

Comparação de Sequências

Said Sadique Adi

FACOM - UFMS

16 de setembro de 2014

Conceitos Básicos

Sequência

Uma **sequencia** s é uma sucessão ordenada de caracteres ou símbolos de um certo conjunto Σ , comumente denominado **alfabeto**.

Denotamos por Σ^* o conjunto incluindo todas as sequências finitas construídas sobre o alfabeto Σ .

Tamanho de uma sequência

O **tamanho** de uma sequencia s , denotado por $|s|$, é o número de símbolos componentes dessa sequencia.

Representamos por $s[i]$ o símbolo presente na i -ésima posição de uma sequencia s .

Dada uma sequência s qualquer, os seguintes conceitos estão relacionados a ela.

Conceitos Básicos

Subsequência

Uma **subsequência** de s corresponde a uma sequência obtida a partir de s por meio da remoção de alguns de seus caracteres.

Segmento

Um **segmento** $s[i..j]$, com $i \geq 1$ e $j \leq |s|$, de s é uma sequência formada pelos símbolos consecutivos de s que se iniciam na i -ésima e terminam na j -ésima posição dessa sequência.

Prefixo

Um **prefixo** de s é um segmento de s da forma $s[1..i]$, com $i \leq |s|$.

Sufixo

Um **sufixo** de s é um segmento de s da forma $s[i..|s|]$, com $i \geq 1$.

Conceitos Preliminares

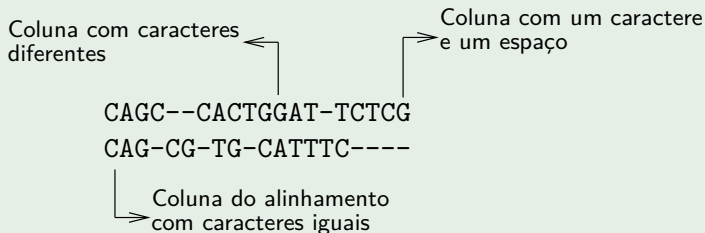
Como comparar duas sequências?

A forma mais comum de se comparar duas sequências é por meio de um processo chamado *alinhamento de sequências*.

Alinhamento de sequências

Alinhar duas sequências s e t consiste em inserir espaços ('-') em locais arbitrários dessas sequências de modo que elas fiquem do mesmo tamanho.

Exemplo de um alinhamento de duas sequências



Conceitos Preliminares

Mais alguns conceitos

Seja Σ um alfabeto qualquer tal que $\{'-\}' \notin \Sigma$.

- Chamamos de $\bar{\Sigma}$ o alfabeto contendo os símbolos de Σ mais o símbolo $\{'-\}'$ ($\bar{\Sigma} = \Sigma \cup \{'-\}'$)
- Para toda sequência $s \in \bar{\Sigma}^*$, seja $s|_{\Sigma}$ a sequência s restrita ao alfabeto Σ (ou seja, a sequência resultante da remoção de todos os espaços presentes em s).

Alinhamento de sequências - definição formal

Um **alinhamento** A de duas sequências s e $t \in \Sigma^*$ é um par (\bar{s}, \bar{t}) , com \bar{s} e $\bar{t} \in \bar{\Sigma}^*$ tal que:

- 1 $|\bar{s}| = |\bar{t}|$;
- 2 $\bar{s}|_{\Sigma} = s$ e $\bar{t}|_{\Sigma} = t$;
- 3 Não existe i tal que $\bar{s}[i] = \bar{t}[i] = \{'-\}'$.

Conceitos Preliminares

matches, mismatches e spaces

Colunas do alinhamento que incluem caracteres iguais, diferentes ou espaço são comumente chamadas de **matches**, **mismatches** e **spaces**, respectivamente

Função de pontuação

Uma **função de pontuação** $w : \bar{\Sigma} \times \bar{\Sigma} \rightarrow \mathbb{R}$ corresponde a uma função que associa um certo valor real para cada par de símbolos (μ, ν) em $\bar{\Sigma}$.

Pontuação de um alinhamento

Dada uma função de pontuação w qualquer, a **pontuação** de um alinhamento $A = (\bar{s}, \bar{t})$ de s e t , denotada por $\text{Score}_w(\bar{s}, \bar{t})$, é definida como:

$$\text{Score}_w(\bar{s}, \bar{t}) = \sum_{i=1}^{|\bar{s}|} w(\bar{s}[i], \bar{t}[i]).$$

Conceitos Preliminares

Similaridade

A **similaridade** de duas sequências s e t dada uma função de pontuação w , denotada por $\text{sim}_w(s, t)$, é a maior pontuação dentre as pontuações de todos os possíveis alinhamentos de s e t . Ou seja:

$$\text{sim}_w(s, t) = \max\{\text{Score}_w(\bar{s}, \bar{t})\},$$

onde (\bar{s}, \bar{t}) é um alinhamento de s e t .

Alinhamento ótimo

Um alinhamento $A = (\bar{s}, \bar{t})$ de s e t é dito **alinhamento ótimo** com respeito a uma função w se $\text{Score}_w(\bar{s}, \bar{t}) = \text{sim}_w(s, t)$.

Dadas as definições acima, o problema de medir quão parecidas são duas sequências pode ser formulado como o seguinte problema de otimização.

Problema do Alinhamento de Duas Sequências

Problema do Alinhamento de Duas Sequências - PADS

Dadas duas sequências s e t construídas sobre um alfabeto Σ qualquer tal que $\{-\} \notin \Sigma$ e uma função de pontuação w , encontrar o valor da similaridade $\text{sim}_w(s, t)$ dessas sequências.

Melhor alinhamento das sequências do exemplo anterior

CAGCCACTGGATTCTCG

| | | | ! ! / | | / | | | ! | | |

CAGC--GTGCATT-TC-

similaridade = 2

função de pontuação: $w(\mu, \mu) = 5$, $w(\mu, \nu) = -4$, $w(\mu, -) = -10$,
 $w(-, \nu) = -10$

Resolvendo o PADS

Método da força bruta

Uma primeira ideia é utilizar a força bruta para resolver o problema: gere todos os alinhamento possíveis de duas sequências e devolva a pontuação daquele de maior valor.

Método extremamente ineficiente

Um algoritmo baseado na força bruta é muito ineficiente já que o número de possíveis alinhamentos de duas sequências de tamanho n é muito grande. Mais especificamente:

$$N(n) = \sum_{i=0}^n \frac{(2n-i)!}{(n-i)!(n-i)!i!}.$$

n	0	1	2	3	4	5	6	7	8	9	10
$N(n)$	1	3	13	63	321	1683	8989	48639	265729	1462563	8097453

Resolvendo o PADS

Solução mais eficiente

Uma solução mais rápida para o problema pode ser obtida observando que ele possui a *propriedade das subsoluções ótimas*

Propriedade das subsoluções ótimas para o problema

Se $\mathcal{A} = (A'A'')$ é um alinhamento ótimo de s e t , e A' e A'' são dois pedaços desse alinhamento, com A' correspondendo a um alinhamento de um prefixo s' de s e um prefixo t' de t e A'' a um alinhamento de um sufixo s'' de s e um sufixo t'' de t , então ambos os alinhamentos são ótimos.

Essa propriedade pode ser provada por contradição supondo que A' ou A'' não são ótimos e, a partir disso, mostrando que \mathcal{A} não é ótimo.

Resolvendo o PADS

Utilizando a propriedade das subsoluções ótimas, podemos considerar a última coluna do alinhamento \mathcal{A} como A'' . Nesse caso, observe que existem apenas três possibilidades para a escolha dos caracteres componentes dessa coluna:

- emparelhar o último caractere de s com um espaço (em t);
- emparelhar o último caractere de t com um espaço (em s);
- emparelhar os dois últimos caracteres de s e t .

Disso, podemos pensar nos passos recursivos do algoritmo a seguir para o cálculo da similaridade de duas sequências s e t .

Algoritmos para o PADS

Algoritmo 1 CALCULA_SIM_RECURSIVO(s, t, w)

Entrada: duas sequências s e t e uma função de pontuação w .

Saída: valor da similaridade de s e t com respeito a w ;

```
1:  $m \leftarrow |s|$ ;  $n \leftarrow |t|$ ; Score  $\leftarrow 0$ ;  
2: se  $m = 0$  então  
3:   para  $i \leftarrow 1$  até  $n$  faça  
4:     Score  $\leftarrow$  Score +  $w(-, t[i])$ ;  
5:   fimpara  
6:   devolva Score;  
7: fimse  
8: se  $n = 0$  então  
9:   para  $i \leftarrow 1$  até  $m$  faça  
10:    Score  $\leftarrow$  Score +  $w(s[i], -)$ ;  
11:   fimpara  
12:   devolva Score;  
13: fimse  
14: Score1  $\leftarrow$  CALCULA_SIM_RECURSIVO( $s[1..m-1], t[1..n], w$ ) +  $w(s[m], -)$ ;  
15: Score2  $\leftarrow$  CALCULA_SIM_RECURSIVO( $s[1..m-1], t[1..n-1], w$ ) +  $w(s[m], t[n])$ ;  
16: Score3  $\leftarrow$  CALCULA_SIM_RECURSIVO( $s[1..m], t[1..n-1], w$ ) +  $w(-, t[n])$ ;  
17: Devolva max{Score1, Score2, Score3};
```

Algoritmos para o PADS

Sobre o algoritmo recursivo

O algoritmo mostrado não possui uma aplicabilidade prática dada sua complexidade de tempo exponencial no tamanho das sequências de entrada.

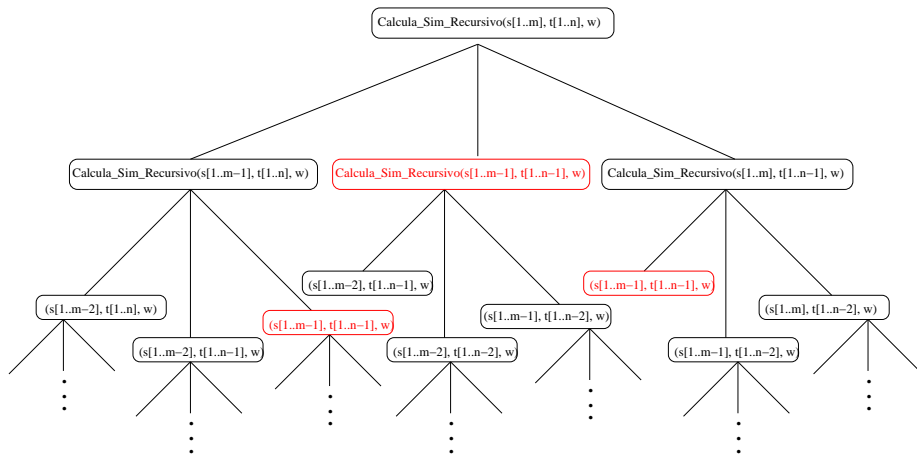
Complexidade do algoritmo recursivo

O desenvolvimento da recorrência $T(m, n) = T(m - 1, n) + T(m - 1, n - 1) + T(m, n - 1)$, representativa do tempo consumido pelo algoritmo, possui como resultado $T(m, n) = \Omega(3^n)$, no caso em que $m = n$.

Note que o Algoritmo 1 realiza cálculos redundantes.

- A similaridade de $s[1..m - 1]$ e $t[1..n - 1]$, por exemplo, é calculada três vezes por `CALCULA_SIM_RECURSIVO`: na chamada com os parâmetros $(s[1..m - 1], t[1..n - 1], w)$, $(s[1..m], t[1..n - 1], w)$ e $(s[1..m - 1], t[1..n], w)$.

Algoritmos para o PADS



Algoritmo Eficiente para o PADS

Se determinadas em uma certa ordem e armazenadas corretamente, as soluções dos subproblemas redundantes (similaridades de partes das sequências) podem ser reutilizadas para a solução eficiente do problema principal (similaridade das sequência como um todo).

As ideias acima constituem o núcleo de uma estratégia para resolução de problemas conhecida como *programação dinâmica*.

Resolvendo o problema com programação dinâmica

Por meio da programação dinâmica, a similaridade de duas sequências s e t , com $|s| = m$ e $|t| = n$, pode ser calculada eficientemente guardando-se as soluções dos subproblemas em uma matriz M , de dimensão $(m+1) \times (n+1)$, indexada por $\{0 \dots m\}$ e $\{0 \dots n\}$, conhecida como *matriz de alinhamento*.

Algoritmo Eficiente para o PADS

Matriz de alinhamento

Cada célula $M[i][j]$ da matriz de alinhamento guarda a similaridade dos prefixos de s e t que incluem, respectivamente, os i e j primeiros caracteres dessas sequências.

Após o preenchimento total da matriz, a similaridade procurada pode ser encontrada na última célula $M[|s|][|t|]$ de M

Preenchendo a matriz

Para que todas as células de M possam ser calculadas eficientemente, ela deve ser preenchida linha a linha, em ordem crescente de suas colunas.

Um preenchimento coluna a coluna, em ordem crescente de linhas, também funcionaria, mas seria mais lento na prática.

Algoritmo Eficiente para o PADS

Preenchendo a matriz

- a primeira posição de M é preenchida com zero, e corresponde à similaridade de duas sequências vazias;
- a primeira linha de M é preenchida com múltiplos dos valores associados às colunas do alinhamento com um espaço ($M[0][k] = k \times w(-, t[k])$, para $0 < k \leq |t|$);
- a primeira coluna de M também é preenchida com múltiplos dos valores associados às colunas com um espaço ($M[k][0] = k \times w(s[k], -)$, para $0 < k \leq |s|$);
- as outras posições de M são preenchidas com base na seguinte recorrência:

$$M[i][j] = \max \begin{cases} M[i][j-1] + w(-, t[j]); \\ M[i-1][j-1] + w(s[i], t[j]); \\ M[i-1][j] + w(s[i], -). \end{cases} \quad (1)$$

Algoritmo Eficiente para o PADS

Exemplo de matriz de alinhamento preenchida

Matriz de alinhamento preenchida para duas sequências $s = \text{ACGT}$ e $t = \text{ACC}$, e uma função de pontuação w tal que $w(\mu, \mu) = 1$, $w(\mu, \nu) = -1$ e $w(\mu, -) = w(-, \nu) = -2$

		A	C	C	
		1	2	3	
	0	0	← -2	← -4	← -6
A	1	↑ -2	↖ 1	← -1	← -3
C	2	↑ -4	↑ -1	↖ 2	← 0 ↖
G	3	↑ -6	↑ -3	↑ 0	↖ 1
T	4	↑ -8	↑ -5	↑ -2	↑ -1 ↖

Observe que as colunas e linhas da matriz encontram-se indexadas também pelas sequências dadas como entrada.

Algoritmo de Needleman-Wunsch

Seguindo a Recorrência 1, o algoritmo abaixo, devido a Needleman e Wunsch, calcula eficientemente a similaridade de duas sequências s e t dada uma função de pontuação w .

Algoritmo 2 CALCULA_SIMILARIDADE(s, t, w)

Entrada: duas sequências s e t e uma função de pontuação w .

Saída: valor da similaridade de s e t com respeito a w ;

```

1: para  $k \leftarrow 0$  até  $|t|$  faça
2:    $M[0][k] \leftarrow k \times w(-, t[k]);$ 
3: fimpara
4: para  $k \leftarrow 0$  até  $|s|$  faça
5:    $M[k][0] \leftarrow k \times w(s[k], -);$ 
6: fimpara
7: para  $i \leftarrow 1$  até  $|s|$  faça
8:   para  $j \leftarrow 1$  até  $|t|$  faça
9:      $M[i][j] \leftarrow \max(M[i-1][j] + w(s[i], -), M[i-1][j-1] + w(s[i], t[j]),$ 
10:       $M[i][j-1] + w(-, t[j]));$ 
11:   fimpara
12: fimpara
13: Devolva  $M[|s|][|t|];$ 

```

Algoritmo de Needleman-Wunsch

Complexidade do algoritmo

Como o Algoritmo 2 envolve apenas o preenchimento de uma matriz de dimensão $(m+1) \times (n+1)$, podemos concluir que ele realiza sua tarefa em tempo e espaço $O(mn)$

Melhoras na complexidade de espaço do algoritmo

Com algumas modificações no algoritmo apresentado, a similaridade de duas sequências pode ser calculada em espaço $O(n)$, onde n é o tamanho da menor sequência.

Algoritmo de Hirschberg

Com base na estratégia de divisão e conquista, Hirschberg propôs em 1975 um algoritmo que permite determinar a similaridade e o alinhamento de duas sequências em espaço $O(n)$, onde n é o tamanho da menor sequência.

Determinação de um Alinhamento Ótimo

Sobre o alinhamento ótimo

Um alinhamento ótimo pode ser construído iniciando-se na posição $M[|s|][|t|]$ da matriz e voltando-se nela pelo caminho inverso daquele utilizado para seu preenchimento, até que a posição $M[0][0]$ seja alcançada.

Construindo o alinhamento ótimo

Dada uma posição qualquer $M[i][j]$ da matriz de alinhamento, sabemos que ela foi calculada de acordo com uma das três possibilidades abaixo:

- 1 utilizando valor em $M[i-1][j]$; nesse caso a coluna do alinhamento inclui um espaço em t emparelhado com o caractere $s[i]$;
- 2 ou utilizando o valor em $M[i-1][j-1]$; nesse caso a coluna do alinhamento inclui o caractere $s[i]$ emparelhado com o caractere $t[j]$;
- 3 ou utilizando o valor em $M[i][j-1]$; nesse caso a coluna do alinhamento inclui um espaço em s emparelhado com o caractere $t[j]$.

Determinação de um Alinhamento Ótimo

Exemplo de um alinhamento ótimo das sequências $s = \text{ACGT}$ e $t = \text{ACC}$

ACGT

AC-C

A determinação de qual célula foi utilizada para o cálculo de $M[i][j]$ pode ser feita por meio de um cálculo simples ou guardando-se um “ponteiro” para essa célula.

Mais de um alinhamento ótimo

Note que podem existir mais de um alinhamento ótimo de duas sequências.

ACGT

ACC-

é outro alinhamento ótimo das sequências $s = \text{ACGT}$ e $t = \text{ACC}$.

Algoritmo para Construir um Alinhamento Ótimo

O Algoritmo 3 recebe como entrada duas sequências s e t , uma matriz de alinhamento preenchida e devolve um alinhamento ótimo de s e t .

Algoritmo 3 CONSTROI_ALINHAMENTO(s, t, M)

Entrada: duas sequências s e t e uma matriz de alinhamento M .

Saída: um dos alinhamentos ótimos de s e t .

```

1:  $i \leftarrow |s|$ ;  $j \leftarrow |t|$ ;  $k \leftarrow 0$ ;
2: enquanto  $i \neq 0$  e  $j \neq 0$  faça
3:   se  $M[i][j] = M[i-1][j-1] + w(s[i], t[j])$  então
4:      $\bar{s}[k] \leftarrow s[i]$ ;  $\bar{t}[k] \leftarrow t[j]$ ;  $i \leftarrow i-1$ ;  $j \leftarrow j-1$ ;  $k \leftarrow k+1$ ;
5:   senão se  $M[i][j] = M[i][j-1] + w(-, t[j])$  então
6:      $\bar{s}[k] \leftarrow '-'$ ;  $\bar{t}[k] \leftarrow t[j]$ ;  $j \leftarrow j-1$ ;  $k \leftarrow k+1$ ;
7:   senão se  $M[i][j] = M[i-1][j] + w(s[i], -)$  então
8:      $\bar{s}[k] \leftarrow s[i]$ ;  $\bar{t}[k] \leftarrow '-'$ ;  $i \leftarrow i-1$ ;  $k \leftarrow k+1$ ;
9:   fimse
10: fimenquanto
11: enquanto  $i \neq 0$  faça  $\bar{s}[k] \leftarrow s[i]$ ;  $\bar{t}[k] \leftarrow '-'$ ;  $i \leftarrow i-1$ ;  $k \leftarrow k+1$ ;
12: enquanto  $j \neq 0$  faça  $\bar{s}[k] \leftarrow '-'$ ;  $\bar{t}[k] \leftarrow t[j]$ ;  $j \leftarrow j-1$ ;  $k \leftarrow k+1$ ;
13: Devolva  $(\bar{s}, \bar{t})$  invertidos;
```

Algoritmo para Construir um Alinhamento Ótimo

Complexidade do algoritmo

Como a cada iteração o algoritmo `CONSTROI_ALINHAMENTO` anda uma linha para cima ou uma coluna para esquerda na matriz, teremos um total de $m + n$ iterações do algoritmo até que a posição $M[0][0]$ seja alcançada. Disso, o alinhamento das duas sequências de entrada pode ser construído em tempo $O(m + n)$.

Alinhamento global de suas sequências

Os alinhamentos de duas sequências visto até aqui são ditos alinhamentos **globais** (por considerarem as sequências como um todo).

O PADS é também conhecido como **Problema do Alinhamento Global de Duas Sequências**.

Alinhamento Local

Apesar de útil, um alinhamento global pode não revelar segmentos parecidos de duas sequências sendo comparadas.

Ex.: sequências $s = \text{CTTCAGCACTTGGATTCTCGG}$ e $t = \text{AGCCACCTGCGGC}$

Um alinhamento global ótimo de s e t ($w(\mu, \mu) = 1$, $w(\mu, \nu) = -1$ e $w(\mu, -) = w(-, \nu) = -2$) é:

```
CTTCAGCACTTGGATTCTCGG
AGCCA-C-CT-G----CG-GC
```

Note porém que as sequências possuem segmentos bem parecidos que não conseguimos identificar através do alinhamento global ótimo.

```
-----CACTTG-----
-----CACCTG-----
```

Alinhamento Local

Alinhamento local

Um **alinhamento local** de duas sequências é um alinhamento que envolve apenas segmentos delas, e não as sequências como um todo.

Problema do Alinhamento Local de Duas Sequências - PALDS

Dadas duas sequências s e t construídas sobre um alfabeto Σ qualquer tal que $\{-\} \notin \Sigma$ e uma função de pontuação w , encontrar um segmento s' de s e um segmento t' de t tal que $\text{sim}_w(s', t')$ seja máxima.

Resolvendo o PALDS

O algoritmo para determinação do alinhamento local ótimo de duas sequências s e t é devido a Smith e Waterman e corresponde a uma variante do algoritmo de Needleman-Wunsch.

Alinhamento Local

Resolvendo o PALDS (também) com programação dinâmica

Dada uma matriz M de alinhamento de dimensão $(m + 1) \times (n + 1)$, cujas células armazenam a pontuação do melhor alinhamento (global) envolvendo um sufixo de $s[1..i]$ e um sufixo de $t[1..j]$, ela pode ser preenchida de acordo com a seguinte recorrência:

$$M[i][j] = \max \begin{cases} M[i][j - 1] + w(-, t[j]); \\ M[i - 1][j - 1] + w(s[i], t[j]); \\ M[i - 1][j] + w(s[i], -); \\ 0. \end{cases} \quad (2)$$

A possibilidade de se escolher o zero justifica-se pelo fato de sempre existir um alinhamento de sufixos vazios de $s[1..i]$ e $t[1..j]$, de pontuação zero.

Pelo mesmo motivo apresentado acima, a primeira linha e coluna da matriz são preenchidas com zero.

Alinhamento Local

Pontuação do alinhamento local ótimo

Preenchida a matriz, a pontuação do alinhamento local ótimo de s e t corresponde ao maior valor que pode ser encontrado nas células de M .

Complexidade do algoritmo

Como a determinação da pontuação de um alinhamento local ótimo de s e t , com $|s| = m$ e $|t| = n$, envolve o preenchimento de uma matriz de dimensão $(m+1) \times (n+1)$ e uma busca pelo maior valor nesta matriz, ela também pode ser feita em tempo e espaço $O(mn)$.




Construindo o alinhamento local ótimo

Um alinhamento local ótimo pode ser construído iniciando-se na célula de maior valor e terminando em uma célula contendo o valor zero.

Alinhamento Local

Exemplo de matriz de alinhamento local preenchida

Matriz de alinhamento local preenchida para duas sequências $s = \text{ACGT}$ e $t = \text{ACC}$, e uma função de pontuação w tal que $w(\mu, \mu) = 1$, $w(\mu, \nu) = -1$ e $w(\mu, -) = w(-, \nu) = -2$:

		A	C	C
		1	2	3
	0	0	0	0
A	1	0	 1	0
C	2	0	0	 2
G	3	0	0	 1
T	4	0	0	0

Alinhamento Semiglobal

Apesar de úteis, os alinhamentos globais e locais não revelam semelhanças entre duas sequências quando uma delas é muito parecida com um segmento da outra.

Ex.: sequências $s = \text{CTTCAGCACTTGGATTCTCGG}$ e $t = \text{AGCCACCTGCGGC}$

Um alinhamento global (à esquerda) e local (à direita) ótimo de s e t ($w(\mu, \mu) = 1$, $w(\mu, \nu) = -1$ e $w(\mu, -) = w(-, \nu) = -2$) é:

CTTCAGCACTTGGATTCTCGG	-----CACTTG-----
AGCCA-C-CT-G----CG-GC	-----CACCTG-----

Note porém que t é muito parecida com um segmento de s , que fica claro ao olharmos para o seguinte alinhamento de s e t (que não é ótimo):

CTTCAG-CACTTG-GATTTCTCGG
----AGCCACCTGCGGC-----

Alinhamento Semiglobal

Alinhamento semiglobal

Um **alinhamento semiglobal** de duas sequências é um alinhamento global cuja pontuação não considera colunas com espaços nas extremidades do alinhamento.

Problema do Alinhamento Semiglobal de Duas Sequências - PASDS

Dadas duas sequências s e t construídas sobre um alfabeto Σ qualquer tal que $\{-\} \notin \Sigma$ e uma função de pontuação w , encontrar o valor da similaridade $\text{sim}_w(s, t)$ delas sem considerar espaços em ambas as pontas de ambas as sequências.

Resolvendo o PASDS

O algoritmo para determinação do alinhamento semiglobal ótimo de duas sequências s e t também corresponde a uma variante do algoritmo de Needleman-Wunsch.

Alinhamento Semiglobal

Resolvendo o PASDS (também) com programação dinâmica

De fato, a recorrência para se resolver o PASDS é a mesma utilizada para resolver o problema do alinhamento global de duas sequências. As únicas diferenças são:

- na inicialização da matriz, a primeira linha e coluna são preenchidas com zero;
- o valor da similaridade procurada corresponde ao maior valor presente na última linha ou coluna da matriz.

Alinhamento Semiglobal

Complexidade do algoritmo

Como a determinação da pontuação de um alinhamento semiglobal ótimo de s e t , com $|s| = m$ e $|t| = n$, envolve o preenchimento de uma matriz de dimensão $(m + 1) \times (n + 1)$ e uma busca pelo maior valor na última linha e coluna da matriz, ela também pode ser feita em tempo e espaço $O(mn)$.

Construindo um alinhamento semiglobal ótimo

Um alinhamento semiglobal ótimo pode ser construído iniciando-se na célula de maior valor presente na ultima linha ou coluna da matriz e voltando-se nela até que a célula $M[0][0]$ seja alcançada.

Falando Sobre a Similaridade

Medida de difícil interpretação

Interpretar o valor da similaridade de duas sequências é difícil. O que significa dizer que a similaridade de duas sequências é 200, por exemplo?

Mais ainda

Da forma como foi definida, é bem possível que um bom alinhamento de duas sequências longas tenha uma pontuação maior que um ótimo alinhamento de duas sequências pequenas:

Mesmo assim....

- a similaridade ainda é útil para a comparação relativa de n sequências de um conjunto;
- é fácil, a partir de um alinhamento ótimo, calcular o grau de semelhança entre duas sequências: conte o número de colunas com caracteres iguais e divida pelo tamanho do alinhamento.

Falando Sobre Funções de Pontuação

Como escolher uma função de pontuação

Na prática, ao comparamos duas sequências, estamos interessados em alinhamentos que revelem semelhanças entre elas. Por isso:

- é comum fazer uso de funções de pontuação que penalizem *mismatches* e *spaces* e valorizem *matches*;
- a função de pontuação a ser utilizada pode variar de acordo com as sequências sendo comparadas;
 - na comparação de sequências de DNAs, é comum o uso de uma matriz denominada EDNAFULL;
 - na comparação de sequências de proteínas, é comum o uso de matrizes denominadas PAM e BLOSUM.

Distância de Edição

Distância de edição

Além da similaridade, existe uma outra medida utilizada para comparar duas sequências denominada **Distância de Edição**, que mede a quantidade de operações necessárias para transformar uma sequência na outra.

Conceitos relacionados

Seja uma função $d : E \times E \mapsto \mathbb{R}$, onde $E \neq \emptyset$. Dizemos que d é uma **métrica** sobre E se, e somente se, $\forall x, y, z \in E$, temos que:

- $d(x, y) \geq 0$ (positividade);
- $d(x, y) = 0 \iff x = y$ (separação);
- $d(x, y) = d(y, x)$ (simetria);
- $d(x, z) \leq d(x, y) + d(y, z)$ (desigualdade triangular).

Distância de Edição

Operações de edição

Podemos usar três operações, ditas **operações de edição**, para transformar uma sequência em outra: inserção, substituição e remoção de um caractere.

Métrica para distância

Uma métrica $\delta : \bar{\Sigma} \times \bar{\Sigma} \mapsto \mathbb{R}$ funciona como uma função de pontuação para as operações de edição: dados dois caracteres $a, b \in \Sigma$, o valor da substituição de a por b é dado por $\delta(a, b)$, o valor da remoção de um a é dado por $\delta(a, -)$ e o valor da inserção de um b é dado por $\delta(-, b)$.

Distância de Edição - definição formal

Dadas duas sequências s e t construídas sobre um alfabeto Σ e uma métrica $\delta : \bar{\Sigma} \times \bar{\Sigma} \mapsto \mathbb{R}$, a soma dos valores da menor sucessão de operações de edição que transformam s em t é chamada de **δ -distância de edição** e denotada por $d_\delta(s, t)$.

Distância de Edição

Distância de Levenshtein

A métrica δ sobre $\bar{\Sigma}$ tal que para todo $a, b \in \bar{\Sigma}$, $\delta(a, b) = 0$ se $a = b$ e $\delta(a, b) = 1$ caso contrário é chamada **distância de Levenshtein** e é denotada por d_E .

Similaridade \times Distância de Edição

Enquanto que a similaridade **mede quão parecidas** duas sequências são, a distância de edição **mede quão diferentes** duas sequências são.

Como calcular a distância de edição entre duas sequências

Podemos usar a mesma ideia do algoritmo que calcula a similaridade global de duas sequências para calcular a distância de edição entre elas:

- basta usar a métrica δ no lugar da função de pontuação w e tomar o mínimo dentre $M[i][j - 1] + \delta(-, t[j])$, $M[i - 1][j - 1] + \delta(s[i], t[j])$, $M[i - 1][j] + \delta(s[i], -)$ durante o preenchimento da matriz.

Programas de Alinhamento

Programas de Alinhamento

Existem vários programas de alinhamento de sequências disponíveis:

- [http://www.ebi.ac.uk/Tools/psa/;](http://www.ebi.ac.uk/Tools/psa/)
- [http://www.ch.embnet.org/software/LALIGN_form.html;](http://www.ch.embnet.org/software/LALIGN_form.html)
- [http://ffas.ljcrf.edu/ffas-cgi/cgi/pair_aln.pl?ses=;](http://ffas.ljcrf.edu/ffas-cgi/cgi/pair_aln.pl?ses=)
- ...

Apesar de teoricamente eficientes, programas baseados nos algoritmos de alinhamento vistos são proibitivos na prática.

Heurísticas para Comparação de Sequências

Princípio das heurísticas para comparação de sequências

- “algoritmos” onde regras empíricas sobre comparação de sequências são usadas na busca de uma boa solução;
Ex. Se duas sequências possuem mais do que 100 aminácidos (nucleotídeos), você pode considerá-las homólogas se 25% dos seus aminoácidos (75% dos seus nucleotídeos) são idênticos.
- quase sempre encontram uma boa solução para o problema de se comparar duas sequências, apesar de não garantir a solução ótima;
- são de 50 a 100 vezes mais rápidas que os algoritmos apresentados anteriormente.

Algoritmo versus Heurística

Os algoritmos sempre devolvem a solução ótima, mas a um custo alto (de tempo e memória); já as heurísticas devolvem uma solução boa, mas a um baixo custo (de tempo e memória).

BLAST

Sobre o BLAST

- Uma das heurísticas de comparação de sequências mais utilizados no mundo;
- Muito útil na comparação de uma sequência com um base de dados de sequências;
- Baseada na busca por similaridades locais de sequências.

Sequências *query* e *subject*

No BLAST, a sequência *query* corresponde à sequência sendo analisada, enquanto que a(s) sequência(s) *subject* corresponde(m) à(s) sequência(s) encontradas na base de dados com a(s) qual(is) a *query* guarda semelhanças.

BLAST

www.ncbi.nlm.nih.gov/blast/Blast.cgi?CMD=Web&PAGE_TYPE=BlastHome

shustring sequences

BLAST® Basic Local Alignment Search Tool

Home Recent Results Saved Strategies Help

My NCBI Sign In (Registered)

NCBI BLAST Home

BLAST finds regions of similarity between biological sequences. [more...](#)

New DELTA-BLAST, a more sensitive protein-protein search [Go](#)

BLAST Assembled RefSeq Genomes

Choose a species genome to search, or [list all genomic BLAST databases](#).

- Human
- Mouse
- Rat
- Arabidopsis thaliana*
- Oryza sativa*
- Bos taurus*
- Danio rerio*
- Drosophila melanogaster*
- Gallus gallus*
- Pan troglodytes*
- Microbes
- Apis mellifera*

Basic BLAST

Choose a BLAST program to run.

nucleotide_blast	Search a nucleotide database using a nucleotide query Algorithms: blastn, megablast, discontinuous megablast
protein_blast	Search protein database using a protein query Algorithms: blastp, psi-blast, phi-blast, delta-blast
blastx	Search protein database using a translated nucleotide query
tblastn	Search translated nucleotide database using a protein query
tblastx	Search translated nucleotide database using a translated nucleotide query

Specialized BLAST

Choose a type of specialized search (or database name in parentheses.)

Your Recent Results [New](#)

[\(3\) - seq](#)

[\(2\) - seq](#)

[seq](#)

News

[BLAST 2.2.27+ released](#)

A new version of the stand-alone BLAST applications has been released.
Mon, 10 Sep 2012 14:00:00 EST

[More BLAST news...](#)

Tip of the Day

[How to save custom search pages.](#)

So you have made a few BLAST searches and after adjusting the database, organism limits and maybe a few Algorithm Parameters you arrive at what you think is a good search strategy.

[More tips...](#)

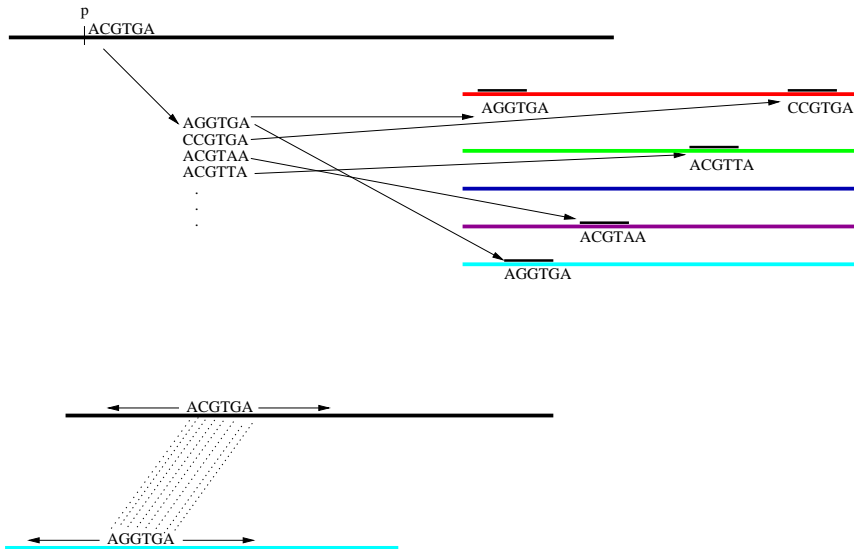
`http://blast.ncbi.nlm.nih.gov/Blast.cgi.`

Ideia de Funcionamento do BLAST

Passos do BLAST

- 1 Para cada posição p da sequência *query*, constrói-se uma lista de palavras de tamanho w que pontuam acima de um valor T quando pareadas com a palavra começando em p ;
- 2 Para cada palavra da lista, identifica-se todos os “casamentos exatos” com segmentos das sequências da base de dados;
- 3 Cada casamento exato, denominado *hit*, é estendido em ambas as direções, com base na sequência *query*, parando quando a pontuação atingir um limitante inferior; o resultado dessa extensão é chamado *HSP* (de *high scoring pair*);
 - observação: dois *hits* muito próximos podem ser juntados para depois serem estendido.
- 4 Devolve todos os HSPs com uma pontuação acima de um limitante inferior I ;

Ideia do funcionamento do BLAST



Motivação e Conceitos Preliminares

Motivação

Em algumas ocasiões, a comparação de duas sequência pode não revelar regiões conservadas dessas sequências.

Alinhamento de várias sequências

Uma generalização natural da tarefa de comparar duas sequências consiste em alinhar não apenas duas, mas um número qualquer $k > 2$ sequencias.

Alinhamento múltiplo

Inserção de espaços em posições arbitrárias de um número k de sequencias de forma que todas elas fiquem do mesmo tamanho e que nenhuma coluna do alinhamento seja composta apenas de espaços.

Alinhamento Múltiplo

Alinhamento múltiplo - definição formal

Um alinhamento múltiplo A de k seqüências s_1, s_2, \dots, s_k consiste de uma k -tupla $A = (\bar{s}_1, \bar{s}_2, \dots, \bar{s}_k)$, com $\bar{s}_1, \dots, \bar{s}_k \in \bar{\Sigma}^*$ tal que:

- 1 $|\bar{s}_1| = |\bar{s}_2| = \dots = |\bar{s}_k|$;
- 2 $\bar{s}_i|_{\Sigma} = s_i$ para todo $i = 1, \dots, k$;
- 3 Não existe j tal que $\bar{s}_i[j] = '-'$ para todo $i = 1, \dots, k$.

Função de pontuação

Nesse contexto, temos uma **função de pontuação** $w : (\bar{\Sigma})^k \rightarrow \mathbb{R}$ que associa um certo valor real para cada k -tupla de símbolos (μ_1, \dots, μ_k) em $\bar{\Sigma}$.

Conceitos Preliminares

Pontuação de um alinhamento múltiplo

Dada uma função de pontuação w qualquer, a **pontuação** $\text{Score}_w(A)$ de um alinhamento múltiplo qualquer $A = (\bar{s}_1, \dots, \bar{s}_k)$ de k sequencias é definida como a soma da pontuação de cada uma de suas colunas.

Similaridade

A **similaridade** $\text{sim}_w(s_1, \dots, s_k)$ de k sequencias s_1, \dots, s_k , dada uma função de pontuação w , é definida como a maior pontuação dentre as pontuações de todos os possíveis alinhamentos múltiplos de s_1, \dots, s_k .

$$\text{sim}_w(s_1, \dots, s_k) = \max\{\text{Score}_w(\bar{s}_1, \dots, \bar{s}_k)\},$$

onde $(\bar{s}_1, \dots, \bar{s}_k)$ é um alinhamento múltiplo de s_1, \dots, s_k .

Conceitos Preliminares

Alinhamento múltiplo ótimo

Um **alinhamento múltiplo ótimo** A^* de k sequencias s_1, s_2, \dots, s_k é definido como um alinhamento tal que $\text{Score}_w(A^*) = \text{sim}_w(s_1, \dots, s_k)$.

Problema do Alinhamento Múltiplo - PAM

Dadas k sequencias s_1, \dots, s_k construídas sobre um alfabeto Σ qualquer tal que $\{-\} \notin \Sigma$ e uma função de pontuação w , encontrar o valor da similaridade $\text{sim}_w(s_1, \dots, s_k)$ dessas sequências.

Resolução do PAM e complexidade

Resolvendo o PAM

O PAM também pode ser resolvido por programação dinâmica usando uma matriz de alinhamento M de k dimensões, tal que

- Cada célula $M[i][j][u] \dots [v]$ da matriz de alinhamento guarda a similaridade dos prefixos de $s_1, s_2, s_3, \dots, s_k$ que incluem, respectivamente, os i, j, u, \dots, v primeiros caracteres das sequências.

Dificuldade do problema

No PAM, a estratégia de programação dinâmica leva a um algoritmo ineficiente, com complexidade de tempo igual a $O(n^k 2^k)$, onde n é o tamanho das sequências.

Complexidade do problema

De fato, o PAM é NP-completo!

PAM com Função de Pontuação SP

Função de pontuação SP

Dada uma função de pontuação $w : \bar{\Sigma} \times \bar{\Sigma} \rightarrow \mathbb{R}$, a **função de pontuação SP** corresponde a

$$SP_w(C) = \sum_{1 \leq i \leq i' \leq k} w(C[i], C[i']),$$

onde C é uma coluna do alinhamento e $C[i]$ denota o i -ésimo caractere dessa coluna.

Pontuação de um alinhamento múltiplo sob a função de pontuação SP

A pontuação de um alinhamento múltiplo A com base na função de pontuação SP corresponde à soma da pontuação de cada uma de suas colunas dada por SP_w .

PAM com Função de Pontuação SP

Alinhamento de duas sequências induzido por um alinhamento múltiplo

Um alinhamento de duas sequências s_i e s_j **induzido** por um alinhamento múltiplo A ($A|_{s_i s_j}$) corresponde ao alinhamento obtido pela remoção de todos os elementos da k -tupla $A = (\bar{s}_1, \dots, \bar{s}_k)$, com exceção de \bar{s}_i e \bar{s}_j , e então das possíveis posições i dessas sequências tais que $\bar{s}[i] = - = \bar{t}[i]$.

Pontuação de um alinhamento com a função de pontuação SP

dado um alinhamento múltiplo A de k sequências s_1, \dots, s_k , sua pontuação com base na função de pontuação SP também pode ser definida como

$$\text{Score}_{\text{SP}_w}(A) = \sum_{i=1}^k \sum_{j=1}^{i-1} \text{Score}_w(A|_{s_i s_j}).$$

Com a função de pontuação SP, o PAM também é NP-completo!

Aproximação para o PAM com Função de Pontuação SP

Um algoritmo de aproximação para o problema do alinhamento múltiplo com função de pontuação SP foi desenvolvido por Gusfield em 1993.

Alinhamentos compatíveis

Um alinhamento múltiplo A de k seqüências $S = (s_1, \dots, s_k)$ é **compatível** com um alinhamento A' de duas seqüências quaisquer s_i e s_j em S se a pontuação do alinhamento induzido por A dessas duas seqüências é igual à pontuação de A' .

Teorema

Dada uma árvore qualquer T onde cada nó v representa uma seqüência s_v e cada aresta uv um alinhamento ótimo das seqüências s_u e s_v , existe um alinhamento múltiplo de todas as seqüências representadas pelos vértices de T que é compatível com os alinhamentos de pares representados pelas arestas dessa árvore.

Aproximação para o PAM com Função de Pontuação SP

Prova algorítmica

Algoritmo que, dada uma árvore qualquer T , devolve em tempo polinomial um alinhamento múltiplo das sequências representadas pelos vértices de T compatível com os alinhamentos de pares representados pelas suas arestas.

Definições preliminares

Dada uma sequência qualquer s e um conjunto $\mathcal{A} = \{A_1, \dots, A_n\}$ de alinhamentos de s e outras n sequências:

- $p_{s_{i,j}}$: posição do caractere $s[j]$ em um alinhamento $A_i \in \mathcal{A}$;
- $z_{s_{i,j}}$: número de espaços imediatamente anteriores ao caractere $s[j]$ em A_i ($z_{s_{i,j}} = p_{s_{i,j}} - p_{s_{i,j-1}} - 1$);
- $z_{s_{1..n,j}}^*$: maior número de espaços imediatamente anteriores ao caractere $s[j]$ tomando-se o conjunto $\mathcal{A} = \{A_1, \dots, A_n\}$ ($z_{s_{1..n,j}}^* = \max z_{s_{i,j}}$ para $i = 1, \dots, n$).

Aproximação para o PAM com Função de Pontuação SP

Passos gerais do Algoritmo

Seja s uma sequência de tamanho m envolvida em pelo menos dois alinhamentos distintos $A_1 = (\bar{s}, \bar{t})$ e $A_2 = (\bar{s}, \bar{u})$ representados por duas arestas, st e su , de T respectivamente.

- 1 Tome um desses alinhamentos ($A_1 = \{\bar{s}, \bar{t}\}$, por exemplo) e considere a sequência u ligada à s por meio da aresta su representativa de A_2 ;
- 2 determine o valor de $z_{s_{1..2},j}^*$ tomando-se o conjunto $A = \{A_1, A_2\}$, para todo $1 \leq j \leq m+1$ (considere que existe um caractere fictício após a última posição da sequência);
- 3 insira um total de $z_{s_{1..2},j}^* - z_{s_i,j}$ colunas formadas apenas por espaços nas posições $p_{s_i,j} - 1$ do alinhamento A_i com o menor número de espaços imediatamente anteriores a $s[j]$, para todo $0 \leq j \leq m+1$;
 - 1 Sejam \bar{A}_1 e \bar{A}_2 os alinhamentos obtidos após a inserção das colunas contendo apenas espaços e \bar{s} , \bar{t} e \bar{u} as sequências desses alinhamentos.
- 4 estenda o alinhamento \bar{A}_1 pela adição da sequência \bar{u} e remova de T a aresta su ;
- 5 repita os passos acima (com $A_1 = \bar{A}_1 + \bar{u}$ e $A_2 =$ alinhamento qualquer envolvendo uma sequência já alinhada em A_1) até que T não possua mais arestas.

Aproximação para o PAM com Função de Pontuação SP

Complexidade do algoritmo

Tomando-se k como o número total de arestas da árvore e l como o tamanho das sequências alinhadas em A_1 no final do algoritmo, temos que o tempo total consumido por ele é de $O(kl) + O(kl) + O(kl^2) = O(kl^2)$.

Aproximação para o PAM de razão $2 - 2/k$

- 1 alinhe as sequências de entrada duas e duas e determine aquela mais parecida com as outras (sequência s_g);
- 2 agrupe os alinhamentos ótimos de s_g com todas as outras sequências utilizando os passos anteriores, em um alinhamento envolvendo as k sequências de entrada.

Aproximação para o PAM com Função de Pontuação SP

O que significa uma aproximação para o PAM de razão $2 - 2/k$?

Significa que, para qualquer instância do PAM, o algoritmo proposto vai calcular uma solução que seja, no máximo, $2 - 2/k$ vezes maior que a solução ótima.

Sobre a aproximação para o PAM

É importante observar que a aproximação descrita é válida somente para funções de pontuação que sejam métricas, como é o caso da distância de edição.