Global Alignment

Ben Langmead

JOHNS HOPKINS
WHITING SCHOOL
of ENGINEERING

Department of Computer Science

Please sign guestbook (www.langmead-lab.org/teaching-materials) to tell me briefly how you are using the slides. For original Keynote files, email me (ben.langmead@gmail.com).

Generalizing edit distance

What if it doesn't make sense for every edit to cost 1?

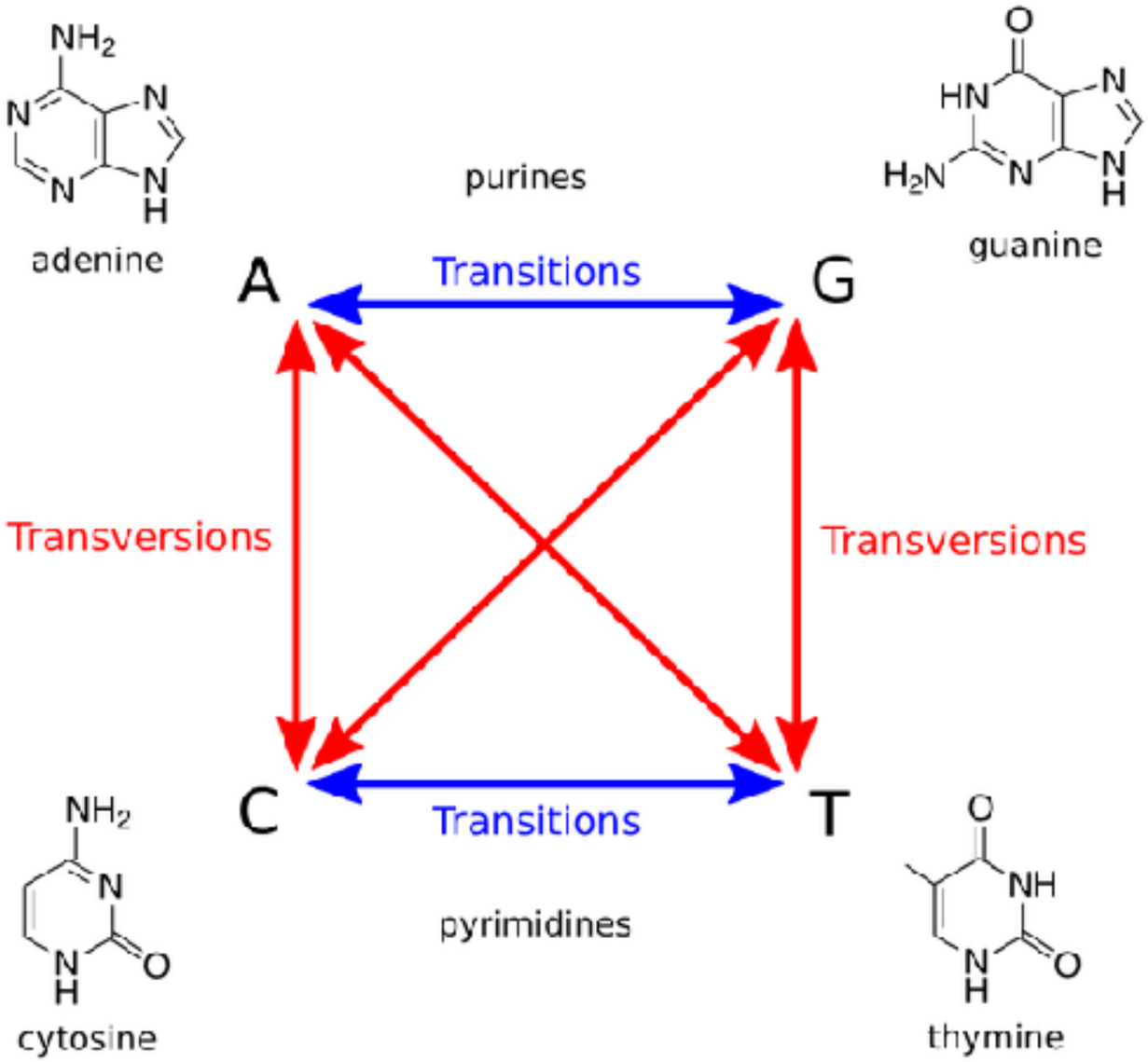
If you compare two human genomes, you see some kinds of sequence differences more often than others

Generalizing edit distance

Transitions are A ↔ G and C ↔ T changes

Transversions are A ↔ C, A ↔ T, C ↔ G, G ↔ T

For random mutations, transitions should be half as frequent as transversions...



...***but*** if you compare two humans, *transition to transversion ratio* (*ti/tv*) is ~2.1

Generalizing edit distance

```
G G G T A G C G G G T T T A A C
| | | | |   | | | | | | | | |
G G G T A A C G G G T T T A A C
```

Human substitution rate ≈ 1 in 1,000

```
G G G T A G C G G G T T T A A C
| | | | |     | | | | | | | | |
G G G T A - - G G G T T T A A C
```

Small-gap rate is ≈ 1 in 3,000

Wanted: keep basic edit distance idea and algorithm, but give different weights to different events according to likelihood

Penalty function

	A	C	G	T	-
A	0	4	2	4	8
C	4	0	4	2	8
G	2	4	0	4	8
T	4	2	4	0	8
-	8	8	8	8	

2 *Transitions* (A↔G, C↔T)

4 *Transversions*

8 *Gaps*

Global alignment

Let $D[0, j] = \sum_{k=0}^{j-1} s(-, y[k])$, and let $D[i, 0] = \sum_{k=0}^{i-1} s(x[k], -)$

Otherwise, let $D[i, j] = \min \begin{cases} D[i-1, j] + s(x[i-1], -) \\ D[i, j-1] + s(-, y[j-1]) \\ D[i-1, j-1] + s(x[i-1], y[j-1]) \end{cases}$

$s(a, b)$ assigns a cost to a particular gap or substitution

$s(a, b)$:

	A	C	G	T	-
A	0	4	2	4	8
C	4	0	4	2	8
G	2	4	0	4	8
T	4	2	4	0	8
-	8	8	8	8	

2 *Transitions* (A↔G, C↔T)

4 *Transversions* (everything else)

8 *Gaps*

Global alignment: implementation

```
from numpy import zeros

def globalAlignment(x, y, s):
    """ Calculate global alignment value of sequences x and y using
        dynamic programming.  Return global alignment value. """
    D = zeros((len(x)+1, len(y)+1), dtype=int)
    for j in range(1, len(y)+1):
        D[0, j] = D[0, j-1] + s('-', y[j-1])
    for i in range(1, len(x)+1):
        D[i, 0] = D[i-1, 0] + s(x[i-1], '-')
for i in range(1, len(x)+1):
    for j in range(1, len(y)+1):
        D[i, j] = min(D[i-1, j-1] + s(x[i-1], y[j-1]), # diagonal
                      D[i-1, j  ] + s(x[i-1], '-'),    # vertical
                      D[i  , j-1] + s('-',    y[j-1])) # horizontal
return D, D[len(x), len(y)]
```

Use of new penalty function

Similar to edit distance

http://bit.ly/CG_DP_Global

Global alignment: implementation

```
def exampleCost(xc, yc):
    """ Cost function assigning 0 to match, 2 to transition, 4 to
        transversion, and 8 to a gap """
    if xc == yc: return 0 # match
    if xc == '-' or yc == '-': return 8 # gap
    minc, maxc = min(xc, yc), max(xc, yc)
    if minc == 'A' and maxc == 'G': return 2 # transition
    elif minc == 'C' and maxc == 'T': return 2 # transition
    return 4 # transversion
```

	A	C	G	T	-
A	0	4	2	4	8
C	4	0	4	2	8
G	2	4	0	4	8
T	4	2	4	0	8
-	8	8	8	8	

http://bit.ly/CG_DP_Global

Global alignment: dynamic programming

globalAlignment
initialization:

```
D = zeros((len(x)+1, len(y)+1), dtype=int)
for j in range(1, len(y)+1):
    D[0, j] = D[0, j-1] + s('-', y[j-1])
for i in range(1, len(x)+1):
    D[i, 0] = D[i-1, 0] + s(x[i-1], '-')
```

	ϵ	T	A	T	G	T	C	A	T	G	C
ϵ	**0**	**8**	**16**	**24**	**32**	**40**	**48**	**56**	**64**	**72**	**80**
T	**8**										
A	**16**										
C	**24**										
G	**32**										
T	**40**										
C	**48**										
A	**56**										
G	**64**										
C	**72**										

$s(a, b)$

	A	C	G	T	-
A	0	4	2	4	8
C	4	0	4	2	8
G	2	4	0	4	8
T	4	2	4	0	8
-	8	8	8	8	

Global alignment: dynamic programming

globalAlignment
loop:

```
for i in range(1, len(x)+1):
    for j in range(1, len(y)+1):
        D[i, j] = min(D[i-1, j-1] + s(x[i-1], y[j-1]), # diagonal
                      D[i-1, j  ] + s(x[i-1], '-'),    # vertical
                      D[i  , j-1] + s('-',    y[j-1])) # horizontal
```

	ϵ	T	A	T	G	T	C	A	T	G	C
ϵ	0	8	16	24	32	40	48	56	64	72	80
T	8	0	8	16	24	32	40	48	56	64	72
A	16	8	0	8	16	24	32	40	48	56	64
C	24	16	8	?							
G	32										
T	40										
C	48										
A	56										
G	64										
C	72										

$s(a, b)$

	A	C	G	T	-
A	0	4	2	4	8
C	4	0	4	2	8
G	2	4	0	4	8
T	4	2	4	0	8
-	8	8	8	8	

Global alignment: dynamic programming

globalAlignment
loop:

```
for i in range(1, len(x)+1):
    for j in range(1, len(y)+1):
        D[i, j] = min(D[i-1, j-1] + s(x[i-1], y[j-1]), # diagonal
                      D[i-1, j  ] + s(x[i-1], '-'),    # vertical
                      D[i  , j-1] + s('-',    y[j-1])) # horizontal
```

	ϵ	T	A	T	G	T	C	A	T	G	C
ϵ	0	8	16	24	32	40	48	56	64	72	80
T	8	0	8	16	24	32	40	48	56	64	72
A	16	8	0	8	16	24	32	40	48	56	64
C	24	16	8	2	10	18	24	32	40	48	56
G	32	24	16	10	2	10	18	26	34	40	48
T	40	32	24	16	10	2	10	18	26	34	42
C	48	40	32	24	18	10	2	10	18	26	34
A	56	48	40	32	26	18	10	2	10	18	26
G	64	56	48	40	32	26	18	10	6	10	18
C	72	64	56	48	40	34	26	18	12	10	10

Optimal global alignment value

$s(a, b)$

	A	C	G	T	-
A	0	4	2	4	8
C	4	0	4	2	8
G	2	4	0	4	8
T	4	2	4	0	8
-	8	8	8	8	

Global alignment: getting the alignment

Traceback works just as it did for edit distance

	ϵ	T	A	T	G	T	C	A	T	G	C
ϵ	0	8	16	24	32	40	48	56	64	72	80
T	8	0	8	16	24	32	40	48	56	64	72
A	16	8	0	8	16	24	32	40	48	56	64
C	24	16	8	2	10	18	24	32	40	48	56
G	32	24	16	10	2	10	18	26	34	40	48
T	40	32	24	16	10	2	10	18	26	34	42
C	48	40	32	24	18	10	2	10	18	26	34
A	56	48	40	32	26	18	10	2	10	18	26
G	64	56	48	40	32	26	18	10	6	10	18
C	72	64	56	48	40	34	26	18	12	10	10

```
T A C G T C A - G C
|  |  | | |  |  |
T A T G T C A T G C
```

+2 (transition)

+8 (gap)

Global alignment: summary

Matrix-filling dynamic programming algorithm is $O(mn)$ time and space

FillIng matrix is $O(mn)$ space and time, yields global alignment value

Traceback is $O(m + n)$ time, yields optimal alignment

Global alignment: scoring functions

Where do these penalty functions come from?

	A	C	G	T	-
A	0	4	2	4	8
C	4	0	4	2	8
G	2	4	0	4	8
T	4	2	4	0	8
-	8	8	8	8	

They can be based on:

Expected frequency of the different mutational events

How *interchangeable* are the alternatives are from a biological perspective

Does the substitution change the *shape* or *function* of the molecule

Prevalence of simple (linear, constant, affine) gap penalties is mostly because that's what we can do efficiently, as discussed in HW4

One occasionally sees more general (e.g. convex) gap penalties

BLOSUM62

Some amino acid substitutions have a smaller impact on structure & function than others. BLOSUM62 elements are, roughly speaking, log-odds of observing these substitutions between two highly related proteins

	Ala	Arg	Asn	Asp	Cys	Gln	Glu	Gly	His	Ile	Leu	Lys	Met	Phe	Pro	Ser	Thr	Trp	Tyr	Val
Ala	4																			
Arg	-1	5																		
Asn	-2	0	6																	
Asp	-2	-2	1	6																
Cys	0	-3	-3	-3	9															
Gln	-1	1	0	0	-3	5														
Glu	-1	0	0	2	-4	2	5													
Gly	0	-2	0	-1	-3	-2	-2	6												
His	-2	0	1	-1	-3	0	0	-2	8											
Ile	-1	-3	-3	-3	-1	-3	-3	-4	-3	4										
Leu	-1	-2	-3	-4	-1	-2	-3	-4	-3	2	4									
Lys	-1	2	0	-1	-3	1	1	-2	-1	-3	-2	5								
Met	-1	-1	-2	-3	-1	0	-2	-3	-2	1	2	-1	5							
Phe	-2	-3	-3	-3	-2	-3	-3	-3	-1	0	0	-3	0	6						
Pro	-1	-2	-2	-1	-3	-1	-1	-2	-2	-3	-3	-1	-2	-4	7					
Ser	1	-1	1	0	-1	0	0	0	-1	-2	-2	0	-1	-2	-1	4				
Thr	0	-1	0	-1	-1	-1	-1	-2	-2	-1	-1	-1	-1	-2	-1	1	5			
Trp	-3	-3	-4	-4	-2	-2	-3	-2	-2	-3	-2	-3	-1	1	-4	-3	-2	11		
Tyr	-2	-2	-2	-3	-2	-1	-2	-3	2	-1	-1	-2	-1	3	-3	-2	-2	2	7	
Val	0	-3	-3	-3	-1	-2	-2	-3	-3	3	1	-2	1	-1	-2	-2	0	-3	-1	4

Rare; larger effect on structure/function

Common; modest effect on structure/function

negative

positive

Matrix is symmetric

Amino acids