MO640/MC668

Guilherme P. Telles

IC-Unicamp

Avisado está

- Estes slides são incompletos.
- Estes slides contêm erros.

Parte I

Alinhamento e similaridade

Notação

- Um alfabeto Σ é um conjunto finito de símbolos (ou letras ou caracteres).
- Uma cadeia (ou seqüência) é uma sucessão ordenada de caracteres de um alfabeto.
- O tamanho de uma cadeia s, $|s|$, é o número de caracteres em s.
- Os símbolos em uma cadeia s são indexados de 1 até $|s|$.
- $s[i]$ é o caractere que ocupa a i-ésima posição em s.
- A cadeia vazia tem tamanho 0 e é denotada ε.

Notação

- A concatenação de duas cadeias s e t é a cadeia formada pelos símbolos de s seguidos pelos símbolos de t.
- É denotada pela justaposição. A concatenação de s e t é st.

Notação

- Uma subcadeia de uma cadeia s é uma cadeia formada por caracteres consecutivos de s.
- Se t é subcadeia de s, s é supercadeia de t.
- Denotamos $s[i, j]$ a subcadeia que contém os caracteres de i a j inclusive.
- Definimos que $s[i, j] = \varepsilon$ para qualquer combinação estranha de índices i e j: fora dos limites da cadeia, $i > j$ etc.

Notação

- Uma subsqüência de uma cadeia s é uma cadeia formada pela remoção de caracteres de s.
- Se t é subseqüência de s, s é superseqüência de t.

Notação

- Um prefixo de uma cadeia s é uma subcadeia da forma $s[1, i]$.
- A cadeia vazia ε é prefixo de qualquer cadeia.
- Um sufixo de uma cadeia s é uma subcadeia da forma $s[i, |s|]$.
- A cadeia vazia ε é sufixo de qualquer cadeia.

Notação

- O operador κ remove símbolos à esquerda ou à direita.

  $abc\kappa = ab$
  $\kappa abc = bc$
  $ab\kappa c$ é ambíguo.

Comparação de cadeias

- *Comparação de seqüências* ou *comparação de cadeias* são nomes genéricos para vários problemas envolvendo cadeias de caracteres.
- O problema pai-de-todos provavelmente é o *problema do casamento exato de cadeias*: dadas uma cadeia p chamada padrão e uma cadeia mais longa t chamada texto, encontrar todas as ocorrências de p em t.
- Vamos estudar alguns problemas dessa classe.

Comparação de seqüências biológicas

- Queremos identificar regiões parecidas e diferentes em seqüências.
- Acredita-se que seqüências similares implicam em função similar.
- Queremos admitir casamento de seqüências que têm mutações uma em relação à outra.

```
SEKLLTWGFR--------FFETVT-PIKPDATFVTQRVWFGDKSEVNLGAGEAGSV
SEKL  W +         FFE++T ++P + V V            +NLG
SEKLYAWNPKSTYIKSPPFFESLTLDLQPPKSIVDAYVL------LNLGDSVTTDH
```

Diferenças

- Queremos admitir:
  - Substituição de letras.
  - Inserção de letras.
  - Remoção de letras.
- Podemos definir o problema da comparação inexata entre duas cadeias através de alinhamentos e similaridade.

Alinhamento global

- Um *alinhamento global* entre duas cadeias s e t consiste na inserção de zero ou mais espaços em posições de s e de t, resultando em cadeias s' e t' tais que
  - s' e t' tenham o mesmo tamanho,
  - a remoção dos espaços de s' e t' resulte em s e t,
  - se $s'[i]$ for um espaço então $t'[i]$ não é um espaço e vice-versa.
- Vamos dizer que $s'[i]$ e $t'[i]$ formam uma *coluna* do alinhamento.

Pontuação

- Podemos definir uma pontuação para um alinhamento como

$$score(s', t') = \sum_{i=1}^{|s'|} c_i$$

onde c_i é a pontuação de uma coluna dada por um sistema de pontuação.

Pontuação de uma coluna

- A forma mais simples para atribuir pontuação a uma coluna de um alinhamento é:
  - um valor constante para a pontuação de dois caracteres iguais alinhados (match, $\mathcal{M}$),
  - um valor constante para dois caracteres diferentes aninhados (mismatch, $\mathcal{m}$) e
  - um valor constante para espaço (gap, $\mathcal{g}$).
- Outros esquemas para pontuação de espaços e de caracteres serão considerados em algum momento.

DNA e proteínas

- Um esquema bastante usado para DNA, obtido a partir da análise de alinhamentos pequenos: $\mathcal{M} = 1$, $m = -1$ e $g = -2$.
- Para proteínas temos as matrizes PAM e BLOSUM, que veremos mais de perto em algum momento.

Similaridade

- É o valor máximo de pontuação obtido por um alinhamento entre duas cadeias.
- Em outras palavras, é a pontuação de um alinhamento ótimo entre duas cadeias.

Problema

- Calcular a similaridade entre duas cadeias e construir um alinhamento com tal similaridade.

Solução por força-bruta

- Solução por força-bruta: gerar todos os alinhamentos possíveis entre as duas cadeias, calcular a pontuação de cada um e escolher o melhor.
  - Inviável porque o número de alinhamentos é exponencial.

Por indução

- Sejam s e t, $|s| = n$ e $|t| = m$.
- Vamos supor que temos um oráculo que sabe calcular a similaridade se as cadeias têm tamanhos

  $n - 1, m$ ou
  $n, m - 1$ ou
  $n - 1, m - 1$.
- Há três possibilidades para a última coluna do alinhamento entre $s[1, n]$ e $t[1, m]$:
  - podemos alinhar $s[n]$ e um espaço.
  - podemos alinhar um espaço e $t[m]$.
  - podemos alinhar $s[n]$ e $t[m]$.
-
- Dentre essas possibilidades, ficamos com a que nos dá a maior pontuação.

Por indução

- Se uma das duas seqüências s ou t têm tamanho 0 então sabemos calcular a similaridade.

Algoritmo recursivo

- Podemos escrever um programa recursivo para encontrar a similaridade diretamente da indução.

```
SIM(s[1,n], t[1,m])
1  if i == 0
2        return j * g;
3  if j == 0
4        return i * g;
5  l = SIM(s[1,n], t[1,m-1]) + g
6  u = SIM(s[1,n-1], t[1,m]) + g
7  d = SIM(s[1,n-1], t[1,m-1]) + (s[n] == t[m] ? M : m)
8  return max(l, u, d)
```

Algoritmo recursivo

- Para resolver um problema de tamanho $n + m$ o algoritmo resolve três problemas de tamanhos $n + m - 1$, $n + m - 1$ e $n - 1 + m - 1$.
- O algoritmo é exponencial, $\Omega(3^{\min\{n,m\}})$.

Número de subproblemas

- Para seqüências de tamanhos n e m há apenas $(n+1) \times (m+1)$ subproblemas.
- Então o algoritmo recursivo resolve o mesmo subproblema muitas vezes.

Programação dinâmica

- Uma matriz A armazena todas as soluções para os subproblemas.
- A célula $A[i, j]$ tem a similaridade entre $s[0, i]$ e $t[0, j]$.
- A similaridade entre s e t aparece na célula $A[|s|, |t|]$.

Construção de A

$$A[i,0] = i \cdot g$$
$$A[0,j] = j \cdot g$$

$$A[i,j] = \max \begin{cases} A[i-1,j] + g \\ A[i,j-1] + g \\ A[i-1,j-1] + p(i,j) \end{cases}$$

$$\text{onde } p(i,j) = \begin{cases} m \text{ se } s[i] = t[j] \\ n \text{ se } s[i] \neq t[j] \end{cases}$$

Algoritmo

SIMILARITY(s, t)

```
1   for i = 0 to n
2       A[i, 0] = i * g
3   for j = 0 to m
4       A[0, j] = j * g
5   for i = 1 to n
6       for j = 1 to m
7           l = A[i, j − 1] + g
8           u = A[i − 1, j] + g
9           d = A[i − 1, j − 1] + p(i, j)
10          A[i, j] = max(l, u, d)
11  return A[n, m]
```

Custo

- Cada célula da matriz é preenchida em tempo constante. São $(n+1)(m+1)$ células e então o algoritmo leva tempo $\Theta(nm)$.
- O algoritmo usa memória $\Theta(nm)$: a matriz e algumas variáveis.

Alinhamentos

- A partir da matriz podemos construir um (ou mais) alinhamentos ótimos entre as cadeias:
  - Usando uma outra matriz para registrar o caminho percorrido para preencher as células.
  - Fazendo as contas na ordem inversa.
- Empates na pontuação máxima significam mais de uma possibilidade para o alinhamento.

Algoritmo

$\textsc{Align}(i, j, k)$

1 **if** $i == 0$ and $j == 0$
2 $\quad k = 0$
3 **else if** $i > 0$ and $A[i, j] == A[i - 1, j] + g$
4 $\quad \textsc{Align}(i - 1, j, k)$
5 $\quad k = k + 1$
6 $\quad s'[k] = s[i]$
7 $\quad t'[k] = -$
8 **else if** $i > 0$ and $j > 0$ and $A[i, j] = A[i - 1, j - 1] + p(i, j)$
9 $\quad \textsc{Align}(i - 1, j - 1, k)$
10 $\quad k = k + 1$
11 $\quad s'[k] = s[i]$
12 $\quad t'[k] = t[j]$
13 **else**
14 $\textsc{Align}(i, j - 1, k)$
15 $k = k + 1$
16 $s'[k] = -$
17 $t'[k] = t[j]$

Algoritmo

- O algoritmo está resolvendo os empates dando preferência para a célula de cima, depois para a diagonal e por último para a da esquerda.

Custo

- O algoritmo percorre um caminho na matriz, de tamanho máximo $n + m$. Para cada célula nesse caminho realiza trabalho constante.
- O custo desse algoritmo é $O(n + m)$.
- Uma versão iterativa vai usar memória constante.

Needleman-Wunsch

- S.B. Needleman e C.D. Wunsch, 1970, foram os primeiros a abordar o problema do alinhamento global.
- Em biologia molecular computacional, o problema do alinhamento global é chamado de Needleman-Wunsch.

Alinhamento global vs. local

- Um alinhamento global pode diluir uma região de boa similaridade local. No exemplo abaixo, -4/6.

```
GTCT--GGC-GCTAAT
 |||   | | | |   |
-TCTACGACTGGTGCT
```

```
CTGGCGCT
|||| |||
CTGGTGCT
```

Alinhamentos locais

- Um alinhamento local entre duas cadeias s e t é um alinhamento entre uma subcadeia de s e uma subcadeia de t.
- Alinhamentos locais são objetos de grande interesse em bioinformática.
- Para calcular alinhamentos locais podemos usar uma adaptação do algoritmo para alinhamentos locais.

Smith-Waterman

- T.F. Smith e M.S. Waterman, 1981, foram os primeiros a abordar o problema do alinhamento local.
- Em biologia molecular computacional, o problema do alinhamento local é chamado de Smith-Waterman.

PD para alinhamentos locais

- O alinhamento com a cadeia vazia tem pontuação 0.

$$A[i, 0] = 0$$
$$A[0, j] = 0$$

- Além das três possibilidades para extensão do alinhamento temos a pontuação 0 que significa que não há similaridade até $s[i]$ e $t[j]$:

$$A[i, j] = \max \begin{cases} A[i-1, j] + g \\ A[i-1, j-1] + p(i, j) \\ A[i, j-1] + g \\ 0 \end{cases}$$

PD para alinhamentos locais

- A maior pontuação local estará em alguma célula de A.
- A célula $A[i, j]$ da matriz armazena a melhor pontuação de um alinhamento entre uma subcadeia de $s[k, i]$ e uma subcadeia $t[\ell, j]$.

Algoritmos

- O algoritmo para preencher a matriz é essencialmente o mesmo.
- O algoritmo para obter o alinhamento precisa primeiro encontrar a célula com maior valor de pontuação e depois alinhar até encontrar uma célula de valor zero.
- A partir da matriz pode-se encontrar outros alinhamentos locais.
- Um algoritmo interessante é o de M.S. Waterman e M. Eggert (1987) que encontra todos os alinhamentos locais não sobrepostos com pontuação acima de um certo limiar.

Alinhamento global vs. semi-global

- Um alinhamento global pode diluir uma região de boa similaridade sufixo-prefixo.

```
CAGCACTGGGATTAGAC
 | |     | |
TACCTGC-GCAGCGTGG
```

```
--------CAGCACTGGGATTAGAC
        ||||  |||
TACCTGCGCAGCG-TGG--------
```

Alinhamento global vs. semi-global

- Um alinhamento global pode diluir uma região de boa similaridade entre uma cadeia e uma cadeia menor.

```
CAGCACTTGGATTCTCGG
||||      | |     ||
CAGC-----G-T----GG
```

```
CAGCA-CTTGGATTCTCGG
   ||  | |||
---CAGCGTGG--------
```

Alinhamento semi-global

- Um alinhamento semi-global é um alinhamento entre um prefixo de s e um sufixo de t ou vice-versa, ou um alinhamento em que s está contido em t ou vice-versa.

PD para semi-globais

- Podemos encontrar alinhamentos semi-globais usando uma variação do algoritmo para alinhamentos globais.
- A inicialização da primeira linha é com zeros para admitir espaços no início de s.
- A inicialização da primeira coluna é com zeros para admitir espaços no início de t.
- A similaridade estará em alguma célula da última coluna ou da última linha da matriz A, para admitir espaços no fim de s ou de t.

Buracos

Buracos

- Vamos chamar uma subcadeia maximal de espaços na mesma cadeia de um alinhamento de *buraco*.
- Em biologia molecular há interesse em penalizar os buracos de formas diferentes, por exemplo, penalizando buracos maiores proporcionalmente menos que buracos menores.

Penalização de buracos

- Vamos denotar a função de penalização de buracos de $w(i)$, onde $i \geq 0$ é o tamanho do buraco.
- A função que usamos até agora é $w(i) = ig$.

Subproblemas

- Quando usamos uma função qualquer para os buracos, $w(i)$ não precisa ter relação alguma com $w(i-1)$.
- Ou seja, se removermos uma coluna do alinhamento, a pontuação não precisa mais ser a soma da pontuação da coluna e do alinhamento restante.
- Mas se a remoção da coluna coincide com o fim de um buraco, podemos somar a pontuação da coluna com o alinhamento restante.

Subproblemas

- Há três possibilidades para o alinhamento entre $s[1, i]$ e $t[1, j]$:
  - O alinhamento termina com um buraco em s e então $s[i]$ está alinhado com um símbolo à esquerda de $t[j]$.
  - O alinhamento termina com um buraco em t e então $t[j]$ está alinhado com um símbolo à esquerda de $s[i]$.
  - O alinhamento termina com $s[i]$ alinhado com $t[j]$.

PD para alinhamento global

- Para $|s| = n$ e $|t| = m$, usamos três matrizes de dimensões $(n+1) \times (m+1)$:
  1. C para guardar a pontuação de alinhamentos que terminam com casamento de um símbolo de s com um símbolo de t.
  2. S para guardar a pontuação de alinhamentos que terminam com espaços em s.
  3. T para guardar a pontuação de alinhamentos que terminam com espaços em t.

Inicialização

$$C[0,0] = 0$$

$$S[0,j] = -w(j)$$

$$T[i,0] = -w(i)$$

Demais células $= -\infty$

Construção das matrizes

$$C[i,j] = p(i,j) + \max \begin{cases} C[i-1, j-1] \\ S[i-1, j-1] \\ T[i-1, j-1] \end{cases}$$

$$S[i,j] = \max \begin{cases} C[i, j-k] - w(k), \text{ para } 1 \leq k \leq j \\ T[i, j-k] - w(k), \text{ para } 1 \leq k \leq j \end{cases}$$

$$T[i,j] = \max \begin{cases} C[i-k, j] - w(k), \text{ para } 1 \leq k \leq i \\ S[i-k, j] - w(k), \text{ para } 1 \leq k \leq i \end{cases}$$

Similaridade e alinhamento

- A similaridade global aparece na célula

$$\max(C[m,n], S[m,n], T[m,n]).$$

- Como sempre, o alinhamento pode ser construído voltando nas matrizes ou registrando o caminho em outra matriz.

Complexidade

- As operações de preenchimento da matriz dominam o algoritmo.
- Para computar $C[i, j]$, $S[i, j]$ e $T[i, j]$ fazemos $3 + 2j + 2i$ acessos a células das matrizes. No total temos

$$\sum_{i=1}^{m} \sum_{j=1}^{n} (3 + 2j + 2i) = O(m^2 n + m n^2)$$

Funções afins para buracos

- Uma função *afim* é uma função da forma

$$w(k) = g_o + kg_e$$

  para $k \geq 1$ e $w(0) = 0$.
- O primeiro espaço custa $g_o + g_e$ e espaços adicionais custam g_e.
- Se $g_o > 0$ e $g_e > 0$ então w também é subaditiva, isto é o custo de k espaços isolados é maior que o de k espaços juntos.

Funções afins

- Essas funções são muito usadas na prática para alinhamento de cadeias biológicas.
- A PD é mais simples que aquela das funções arbitrárias: como $w(i) = w(i-1) + g_e$ para $i > 0$ então basta determinar se um buraco está começando ou se está sendo estendido, mas não é necessário saber o tamanho dele.

Inicialização

$$C[0,0] = 0$$
$$C[i,0] = -\infty, 1 \leq i \leq m$$
$$C[0,j] = -\infty, 1 \leq j \leq n$$

$$S[i,0] = -\infty, 0 \leq i \leq m$$
$$S[0,j] = -(g_o + jg_e), 1 \leq j \leq n$$

$$T[i,0] = -(g_o + ig_e), 1 \leq i \leq m$$
$$T[0,j] = -\infty, 0 \leq j \leq n$$

Construção das matrizes

$$C[i,j] = p(i,j) + \max\begin{cases} C[i-1,j-1] \\ S[i-1,j-1] \\ T[i-1,j-1] \end{cases}$$

$$S[i,j] = \max\begin{cases} C[i,j-1] - (g_o + g_e) \\ S[i,j-1] - g_e \\ T[i,j-1] - (g_o + g_e) \end{cases}$$

$$T[i,j] = \max\begin{cases} C[i-1,j] - (g_o + g_e) \\ S[i-1,j] - (g_o + g_e) \\ T[i-1,j] - g_e \end{cases}$$

Similaridade e alinhamento

- A similaridade global aparece na célula

$$\max(C[m,n], S[m,n], T[m,n]).$$

- Como sempre, o alinhamento pode ser construído voltando nas matrizes ou registrando o caminho em outra matriz.

Complexidade

- As operações de preenchimento da matriz dominam o algoritmo.
- Para computar $C[i, j]$, $S[i, j]$ e $T[i, j]$ fazemos 9 acessos a células das matrizes. São $O(nm)$ células e então no total temos tempo $O(nm)$.

Comparação de cadeias similares

Alinhamento global com k diferenças

- O problema é encontrar um alinhamento global entre duas cadeias s e t com no máximo k diferenças (espaços ou mismatches).
- Se s e t são cadeias parecidas, então um alinhamento entre s e t não se desvia muito da diagonal principal da matriz e podemos usar um algoritmo mais eficiente.

Algoritmo

- O tal algoritmo mais eficiente preenche apenas uma faixa de largura $2k + 1$ em torno da diagonal principal.
- Nenhuma célula fora dessa região é visitada.
- O algoritmo preenche apenas as células $A[i, j]$ tais que

$$-k \leq i - j \leq k$$

- Se o alinhamento resultante tem no máximo k diferenças então a pontuação em $A(n, m)$ é maior ou igual à pontuação máxima com mais que k diferenças. E vice-versa.

Complexidade

- Para que o problema tenha solução, n e m devem ser tais que $|n - m| \leq k$.
- Então o algoritmo preenche $O(kn) = O(km)$ células. O custo para preencher uma célula é constante.

k desconhecido

- Se k não é conhecido, pode-se usar uma busca binária.
- Começando com $k = 1$ o algoritmo computa $A[n, m]$ e verifica se esse valor é maior ou igual ao melhor valor de pontuação para um alinhamento com 1 diferença. Se for, termina.
- Senão o algoritmo dobra o valor de k e recomputa A.

Complexidade

- Seja k^* o número máximo de diferenças entre as cadeias s e t.
- Seja k' o valor máximo de k alcançado pelo algoritmo. Então $k' \leq 2k^*$.
- O trabalho total realizado pelo algoritmo foi

$$O(k'n + k'n/2 + k'n/4 + \ldots + n) = O(k'n) = O(k^*n).$$

- Logo, se o número de diferenças é pequeno então esse algoritmo é mais rápido e usa menos memória que a PD convencional.

Espaço linear

Similaridade em espaço linear

- O algoritmo de PD para alinhamentos pode calcular o valor da similaridade usando apenas duas linhas da matriz.
- Ou apenas uma linha mais uma variável.

Alinhamento em espaço linear

- Também é possível obter o alinhamento em espaço linear sem aumento assintótico do tempo de execução.

Duas cadeias

- Vamos considerar as cadeias abaixo, como exemplo.

```
TACTTAG
-AGTTC-
```

Matriz para prefixos

- Usando PD podemos construir a matriz A de similaridades de prefixos:

	e	G	A	T	T	C	A	T
e	0	-2	-4	-6	-8	-10	-12	-14
C	-2	-1	-3	-5	-7	-7	-9	-11
T	-4	-3	-2	-2	-4	-6	-8	-8
T	-6	-5	-4	-1	-1	-3	-5	-7
G	-8	-5	-6	-3	-2	-2	-4	-6
A	-10	-7	-4	-5	-4	-3	-1	-3

Matriz para sufixos

- Usando PD no sentido contrário podemos construir a matriz B de similaridades de sufixos:

	G	A	T	T	C	A	T	e
C	-3	-1	-2	-4	-3	-6	-7	-10
T	-4	-2	0	-2	-5	-4	-5	-8
T	-7	-5	-3	-1	-3	-4	-3	-6
G	-8	-8	-6	-4	-2	-2	-3	-4
A	-11	-9	-7	-5	-3	-1	-1	-2
e	-14	-12	-10	-8	-6	-4	-2	0

Matriz de cortes

- Somando as duas matrizes obtemos a matriz de cortes C.

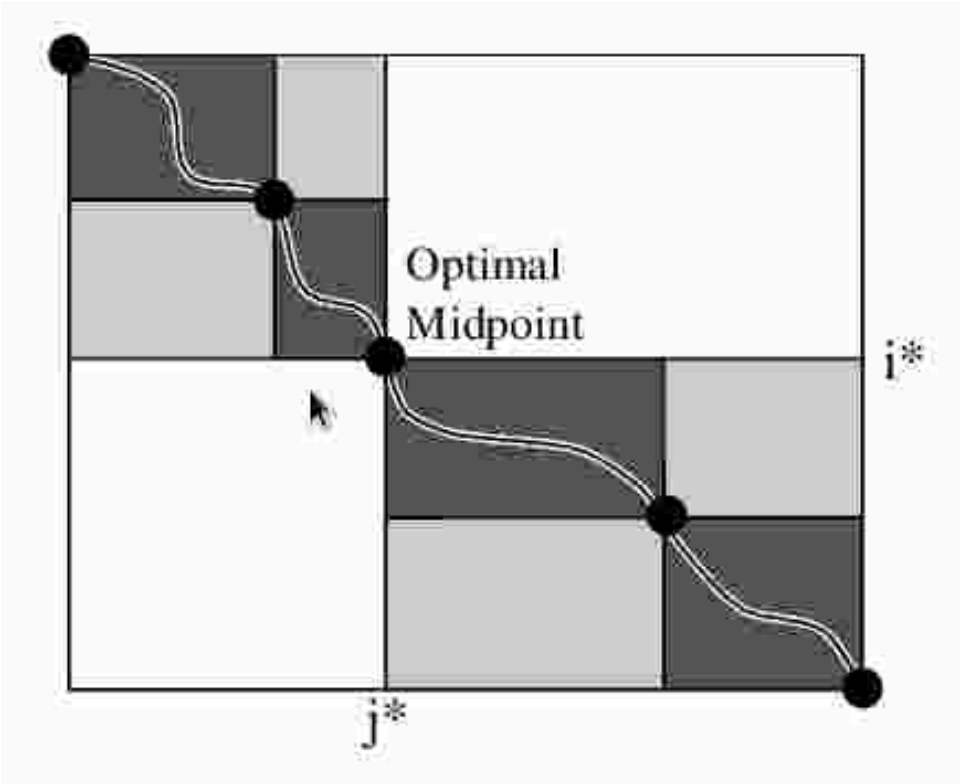
		G	A	T	T	C	A	T
		-3	-3	-7	-12	-11	-14	-19
C		-8	-5	-3	-7	-10	-11	-14
T		-13	-10	-5	-3	-5	-8	-11
T		-14	-13	-8	-5	-3	-5	-8
G		-19	-14	-13	-10	-7	-3	-3
A								

- Essa matriz é tal que, para uma linha i, a célula (i, k) em C de valor máximo está no alinhamento entre s e t.

Algoritmo

1. Use o algoritmo para alinhamento em espaço linear (prefixos) com $s[1, n/2]$ e $t[1, m]$. Ao preencher a linha $n/2$, registre os apontadores também.
2. Use o algoritmo para alinhamento em espaço linear (sufixos) com $s[n/2, n]$ e $t[1, m]$. Ao preencher a linha $n/2$, registre os apontadores também.
3. Some as duas linhas $n/2$ e encontre a célula com o máximo, na coluna k.
4. Use a linha $n/2$ (prefixos) para retroceder enquanto for possível, até a célula k_l. Use a linha $n/2$ (sufixos) para avançar enquanto for possível, até a célula k_r.
5. Essas células definem um subcaminho ótimo $A[n/2, k_l] \ldots A[n/2, k_r]$.
6. Resolva os dois subproblemas $s[1, n/2 - 1], t[1, k_l]$ e $s[n/2 + 1, n], t[k_r, m]$ e combine os caminhos.

Subproblemas



[Myers e Miller, 1988]

Complexidade

- Seja $T(n, m)$ o número de vezes que o valor da similaridade máxima é calculado entre prefixos de t ou entre sufixos de t. Podemos mostrar por indução em m que $T(n, m) \leq 2nm$.
- Para $m > 1$, serão necessárias no máximo
  1. $\frac{n}{2}m$ cálculos de similaridades máximas entre $s[1, n/2]$ e $t[1, k]$
  2. $\frac{n}{2}m$ cálculos de similaridades máximas entre $s[n/2, n]$ e $t[k, m]$

Complexidade

- Serão feitas duas chamadas recursivas causando no máximo $T(n/2, k)$ e $T(n/2, n-k)$ cálculos de similaridades máximas.
- Somando, temos

$$\begin{aligned} T(m,n) &\leq \frac{nm}{2} + \frac{nm}{2} + T(n/2,k) + T(n/2, n-k) \\ &\leq nm + nk + n(m-k) \\ &= 2nm \end{aligned}$$

BLAST

Heurísticas

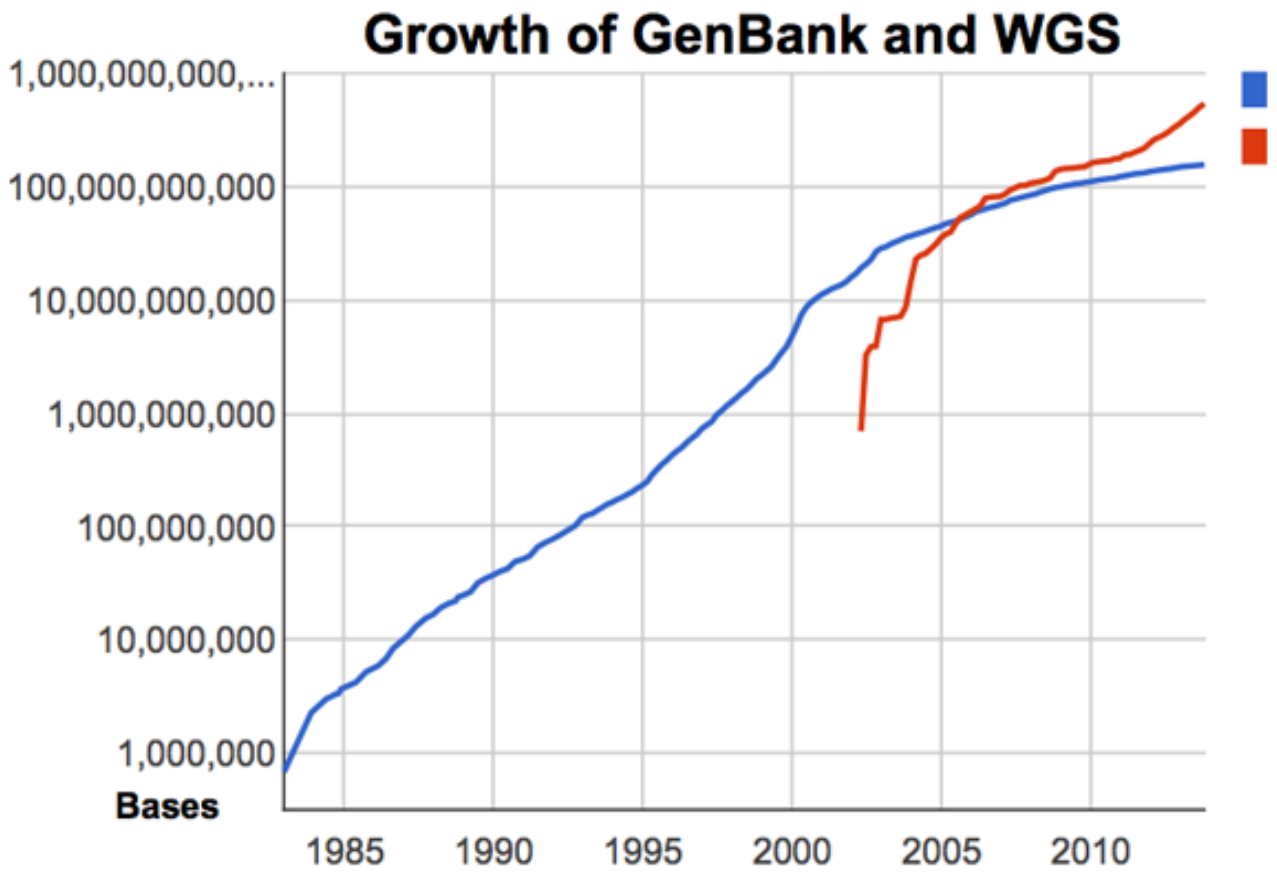
O algoritmo exato baseado em PD é muito custoso quando:

- temos um conjunto grande de cadeias para comparar entre si.
- queremos comparar uma cadeia contra muitas outras em um banco de cadeias.

Banco de cadeias

- Conjunto de cadeias de DNA, RNA ou proteína em formato digital. Não está necessariamente associado a um SGBD. Pode ser, por exemplo, um arquivo texto.
- Genbank: coleção das seqüências anotadas do NIH.
- WGS: montagens incompletas de genomas.

Genbank WGS


Growth of GenBank and WGS
1,000,000,000,...
100,000,000,000
10,000,000,000
1,000,000,000
100,000,000
10,000,000
1,000,000
Bases
1985
1990
1995
2000
2005
2010

BLAST

- BLAST é um acrônimo de Basic Local Alignment Search Tool.
- Heurística para encontrar alinhamentos locais entre cadeias.
- Fundamentada em modelos estatísticos.
- Tem sensibilidade e desempenho bastante bons.

BLAST

- Família de programas: BLAST (90), gapped-BLAST (97), psi-BLAST (97), megaBLAST e outros.
- Existem várias implementações disponíveis de BLAST.
- 38.380 (90) e 36.410 (97) citações em 2014.

Comparação: PD x BLAST

- Uma proteína contra 2.831 outras
- Algoritmo PD exato (programa swat): 4.19 segundos
- Algoritmo BLAST (programa blastall): 0.17 segundo (+0.34 segundo para pré-processar o banco)

Observação

- Se um alinhamento entre duas cadeias é estatisticamente significativo (isto é, tem baixa probabilidade de ser aleatório), ele provavelmente contém subcadeias alinhadas nas duas cadeias com alta similaridade.

Idéia

- Palavra: cadeia pequena, normalmente de 3 aminoácidos.
- Procurar palavras que alinham bem nas duas cadeias e tentar fazer esse alinhamento crescer.

Entrada para o Gapped-Blast

- Duas cadeias s e t.
- Uma matriz de substituições S, que especifica a pontuação para alinhar pares de aminoácidos.

Passos do Gapped-BLAST

- 1- Compila, em cada cadeia, uma lista de palavras que quando alinhadas com alguma outra podem pontuar acima de um limiar T.
- 2- Busca alinhamentos entre palavras selecionadas no Passo 1 que satisfaçam certas condições.
- 3- Estende os alinhamentos encontrados no Passo 2.

Passo 1: Busca de hits

- O algoritmo procura palavras de s e t que pontuam acima de T quando alinhadas sem espaços (hit).
- Palavras são subcadeias. Para aminoácidos têm tamanho 3.
- O banco de cadeias t é pré-processado e uma tabela de hashing é construída com as cadeias dele.

Passo 2: Formação de HSPs

- Um *HSP* (high scoring pair) é um alinhamento sem espaços entre s e t cuja pontuação não pode ser melhorada pelo aumento ou diminuição do alinhamento.
- O algoritmo procura pares de hits que não se sobrepõem e estão na mesma diagonal à distância no máximo A.
  - A diagonal de um hit (s_i, t_j) é $i - j$.
  - A distância entre dois hits (s_i, t_j) e (s_k, t_l) é $|s_i - s_k|$.
- Se algum hit se sobrepõe ao considerado mais recentemente, ele é desconsiderado.
- O algoritmo mantém um array que registra o hit considerado mais recentemente em cada diagonal.

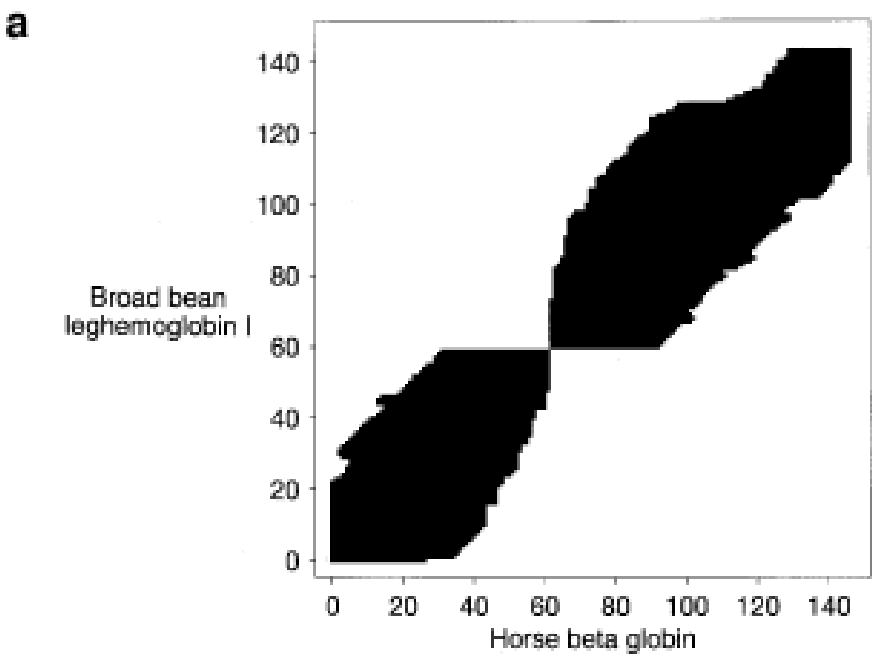
Passo 2: Formação de HSPs

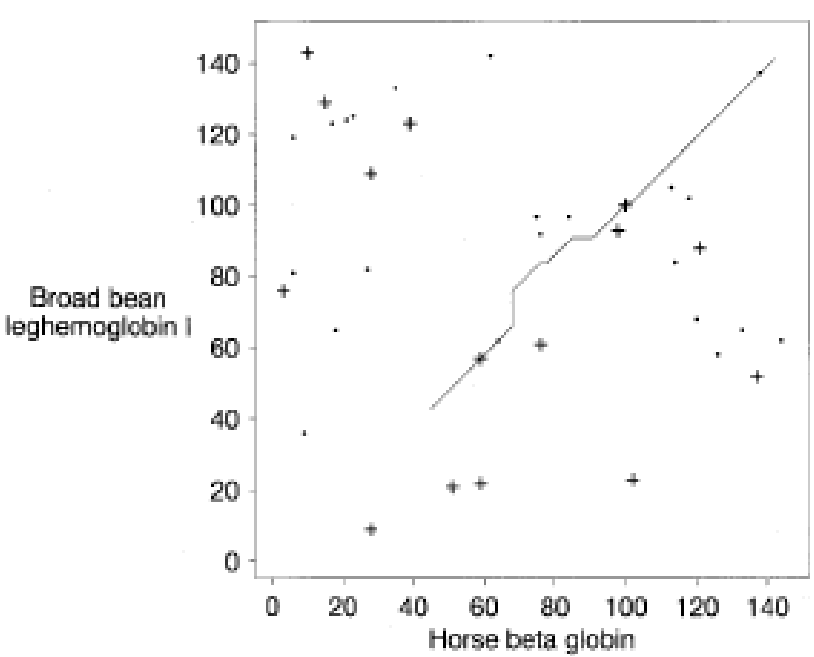
- O alinhamento é estendido nas duas direções sem adicionar espaços enquanto a pontuação não cair mais que X_u em relação ao melhor valor obtido.
- Se o alinhamento obtido tiver pontuação maior que S_g, então ele passa para a próxima fase.

Extensão dos alinhamentos com espaços

- O algoritmo seleciona uma semente.
  - Par central do subalinhamento de tamanho 11 de melhor pontuação ou
  - par central se o HSP tiver menos que 11 colunas.
- Usa uma variação do algortimo de PD que estende a semente nas duas direções.
- Preenche apenas as células em que a pontuação não diminui mais que um certo limiar X_g.

Matriz


a
Broad bean leghemoglobin I
0
20
40
60
80
100
120
140
Horse beta globin


b
Broad bean leghemoglobin I
0
20
40
60
80
100
120
140
Horse beta globin

Saída do gapped-BLAST

- Alinhamentos com espaços que tenham relevância estatística aceitável.
- O valor da relevância estatística para cada alinhamento.
- A pontuação normalizada para cada alinhamento.

Saída do gapped-BLAST

- Os resultados produzidos pelo BLAST são avaliados de acordo com um modelo estatístico de cadeias aleatórias.
- O modelo supõe
  - letras geradas independentemente com base em uma probablidade de ocorrência calculada a partir de proteínas reais.
  - Um esquema de pontuação que tem pelo menos um valor positivo e que a pontuação esperada para o alinhamento de um par seja negativa.

Relevância estatística

- O E-value é o número de alinhamentos que podem ser produzidos ao acaso entre cadeias de tamanhos m e n:

$$E = Kmne^{-\lambda S}$$

onde S é a pontuação do alinhamento, K e λ são calculados (e estimados) de acordo com as probabilidades de substituição e com a pontuação de cada substituição.

Pontuação normalizada

- A pontuação normalizada S', independente de K e λ, é

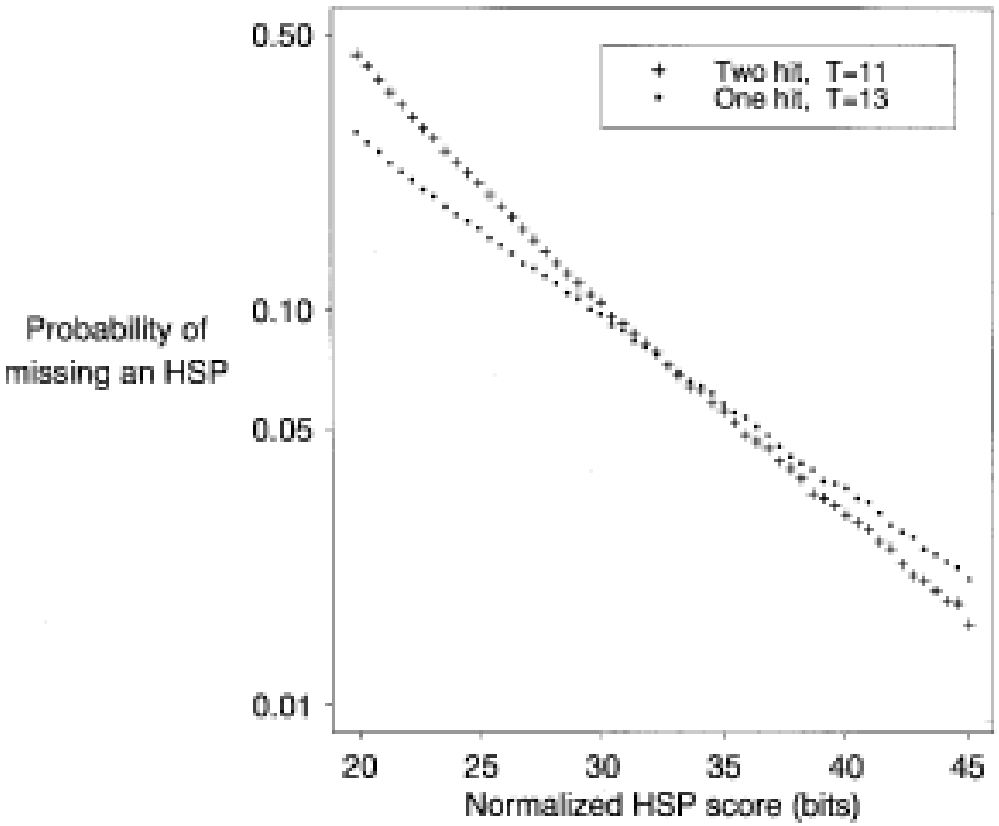
$$S' = \frac{\lambda S - \ln K}{\ln 2}$$

e o E-value correspondente é

$$E = mn2^{-S}$$

Acurácia

- A probabilidade de perder HSPs foi calculada para $K = 0.134$ e $\lambda = 0.3176$ usando a matriz BLOSUM-62.



Complexidade

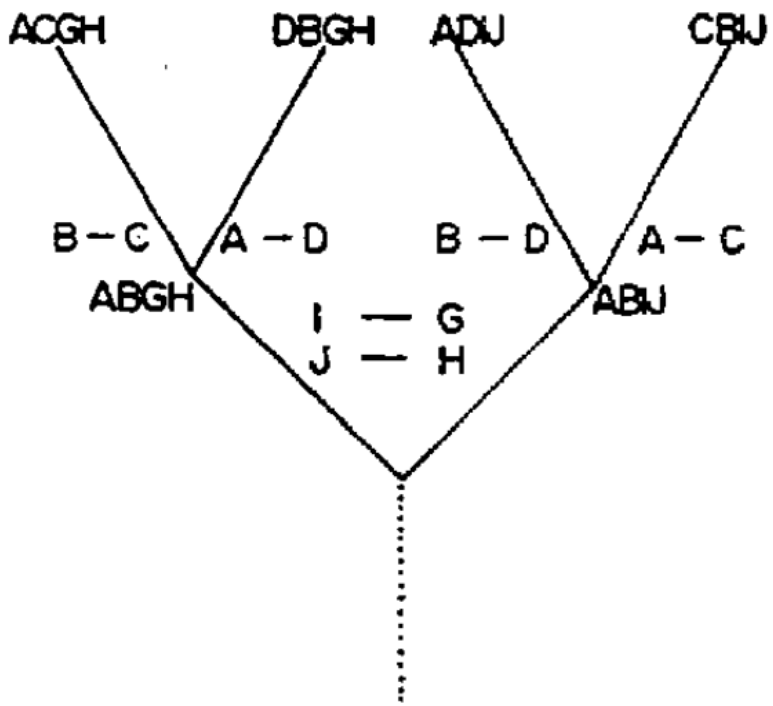
- No pior caso $O(nm)$.
- Na prática, muito aquém desse limite.

Esquemas de pontuação para amino-ácidos

- Baseados em seqüências conhecidas.
- Tentam refletir a similaridade de funções dos amino-ácidos e suas relações na moléculas, ao invés de mudanças nas seqüências gênicas.
  - Tamanho,
  - forma,
  - concentração de cargas,
  - conformação,
  - habilidade de formar ligações salinas, hidrofóbicas e de hidrogênio.

PAM

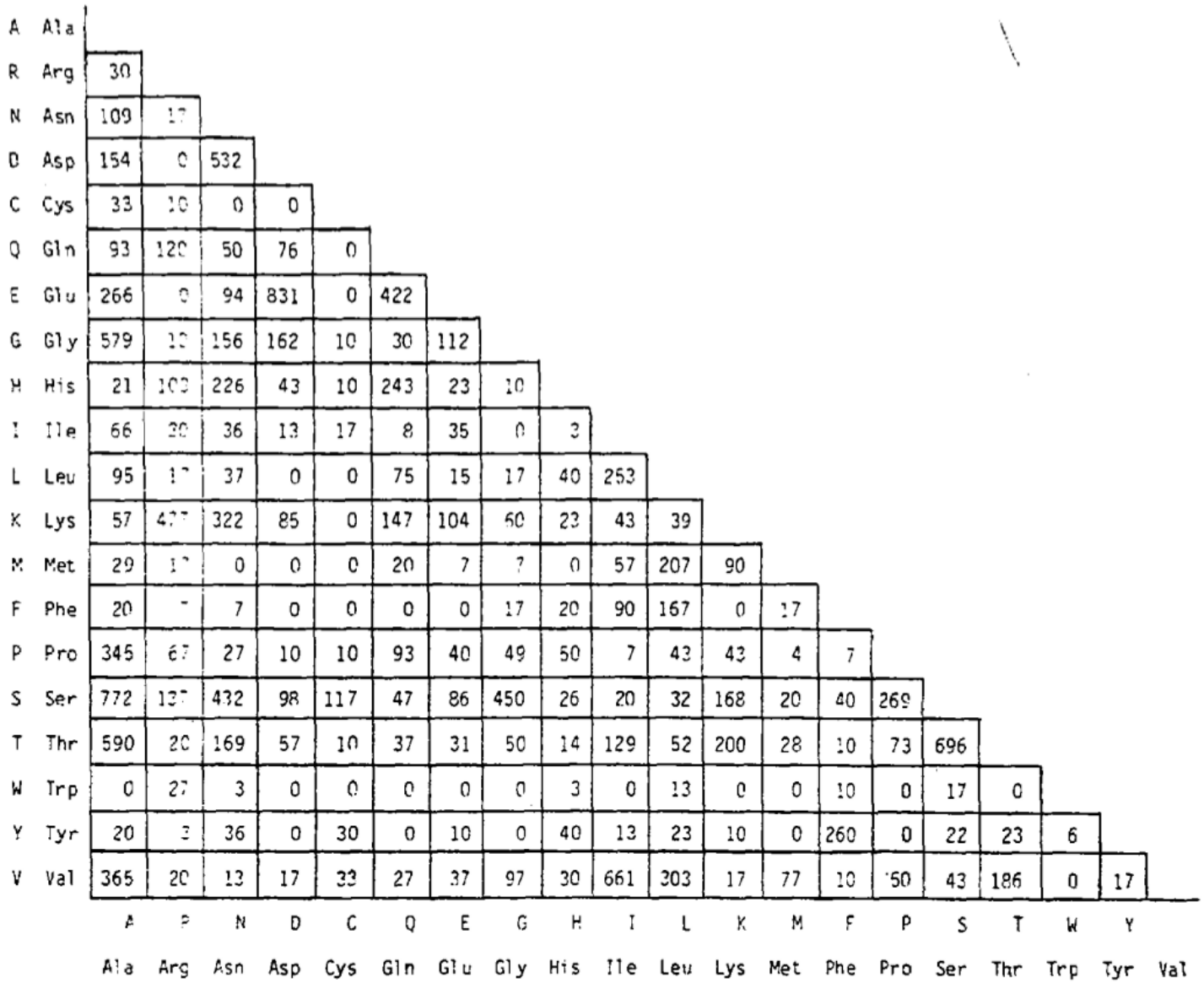
- Uma mutação pontual aceita é uma substituição de um aminoácido por outro que foi aceita pela seleção natural.
- Uma forma de obter informação acerca das mutações aceitas é comparando seqüências com ancestralidade inferida.



Matrizes PAM

- A família de matrizes PAM foi construída a partir de 1572 mudanças em 71 árvores evolutivas de proteínas com relacionamento evolutivo próximo.

Matriz de mutações pontuais aceitas A x10



		A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
A	Ala																				
R	Arg	30																			
N	Asn	109	17																		
D	Asp	154	0	532																	
C	Cys	33	10	0	0																
Q	Gln	93	120	50	76	0															
E	Glu	266	0	94	831	0	422														
G	Gly	579	10	156	162	10	30	112													
H	His	21	100	226	43	10	243	23	10												
I	Ile	66	20	36	13	17	8	35	0	3											
L	Leu	95	17	37	0	0	75	15	17	40	253										
K	Lys	57	427	322	85	0	147	104	60	23	43	39									
M	Met	29	17	0	0	0	20	7	7	0	57	207	90								
F	Phe	20	7	7	0	0	0	0	17	20	90	167	0	17							
P	Pro	345	67	27	10	10	93	40	49	50	7	43	43	4	7						
S	Ser	772	137	432	98	117	47	86	450	26	20	32	168	20	40	269					
T	Thr	590	20	169	57	10	37	31	50	14	129	52	200	28	10	73	696				
W	Trp	0	27	3	0	0	0	0	0	3	0	13	0	0	10	0	17	0			
Y	Tyr	20	3	36	0	30	0	10	0	40	13	23	10	0	260	0	22	23	6		
V	Val	365	20	13	17	33	27	37	97	30	661	303	17	77	10	50	43	186	0	17	
		A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
		Ala	Arg	Asn	Asp	Cys	Gln	Glu	Gly	His	Ile	Leu	Lys	Met	Phe	Pro	Ser	Thr	Trp	Tyr	Val

Mutabilidade relativa

- A probabilidade de cada aminoácido mudar num intervalo evolutivo curto é chamada de mutabilidade relativa.
- É o número de vezes que um aminoácido mudou dividido pelo número de vezes que ele aparece nas cadeias.

Aligned sequences	A D A A D B		
Amino acids	A	B	D
Changes	1	1	0
Frequency of occurrence (total composition)	3	1	2
Relative mutability	.33	1	0

Mutabilidade relativa normalizada m_j, Ala=100

Relative Mutabilities of the Amino Acids[a]

Asn	134	His	66
Ser	120	Arg	65
Asp	106	Lys	56
Glu	102	Pro	56
Ala	100	Gly	49
Thr	97	Tyr	41
Ile	96	Phe	41
Met	94	Leu	40
Gln	93	Cys	20
Val	74	Trp	18

Freqüência normalizada

- A freqüência normalizada dos aminoácidos nos dados de mutações pontuais aceitas é f_i.

Normalized Frequencies of the Amino Acids in the Accepted Point Mutation Data

Gly	0.089	Arg	0.041
Ala	0.087	Asn	0.040
Leu	0.085	Phe	0.040
Lys	0.081	Gln	0.038
Ser	0.070	Ile	0.037
Val	0.065	His	0.034
Thr	0.058	Cys	0.033
Pro	0.051	Tyr	0.030
Glu	0.050	Met	0.015
Asp	0.047	Trp	0.010

Probabilidade de substituição

- A partir da combinação da informação sobre os tipos de mutações e da mutabilidade relativa, pode-se calcular a probabilidade de que um aminoácido j seja substituído pelo aminoácido i em um intervalo de 1 PAM

$$M_{ji} = \frac{\lambda m_j A_{ji}}{\sum_i A_{ji}},$$

e

$$M_{jj} = 1 - \lambda m_j.$$

- λ é uma constante de proporcionalidade calculada para que haja 1 mutação a cada 100 aminoácidos. Se há 99% de aminoácidos conservados então λ deve satisfazer

$$0.01 = \lambda \sum_i f_i m_i.$$

1 PAM (x10.000)

ORIGINAL AMINO ACID

REPLACEMENT AMINO ACID	A Ala	R Arg	N Asn	D Asp	C Cys	Q Gln	E Glu	G Gly	H His	I Ile	L Leu	K Lys	M Met	F Phe	P Pro	S Ser	T Thr	W Trp	Y Tyr	V Val
A Ala	9867	2	9	10	3	8	17	21	2	6	4	2	6	2	22	35	32	0	2	18
R Arg	1	9913	1	0	1	10	0	0	10	3	1	19	4	1	4	6	1	8	0	1
N Asn	4	1	9822	36	0	4	6	6	21	3	1	13	0	1	2	20	9	1	4	1
D Asp	6	0	42	9859	0	6	53	6	4	1	0	3	0	0	1	5	3	0	0	1
C Cys	1	1	0	0	9973	0	0	0	1	1	0	0	0	0	1	5	1	0	3	2
Q Gln	3	9	4	5	0	9876	27	1	23	1	3	6	4	0	6	2	2	0	0	1
E Glu	10	0	7	56	0	35	9865	4	2	3	1	4	1	0	3	4	2	0	1	2
G Gly	21	1	12	11	1	3	7	9935	1	0	1	2	1	1	3	21	3	0	0	5
H His	1	8	18	3	1	20	1	0	9912	0	1	1	0	2	3	1	1	1	4	1
I Ile	2	2	3	1	2	1	2	0	0	9872	9	2	12	7	0	1	7	0	1	33
L Leu	3	1	3	0	0	6	1	1	4	22	9947	2	45	13	3	1	3	4	2	15
K Lys	2	37	25	6	0	12	7	2	2	4	1	9926	20	0	3	8	11	0	1	1
M Met	1	1	0	0	0	2	0	0	0	5	8	4	9874	1	0	1	2	0	0	4
F Phe	1	1	1	0	0	0	0	1	2	8	6	0	4	9946	0	2	1	3	28	0
P Pro	13	5	2	1	1	8	3	2	5	1	2	2	1	1	9926	12	4	0	0	2
S Ser	28	11	34	7	11	4	6	16	2	2	1	7	4	3	17	9840	38	5	2	2
T Thr	22	2	13	4	1	3	2	2	1	11	2	8	6	1	5	32	9871	0	2	9
W Trp	0	2	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	9976	1	0
Y Tyr	1	0	3	0	3	0	1	0	4	1	1	0	0	21	0	1	1	2	9945	1
V Val	13	2	1	1	3	2	2	3	3	57	11	1	17	1	3	2	10	0	2	9901

Mutação de uma cadeia

- Dada uma seqüência de aminoácidos, uma seqüência mutada a partir dela com distância média 1 PAM pode ser obtida com uma simulação.
- Vamos supor que o primeiro aminoácido da cadeia seja Ala. Escolhe-se um número aleatório x entre 0 e 1. Se $0 \leq x < 0.9867$ então Ala não muda, se $0.9867 \leq x < 0.9868$ então Ala é substituída por Arg, $0.9868 \leq x < 0.9872$ então Ala é substituída por Asp e assim por diante. Esse processo é repetido para cada aminoácido na cadeia.

N PAMs

- A matriz 1 PAM pode ser multiplicada por ela mesma N vezes para obter uma matriz que prevê as substituições que serão encontradas depois de N períodos evolutivos sobre uma cadeia típica.

250 PAM

- A matriz 250 PAM tem apenas 1 aminoácido em cada 5 sem mudança, mas a freqüência é bem diferente entre eles.

250 PAM (x100)

ORIGINAL AMINO ACID

REPLACEMENT AMINO ACID	A Ala	R Arg	N Asn	D Asp	C Cys	Q Gln	E Glu	G Gly	H His	I Ile	L Leu	K Lys	M Met	F Phe	P Pro	S Ser	T Thr	W Trp	Y Tyr	V Val
A Ala	13	6	9	9	5	8	9	12	5	8	6	7	7	4	11	11	11	2	4	9
R Arg	3	17	4	3	2	5	3	2	6	3	2	9	4	1	4	4	3	7	2	2
N Asn	4	4	6	7	2	5	6	4	6	3	2	5	3	2	4	5	4	2	3	3
D Asp	5	4	8	11	1	7	10	5	6	3	2	5	3	1	4	5	5	1	2	3
C Cys	2	1	1	1	52	1	1	2	2	2	1	1	1	1	2	3	2	1	4	2
Q Gln	3	5	5	6	1	10	7	3	7	2	3	5	3	1	4	3	3	1	2	3
E Glu	5	4	7	11	1	9	12	5	6	3	2	5	3	1	4	5	5	1	2	3
G Gly	12	5	10	10	4	7	9	27	5	5	4	6	5	3	8	11	9	2	3	7
H His	2	5	5	4	2	7	4	2	15	2	2	3	2	2	3	3	2	2	3	2
I Ile	3	2	2	2	2	2	2	2	2	10	6	2	6	5	2	3	4	1	3	9
L Leu	6	4	4	3	2	6	4	3	5	15	34	4	20	13	5	4	6	6	7	13
K Lys	6	18	10	8	2	10	8	5	8	5	4	24	9	2	6	8	8	4	3	5
M Met	1	1	1	1	0	1	1	1	1	2	3	2	6	2	1	1	1	1	1	2
F Phe	2	1	2	1	1	1	1	1	3	5	6	1	4	32	1	2	2	4	20	3
P Pro	7	5	5	4	3	5	4	5	5	3	3	4	3	2	20	6	5	1	2	4
S Ser	9	6	8	7	7	6	7	9	6	5	4	7	5	3	9	10	9	4	4	6
T Thr	8	5	6	6	4	5	5	6	4	6	4	6	5	3	6	8	11	2	3	6
W Trp	0	2	0	0	0	0	0	0	1	0	1	0	0	1	0	1	0	55	1	0
Y Tyr	1	1	2	1	3	1	1	1	3	2	2	1	2	15	1	2	2	3	31	2
V Val	7	4	4	4	4	4	4	5	4	15	10	4	10	5	5	5	7	2	4	17

Percentual de mudança

- Para cada matriz de probabilidade de mutação podemos calcular o percentual de aminoácidos que sofrerão mudança em média:

$$100(1 - \sum_i f_i M_{ii}).$$

Correspondência entre percentual de diferenças e distância PAM

Correspondence between Observed Differences and the Evolutionary Distance

Observed Percent Difference	Evolutionary Distance in PAMs
1	1
5	5
10	11
15	17
20	23
25	30
30	38
35	47
40	56
45	67
50	80
55	94
60	112
65	133
70	159
75	195
80	246
85	328

Matriz de possibilidades

- M_{ji} é a probabilidade de que o aminoácido j mude para i.
- f_i é a probabilidade de que o aminoácido i ocorra ao acaso na cadeia mutada.
- Uma matriz de possibilidades (odds matrix) pode ser calculada como

$$Rji = \frac{M_{ji}}{f_i}.$$

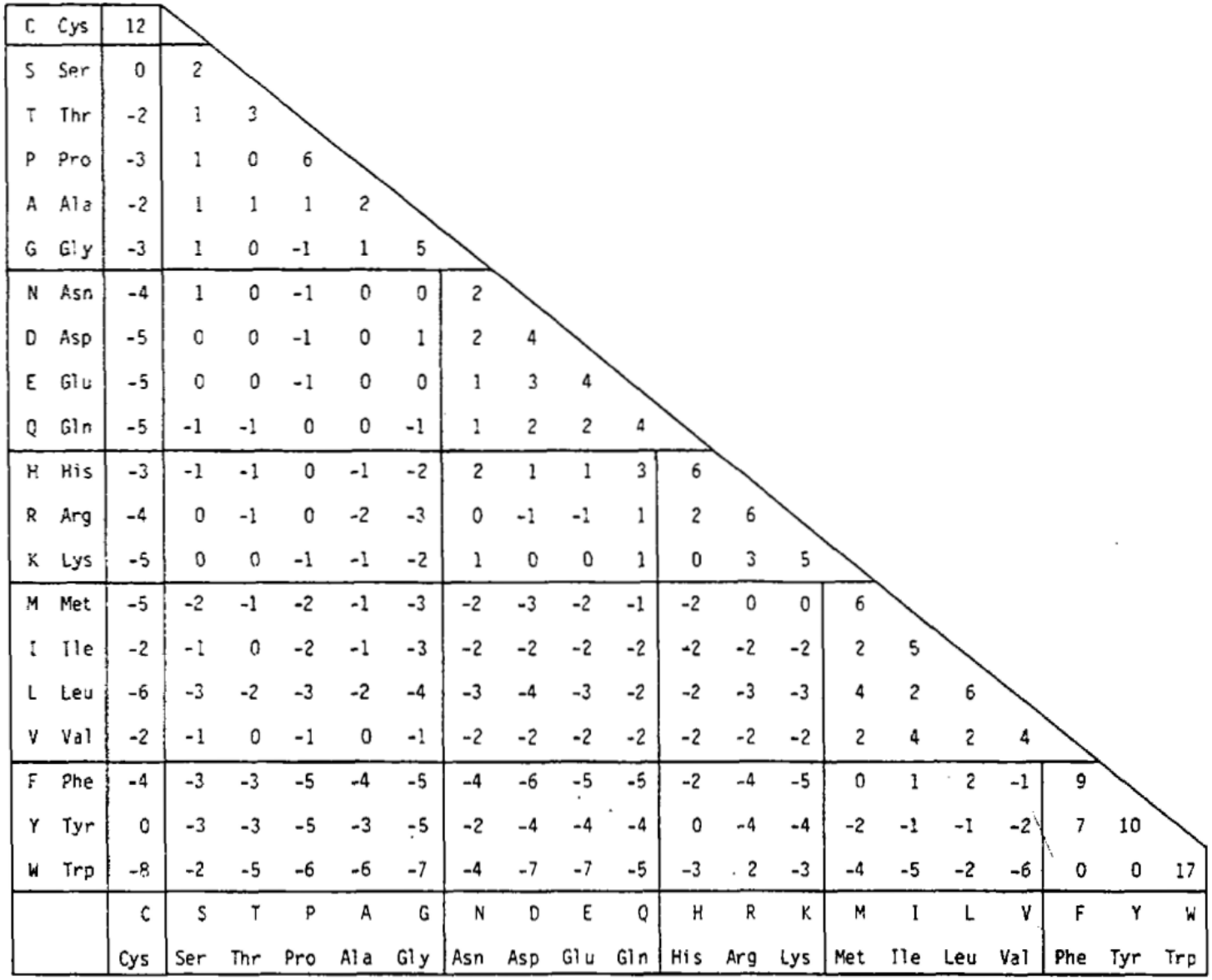
Matriz de pontuação

- A matriz de pontuação para alinhamentos é obtida fazendo

$$round(10 \log_{10} R_{ij}).$$

- Assim os logs das possibilidades podem ser somados normalmente nos alinhamentos (o que corresponde ao log da multiplicação das possibilidades).
- A multiplicação por 10 é para amenizar os efeitos do arredondamento.

PAM 250



C	Cys	12																		
S	Ser	0	2																	
T	Thr	-2	1	3																
P	Pro	-3	1	0	6															
A	Ala	-2	1	1	1	2														
G	Gly	-3	1	0	-1	1	5													
N	Asn	-4	1	0	-1	0	0	2												
D	Asp	-5	0	0	-1	0	1	2	4											
E	Glu	-5	0	0	-1	0	0	1	3	4										
Q	Gln	-5	-1	-1	0	0	-1	1	2	2	4									
H	His	-3	-1	-1	0	-1	-2	2	1	1	3	6								
R	Arg	-4	0	-1	0	-2	-3	0	-1	-1	1	2	6							
K	Lys	-5	0	0	-1	-1	-2	1	0	0	1	0	3	5						
M	Met	-5	-2	-1	-2	-1	-3	-2	-3	-2	-1	-2	0	0	6					
I	Ile	-2	-1	0	-2	-1	-3	-2	-2	-2	-2	-2	-2	-2	2	5				
L	Leu	-6	-3	-2	-3	-2	-4	-3	-4	-3	-2	-2	-3	-3	4	2	6			
V	Val	-2	-1	0	-1	0	-1	-2	-2	-2	-2	-2	-2	-2	2	4	2	4		
F	Phe	-4	-3	-3	-5	-4	-5	-4	-6	-5	-5	-2	-4	-5	0	1	2	-1	9	
Y	Tyr	0	-3	-3	-5	-3	-5	-2	-4	-4	-4	0	-4	-4	-2	-1	-1	-2	7	10
W	Trp	-8	-2	-5	-6	-6	-7	-4	-7	-7	-5	-3	2	-3	-4	-5	-2	-6	0	0	17
		C	S	T	P	A	G	N	D	E	Q	H	R	K	M	I	L	V	F	Y	W
		Cys	Ser	Thr	Pro	Ala	Gly	Asn	Asp	Glu	Gln	His	Arg	Lys	Met	Ile	Leu	Val	Phe	Tyr	Trp

Matriz de pontuação

- Pontuação acima de 1 indica um par de aminoácidos que se substituem com mais freqüência que seqüências com a mesma composição se substuiriam ao acaso.
- Pontuação abaixo de 1 indica um par de aminoácidos que se substituem com menos freqüência que seqüências com a mesma composição se substuiriam ao acaso.
- Em geral, PAMs menores (40,60,100) são melhores para localizar similaridades locais altas em alinhamentos curtos e PAMs maiores (200,250) são melhores para localizar similaridades mais baixas em alinhamentos mais longos.
- Existem atualizações das PAM: GCB e TJJ.

Matrizes BLOSUM

- Block substitution matrices.
- Tenta representar relacionamentos distantes explicitamente, ao invés de inferí-los, como nas PAM.
- Construídas a partir de blocos de alinhamentos que representam regiões conservadas de proteínas.
- Geradas a partir de um conjunto de dados maior do que o usado nas PAM.

Blocos

- Um bloco é um conjunto de colunas sem espaços entre elas que representam um alinhamento local entre várias seqüências.
- As matrizes foram construídas a partir de 504 grupos de seqüências não redundates dos bancos Prosite e Swiss-prot, que geraram 2205 blocos inicialmente.

Construção das matrizes

- As matrizes BLOSUM são construídas com base nas probabilidades observada e de ocorrência de pares de aminiácidos nos blocos.
- Um limiar de similaridade entre as seqüências define cada membro da família de matrizes.

Freqüências de ocorrência

- Para cada coluna de cada bloco, as seqüências são consideradas uma a uma. Para cada seqüência adicionada, o número de matches e mismatches com as seqüências adicionadas anteriormente é contado em cada coluna.
- O resultado é uma tabela de freqüências de ocorrência de cada par de aminoácidos nos blocos. Pares ab e ba são o mesmo par.

Exemplo

- ADM
  ADM
  ADM
  AEM
  ADI
  AEM
  ADI
  ADM
  AEI
  SDM
- $f_{AA} = 36,\ f_{AS} = 9,\ F_{SS} = 0.$

Probabilidade observada

- A probabilidade observada de cada par de aminoácidos é

$$q_{ab} = \frac{f_{ab}}{\sum_a \sum_b f_{ab}}$$

- No exemplo, $q_{AA} = \frac{36}{45} = 0.8$, $q_{AS} = \frac{9}{45} = 0.2$ e $q_{SS} = 0$.

Probabilidade de ocorrência

- A probabilidade de ocorrência de cada aminoácido em um par é

$$p_a = q_{aa} + \sum_{a \neq b} \frac{q_{ab}}{2}$$

- No exemplo, $p_A = \frac{36}{45} + \frac{\frac{36}{45}}{2} = 0.9$, $p_S = 0 + \frac{\frac{9}{45}}{2} = 0.1$.

Probabilidade de ocorrência

- A probabilidade de ocorrência de um par é

$$e_{ab} = \begin{cases} p_a p_b \text{ se } a = b \\ p_a p_b + p_b p_a \text{ se } a \neq b \end{cases}$$

- No exemplo, $e_{AA} = 0.9 \times 0.9 = 0.81$, $e_{AS} = 2 \times (0.9 \times 0.1) = 0.18$ e $e_{SS} = 0.1 \times 0.1 = 0.01$.

Matriz de pontuação

- A matriz de pontuaão é

$$S_{ab} = round(2 \log_2 \frac{q_{ab}}{e_{ab}})$$

- Se a freqüência de substituição para um par ab é maior do que a esperada ao acaso, $S_{ab} > 0$. Se for menor, $S_{ab} < 0$.

Família de matrizes

- A família de matrizes BLOSUM é gerada agrupando seqüências de acordo com o percentual de letras idênticas entre elas.
- A matriz BLOSUM-X é gerada da seguinte maneira. Suponha que a seqüência S_1 está em sendo considerada para um bloco.
  - Se a seqüência S_2 tem pelo menos X% de identidade com S_1, S_1 e S_2 são colocadas no mesmo grupo e as contribuições de S_2 para a tabela de freqüência de ocorrências são feita pela média em relação ao número de seqüências no grupo (no caso, $/2$).
  - Se a seqüência S_3 tem pelo menos X% de identidade com S_1, S_3 é colocadas no mesmo grupo que S_1 mesmo que S_3 não tenha X% de identidade com as demais seqüências no grupo (no caso, S_2), e as contribuições de S_3 para a tabela de freqüência de ocorrências são feita pela média em relação ao número de seqüências no grupo (no caso, $/3$).

Família de matrizes

- No exemplo, se considerarmos X=60, oito das nove seqüências com A serão agrupadas e a contribuição da coluna é equivalente a uma coluna com dois As e um S.

Relação com as PAM

- As matrizes BLOSUM e PAM foram comparadas usando a informação mútua média ou entropia relativa

$$H = \sum_{i=1}^{20} \sum_{j=1}^{i} q_{ij} \times s_{ij}.$$

- BLOSUM-45 é comparável a PAM-250 ($H \sim 0.4$) e BLOSUM-80 é comparável a PAM-120 ($H \sim 1$).
- Experimentos com seqüências conhecidas mostram que BLOSUM-62 perde menos alinhamentos usando Blast, Fast e PD que outras BLOSUM e que a PAM-140, a melhor PAM no teste. Também foram melhores que as GCB e JTT.
- A BLOSUM-62 é a matriz padrão do Blast.

Distância

- Distância (ou métrica) $d(x, y)$ é uma relação tal que
  - $d(x, x) = 0$ se $x = y$ e $d(x, y) > 0$ se $x \neq y$.
  - $d(x, y) = d(y, x)$
  - $d(x, y) \leq d(x, z) + d(z, y)$.

Distância entre cadeias

- Uma medida da diferença entre duas cadeias s e t pode ser definida como o menor número de operações de substituição, inserção e remoção que transfoma s em t.
- O custo de uma substituição é $c : \Sigma \times \Sigma \mapsto \mathbb{R}$.
- O custo de uma inserção ou remoção é h.
- Para ser uma distância devem valer para todo $x, y, z \in \Sigma$,

$$c(x, y) = c(y, x)$$
$$c(x, y) > 0 \text{ se } x \neq y$$
$$h > 0$$
$$c(x, z) \leq c(x, y) + c(y, z)$$

Distância de edição

- Fazendo

$$c(x, y) = \begin{cases} 0 & \text{se } x = y \\ 1 & \text{se } x \neq y \end{cases}$$

e

$$h = 1$$

temos a *distância de edição* (ou *distância de Levenshtein*).

Similaridade e distância

- Distância e similaridade podem ser usadas para computar alinhamentos.
- Distância não funciona bem para alinhamentos locais: o menor número de operações de edição será 0 para todos os pares de subseqüências iguais.
- Em um esquema de similaridade aditiva, quebrando o alunhamento em dois blocos a soma das pontuações dos blocos é igual à pontuação inteiro.
- Um esquema de similaridade aditivo para o alinhamento de duas cadeias e uma distância entre duas cadeias está relacionada da seguinte forma:

$$sim(\mathcal{A}_{s,t}) + dist(s,t) = M/2(m+n)$$

para uma constante M.