CS 466
Introduction to Bioinformatics
Lecture 14

Mohammed El-Kebir
October 11, 2019
Outline

• Recap: RNA Secondary Structure Prediction
• Protein Contact Map Overlap

Reading:
• Lecture notes
Nussinov Algorithm – Dynamic Programming

**Problem:** Given RNA sequence $v \in \{A, U, C, G\}^n$, find a *pseudoknot-free secondary structure* with the maximum number of complementary base pairings

Let $s[i, j]$ denote the maximum number of pseudoknot-free complementary base pairings in subsequence $v_i, ..., v_j$

$s[i, j] = \max \left\{ 
\begin{array}{l}
0, \\
 s[i + 1, j - 1] + 1, \\
 s[i + 1, j - 1], \\
 s[i + 1, j], \\
 s[i, j - 1], \\
 \max_{i < k < j} \{s[i, k] + s[k + 1, j]\}, \\
\end{array}
\right.$

if $i \geq j$,
if $i < j$ and $(v_i, v_j) \in \Gamma$, (1)
if $i < j$ and $(v_i, v_j) \notin \Gamma$, (1*)
if $i < j$, (2)
if $i < j$, (3)
if $i < j$, (4)

**Question:** Which case is redundant?
Nussinov Algorithm – Traceback Step

Push (1, n) onto stack
Repeat until stack is empty:
  pop (i,j)
  if i ≥ j continue
  else if s[i+1,j] = s[i,j]
    push (i+1,j)
  else if s[i,j-1] = S[i,j]
    push (i,j-1)
  else if s[i+1,j-1] + 1 = s[i,j]
    record (i,j) base pair
    push (i+1,j-1)
  else for k = i+1 to j-1
    if s[i,k]+s[k+1,j] = s[i,j]
      push (k+1,j)
      push (i,k)
    break (for loop)
Nussinov Algorithm – Traceback Step

Push (1, \( n \)) onto stack
Repeat until stack is empty:
- pop \((i, j)\)
  - if \( i \geq j \) continue
  - else if \( s[i+1, j] = s[i, j] \)
    - push \((i+1, j)\)
  - else if \( s[i, j-1] = S[i, j] \)
    - push \((i, j-1)\)
  - else if \( s[i+1, j-1] + 1 = s[i, j] \)
    - record \((i, j)\) base pair
    - push \((i+1, j-1)\)
  - else for \( k = i+1 \) to \( j-1 \)
    - if \( s[i, k] + s[k+1, j] = s[i, j] \)
      - push \((k+1, j)\)
      - push \((i, k)\)
      - break (for loop)

BackTrack\((i, j)\)
if \( i < j \)
- if \( s[i+1, j] = s[i, j] \)
  - BackTrack\((i+1, j)\)
- else if \( s[i, j-1] = S[i, j] \)
  - BackTrack\((i, j-1)\)
- else if \( s[i+1, j-1] + 1 = s[i, j] \)
  - Output \((i, j)\)
  - BackTrack\((i+1, j-1)\)
else for \( k = i+1 \) to \( j-1 \)
- if \( s[i, k] + s[k+1, j] = s[i, j] \)
  - BackTrack\((k+1, j)\)
  - BackTrack\((i, k)\)
  - break (for loop)
Outline

• Recap: RNA Secondary Structure Prediction
• Protein Contact Map Overlap

Reading:

• Lecture notes
Central Dogma of Molecular Biology

Three fundamental molecules:

1. **DNA**
   Information storage.

2. **RNA**
   Old view: Mostly a “messenger”.
   New view: Performs many important functions, through **3-D structure**!

3. **Protein**
   Perform most cellular functions (biochemistry, signaling, control, etc.)

DNA → RNA → Protein

*First proposed by Francis Crick in 1956.*
A GUIDE TO THE TWENTY COMMON AMINO ACIDS

AMINO ACIDS ARE THE BUILDING BLOCKS OF PROTEINS IN LIVING ORGANISMS. THERE ARE OVER 500 AMINO ACIDS FOUND IN NATURE - HOWEVER, THE HUMAN GENETIC CODE ONLY DIRECTLY ENCODES 20. ‘ESSENTIAL’ AMINO ACIDS MUST BE OBTAINED FROM THE DIET, Whilst NON-ESSENTIAL AMINO ACIDS CAN BE SYNTHESISED IN THE BODY.

Chart Key:
- ALIPHATIC
- AROMATIC
- ACIDIC
- BASIC
- HYDROXYLIC
- SULFUR-CONTAINING
- AMIDIC
- NON-ESSENTIAL
- ESSENTIAL

**Chemical Structure**
- Single letter code
- Three letter code
- Diet codes

**Name**
- ALANINE
- GLYCINE
- ISOLEUCINE
- LEUCINE
- PROLINE
- VALINE
- PHENYLALANINE
- TRYPTOPHAN
- TYRONE
- ASPARTIC ACID
- GLUTAMIC ACID
- ARGinine
- HISTidine
- LYSine
- SERine
- THREONINE
- CYSTEINE
- METHIONINE
- ASPARAGINE
- GLUTAMINE

**Note:** This chart only shows those amino acids for which the human genetic code directly codes for. Selenocysteine is often referred to as the 21st amino acid, but is encoded in a special manner. In some cases, distinguishing between asparagine/aspartic acid and glutamine/glutamic acid is difficult. In these cases, the codes asx (B) and glx (Z) are respectively used.

© COMPOUND INTEREST 2014 - WWW.COMPOUNDCHEM.COM | Twitter: @compoundchem | Facebook: www.facebook.com/compoundchem
Shared under a Creative Commons Attribution-NonCommercial-NoDerivatives licence.
Protein Structure Prediction
Example

• http://pdb101.rcsb.org/motm/218
What is functionally important is conserved throughout evolution

- It can activate DNA repair proteins when DNA has sustained damage.
- It can arrest growth by holding the cell cycle at the G1/S regulation point on DNA damage.
- It can initiate apoptosis (i.e., programmed cell death) if DNA damage proves to be irreparable.
- It is essential for the senescence response to short telomeres.

Sequence → Structure → Function

What is functionally important is conserved throughout evolution
What is functionally important is conserved throughout evolution.
How to Compare Two Protein Sequences?

**TP53 (Human)**

```
1 meepqsdpsv epplsqetfs dlwkllpenn vlspqpsam ddllplspddi eqwftedpgp
61 deaprmpeaa ppvapapaap tpaapapaps wplsssvpsq ktyqqgsgfr lgflhsgtak
121 svtctyspal nkmcqlakt cpvqlwvdst pppgtrvram aiykqssqht evvrrcphhe
181 rcsdsdglap npghlirvegn lrveylddrn tfrhsvvvpy eppevsgdct tihynymcns
241 scmgmnrrp iltiileds sgnllgrnsf evrvcagpgr drrteeenlr kkgephhelp
301 pgstkralpn ntssspqppkk kplgeyftl qirgrerfem frelnealel kdaaqkepg
361 gsrahshhlk skkgqstsrh kklmfktgep dsd
```

**p53 (Mouse)**

```
1 mtameesqsd islelplsge tfslwklldp pedilpsphc mddlllpqdv eeffegpsea
61 lrvsgapaaq dpvtetpvgp avapatpwpl ssfvsqkty qgnyqfhlgf lqsrgtaksvm
121 ctysspnlknk fcqlaktcpl qlwvsatppa gsrvramaiy kksqhmtevv rrcphhercs
181 dgdgllpqqh rirvegnylp eyledrgtrf hsvvyveyg eayseyttih ykymcsnncsm
241 ggmnrrpilt ilitedssgn llgrdsfevr vcacpgrdrr teeenfrke vlcpelppgs
301 akralptcts asppkkkpl dgeyftlkir grkrfeqfrem frelnealel kdaaqkepg
361 ahssylktkk gqstsrhkkt mvkkvpgdsd
```
How to Compare Two Protein Sequences?

Global Alignment problem: Given strings $v \in \Sigma^m$ and $w \in \Sigma^n$ and scoring function $\delta$, find alignment of $v$ and $w$ with maximum score

[Needleman-Wunsch algorithm]

Local Alignment problem: Given strings $v \in \Sigma^m$ and $w \in \Sigma^n$ and scoring function $\delta$, find a substring of $v$ and a substring of $w$ whose alignment has maximum global alignment score $s^*$ among all global alignments of all substrings of $v$ and $w$

[Smith-Waterman algorithm]
How to Compare Two Protein Structures?

**FIG. 1.** An optimal alignment of two 4Å threshold contact maps of proteins 1bpi and 1knt.

1knt-4.0.cm: 55 residues with 43 contacts.
31 shared contacts.
1bpi-4.0.cm: 58 residues with 53 contacts.
Contact Map Overlap: Example Instance

FIG. 2. An alignment of value 5.

FIG. 2. Relationship between a matching in a bipartite graph B and a feasible path in the corresponding grid graph $B'$. (Left) Two contact maps $G_1$ and $G_2$, and a matching in the bipartite graph $B$ (in the grey area). Note that $B$ is a complete graph, but for the sake of simplicity only the edges of the considered matching ($M = \{(1,1)(2,3),(3,4)(5,5)\}$) are visualized. According to (1), $w(M) = 2$. (Right) The same matching is visualized in the grid alike graph $B'$ as an increasing set of vertices $\{(1,1)(2,3),(3,4)(5,5)\}$ which we call a feasible path. It activates the arcs $((1,1)(2,3) \text{ and } (3,4)(5,5))$. The score of the path is the number of these arcs (i.e., 2 in this case).
Integer Linear Programming

Figure 2.1: In gray, a polyhedron that is described by six constraints $a_i x \leq b_i$. The objective function $c^T x$ increases in the direction in which the arrow points. The optimal solution $x^*$ is denoted by a star. Left: The linear program. Right: The integer linear program. Here, only the integer points within the polyhedron are feasible, which are colored black. The LP relaxation of this ILP is the LP problem that is visualized on the left side. The optimal objective function value of the LP relaxation is always an upper bound on the optimal objective function value of the ILP.

Separating Cutting Planes

Figure 2.2: The cutting plane method solves an ILP problem. The gray area denotes the polyhedron described by the constraints of the current relaxed problem. The dashed line denotes the objective function $c^T x$ which increases in the direction in which the arrow points. The integer feasible solutions are colored black. Solving the LP relaxation, we obtain the relaxed solution $x^0$. After adding a cutting plane (dotted line), we obtain a new relaxed solution $x^1$. Finally, after adding a second cutting plane, we obtain solution $x^2$ which has integer value and is thus the optimal solution $x^*$ of the ILP. A cutting plane method solves only the LP relaxation of an ILP, but here, since the optimal solution has integer value, the solution of the LP relaxation is also the solution of the ILP.
Cutting Plane Method

Figure 2.2: The cutting plane method solves an ILP problem. The gray area denotes the polyhedron described by the constraints of the current relaxed problem. The dashed line denotes the objective function $c^T x$ which increases in the direction in which the arrow points. The integer feasible solutions are colored black. Solving the LP relaxation, we obtain the relaxed solution $x^0$. After adding a cutting plane (dotted line), we obtain a new relaxed solution $x^1$. Finally, after adding a second cutting plane, we obtain solution $x^2$ which has integer value and is thus the optimal solution $x^*$ of the ILP. A cutting plane method solves only the LP relaxation of an ILP, but here, since the optimal solution has integer value, the solution of the LP relaxation is also the solution of the ILP.

Algorithm 2 Solving an LP problem using the cutting plane method.

1: $P$ // The original problem
2: $P^t$ // The relaxed problem in iteration $t$
3: $t ← 0$ // Iteration
4: while True do
5:     Compute optimal solution $x^t$ for $P^t$
6:     if $x^t$ feasible for $P$ then
7:         return $x^t$
8:     else
9:         Find a cutting plane $a_i x ≤ b_i$ that all solutions of $P$ satisfy, but not $x^t$
10:        $P^{t+1} ← P^t$ with additional constraint $a_i x ≤ b_i$
11:        $t ← t + 1$
12:    end if
13: end while
Branch & Cut: Solving an ILP

• Whiteboard
Figure 3.8: Example of the graphs in which we use shortest path computations to detect violated constraints. Left: The alignment graph $G = (V, E)$. Center: The graph $G' = (V', E')$ in which we identify a violated constraint (3.8). The shortest decreasing path is colored black, it is $C = \{4.1, 4.2, 3.2, 2.2, 2.3, 1.3\}$. If for the this path $\sum_{i,k \in C} \bar{x}_{ik} > 1$ holds, we identified a violated constraint (3.8).