An alignmnet ends either with
(1) a match/mismatch
(2) a gap in the first sequence
(3) a gap in the second sequence

```
S1:   ATCGCTGGCATAC
S2:   TTCCTAGCCTAC
```

```
ATCGCT        ATCGCT-       ATCGC-T
TTCCTA        -TTCCTA       TTCCTA-
```

use the opt. alignment of S1[1..5] and S2[1..5].

use the opt. alignment of S1[1..6] and S2[1..5].

use the opt. alignment of S1[1..5] and S2[1..6].

One of the alignments is optimal !

The recurrence relation

	ATCGCT TTCCTA	ATCGCT– –TTCCTA	ATCGC–T TTCCTA–
Edit steps	D(5,5)+1	D(6,5)+1	D(5,6)+1

$$D(6,6) = \min \begin{cases} D(5,5)+1 \\ D(6,5)+1 \\ D(5,6)+1 \end{cases}$$

The general recurrence relation

$$D(i,j) = \min \begin{cases} D(i-1,j-1)+t(i,j) \\ D(i,j-1)+1 \\ D(i-1,j)+1 \end{cases}$$

$t(i,j)=0$ if $S1(i)= S2(1)$ "match"

$t(i,j)=1$ if $S1(i) \neq S2(1)$ "mismatch"

"Calculate D(3,4)" is a subproblem of "calculate D(5,5)"

"Calculate D(3,4)" is also a subproblem of "calculate D(12,15)"

Idea:
We solve "calculate D(3,4)" only once

We start with solving easy problems like "calculate D(1,1)" or even "calculate D(0,0),D(0,1),D(1,0) ..."

BOTTOM-UP COMPUTATION

INITIALIZATION

			W	R	I	T	E	R	S
		0	1	2	3	4	5	6	7
	0	0	1	2	3	4	5	6	7
V	1	1							
I	2	2							
N	3	3							
T	4	4							
N	5	5							
E	6	6							
R	7	7							

Align the first 0 characters of S1 to the first 2 characters of S2:

S1: WRITERS

S2: VINTERS

VI...
--...

This results in 2 insertions.

Tabular calculation

			W	R	I	T	E	R	S
		0	1	2	3	4	5	6	7
	0	0	1	2	3	4	5	6	7
V	1	1	1	2	3	4	5	6	7
I	2	2	2	2	2	3	4	5	6
N	3	3	3	3	3	3	4	5	6
T	4	4	4	4	4	?			
N	5	5							
E	6	6							
R	7	7							

			W	R	I	T	E	R	S
		0	1	2	3	4	5	6	7
	0	0	1	2	3	4	5	6	7
V	1	1	1	2	3	4	5	6	7
I	2	2	2	2	2	3	4	5	6
N	3	3	3	3	3	3	4	5	6
T	4	4	4	4	4	3	4	5	6
N	5	5	5	5	5	4	4	5	6
E	6	6	6	6	6	5	4	5	6
R	7	7	7	6	7	6	5	4	5

Edit distance
of S1 and S2

THE TRACEBACK

			W	R	I	T	E	R	S
		0	1	2	3	4	5	6	7
	0	0	1	2	3	4	5	6	7
V	1	1	1	2	3	4	5	6	7
I	2	2	2	2	2	3	4	5	6
N	3	3	3	3	3	3	4	5	6
T	4	4	4	4	4	3	4	5	6
N	5	5	5	5	5	4	4	5	6
E	6	6	6	6	6	5	4	5	6
R	7	7	7	6	7	6	5	4	5

RETRIEVING COOPTIMAL ALIGNMENTS

			W	R	I	T	E	R	S
		0	1	2	3	4	5	6	7
	0	0	1	2	3	4	5	6	7
V	1	1	1	2	3	4	5	6	7
I	2	2	2	2	2	3	4	5	6
N	3	3	3	3	3	3	4	5	6
T	4	4	4	4	4	3	4	5	6
N	5	5	5	5	5	4	4	5	6
E	6	6	6	6	6	5	4	5	6
R	7	7	7	6	7	6	5	4	5

```
WRIT-ERS
VINTNER-
*** *  *
```

```
WRI-T-ERS
V-INTNER-
** * *  *
```

```
WRI-T-ERS
-VINTNER-
** * *  *
```

The big O

Consider an algorithm which takes n sequences of lengths l1,l2,...ln as input.

The algorithm has time complexity O(g(l1,l2,...ln)) if it needs less then C*g(l1,l2,...,ln) computation steps C is a constant independent of the lengths of the input sequences.

The algorithm has space complexity O(g(l1,l2,...,ln), if it uses less then C'*g(l1,l2,...,ln)) units of memory.

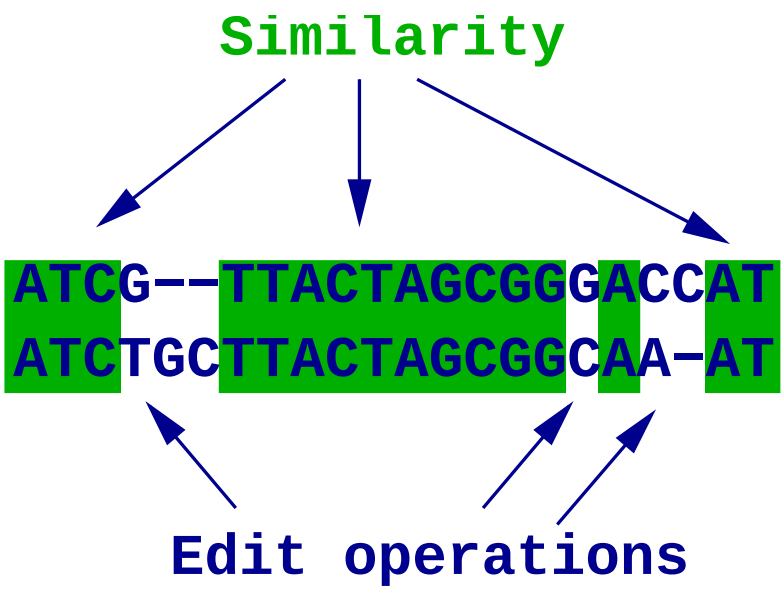
Time and space complexity of the basic dynamic programming algorithm for minimal edit distance alignments

Let's say the two sequences have lengths n and m.
In the tabular calculation we construct a table of
(n+1)x(m+1) numbers. (The D(i,j))
Hence the space complexity is O(nm).

According to the recurrence relation we need to
compare three values when filling in a new field.
Hence the time complexity is also O(nm).

Since the length of both sequences is usually in the same range we can write shortly, that both time and space complexity are of order $O(n^2)$.

ATCG--TTACTAGCGGGACCAT
ATCTGCTTACTAGCGGCAA-AT


Similarity
ATCG--TTACTAGCGGGACCAT
ATCTGCTTACTAGCGGCAA-AT
Edit operations

Distance

the less different the more similar

TRIVIAL ? **No.**

TRUE ? **Not always.**

Alphabet: A={a1,a2,a3,...,an}
e.g. A={a,t,c,g}
A=The 20 amino acids

A^n **All sequences of length n that can be formed from characters in A.**

A^* **All sequences that can be formed from characters in A.**

Distance on A u{−}

d(a1,a2) >= 0 small if a1=a2
high if a1≠a2

d(a1,−)
d(−,a2) =g > 0 Costs for a gap

Distance given an alignment

a1 a2 − a4
b1 − b3 b4

d(alignment)=
= d(a1,b1)+d(a2,−)+d(−,a3)+d(a4,b4)

$= \sum_{i} d(ai,bi)$

Distance on sequences A¨

S1, S2 Sequences

d(S1,S2)= minimum (d(alignment))

where the minimum is taken over all possible alignments of S1 and S2.

Example: edit distance

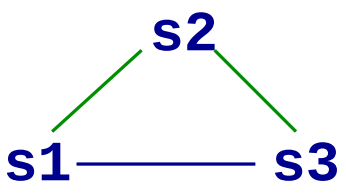
Metric

d(s1,s1)= 0

d(s1,s2)=d(s2,s1) Symmetry

d(s1,s3) <= d(s1,s2)+d(s2,s3)

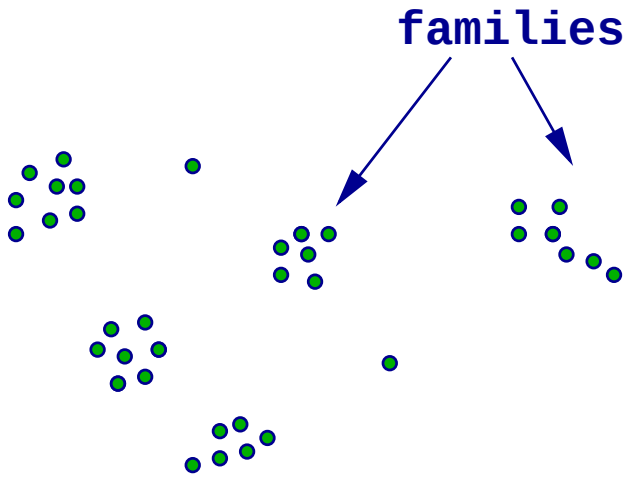
triangular inequality



Idea: Metric on sequence space.

Ok, for edit distance

THE OLD IDEA OF A METRIC ON SEQUENCE SPACE



Problem was put forward in [Ulam 1972]

Ulam, S.: Some combinatorial problems studied experimentally on computing machines. In: Applications of number theory to numerical analysis, ed. Zaremba, S.K. Academic Press, New York and London, 1972.

Score on Au{–}

s(a1,a2) negative
if a1 and a2 are different
positive
if a1 and a2 are similar or identical.

s(a1,–)
s(–,a2) **negative (gap costs)**

Note that distances are never negative, while scores can be both positive and negative.

Score given an alignment

$$s(\text{alignment}) = \sum_i s(ai,bi)$$

Example: $s(ai,ai)=2$
$s(ai,aj)=-1 \quad i \neq j$
$s(ai,-)=s(-,ai)=-5$

ATCG-CC
AT-GAAC

$s=2+2-5+2-5-1+2=-3$

Score on A^*

S(S1,S2)=maximum(s(alignment))
where the maximum is over all possible alignments of S1 and S2.

With the help of scores we can ...

... account for the fact that some amino acids are more similar then others

... place alignment into a likelihood framework

... detect local similarities

File Edit View Go Communicator Help

Members WebMail Connections BizJournal SmartUpdate Mktplace

Bookmarks Location: http://www.dkfz-heidelberg.de/tbi/people/tmueller/model/node1.html

Back Forward Reload Home Search Netscape Print Security Shop Stop

Amino Acid Score Matrix VT160

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
A	3	-1	-1	-1	0	-1	-1	0	-2	-1	-2	-1	-1	-2	0	1	1	-3	-3	0
R	-1	6	0	-1	-2	1	-1	-1	1	-3	-2	3	-2	-3	-1	-1	-1	-1	-2	-2
N	-1	0	5	2	-2	0	0	0	1	-3	-3	1	-2	-3	-2	1	0	-4	-2	-2
D	-1	-1	2	5	-3	0	2	0	0	-4	-4	0	-3	-4	-1	0	-1	-4	-3	-3
C	0	-2	-2	-3	10	-2	-3	-2	-1	-1	-1	-3	-1	-1	-3	0	-1	-1	0	-1
Q	-1	1	0	0	-2	5	1	-2	2	-3	-2	1	-1	-3	0	0	-1	-2	-2	-2
E	-1	-1	0	2	-3	1	5	-1	-1	-3	-3	1	-2	-4	-1	-1	-1	-4	-3	-2
G	0	-1	0	0	-2	-2	-1	6	-2	-4	-4	-2	-3	-4	-2	0	-1	-2	-4	-3
H	-2	1	1	0	-1	2	-1	-2	7	-3	-2	0	-2	-1	-1	-1	-1	-2	2	-3
I	-1	-3	-3	-4	-1	-3	-3	-4	-3	4	2	-3	2	0	-3	-2	0	-2	-2	3
L	-2	-2	-3	-4	-1	-2	-3	-4	-2	2	4	-3	2	1	-2	-2	-1	-1	-1	1
K	-1	3	1	0	-3	1	1	-2	0	-3	-3	5	-2	-3	-1	-1	0	-3	-2	-2
M	-1	-2	-2	-3	-1	-1	-2	-3	-2	2	2	-2	6	0	-3	-2	0	-1	-1	1
F	-2	-3	-3	-4	-1	-3	-4	-4	-1	0	1	-3	0	7	-3	-2	-2	1	4	0
P	0	-1	-2	-1	-3	0	-1	-2	-1	-3	-2	-1	-3	-3	7	0	0	-4	-3	-2
S	1	-1	1	0	0	0	-1	0	-1	-2	-2	-1	-2	-2	0	3	1	-3	-2	-1
T	1	-1	0	-1	-1	-1	-1	-1	-1	0	-1	0	0	-2	0	1	4	-3	-2	0
W	-3	-1	-4	-4	-1	-2	-4	-2	-2	-2	-1	-3	-1	1	-4	-3	-3	12	2	-3
Y	-3	-2	-2	-3	0	-2	-3	-4	2	-2	-1	-2	-1	4	-3	-2	-2	2	8	-2
V	0	-2	-2	-3	-1	-2	-2	-3	-3	3	1	-2	1	0	-2	-1	0	-3	-2	4

DOWNLOAD VT160

PROBABILISTIC FRAMEWORK VIA SCORES

S1: a1 a2 a3 a4, ..., an
S2: b1 b2 b3 b4, ..., bn

S1 and S2 are either related or they are not.

We build separate models for the case of related sequences (E) and the case of unrelated sequences (B)...

E: Evolution
B: Background

... and then compare the probabilities P(Alignment|E) and P(Alignment|B)

Model for related sequences:

$M(ai,aj) = M_{ij}$

= Probability that ai and aj have independently derived from the same ancestor in this position of the sequence.

Higher for similar or even identical amino acids.

Assume positions in the sequences are independent.

a1 a2 a3 a4
b1 b2 b3 b4

$$P(\text{Alignment}|M) = \prod_i M(ai,bi)$$

Model for unrelated sequences
(Background model B)

Assume the letter a_i occurs randomly with probability $q_i = q(ai)$.

We model the relative frequency of amino acids

q(C) is smaller than q(L)

Random alignment: a1 a2 a3 a4 ...
b1 b2 b3 b4 ...

$$P(\text{Alignment}|B) = \prod_i q(ai)*q(bi)$$

Odds ratios

$$\frac{P(\text{Alignment}|E)}{P(\text{Alignment}|B)} = \frac{\prod M_{ij}}{\prod q_i q_j} = \prod_i \frac{M_{ij}}{q_i q_j}$$

$$\text{Log odds} = \sum_i \log\left(\frac{M_{ij}}{q_i q_j}\right)$$

Score: s(ai,aj)

can be both positive and negative

For the score

$$s(ai,aj) = \log\left(\frac{M_{ij}}{q_i q_j}\right) \quad ...$$

... the maximal score alignment is the alignment with the highest odds ratio.

We optimize the alignment such that it is typical for the E model and untypical for the B model.

The general recurrence relation
for maximal score alignments

$$S(i,j) = \max \begin{cases} S(i-1,j-1)+s(S1(i),S2(j)) \\ S(i,j-1)+s(-,S2(i)) \\ S(i-1,j)+s(S1(i),-) \end{cases}$$

S(i,j) = optimal global alignment
score of S1[1..i] and S2[1..j].

INITIALIZATION

			W	R	I	T	E	R	S
		0	1	2	3	4	5	6	7
	0								
V	1								
I	2								
N	3								
T	4								
N	5								
E	6								
R	7								

$$S(0,j)=\sum_{k<=j} s(-,S2(k))$$

$$S(i,0)=\sum_{k<=i} s(S1(k),-)$$

Dynamic programming for maximal score (log odds) alignments and minimal edit distance alignments

(1) Recurrence relation modified

(2) Tabular calculation:
only the initialisation is modified

(3) Traceback is identical

Gaps

```
ATTACGTACTCCATG
ATTACGT----CATG
```

In an edit script we need 4 edit operations for the gap of length 4.

In maximal score alignments we treat the dash "–" like any other character, hence we charge the s(x,–) costs 4 times.

But

In terms of evolution this gap is probably the result of a single deletion or insertion of length 4.

Biological observations:

Gaps are usually longer then just one character

However, long gaps are less frequent than short gaps

Therefore ...

...gaps should be considered as single units

Gap costs should depend on the length of the gap, they should be monotonously growing, but not as fast as the legth itself.

Gap costs should be subadditive:

g(n) gap cost of a gap of length n
n=n1+n2

Subadditivity:

$$g(n) <= g(n1) + g(n2)$$

If not:

Gap is cheaper if it is considered as two successive gaps.

SCORING

Scorematrix for pairs of characters
e.g. VT160

and

Gapcosts g(n)
e.g. g(n)=12+3n

```
MYL--V
M-ACVV
```

Score= vt(M,M)−g(1)+vt(L,A)−g(2)+vt(V,V)

= 6 −12 −2 −15 +4

= −19

GENERAL GLOBAL ALIGNMENT PROBLEM

Given a score matrix and a subadditive gap cost function, calculate the global maximal score alignment.