘interesting’
• Is there a more efficient algorithm not requiring repeated scans?
  – Ruzzo & Tompa solved for WLLs
  – ∃ solution for arbitrary WDAGs?
• A (locally-) maximal(-scoring) segment $I$ is one such that
  – $P1$: no subsegment of $I$ has a higher score than $I$
  – $P2$: no segment properly containing $I$ satisfies $P1$

• Example:

  contained in higher-scoring segment
  maximal-scoring segments

  score = 75, but does not satisfy $P1$
• **Problem**: given $S > 0$, find all maximal segs of score $\geq S$
• Segments are *paths* in a linked-list WDAG with $N+1$ vertices and $N$ edges
• **Highest weight path** is found by dynamic programming;
  in (pseudo-)pseudocode:
  
  ```
  cumul = max = 0; start = 1;
  for (i = 1; i \leq N; i++) {
    cumul += s[i];
    if (cumul \leq 0)
      {cumul = 0; start = i + 1;} /* NOTE RESET TO ZERO */
    else if (cumul \geq max)
      {max = cumul; best_end = i; best_start = start;}
  }
  if (max \geq S) print best_start, best_end, max
  ```
Maximal segments – from cumulative score plot
(without 0 resets)

start (local minimum)

maximal segment

end (local maximum)
• Can find \textit{all} maximal segs of score $\geq S$ using following practical (but \textit{non-optimal}) algorithm:

\begin{verbatim}
cumul = max = 0; start = 1;
for (i = 1; i $\leq$ N; i++) {
    cumul += s[i];
    if (cumul $\geq$ max)
        {max = cumul; end = i;}
    if (cumul $\leq$ 0 or i == N) {
        if (max $\geq$ S) 
            {print start, end, max; i = end; } /* N.B. MUST BACKTRACK! */
        max = cumul = 0; start = end = i + 1;
    }
}
\end{verbatim}
‘backtracked’ region – scanned twice

1\textsuperscript{st} maximal segment

2\textsuperscript{d} maximal segment
• In worst case this is $O(N^2)$ (because of backtracking),
  – but in practice usually $O(N)$ because a given base is usually traversed only a few times
• Ruzzo-Tompa algorithm *guarantees* $O(N)$
- undesirable aspect of maximal segments as defined:
  - single maximal seg may contain two (or more) high-scoring regions, separated by significant negative-scoring regions
  - i.e. two possibly biologically distinct target occurrences get merged into one maximal segment
• Example:

now entire segment has score = 105, & satisfies $P1$ and $P2$
A better problem!

- to avoid this, have max allowed ‘dropoff’ D < 0
- **$D$-segment** is segment without any subsegments of score < D
- **maximal $D$-segment** is $D$-segment I such that
  - $P1$: no subsegment of I has higher score than I
  - $P2$: no $D$-segment properly containing I satisfies $P1$
- Problem: given $S (\geq -D)$, find all maximal $D$-segs of score $\geq S$
  - (algorithm fails if $S < -D$)
Maximal D-segments

maximal segment

1\textsuperscript{st} maximal D-segment

2\textsuperscript{nd} maximal D-segment
• \(O(N)\) algorithm to find all maximal \(D\)-segs:

\[
\text{cumul} = \text{max} = 0; \text{start} = 1;
\]

\[
\text{for } (i = 1; i \leq N; i++) \{
\]

\[
\text{cumul} += s[i];
\]

\[
\text{if } (\text{cumul} \geq \text{max})
\]

\[
\{ \text{max} = \text{cumul}; \text{end} = i; \}
\]

\[
\text{if } (\text{cumul} \leq 0 \text{ or } \text{cumul} \leq \text{max} + D \text{ or } i == N) \{
\]

\[
\text{if } (\text{max} \geq S)
\]

\[
\{ \text{print start, end, max; } \}
\]

\[
\text{max} = \text{cumul} = 0; \text{start} = \text{end} = i + 1; /* \text{NO BACKTRACKING NEEDED!} */
\]

\[
\}
\]

\[
\}
\]
• So more biologically relevant problem is also computationally simpler!

• what are appropriate S and D?
  – mainly an empirical question (based on known examples); altho
    • interpretation via 2-state HMM can be useful
    • Karlin-Altschul theory tells when they are ‘statistically significant’
D-Segments

- Powerful tool for analyzing ‘linear’ data
  - Single sequences (incl. motifs, numerical data)
  - Fixed alignment

- Strengths:
  - Very simple to program
  - Very fast, even for mammalian genomes

- Main limitation:
  - Only allows two types of segments (‘target’ and ‘background’)
    - Essentially a generalization of 2-state HMMs
    - multi-state HMMs are more flexible
Statistical significance of segment scores

• How often does a given score occur by chance in background sequence?

• Can suggest (but not prove!) biological significance
Methods for Assessing Significance of Maximal Segment Scores

1. exact prob dist’n
2. approximate formula (Karlin-Altschul)
3. from simulated sequences
4. from real biological ‘background’ sequences
   – i.e. not having feature in question

1, 2, 3 require probability model approximating biological reality; 4 requires an appropriate dataset
2 is faster than 1 or 3 (and gives ‘intuition’), but involves add’l approximations (ignores ‘edge effects’)
1 requires more complex algorithm
Karlin / Altschul approximation

- for $s(r) = \log_b(t_r / b_r)$, expected # segments of score $\geq S$ in (random) backgd seq of length $N$
  \[ \approx NK b^{-S} \]
- for some constant $K$ (not depending on $S$)
- Note that $b^{-S} = b^{-LLR} = 1 / LR$
  so (apart from $K$) this is essentially the observation in lecture 5:
Average likelihood ratios

- **average LR** (for sites) $\approx$ **average spacing** between occurrences of ‘site-like’ sequences *in background*

- So e.g. for 3’ splice sites
  - if the average LR is 1000, then one expects ‘splice-site-like’ sequences to occur on average once per kb *in background sequence*
  - *N.B.* This says nothing about the frequency of *actual* splice sites! (which could be greater or smaller than 1 per kb), and so doesn’t by itself provide the probability that an *apparent* splice site is an *actual* site.