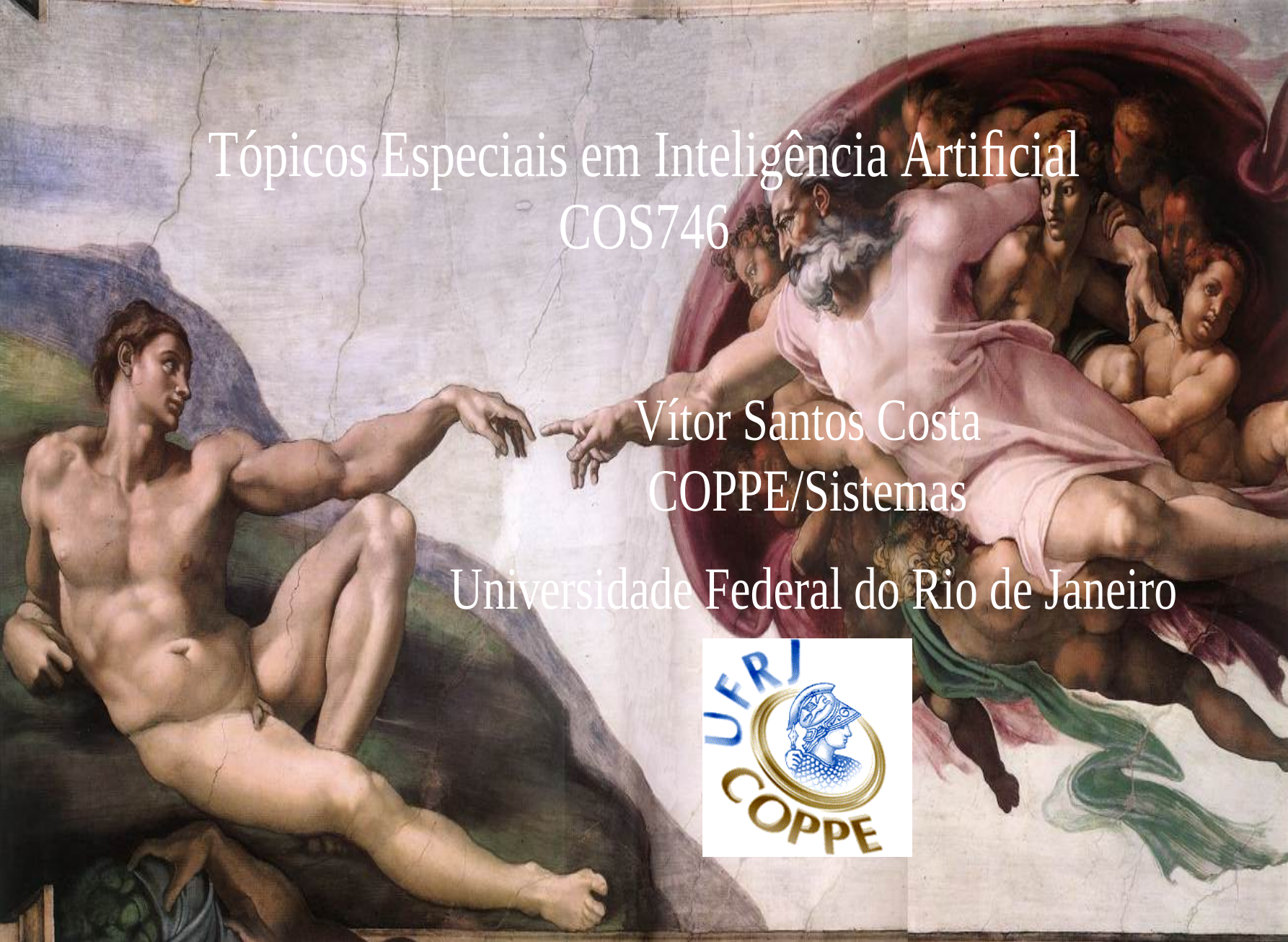


Tópicos Especiais em Inteligência Artificial
COS746

Vítor Santos Costa
COPPE/Sistemas
Universidade Federal do Rio de Janeiro

O Passo de Expectativa

- Agora podemos calcular a probabilidade de que o símbolo i tenha sido produzido pelo estado k, dado uma sequência x:

$$\begin{aligned} Pr(\pi_i|k,x) &= \frac{Pr(\pi_i|k)}{Pr(x)} \\ &= \frac{f_k(i)b_k(i)}{Pr(x)} \\ &= \frac{f_k(i)b_k(i)}{f_N(L)} \end{aligned}$$

O Passo de Expectativa

- A partir daqui podemos calcular o número esperado de vezes que a letra c é emitida pelo estado k.
- De notar que adicionamos o índice j para referir a uma sequência específica no conjunto de treino.

$$n_{k,c} = \sum_{x^j} [\frac{1}{Pr(x^j)} \sum_{\{i|x^j=c\}} f_k^j(i) b_k^j(i)]$$

  - ⋆ Soma sobre todas as sequências x^j no conjunto de treino
  - ⋆ Soma sobre todas as posições onde c aparece em x^j

O Passo de Expectativa

- e podemos calcular o número esperado de vezes que a transição de k para l é usada:

$$n_{k \to l} = \sum_{x^j} \frac{\sum_i f_k^j(i) a_{kl} e_l(x_{i+1}^j) b_l^j(i+1)}{Pr(x^j)}$$

- ou se l é um estado silencioso:

$$n_{k \to l} = \sum_{x^j} \frac{\sum_i f_k^j(i) a_{kl} b_l^j(i)}{Pr(x^j)}$$

O Passo de Maximização

- Seja $n_{k,c}$ o número esperado de emisões de c a partir do estado k para o conjunto de treino.
- Estime novos parâmetros de emissão por:

$$e_k(c) = \frac{n_{k,c}}{\sum_{c'} n_{k,c'}}$$

- Exactamente como no caso simples
- Mas habitualmente fazemos algum *amaciamento*, (ie, adicionar psuedocontagens).

O Passo de Maximização

- Deixe $n_{k \to l}$ ser o número esperado de transições desde o estado k para o estado l para o conjunto de treino
- estime novos parâmetros de transição como:

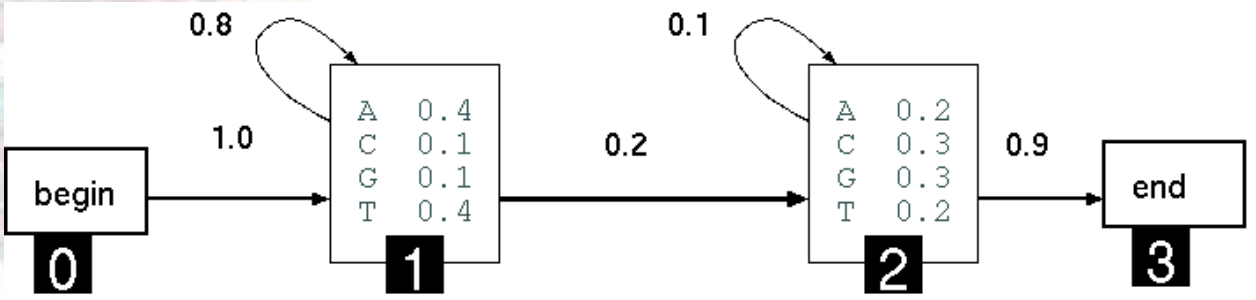
$$a_{kl} = \frac{n_{k \to l}}{\sum_m n_{k \to m}}$$

O Algoritmo de Baum-Welch

- inicializar os parâmetros do HMM
- itere até convergir:
  - ⋆ inicializar $n_{k,c}$.$n_{k\to l}$ com pseudo-contagens
  - ⋆ *Passo-E*: para cada sequência de treino $j = 1 \ldots n$
    - ∗ calcule $f_k(i)$ para a sequência j
    - ∗ calcule $b_k(i)$ para a sequência j
    - ∗ adicione a contribuição da sequência j a $n_{k,c}$,$n_{k\to l}$
  - ⋆ *Passo-M*: atualize os parâmetros do HMM usando $n_{k,c}$,$n_{k\to l}$

Exemplo do Algoritmo de Baum-Welch

- dado
  - ⋆ um HMM com os parâmetros inicializados como em baixo
  - ⋆ sequências de treino `TAG`, `ACG`



- vamos seguir uma iteração de Baum-Welch

Exemplo do Algoritmo de Baum-Welch

- determinando os valores de forward para `TAG`:

$$
\begin{aligned}
f_0(0) &= 1 \\
f_1(1) &= e_1(\texttt{T}) \times a_{01} \times f_0(0) = 0.4 \times 1 = 0.4 \\
f_1(2) &= e_1(\texttt{A}) \times a_{11} \times f_1(1) = 0.4 \times 0.8 \times 0.4 = 0.128 \\
f_2(2) &= e_2(\texttt{A}) \times a_{12} \times f_1(1) = 0.1 \times 0.2 \times 0.4 = 0.008 \\
f_2(3) &= e_2(\texttt{G}) \times (a_{12} \times f_1(2) = 0.1 + a_2 2 \times f_2(2)) = \\
&\quad 0.4 \times (0.0008 + 0.0256) = 0.01056 \\
f_3(3) &= a_{23} \times f_2(3) = 0.9 \times 0.01056 = 0.009504
\end{aligned}
$$

- computamos apenas valores que representam probabilidade não-zero
- da mesma forma podemos computar os valores de forward de `ACG`

Exemplo do Algoritmo de Baum-Welch

- determinando os valores de backward para TAG:

$$
\begin{aligned}
b_3(3) &= 1 \\
b_2(3) &= 1_{23} \times b_3(3) = 0.9 \times 1 = 0.9 \\
b_2(2) &= a_{22} \times e_2(\mathtt{G}) \times b_2(3) = 0.1 \times 0.4 \times 0.9 = 0.036 \\
b_1(2) &= a_{12} \times e_2(\mathtt{G}) \times b_2(3) = 0.2 \times 0.4 \times 0.9 = 0.072 \\
b_1(1) &= a_{11} \times e_1(\mathtt{A}) \times b_1(2) + a_{12} \times e_2(\mathtt{A}) \times b_2(2) = \\
&\quad 0.8 \times 0.4 \times 0.072 + 0.2 \times 0.1 \times 0.036 = 0.02376 \\
b_0(0) &= a_{01} \times e_1(\mathtt{T}) \times b_1(1) = 1.0 \times 0.4 \times 0.02376 = 0.009504
\end{aligned}
$$

- computamos apenas valores que representam probabilidade não-zero
- da mesma forma podemos computar os valores de backward de ACG

Exemplo do Algoritmo de Baum-Welch

- determinando o número esperado de contagens de emissões para o estado 1:

	contribuição de `TAG`		contribuição de `ACG`		pseudo contagem
$n_{1,A} =$	$\frac{f_1(2)b_1(2)}{f_3(3)}$	$+$	$\frac{f_1(1)b_1(1)}{f_3(3)}$	$+$	1
$n_{1,C} =$			$\frac{f_1(2)b_1(2)}{f_3(3)}$	$+$	1
$n_{1,G} =$					1
$n_{1,T} =$	$\frac{f_1(1)b_1(1)}{f_3(3)}$			$+$	1

- note que os valores de forward/backward diferem entre as duas sequências.

Exemplo do Algoritmo de Baum-Welch

- determinando o número esperado de transições para o estado 1 (não usamos pseudo-contagens):

$$n_{1\to 1} = \underbrace{\frac{f_1(1)a_{11}e_1(\texttt{A})b_1(2)}{f_3(3)}}_{\text{contribuição de } \texttt{TAG}} + \underbrace{\frac{f_1(1)a_{11}e_1(\texttt{C})b_1(2)}{f_3(3)}}_{\text{contribuição de } \texttt{ACG}}$$

$$n_{1\to 2} = \frac{f_1(1)a_{12}e_2(\texttt{A})b_2(2)+f_1(2)a_{12}e_2(\texttt{G})b_2(3)}{f_3(3)} + \frac{f_1(1)a_{12}e_2(\texttt{C})b_2(2)+f_1(2)a_{12}e_2(\texttt{G})b_2(3)}{f_3(3)}$$

- da mesma forma, podemos determinar os números esperados de emissão e de transição para o estado 2.

Exemplo do Algoritmo de Baum-Welch

- Determinando as probabilidades para o estado 1:

$$e_1(\mathtt{A}) = \frac{n_{1,A}}{n_{1,A}+n_{1,C}+n_{1,G}+n_{1,T}}$$

$$e_1(\mathtt{C}) = \frac{n_{1,C}}{n_{1,A}+n_{1,C}+n_{1,G}+n_{1,T}}$$

$$\ldots$$

$$a_{11} = \frac{n_{1\to 1}}{n_{1\to 1}+n_{1\to 2}}$$

$$a_{12} = \frac{n_{1\to 2}}{n_{1\to 1}+n_{1\to 2}}$$

Convergência em Baum-Welch

- Alguns critérios de convergência:
  - ⋆ "likelihood" da sequência de treino não muda muito
  - ⋆ número máximo de iterações
- habitualmente converge num número pequeno de iterações
- converge para máximo *local* (na likelihood dos dados dado o modelo)

$$logPr(\text{sequências}|\theta) = \sum_{x^j} logPr(x^j|\theta)$$

Complexidade Computacional dos Algoritmos HMM

- Dado um HMM com S estados e uma sequência de comprimento L, a complexidade dos algoritmos Forward, Backward e de Viterbi é de:

$$O(S^2 L)$$

  - ⋆ Isto assume que os estados são densamente interligados

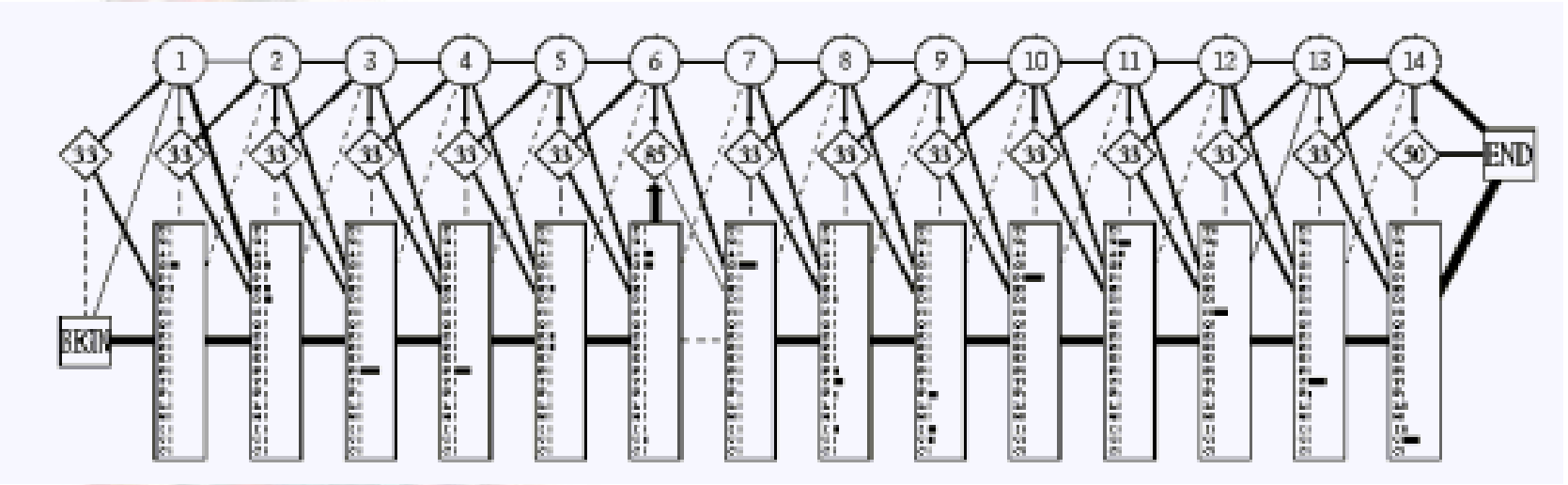
- Dadas M sequências de comprimento L, a complexidade de Baum-Welch em *cada iteração* é:

$$O(M S^2 L)$$

Uma Aplicação de Aprendizagem com Baum-Welch em HMMs

- Modelando famílias de proteínas usando *profile HMMs*
- *profile HMMs* podem ser usados para:
  - determinar uma sequência múltipla de alinhamento para um conjunto de proteínas
  - detectar novo membros de uma família de proteínas
- Aplicação mais importante de HMMs em BioComputação:
  - HMMER: `http://hmmer.wustl.edu/`
  - `http://www.cse.ucsc.edu/research/compbio/ismb99.tutorial.html`
  - `http://www.people.virginia.edu/~wrp/cshl98/`

Um Profile HMM treinado para o Domínio SH3



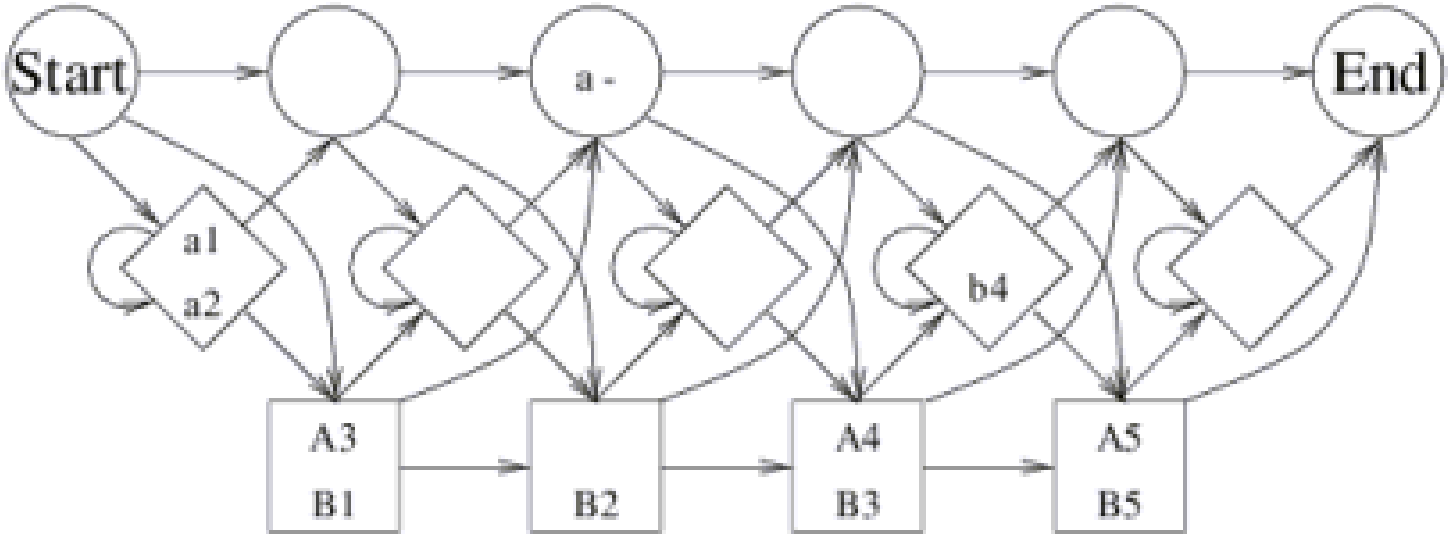
A Estrutura de um Profile HMM

- *Estados de Emparelhamento*: representam posições essencialmente conservadas na família de sequências
- *Estados de Inserção*: representam subsequências que foram inseridas em alguns membros da família
- *Estados de Remoção*: representam subsequências que foram removidas em alguns membros da família

Alinhamento Múltiplo no Domínio SH3

```
G G W W R G d y . g g k k q L W F P S N Y V
I G W L N G y n e t t g e r G D F P G T Y V
P N W W E G q l . . n n r r G I F P S N Y V
D E W W Q A r r . . d e q i G I V P S K - -
G E W W K A q r . . t g q e G F I P F N F V
G D W W L A r s . . s g q t G Y I P S N Y V
G D W W D A e l . . k g r r G K V P D N Y L
- D W W E A r s l s s g h r G Y V P S N Y V
G D W W Y A r s l i t n s e G Y I P S T Y V
G E W W K A r s l a t r k e G Y I P S N Y V
G D W W L A r s l v t g r e G Y V P S N F V
G E W W K A q t . k n g q . G W V P S N Y I
S D W W R V v n l t t r q e G L I P L N F V
L P W W R A r d . k n g q e G Y I P S N Y I
R D W W E F r s k t v y t p G Y Y E S G Y V
E H W W K V k d . q l g n v G Y I P S N Y V
I H W W R V q d . r n g h e G Y V O S S Y L
K D W W K V e v . . n d r q G F V P A A Y V
V G W M P G l n e r t r q r G D F P G T Y V
P D W W E G e l . . n g q r G V F P A S Y V
E N W W N G e i . . g n r k G I F P A T Y V
E E W L E G e c . . k g k v G I F P K V F V
G G W W K G d y . g t r i q Q Y F P S N Y V
D G W W R G s y . . n g q v G W F P S N Y V
Q G W W R G e i . . y g r v G W F P A N Y V
G R W W K A r r . a n g e t G I I P S N Y V
G G W T Q G e l . k s g q k G W A P T N Y L
G D W W E A r s n . t g e n G Y I P S N Y V
N D W W T G r t . . n g k e G I F P A N Y V
```

Como é que os Profile HMMs Representam Famílias de Proteínas

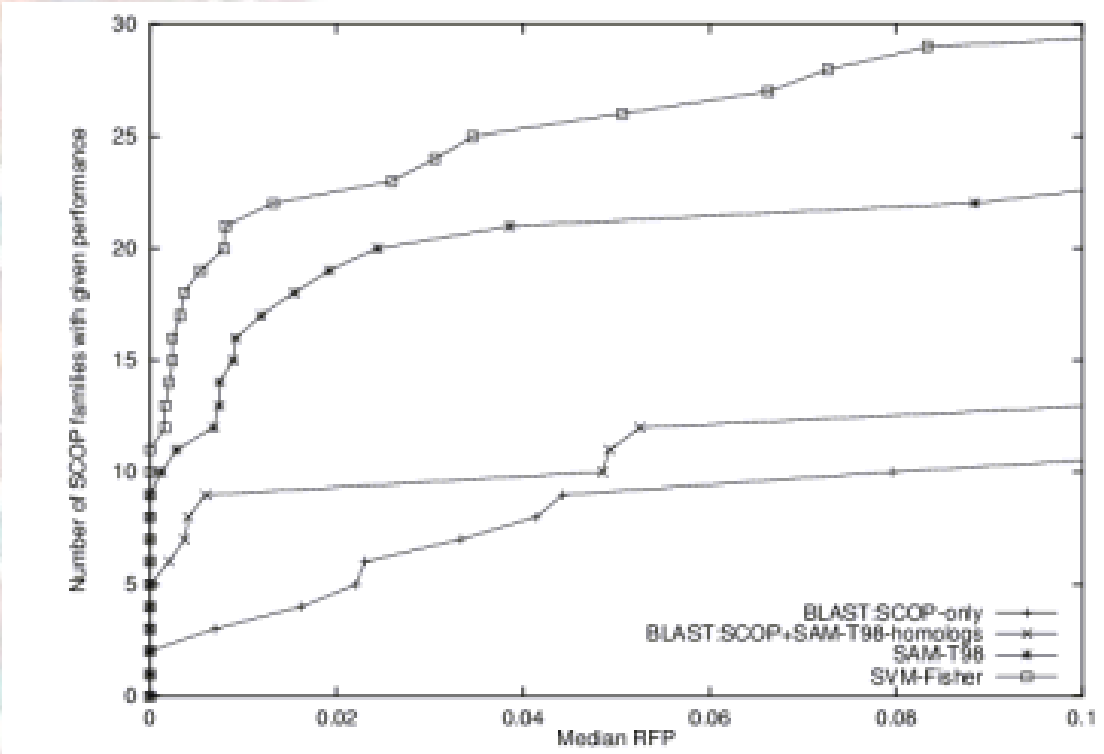



```
a1  a2  A3      -      A4   .  A5
 .   .  B1      B2     B3  b4  B5
```

Classificando Sequências: Três Métodos

- escolha um limite sobre $Pr(x)$ que permita boa discriminação entre casos positivos e casos negativos:
  - ⋆ depende do comprimento de x
- construa um *modelo nulo*; rode uma sequência x pelos dois para ver quem obtém um $Pr(x)$ maior
- construa um conjunto de modelos para famílias disjuntos; rode a sequência de interrogação x por todos os modelos para ver quem obtém o maior $Pr(x)$

Acuracia de Profile HMM



- classificando 2447 proteínas em 33 famílias
- o eixo dos x representa a mediana das sequências negativas que têm scores tão altos como uma sequência positiva para um dado modelo de família

Selecção de Modelo para Profile HMMs

- assumimos que recebemos um modelo de comprimento especificado; como determinar o comprimento?
- heurísticas:
  - ⋆ escolha um comprimento inicial; aprenda parâmetros
  - ⋆ se mais do que $x - del\%$ dos caminhos de Viterbi, vão por posições de remoção na posição k, remova essa posição do modelo
  - ⋆ se mais do que que $x - ins\%$ vão por inserções na posição k, adicione novas posição ao modelo.
  - ⋆ itere

Comentários sobre Modelos de Markov

- Tém muitas aplicações bem-sucedidas em biologia computacional
  - reconhecimento de genes e tarefas associadas
  - modelagem de famílias de proteínas
  - modelagem de motivos
  - e mais
- Existem muitas variantes dos modelos que consideramos aqui:
  - modelos de motivos de tamanho fixo
  - modelos de semi-markov
  - gramáticas estocástias de contexto livre
  - amostragem de Gibbs para aprender parâmetros

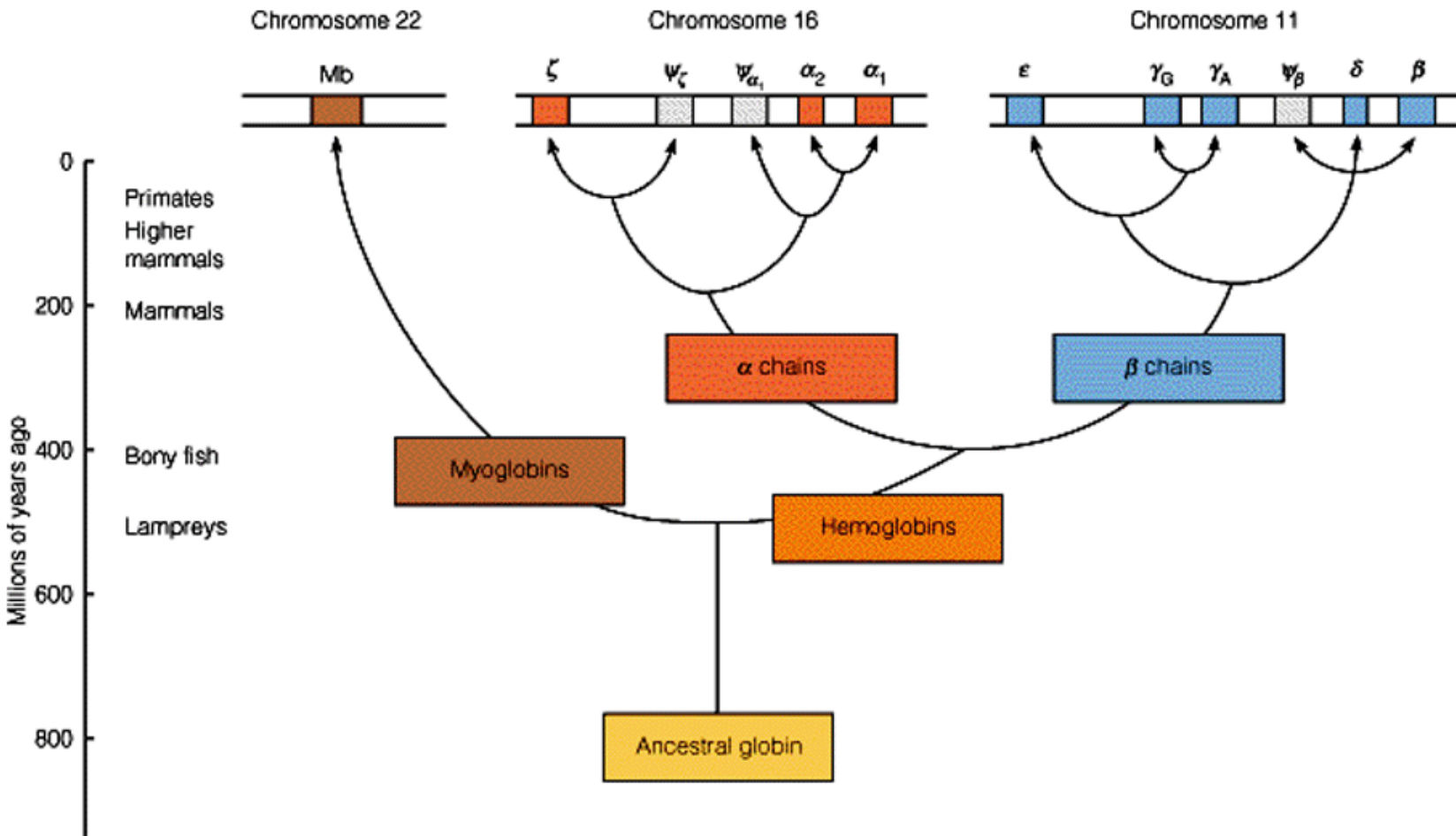
Redes Bayesianas: Mais Tutoriais

- http://www.ai.mit.edu/~murphyk/Bayes/bayes.html
- Dynamic Bayes Nets com uma aplicação: http://www.cs.wisc.edu/~dpage/cs731.html
- Tutoriais de Andrew Moore sobre Statistical Data Mining: http://www-2.cs.cmu.edu/~awm/tutorials/
- PRMs e POMDPs: www.stanford.edu/~grenager
- Estatistica Bayesiana: http://www.stat.cmu.edu/~minka/dynamic.html

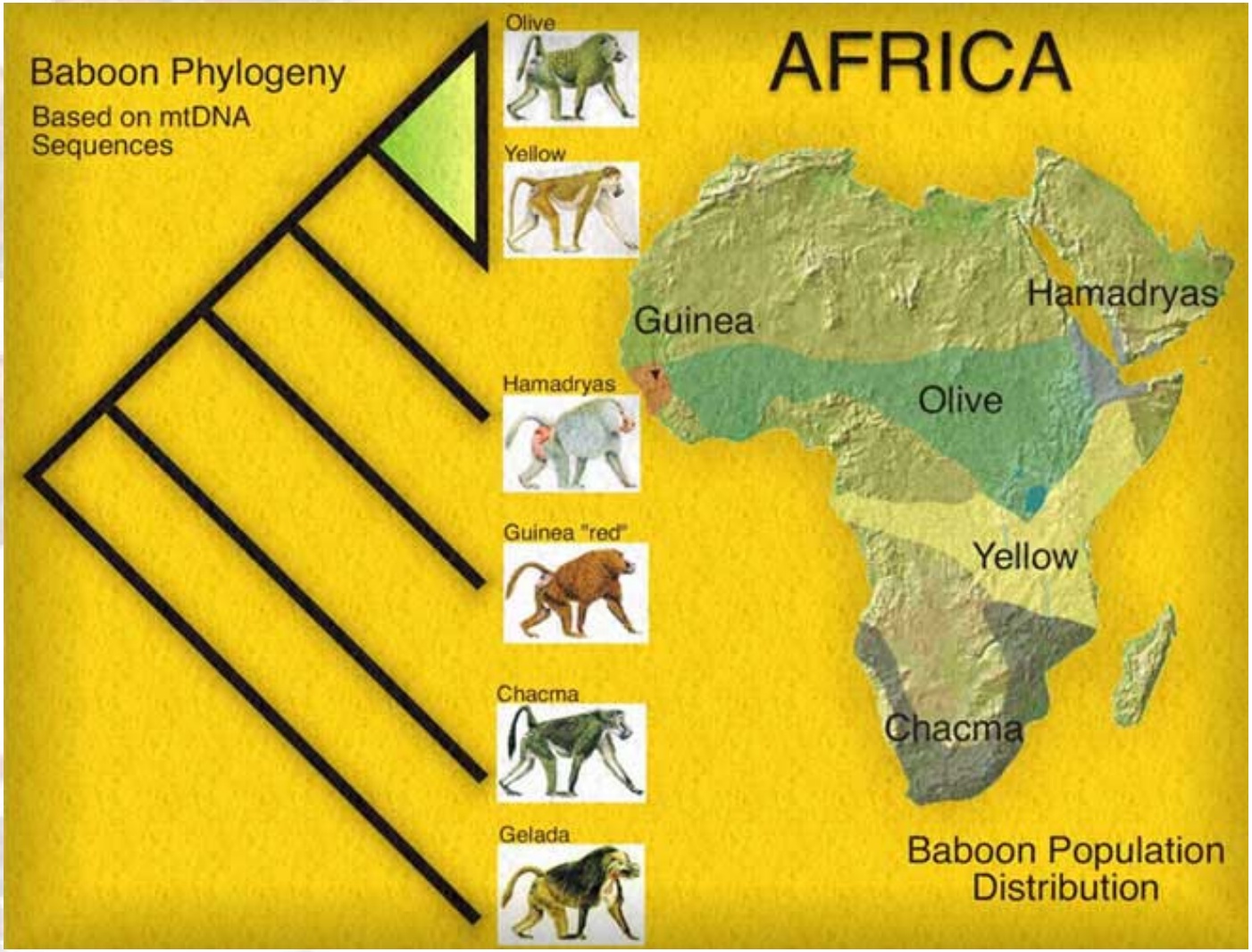
Árvores Filogenéticas

- *Árvore Filogenética*: diagrama mostrando a linha evolucionária de espécies ou de genes
- Porquê usar árvores:
  - ⋆ para entender a ascendência de várias espécies
  - ⋆ para compreender como várias funções evoluiram
  - ⋆ para informar sobre alinhamentos múltiplos

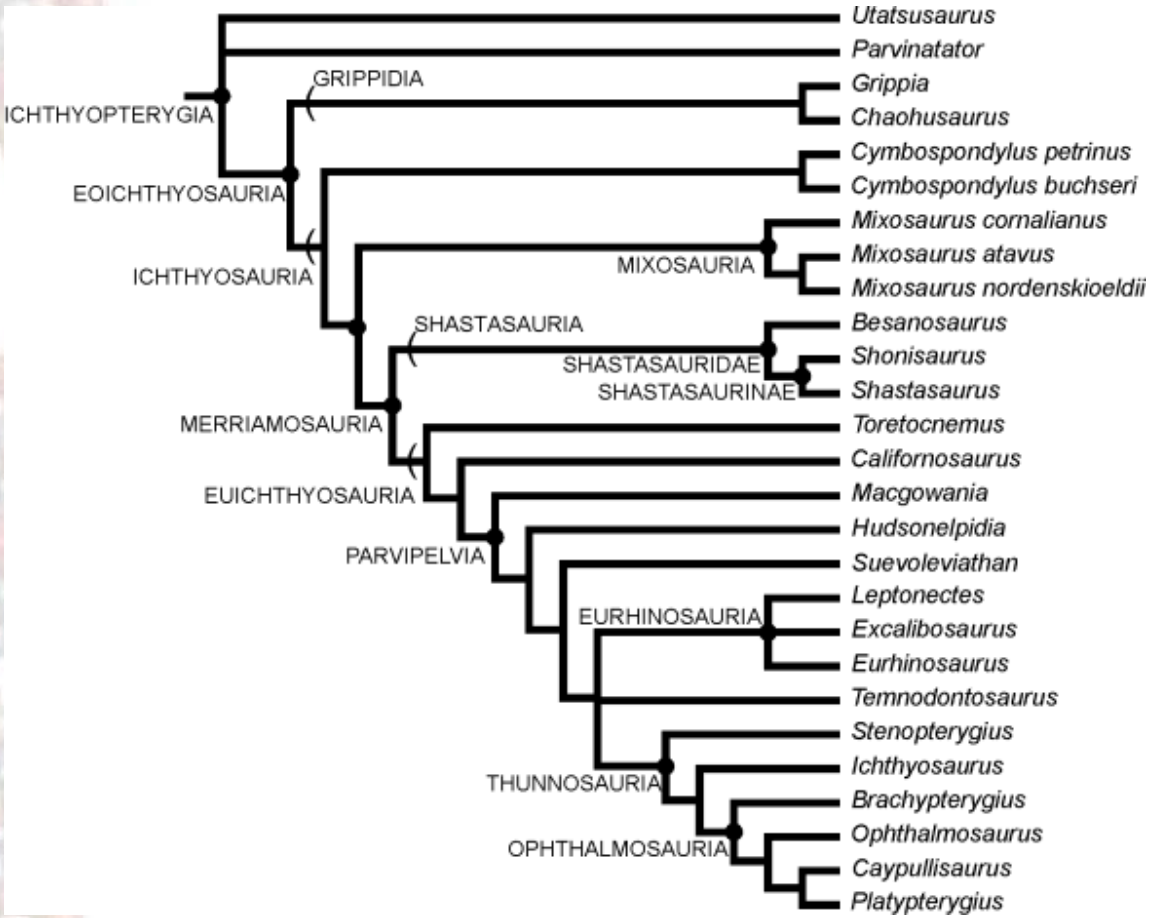
Globin evolution and expression



Exemplo de Filogenia: Babuínos



Exemplo de Filogenia: Ichtiousaurus



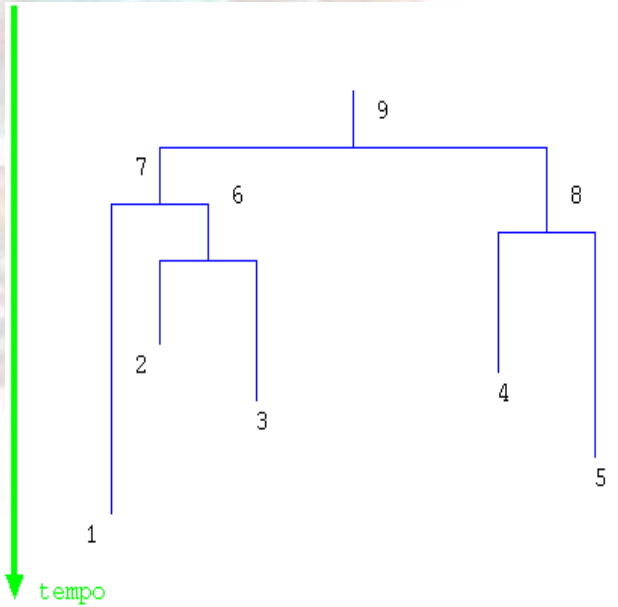
Árvores Filogenéticas: Ideias Básicas

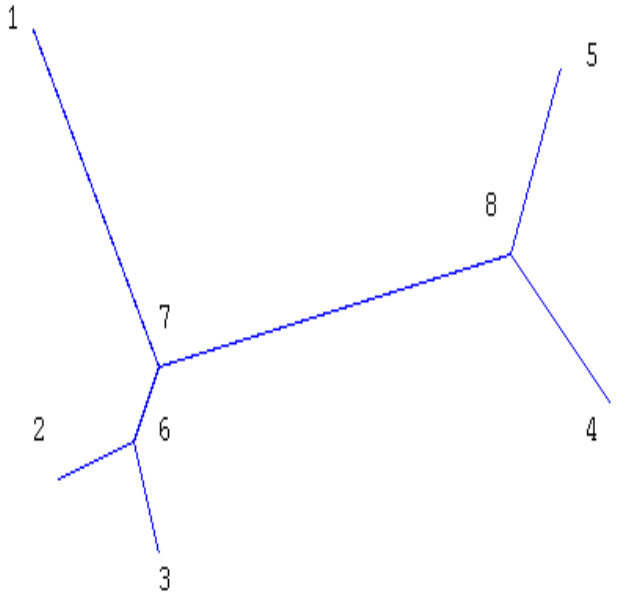
- Folhas representam coisas (genes, individuos/famílias, espécies) sendo comparadas
  - o termo *taxon* é usado para referir a esses elementos quando representam espécies e classificações mais amplas de organismos
  - vamos chamá-las de sequências
- nós internos são hipóteticos antepassados
- numa árvore enraizada, um caminho desde a raíz até a um nó representa um caminho evolucionário
- uma árvore não-enraizada representa relações entre coisas, mas não caminhos evolucionários

Dados para Construir Árvores

- Árvores podem ser construídas de vários tipos de dados:
  - ⋆ *baseados em distâncias*: medidas de distâncias entre espécies/genes
  - ⋆ *baseados em caracteres*: traços morfológicos (eg, pernas), sequências de DNA/proteínas
  - ⋆ *ordem de genes*: ordem linear de genes ortológicos encontrados em genomas dados

Árvores Enraizadas e Não-Enraizadas





Número de Árvores Possíveis

- dadas n sequências, existem $\prod_{i=3}^{n}(2i-5)$ árvores não-enraizadas possíveis
- e $(2n-3)\prod_{i=3}^{n}(2i-5)$ árvores enraizadas

Número de Árvores Possíveis

# sequências (n)	# árvores não-enraizadas	# árvores enraizadas
4	3	15
5	15	105
6	105	945
8	10,395	135,135
10	2,027,025	34,459,425

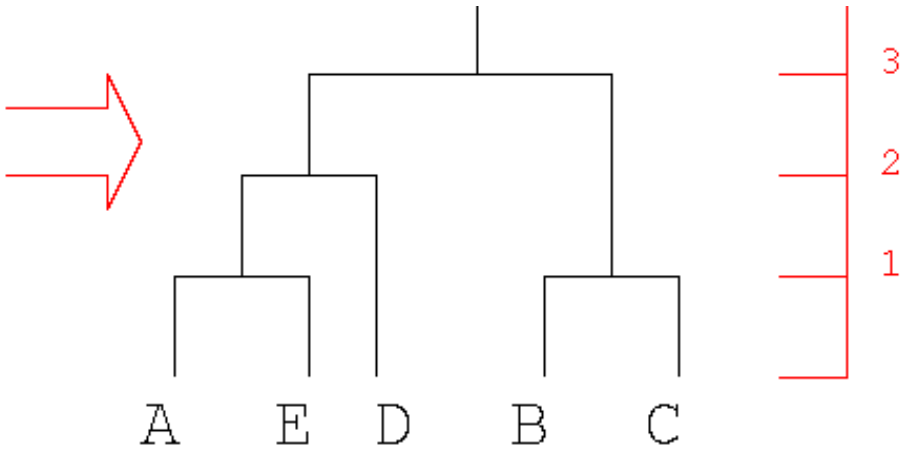
Construção de Árvores Filogenéticas

- Três tipos de métodos gerais:
  - ⋆ *distância*: encontrar uma árvore que explique as distâncias evolucionárias estimadas
  - ⋆ *parsimónia*: encontrar a árvore que requere o número mínimo de alterações para explicar os dados
  - ⋆ *maximum likelihood*: encontrar uma árvore que maximize a likelihood dos dados

Métodos Baseados em Distância

- **Dados:** uma matriz $n \times n$ M onde M_{ij} é a distância entre os objectos i e j
- **faça:** construa uma árvore pesada nas arestas tal que a distância entre as folhas i e j corresponda a M_ij

	A	B	C	D	E
A	0	8	8	5	3
B		0	3	8	8
C			0	8	8
D				0	5
E					0



O Método UPGMA

- Unweighted Pair Group Method using Arithmetic Averages
- Ideia básica:
  - ⋆ Iterativamente tirar duas sequências/clusters e agragá-los
  - ⋆ criar novo nó na árvore para o cluster agregado
- a distância d_{ij} entre os clusters C_i e C_j de sequências é definida como:

$$d_{ij} = \frac{1}{|C_i||C_j|} \sum_{p \in C_i, q \in C_j} d_{pq}$$

ou distância média entre pares de sequências de cada cluster

Algoritmo UPGA

- Dar a cada sequência o seu próprio cluster
- definir uma folha para cada sequência e colocar na altura 0
- enquanto há mais de 2 clusters:
  - ⋆ determinar dois clusters i, j com o menor d_{ij}
  - ⋆ defina um novo cluster $C_k = C_i \cup C_j$
  - ⋆ defina um nó k com filhos i e j, coloque-o na altura $d_{ij}/2$
  - ⋆ substitua os clusters i e j com k
- junte os últimos dois clusters, i e j, pela raíz na altura $d_{ij}/2$

UPGMA

- dado um novo cluster C_k formado pela agregação de C_i e de C_j
- podemos calcular a distância entre C_k e qualquer outro cluster C_i como segue:

$$d_{kl} = \frac{d_{il}|C_i| + d_{jl}|C_j|}{|C_i| + |C_j|}$$

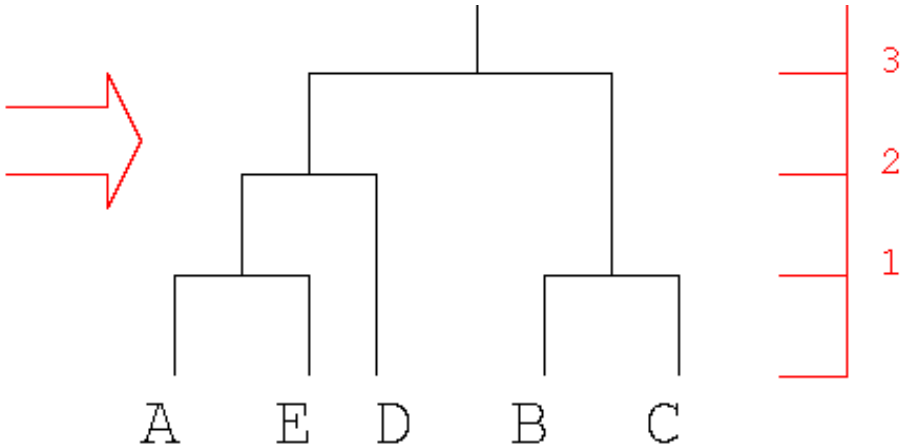
A Premissa do Relógio Molecular e Dados Ultraméricos

- A *premissa do relógio molecular*: divergência das sequências é assumida ocorrer à mesma velocidade em todos os pontos da árvore
- esta premissa não é verdade em geral: pressões evolucionárias variam de acordo com o tempo, organismos, genes num organismo e regiões num gene
- se não podemos assumir esta premissa, os dados são chamados de *ultramericos*

A Premissa do Relógio Molecular e Dados Ultraméricos

- Dados ultraméricos: para qualquer tripla de sequências i, j, k as distâncias ou são todas iguais, ou duas são iguais e a restante é mais pequena.

	A	B	C	D	E
A	0	8	8	5	3
B		0	3	8	8
C			0	8	8
D				0	5
E					0



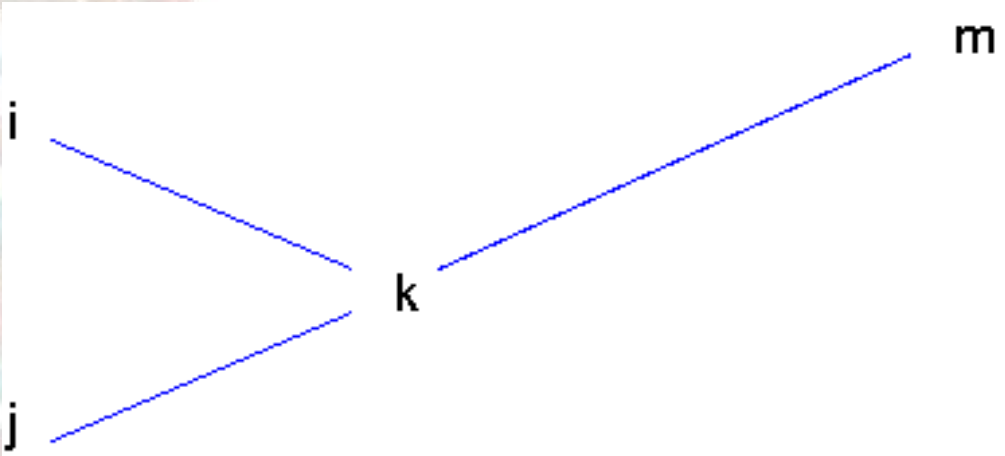
Junção de Vizinhos

- com em UPGMA, construimos uma árvore juntando iterativamente sub-árvores
- diferente de UPGMA:
  - ⋆ não assumimos o relógio molecular
  - ⋆ produz árvore não enraizada
- asumme *aditividade*: a distância entre dois pares de folhas é a soma dos comprimentos dos vértices que fazem a ligação.

Distâncias em Junção de Vizinhos

- dado um novo nó interno k, a distância para outro nó m é dada por:

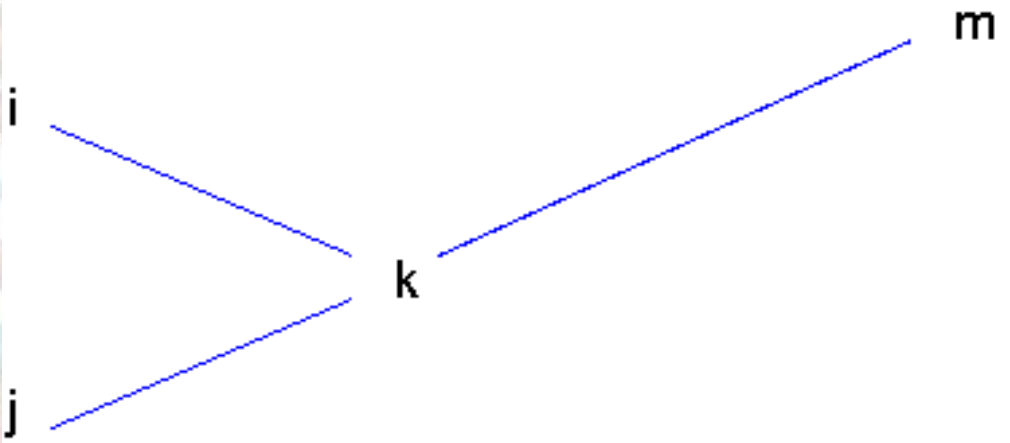
$$d_{km} = \frac{1}{2}(d_{im} + d_{jm} - d_{ij})$$



Distâncias em Junção de Vizinhos

- Podemos calcular a distância de uma folha para o nó pai na segunte forma:

$$d_{ik} = \frac{1}{2}(d_{ij} + d_{im} - d_{jm})$$



$$d_{jk} = d_{ij} - d_{ik}$$

Distâncias em Junção de Vizinhos

- Podemos generalizar esta regra de forma a tomar em conta a distância para todas as outras folhas:

$$d_{ik} = \frac{1}{2}(d_{ij} + r_i - r_j)$$

onde

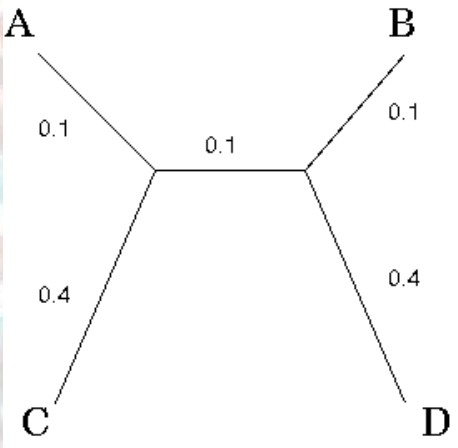
$$r_i = \frac{1}{|L| - 2} \sum_{m \in L} d_{im}$$

e L é o conjunto das folhas

- isto é mais robusto se os dados não forem estritamente aditivos

Juntar que Nós?

- Em cada passo escolhemos um par de nós para juntar. Devemos escolher os nós com o menor d_{ij}?
- Suponhamos que a árvore verdadeira parece como isto e que estamos a escolher os primeiros nós para juntar:



$$d_{AB} = 0.3$$
$$d_{AC} = 0.5$$

- Decisão errada em juntar A e B: precisamos de considerar distância do par até outras folhas.

Juntar que Nós?

- Para evitar o problema escolha o par de nós baseado nas distâncias baseado em D_{ij}:

$$D_{ij} = d_{ij} - (r_i + r_j)$$

$$r_i = \frac{1}{|L| - 2} \sum_{k \in L} d_{ik}$$

Algoritmo de Junção de Vizinhos

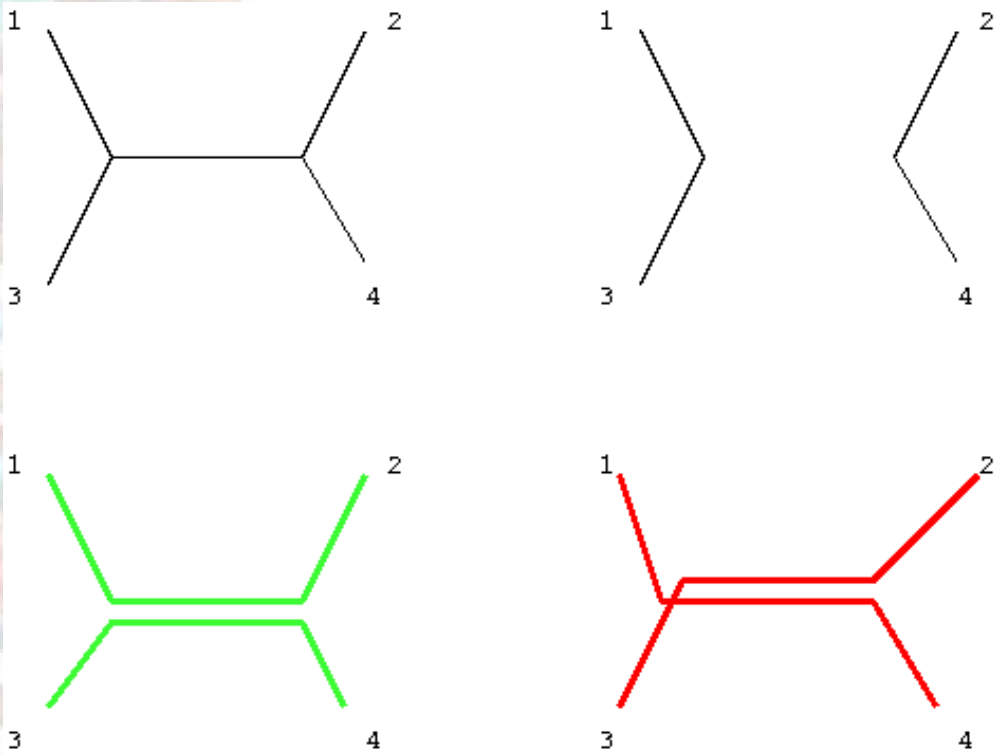
- defina a árvore T como o conjunto de nós folhas
- $L = T$
- enquanto há mais que duas sub-árvores em T:
  - ⋆ escolha o par i, j em L com D_{ij} mínimo
  - ⋆ adicione a T um novo nó agregando i e j
  - ⋆ determine novas distâncias:

$$d_{ik} = \frac{1}{2}(d_{ij} + r_i - r_j)$$
$$d_{jk} = d_{ij} - dik$$
$$d_{km} = \frac{1}{2}(d_{im} + d_{jm} - d_{ij}) \text{para todos os outros} m \in L$$

  - ⋆ remova i e j de L e insira k (processe-o como se uma folha)
- junte as duas árvores restantes, i e j com um vértice de comprimento d_{ij}

Testando Aditividade

- Para qualquer conjunto de qualquer folhas i, j, k, l duas das distâncias $d_{ij} + d_{kl}$, $d_{ik} + d_{jl}$ e $d_{il} + d_{jk}$ devem ser iguais e maiores que a terceira distância



Escolhendo Raízes

- Escolher uma raíz para árvores não-enraizadas é muitas vezes feita usando um “out-group”
- *Outgroup* é uma espécie que se sabe ser mais diferentes das outras espécies do que elas são entre elas.
- o ponto onde o outgroup se junta ao resto da árvore é o melhor candidato para a raíz.

Comentários Sobre Métodos Baseados em Distância

- Se os dados de distância são ultraméricos (e as distâncias são distâncias genuínas), então UPGMA encontra a árvore certa
- Se os dados são aditivos (e as distâncias são distâncias genuínas), então junção de vizinhos identifica a árvore correcta
- senão, os métodos podem não recuperar a árvore correcta, mas são boas heurísticas