Universidade Federal de Pernambuco

Cátia Mesquita Brasil Khouri

Modelos Escondidos de Markov para Classificação de Proteínas

Recife

Dezembro de 2002

Cátia Mesquita Brasil Khouri

Orientação: Profª Dra. Katia Silva Guimarães

Modelos Escondidos de Markov para Classificação de Proteínas

Dissertação apresentada ao Centro de Informática da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

Recife

3 de maio de 2004

Agradecimentos

A Deus, autor e doador de toda manifestação de vida neste planeta, por ter-me concedido realizar esse trabalho.

Ao meu querido esposo Walter Júnior, que em muitas ocasiões abriu mão de mim e de si mesmo, e ao mesmo tempo caminhou comigo, na preparação deste trabalho.

Aos meus filhos, Davi e Daniel, que com compreensão torceram por mim, mesmo nas horas de muita saudade.

À minha orientadora, Profa. Dra. Katia Silva Guimarães, pela sua dedicação, apoio e amizade, que possibilitaram a conclusão deste trabalho com êxito.

A Leonardo Sapucaia, que me ajudou na implementação da ferramenta, e aos demais alunos da graduação em Ciência da Computação da UESB, por seu interesse e estímulo.

À amiga Corina, pelo carinho e apoio nas horas mais difíceis.

Aos colegas do curso, pelo companheirismo e amizade, especialmente a Maísa, nosso ponto de apoio em Recife.

Resumo

A Biologia Molecular apresenta-se como uma área da Biologia bastante fértil em aplicações de técnicas computacionais. A estrutura das moléculas de ácidos nucléicos e proteínas, composta de partículas alinhadas ao longo de uma cadeia, permite-lhes serem tratadas computacionalmente como seqüências de símbolos de um alfabeto finito. O estudo das similaridades existentes entre seqüências distintas de proteínas que desempenham a mesma função pode ajudar a traçar caminhos evolucionários comuns e descobrir semelhanças entre diferentes organismos, que podem levar à compreensão de famílias inteiras, contribuindo para a definição de mecanismos gerais que regem as formas de vida na Terra.

Modelos Escondidos de Markov – HMMs, têm-se apresentado como uma excelente técnica para a comparação de seqüências de proteínas, suportada por uma forte fundamentação matemática. Este processo de modelagem é baseado nas características estatísticas do objeto de estudo, o qual é visto como um processo aleatório parametrizado, cujos parâmetros podem ser determinados de uma maneira bem definida e precisa. No projeto de um HMM, há três problemas fundamentais a serem resolvidos: (1) Avaliação da probabilidade de uma seqüência de observações, dado o HMM; (2) Determinação da melhor seqüência de estados (a mais provável); (3) Ajuste dos parâmetros do modelo, de acordo com a seqüência observada. Neste trabalho é apresentada uma arquitetura de HMM para modelagem de famílias de proteínas, que é implementada com uma técnica de aprendizagem de máquina a qual permite que os parâmetros do modelo, tais como penalidades por remoções, inserções e substituições, sejam aprendidos durante a construção do modelo, sem a introdução de conhecimento prévio.

Para aplicar a técnica, foi desenvolvida uma ferramenta para construção de um HMM capaz de classificar seqüências de proteínas. Foram realizados experimentos com três famílias de proteínas, a saber, globinas, proteinoquinases e GTPases. Para cada família, um HMM foi treinado usando um conjunto de seqüências daquela família. Os resultados dos experimentos mostram que a técnica HMM é capaz de explorar informações estatísticas contidas em uma grande quantidade de seqüências de proteínas de uma mesma família. Os HMM's construídos são capazes de distinguir com um alto grau de precisão seqüências membros de seqüências não membros das famílias modeladas.

Abstract

The Molecular Biology is an area of the Biology quite fertile in applications of computational techniques. The structure of the nucleic acids and protein molecules, composed of particles aligned along a chain, allows it to be dealt with computationally as sequences of symbols of a finite alphabet. The study of the existent similarities among different protein sequences that carry out the same function can help design common evolutionary ways and to discover similarities among different organisms, that can lead to the understanding of whole families, contributing to the definition of general mechanisms that govern the forms of life in the Earth.

Hidden Markov Models – HMMs, have presented themselves as an excellent technique for the comparison of protein sequences, supported by a strong mathematical foundation. This modeling process is based on the statistical characteristics of the subject, which is seen as a parameterized random process, whose parameters can be determined in a well defined and accurate way. In the project of a HMM, there are three fundamental problems that must be solved: (1) Evaluation of the probability of a sequence of observations, given an HMM; (2) Determination of the best sequence of states (the most probable); (3) Adjustment of the parameters of the model, in agreement with the observed sequence. In this work, an HMM architecture is presented for modeling of protein families. The architecture is implemented with a machine learning technique, which allows the parameters of the model, such as penalties for removals, insertions and substitutions, to be learned during the construction of the model, without the introduction of previous knowledge.

To apply the technique, a tool was developed for construction of a HMM capable of classifying protein sequences. Experiments were accomplished with three families of proteins, namely globins, proteinkinases and GTPases. For each family, an HMM was trained using a set of sequences belonging to that family. The results of the experiments show that the HMM technique is capable of exploring the statistic information contained in a great amount of protein sequences belonging to the same family. The HMM's built are capable of distinguishing sequences members of no members of the modeled families with a high degree of precision.

Sumário

Índice de Figuras

Capítulo 1

As seqüências de DNA e proteínas

A Biologia Molecular apresenta-se, atualmente, como uma área da Biologia bastante fértil para a aplicação de técnicas computacionais. A estrutura das moléculas de ácidos nucléicos e proteínas, composta de partículas alinhadas ao longo de uma cadeia, permite-lhes serem tratadas computacionalmente como seqüências de símbolos de um alfabeto finito.

Neste capítulo são apresentadas noções básicas de Biologia Molecular, como a estrutura das moléculas de DNA e proteínas, mecanismos de replicação do DNA e sintetização de proteínas. Em seguida, apresentam-se algumas das técnicas computacionais que têm sido utilizadas no tratamento dos problemas da Biologia Molecular, mais especificamente, comparação de seqüências e identificação de genes.

1.1 Um pouco de Biologia Molecular

Desde quando o homem começou a cultivar lavouras ou criar animais, sabe-se que cada semente cultivada possui os dados necessários para o desenvolvimento de um organismo. Também desde muito cedo se observava que características de uma geração eram passadas para a próxima, mas não se sabia com isso acontecia. Apenas em 1806, Gregory Mendel hipotetizou que traços fenotípicos são o resultado da interação entre partículas discretas, as quais hoje são conhecidas como **genes** [1].

No início do século XX, muitos cientistas já achavam que os cromossomos eram responsáveis pela hereditariedade. Descobriu-se que os cromossomos são compostos de DNA e proteínas, sendo que o DNA é uma molécula linear constituída de apenas 4 subunidades básicas,

enquanto que a proteína é composta de 20 subunidades básicas e possui uma forma tridimensional mais complexa, apresentando várias dobras. Pelo fato do DNA ser bem menos complexo do que a proteína, pareceu aos cientistas da época que uma estrutura tão simples não poderia ser responsável por carregar toda a informação genética de um organismo, de modo que eles atribuíram essa função às proteínas.

Em 1952, Alfred Hershey e Martha Chase [1] mostraram que o DNA é que de fato carrega a informação genética, e em 1953, James Watson e Francis Crick [2] publicaram um artigo descrevendo a estrutura do DNA. A determinação desta estrutura é freqüentemente referenciada como um marco para a biologia molecular moderna.

1.1.1 A Estrutura do DNA

O DNA é uma grande molécula composta de apenas 4 tipos de subunidades, chamadas de **nucleotídeos** (desoxirribonucleotídeos). Cada nucleotídeo é constituído de um açúcar (desoxirribose), um fosfato e uma das quatro bases: Adenina, Citosina, Guanina ou Timina, usualmente denotadas por A, C, G e T, respectivamente. Estas bases dividem-se em duas categorias, as bases maiores – Adenina e Guanina, são **purinas**, enquanto que as menores – Citosina e Timina, são **pirimidinas**. As desoxirriboses formam o que se chama de **esqueleto** da molécula de DNA.

A estrutura básica do DNA é descrita como uma dupla hélice, onde cada hélice ou fita é um polímero de nucleotídeos e ambas giram em torno do mesmo eixo. Em cada fita os nucleotídeos estão organizados de modo a formar uma cadeia (Figura 1. 1).

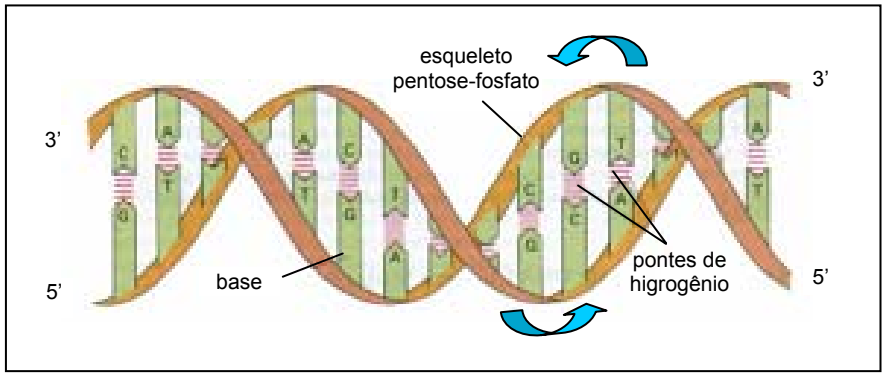


Figura 1. 1 A estrutura do DNA é descrita como uma dupla hélice. Bases complementares se combinam: Adenina-Timina e Guanina-Citosina

As duas hélices são mantidas juntas através de pontes de hidrogênio que ligam as bases de ambas as fitas aos pares, de modo que uma base de uma fita fica lado a lado com uma base da outra fita, formando um par. Os pares de bases são formados obedecendo sempre ao seguinte critério: uma purina será sempre ligada a uma pirimidina e, ainda mais restritamente, guanina forma par com citosina e adenina forma par com timina. As bases guanina com citosina e adenina com timina são por isso chamadas **complementares**. Isto significa que se a seqüência de bases em uma fita é dada, a seqüência da outra fita é automaticamente determinada, pois uma é complementar à outra.

Uma cadeia é construída direcionalmente, devido à estrutura assimétrica dos açúcares que constituem o esqueleto da molécula. Os nucleotídeos são unidos através de ligações fosfato covalentes que ligam o carbono 5 de um grupo desoxirribose ao carbono 3 do grupo adjacente, formando uma cadeia. Cada nucleotídeo é conectado à cadeia na direção do carbono 5 ao carbono 3; por isso diz-se que a fita do DNA vai de 5' (lê-se 5 linha) a 3' (3 linha). A direção das duas fitas é inversa, uma da outra (Figura 1. 1).

Para cada organismo, a molécula de DNA possui quantidades diferentes de bases, que por sua vez são dispostas de modos distintos. A seqüência em que os nucleotídeos estão organizados na fitas é que determina quais as informações genéticas que a célula carrega. Cada nucleotídeo é representado por uma letra no alfabeto de 4 letras – A, C, G, T – que é usado para escrever mensagens genéticas codificadas em forma linear. Uma vez que o alfabeto em questão é composto de 4 letras, são possíveis 4^n diferentes cadeias de comprimento n. O comprimento total do DNA humano é estimado em 3 x 10^9 nucleotídeos, de modo que é possível um número de $4^{(3\times10^9)}$ cadeias diferentes possíveis.

1.1.2 Cromossomos e Genes

Os organismos vivos são divididos em dois grandes grupos: **procariotos** e **eucariotos**. Os procariotos são organismos com uma única célula, que não tem núcleo, enquanto que os eucariotos são organismos mais complexos, cujas células são compostas de um **núcleo** e o líquido celular, chamado **citoplasma**. No núcleo das células eucariotas existem elementos constituídos de estruturas contíguas chamados **cromossomos**, nos quais o DNA é armazenado.

Um **gene** é uma parte específica da seqüência de nucleotídeos ao longo de um cromossomo que geralmente carrega a informação necessária para a construção de uma proteína. Nos seres humanos, genes constituem apenas 5 a 10% do DNA; os outros 95 a 90% constituem-se de seqüências não gênicas, usualmente denominadas '*junk DNA*'. Não se conhece ainda qual a

função dos não gênicos, mas experimentos envolvendo sua supressão demonstraram serem vitais. Estima-se que o homem possui entre 30.000 e 50.000 genes. A expressão do gene é o processo biológico pelo qual a seqüência de DNA gera uma seqüência de proteína. Nem todos os genes numa seqüência de DNA são expressos.

1.1.3 Replicação do DNA

A descoberta da estrutura da molécula do DNA, composta de duas fitas complementares, permitiu também descobrir como se dá o processo de cópia e transferência das informações genéticas de uma célula para sua prole. Uma vez que cada fita de uma molécula é complementar à outra, podemos concluir que ambas carregam a mesma informação genética.

Considerando que *A* e *A'* são fitas que compõem uma determinada molécula de DNA, a fita *A* pode servir de molde para construir-se uma cópia de *A'* da mesma forma que *A'* pode servir de molde para construir uma cópia de *A*. E é exatamente desta maneira que ocorre nos organismos vivos; a fita *A* se separa da fita *A'* e a partir de cada uma é feita uma cópia de sua complementar, ao final do que tem-se duas cópias da molécula original. Desta forma, a informação genética existente no DNA pode ser replicada em mais DNA (Figura 1. 2).

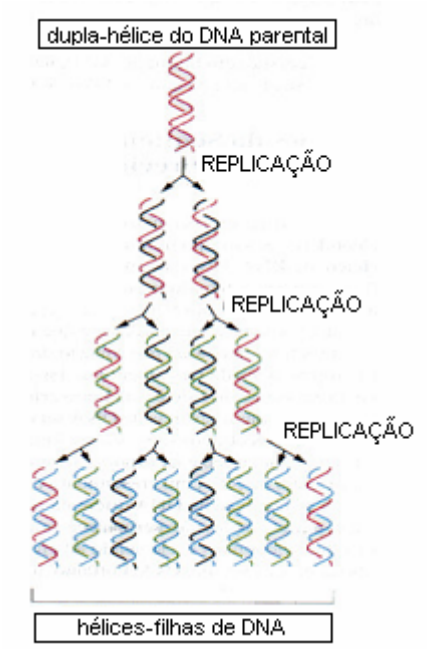


Figura 1. 2 A máquina natural de replicação do DNA

A máquina de replicação natural trabalha com uma margem de menos de 1 erro em cada 10^9 nucleotídeos adicionados. Ocasionalmente, portanto, ela comete um erro, adicionando ou não alguns nucleotídeos, ou colocando um **A** onde deveria colocar **G**, ou um **T** no lugar de um **C**.

Estes erros são chamados de **mutações** e podem ter conseqüências mais ou menos graves, a depender de onde eles ocorrem. Uma mutação pode implicar na ausência de uma proteína vital para a célula, ocasionando sua morte, o que conseqüentemente interromperá a replicação do erro. Outras vezes, a mutação pode não afetar a função da proteína. Muito raramente a mutação produzirá um gene melhorado, com uma função nova, de modo que os organismos com esta mutação terão vantagens sobre os outros, e o gene mutado poderá vir a substituir o gene original na população através de seleção natural. Quando uma base errada é incorporada à nova fita, diz-se que ocorreu uma **substituição**. A adição de uma base ou bases a mais é chamada **inserção**, e a retirada de uma ou mais bases é chamada **remoção**.

1.1.4 Transcrição e RNA

O DNA é relativamente inerte quimicamente. As informações que ele carrega são expressas indiretamente através de outras moléculas: RNAs específicos e proteínas. Estas últimas é que definem as propriedades químicas de uma célula. O RNA é um outro tipo de ácido nucléico (Ácido Ribonucléico) que pode ser encontrado tanto no núcleo quanto no citoplasma da célula (diferente do DNA, que só existe no núcleo). O RNA é um polímero que difere quimicamente em dois aspectos do DNA: (1) o esqueleto do RNA é composto de riboses, ao invés de desoxirriboses; e como no DNA e (2) o RNA possui as bases Adenina, Citosina, Guanina, também presentes no DNA, e Uracil (denotada por U), em substituição à Timina (T); no RNA a base Uracil parea com a Adenina.

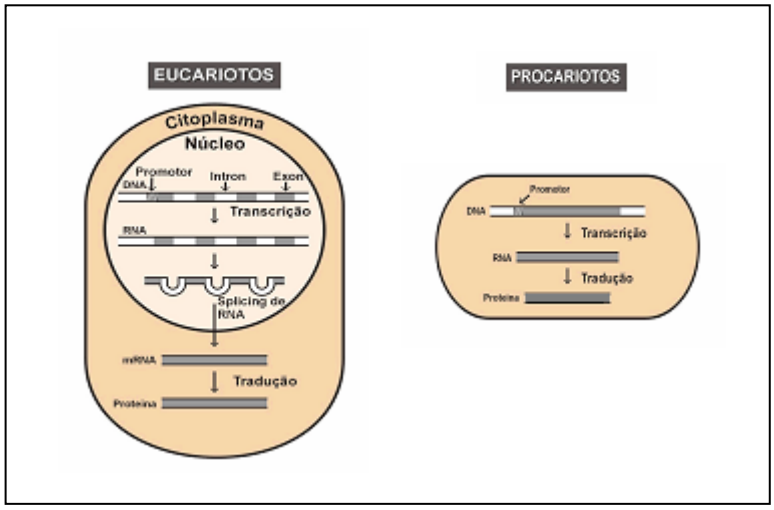


Figura 1. 3 Genes eucariotos e procariotos. Nos eucariotos a transcrição começa no núcleo da célula. Após o *splicing*, o RNA migra para o citoplasma. Nos procariotos, o gene é contínuo.

O RNA retém toda a informação genética do DNA, mas é composto de uma fita única, que é construída a partir de uma fita do DNA usada como molde. A informação existente no DNA é passada para o RNA num processo chamado **transcrição**, promovido pela enzima **RNA**

polimerase. Em organismos eucariotos a transcrição se dá em duas etapas (Figura 1. 3). Próximo à maioria dos genes, existe uma região padrão chamada de **Promotora**, que indica à RNA polimerase onde começar a transcrição, isto é, indica a localização do início do gene. Primeiramente o DNA é integralmente transcrito em RNA, ainda no núcleo da célula. Esta nova molécula é composta de fragmentos alternados chamados ***exons*** (regiões codificadoras) e ***introns*** (regiões não codificadoras). Depois do gene a transcrição pára e algumas dúzias de adenina são adicionadas no final da molécula de RNA para marcar o seu fim. A transcrição ocorre no sentido 5'-3' da fita de RNA (chamado sentido *dowstream*), que corresponde ao sentido 3'-5' da fita molde (chamado sentido *upstream*).

Em seguida, ocorre o '*splicing* do RNA', em que as seqüências não codificadoras de genes (*introns*) são removidas, produzindo uma molécula muito mais curta, chamada de **RNA mensageiro – mRNA**, que no citoplasma da célula servirá de molde para construção de uma proteína. A depender do tecido onde a transcrição ocorre, pode haver uma variação do padrão *splicing* – ***Alternative Splicing***, o que contribui para a diversidade das proteínas no organismo.

	U	C	A	G	
U	Phe	Ser	Tyr	Cys	U
	Phe	Ser	Tyr	Cys	C
	Leu	Ser	STOP	STOP	A
	Leu	Ser	STOP	Trp	G
C	Leu	Pro	His	Arg	U
	Leu	Pro	His	Arg	C
	Leu	Pro	Gln	Arg	A
	Leu	Pro	Gln	Arg	G
A	Ile	Thr	Asn	Ser	U
	Ile	Thr	Asn	Ser	C
	Ile	Thr	Lys	Arg	A
	Met(START)	Thr	Lys	Arg	G
G	Val	Ala	Asp	Gly	U
	Val	Ala	Asp	Gly	C
	Val	Ala	Glu	Gly	A
	Val	Ala	Glu	Gly	G

Figura 1. 4 O código genético Universal. O *codon* AUG, além de ser traduzido no aminoácido Metionina, indica o início da tradução. Os *stop codons* UAA, UAG e UGA indicam fim da tradução.

1.1.5 Tradução e Proteína

Em analogia ao fato de que DNA e RNA são cadeias de nucleotídeos, uma proteína é uma cadeia composta de um outro tipo de subunidade: **aminoácidos**. As proteínas são responsáveis pela maioria das reações químicas executadas na célula e portanto, essenciais para todas as suas

funções. Pode-se dividir as proteínas em dois grupos distintos: **estruturais**, que são aquelas que são utilizadas para confeccionar tecidos, e as **enzimas**, que têm ação catalisadora.

Após a transcrição do DNA em mRNA, este será traduzido em proteína conforme o **código genético**. A seqüência de nucleotídeos em uma seqüência de DNA é que define a seqüência dos aminoácidos na proteína que será produzida. A seqüência do mRNA é 'lida' de modo que cada grupo de três nucleotídeos, chamado *codon*, é traduzido em um aminoácido.

O código genético universal é o mapeamento lógico que especifica em que informação genética a seqüência do DNA implicará. A Figura 1. 4 mostra o código genético que é válido para a maioria dos organismos vivos, sejam eles plantas, bactérias ou humanos. Como cada *codon* é formado por 3 dos 4 nucleotídeos possíveis (com possibilidade de repetição), existem $4^3 = 64$ *codons* possíveis. Entretanto, em geral, apenas 20 aminoácidos distintos são encontrados em proteínas; assim, a maioria dos aminoácidos é codificada por mais de um *codon*, isto é, o código genético é degenerado. Alguns *codons* especiais, chamados ***stop codons***, são usados para sinalizar o término do processo de tradução. O *codon* AUG, também chamado ***start codon***, além de ser traduzido no aminoácido Metionina, representa o início da tradução. O código genético pode ser visto como uma função de 64 *codons* possíveis para 20 aminoácidos mais *stop*.

Existem regiões do DNA que são transcritas em mRNA porém não são traduzidas, isto é, não codificam proteínas. Estas regiões são chamadas **UTR** (*untranslated region*). Em organismos procariotos, um gene tem só uma região codificante, contígua, que fica entre as regiões 3' UTR e 5' UTR. Nos eucariotos, o mRNA é composto de regiões codificantes, que são usadas para a construção de proteínas, separadas por regiões não codificantes – 3' UTR e 5' UTR.

Os ribossomos são construtores celulares que completam o processo de sintetização das proteínas, utilizando pequenas moléculas de RNA chamadas **tRNA – RNA transportador**, que servem de adaptadores entre o mRNA e os aminoácidos. O tRNA mantém, de um lado, um *anticodon* (uma seqüência de três bases de RNA) e, de outro lado, o aminoácido correspondente no código genético. O tRNA carrega os aminoácidos para os ribossomos que se movem ao longo do mRNA, de modo que a cadeia de proteína vai se alongando de um aminoácido por vez, correspondente aos *codons* sucessivos no mRNA. Na Figura 1. 5, uma visão de uma célula num processo de construção da proteína desde a transcrição, no núcleo, até a tradução, no citoplasma.

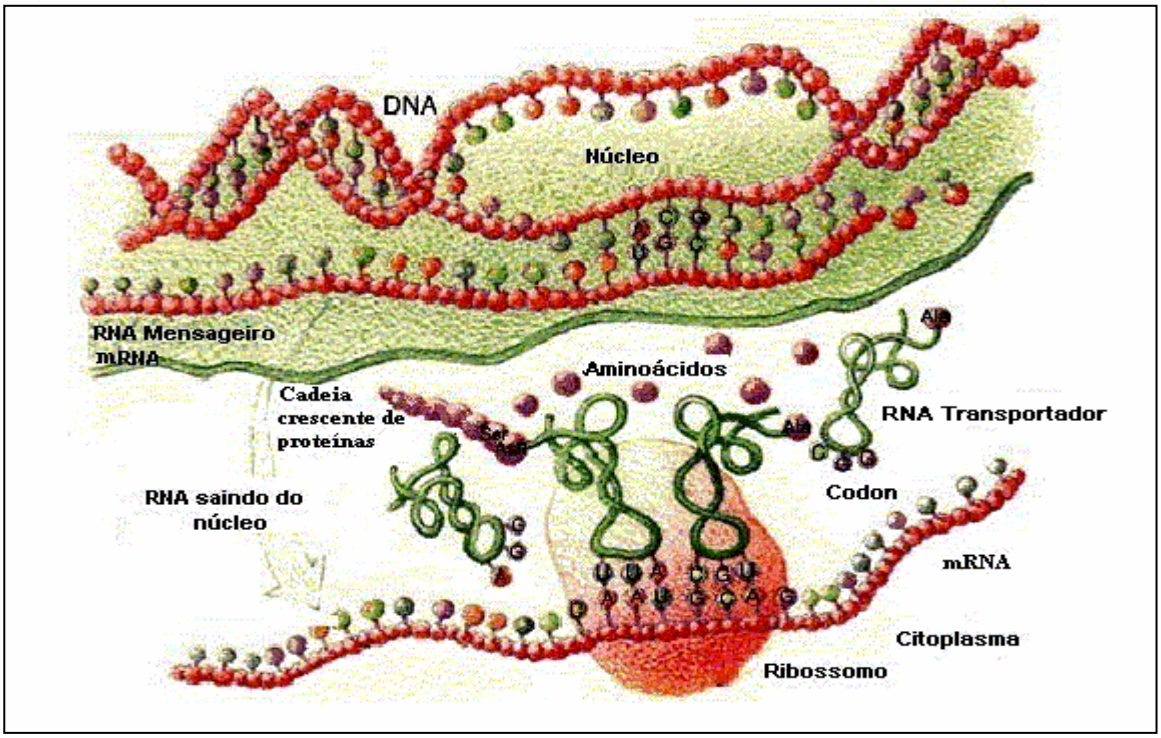


Figura 1. 5 A transcrição de DNA em mRNA ocorre no núcleo da célula. O mRNA move-se para o citoplasma e os ribossomos movem-se ao longo deste enquanto que moléculas de tRNA funcionam como adaptadores entre o anticodon e o aminoácido correspondente, estendendo a cadeia da proteína.

1.1.6 Estrutura da Proteína

Como foi dito acima, uma proteína é uma cadeia de aminoácidos. Diferentemente dos ácidos nucléicos, esta cadeia, em geral, dobra-se em uma conformação específica, determinada pela seqüência de aminoácidos. Existem, no entanto, alguns padrões estruturais que se repetem em partes das macromoléculas de proteínas devido a pontes de hidrogênio que se formam entre seus aminoácidos. Dois padrões são particularmente comuns e são conhecidos como **α-hélice** e **folha-β-pregueada** (ou simplesmente **folha-β**). Uma vez que uma α-hélice isolada em meio aquoso não é normalmente estável, duas α-hélices idênticas podem girar igualmente, uma em torno da outra, formando uma estrutura estável, chamada ***coiled-coil***. As cisteínas (duas unidades do aminoácidos cisteína) existentes numa cadeia normalmente se unem formando pontes dissulfídicas.

Na Figura 1. 6, são mostrados os três seguintes níveis da estrutura da proteína. As α-hélices e folhas-β-pregueadas constituem a **estrutura secundária** de uma proteína. Certas combinações de α-hélices e folhas-β juntas formam unidades compactas chamadas de **domínios da proteína**. Os domínios parecem ser as unidades a partir das quais as proteínas são construídas, sendo que proteínas menores podem conter apenas um domínio. A **estrutura terciária** é a

conformação tridimensional da molécula e pode ser entendida como uma combinação ou um 'empacotamento' de estruturas secundárias dentro de um ou mais domínios, incluindo pontes de heterogêneo e ligações fracas, formando uma subunidade protéica (monômero). Finalmente, a **estrutura quaternária** pode ser vista como um nível mais alto de dobramento, correspondendo à união de subunidades (dímero). Vale ressaltar que nem todas as proteínas possuem estrutura quaternária.

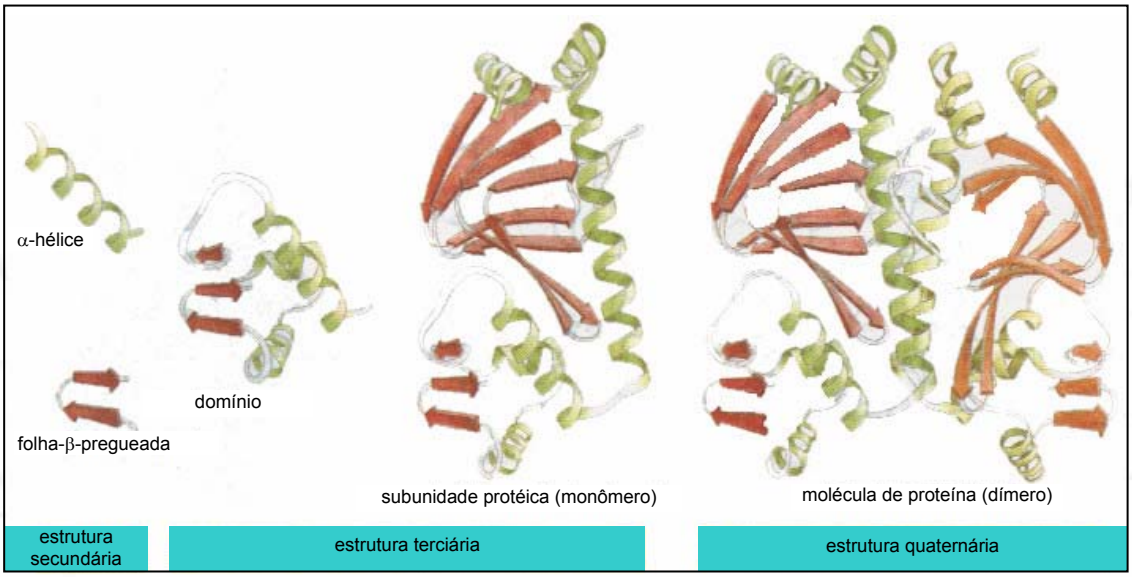


Figura 1. 6 Os níveis de estrutura da proteína. A estrutura primária corresponde simplesmente à seqüência de aminoácidos e as pontes dissulfídicas

A estrutura de uma proteína é praticamente definida pela seqüência primária de aminoácidos, que por sua vez é definida pela seqüência de DNA. As outras três estruturas dimensionais são, normalmente, aquelas com a mínima energia livre. A estrutura 3D de uma proteína determina a sua funcionalidade, portanto, é interessante poder determinar esta estrutura a fim de predizer sua função na célula.

1.1.7 Homologia

A estrutura química de todos os seres vivos é baseada em nucleotídeos, aminoácidos, açúcares e ácidos graxos; além disso, todos os seres vivos sintetizam seus constituintes químicos da mesma maneira, armazenam suas informações genéticas no DNA e as expressam através de RNA e proteínas. Algumas seqüências de nucleotídeos em genes específicos ou algumas seqüências de aminoácidos, apresentam grande similaridade.

Esta similaridade entre proteínas que desempenham a mesma função em organismos semelhantes, bem como nos genes que as codificam, sinaliza para a hipótese de que os

organismos em questão possuem uma mesma origem evolucionária. Comparar as seqüências de aminoácidos destas proteínas pode levar a importantes conclusões sobre a origem dos organismos onde elas se encontram. Seqüências semelhantes podem ser consideradas **homólogas**, se são originadas de um ancestral comum, ou **análogas**, se de ancestrais diferentes. As seqüências homólogas podem ainda ser classificadas como **parálogas**, se pertencem a um mesmo organismo, ou como **ortólogas**, se são de organismos distintos. Pode-se explorar estas semelhanças para traçar caminhos evolucionários comuns e a partir daí descobrir similaridades entre diferentes organismos. Assim o entendimento de um gene ou proteína pode levar à compreensão de famílias inteiras de homólogos, contribuindo para a definição de mecanismos gerais que regem as formas de vida na Terra.

É importante notar que proteínas homólogas podem apresentar seqüências relativamente diferentes, mas a sua forma – estrutura tridimensional – tende a ser semelhante. Duas regiões de seqüências que se apresentam muito similares, são ditas **conservadas**. A funcionalidade de uma proteína é determinada pela sua estrutura tridimensional (composição da seqüência de aminoácidos, formas e dobras). Assim, é importante estudar meios de predizer a estrutura da proteína, com o objetivo de compreender o seu papel nos organismos.

Algumas enzimas apresentam, em sua estrutura tridimensional, certas ‘bolsas’ para acomodar os reagentes. Estas bolsas, chamadas **sítios ativos** geralmente são bem conservadas entre organismos de espécies diferentes. Geralmente, regiões com um alto grau de conservação, desempenham um papel fundamental na atividade da proteína e são chamadas **motivos** (*motifs*).

Capítulo 2

Técnicas Computacionais Aplicadas à Análise de Seqüências de Ácidos Nucléicos e de Proteínas

O termo **genoma** foi empregado pela primeira vez pelo botânico alemão Hans Winkler [3], logo após a Primeira Guerra Mundial. O genoma de um organismo é o conjunto de todos os seus genes. Um dos principais objetivos dos projetos de seqüenciamento genômico é o estudo da estrutura, função e evolução dos genes. Para isso, laboratórios de seqüenciamento trabalham na determinação da seqüência de todos os genes do organismo estudado. A partir dessas seqüências, inúmeros tipos de análise podem ser realizados no sentido de extrair informações significantes sobre esses genes e conseqüentemente, do organismo em questão e mesmo de outros organismos que possam estar com ele relacionados.

Em 1994, em artigos publicados por Marc Wilkins e Keith Williams [3] apareceu pela primeira vez o termo **proteoma**, que diz respeito à expressão de todas as proteínas de um conjunto de cromossomos, isto é, o estudo de um proteoma refere-se ao estudo das proteínas produzidas por um dado genoma. Em um mesmo organismo (multicelular) este conjunto de proteínas varia com o tipo de célula e também com o tempo.

Uma vez que uma molécula de um ácido nucléico ou proteína pode ser vista como uma seqüência de caracteres – *string*, isto é, uma sucessão finita de símbolos pertencentes a um certo

conjunto Σ (o alfabeto), muitas técnicas computacionais têm sido utilizadas para sua análise. Nos últimos anos, a aplicação de tais técnicas tem sido atribuída a uma área de estudos que se dedica à aplicação de ferramentas computacionais aos problemas da biologia, a saber, a **Biologia Computacional**.

Neste contexto, técnicas computacionais têm sido apresentadas para a comparação de seqüências, montagem de fragmentos de seqüências, ou seqüenciamento, construção de árvores filogenéticas, predição da estrutura 3D de moléculas, entre outros. No decorrer deste capítulo serão vistas algumas das técnicas utilizadas para a análise de seqüências de ácidos nucléicos e proteínas, mais especificamente para comparação de seqüências e identificação de genes.

2.1 Alinhamento de seqüências

Uma abordagem que tem se revelado eficiente na análise de seqüências de DNA ou de proteínas é a análise comparativa dessas seqüências. Segundo Duret e Abdeddaim [4], a genômica comparativa é fundamental no estudo dos genes, pois,

> Realmente, a evolução dos organismos vivos pode ser considerada como um experimento em larga escala, contínuo, de mutações de genes. Por mais de três bilhões de anos, os genomas têm sofrido mutações (substituições, inserções, remoções, recombinações). Mutações de remoção são, geralmente, rapidamente eliminadas por seleção natural, enquanto que mutações que não têm efeito filogenético (mutações neutras) podem, por derivação genética aleatória, tornar-se fixos na população. Globalmente, vantagens obtidas das mutações são muito raras e quando resíduos são pobremente conservados durante a evolução geralmente correspondem a regiões que são fracamente constrangidas por seleção. Assim, estudar padrões de mutações através de análises de seqüências homólogas é útil não só para estudar as relações evolucionárias entre seqüências, mas também para identificar os constrangimentos estruturais e funcionais de seqüências (DNA, RNA, ou proteínas).

De fato, a comparação de seqüências é a operação mais básica no estudo de tais seqüências e está presente no estudo de diversos problemas da biologia computacional, dentre os quais citamos:

1. Comparação de resultados de seqüenciamento obtidos por diferentes laboratórios;
2. Identificação de sítios de funcionalidade através da identificação de regiões altamente conservadas;
3. Demonstração de homologias entre seqüências;
4. Construção de árvores filogenéticas a partir das quais pode-se inferir eventos de mutação e reconstruir o relacionamento evolucionário entre seqüências, permitindo a identificação de eventos de duplicação de genes para distinguir ortólogos de parálogos;

5. Busca de similaridades significantes, ainda que fracas, em bases de dados existentes, para identificar membros relativamente distantes de uma mesma família de proteínas – classificação de proteínas;
6. Predição da estrutura secundária de proteínas, podendo inclusive modelar homologias de estruturas conhecidas;
7. Predição de função, uma vez que a estrutura tridimensional das proteínas é freqüentemente muito mais conservada que a estrutura primária e que normalmente estruturas 3D similares implicam em funções similares. Assim, se um gene é determinado homólogo de outro cuja função já é conhecida, pode ser possível inferir a função do novo gene.

Intuitivamente, pode-se dizer que um alinhamento de seqüências consiste em compará-las colocando cada seqüência em uma linha (uma em baixo da outra), de modo que cada coluna seja composta por resíduos (nucleotídeos ou aminoácidos) que derivem de um ancestral comum. Para conseguir este alinhamento é necessário introduzir *gaps* (espaços) dentro das seqüências, forçando a que todas as seqüências fiquem com o mesmo comprimento, e admitir substituições. A introdução de um *gap* numa seqüência pode representar uma remoção nesta seqüência ou uma inserção nas demais seqüências alinhadas. São permitidos *gaps* no início e no final das seqüências também.

A Figura 2. 1 mostra um exemplo de alinhamento de 5 seqüências. O *gap* introduzido na coluna 4 das seqüências 2 e 4 pode representar uma remoção nestas seqüências ou uma inserção do resíduo K nas seqüências 1, 3 e 5. Pode-se inferir deste alinhamento também que na coluna 6 da seqüência 3 o resíduo N foi substituído por W.

	1	2	3	4	5	6	7	8	9	10	11	12
1	G	G	A	K	M	N	M	W	H	A	C	C
2	C	G	A	–	M	N	M	D	H	H	–	C
3	–	Y	A	K	M	W	W	D	–	A	C	–
4	–	Y	I	–	M	N	M	D	H	A	C	–
5	G	Y	A	K	F	N	M	W	C	A	G	C

Figura 2. 1 Alinhamento de 5 seqüências. Um alinhamento de seqüências de proteínas corresponde ao modelo hipotético das mutações que ocorreram durante a evolução.

Obviamente, existem muitos alinhamentos distintos, possíveis, para um mesmo conjunto de seqüências; no entanto, determinar qual o melhor alinhamento, isto é, aquele que melhor retrata o cenário evolucionário, não é trivial. Os algoritmos conhecidos que realizam alinhamento

de duas seqüências são baseados em programação dinâmica e possuem tempo de execução de ordem quadrática [5]. Quando o número de seqüências a serem alinhadas cresce muito, as generalizações dos procedimentos utilizados para duas seqüências redundam em tempos de execução inviáveis. Por esse motivo, algumas das ferramentas atualmente utilizadas para construir alinhamentos de várias seqüências são baseadas em algoritmos heurísticos que, portanto, não podem garantir a melhor solução.

2.1.1 Técnicas para Alinhamento de Seqüências

Antes de se abordar as técnicas para alinhamento de seqüências propriamente ditas, serão apresentadas algumas definições conforme Setubal e Meidanis [5].

(1) Uma **seqüência** ou **cadeia**, é uma sucessão finita de símbolos pertencentes a um alfabeto Σ. O número de caracteres de uma seqüência *s* é denotado por $|s|$. A seqüência vazia possui zero caracteres.

Exemplo.: para o alfabeto Σ = {A, C, G, T}, *s* = TTGCTA e *t* = ACTGCAAT são seqüências. $|s|$ = 6 e $|t|$=8.

(2) Dada uma seqüência *s*, uma **subseqüência** *t*, de *s*, é uma seqüência que pode ser obtida de *s* com a remoção de alguns caracteres.

Exemplo.: GTGC é uma subseqüência de AGCTAGC, enquanto que GTGA não é.

(3) Um trecho de caracteres consecutivos de *s* é uma **subcadeia** de *s*.

Exemplo.: GCT é uma subcadeia de AGCTAGC, enquanto que GCTG não é.

(4) A **concatenação** de duas seqüências *s* e *t* é a seqüência *st* obtida pela justaposição dos caracteres de *t* aos caracteres de *s*. Se *u* = *swt*, diz-se que *s* é um **prefixo** de *u*, *w* é um **fator** de *u* e que *t* é um **sufixo** de *u*.

(5) Uma **biosseqüência** é uma seqüência onde Σ = {A, C, G, T} (DNA), Σ = {A, C, G, U} (RNA) ou Σ = {A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, Y} (Proteínas).

Alinhamento Global – O Algoritmo Básico

Sejam as duas seqüências de proteínas s_1 = PKMAAGWT e s_2 = PKMAGST. Pode-se alinhar estas seqüências da seguinte maneira:

P	K	M	A	A	G	W	T
P	K	M	–	A	G	S	T

Figura 2. 2 Alinhamento de 2 seqüências

Ao alinhar-se as seqüências dadas percebe-se que as únicas diferenças entre elas são: (1) s_1 possui um A a mais que s_2 e (2) a 7ª posição das seqüências é ocupada por caracteres diferentes – W em s_1 e S em s_2.

Da mesma forma que no exemplo da Figura 2. 1, um *gap* teve que ser introduzido em s_2 de modo a que ambas as seqüências ficassem com o mesmo comprimento. Assim é que pode-se definir um **alinhamento global** entre duas seqüências como sendo a inserção de *gaps* em determinados locais das seqüências (inclusive no início ou no final delas) de modo que elas fiquem com o mesmo comprimento, não sendo permitido que ambas possuam *gaps* na mesma posição, isto é, que um *gap* em uma seqüência seja alinhado com um *gap* na outra.

A introdução de gaps em seqüências pode ser vista como conseqüência de eventos mutacionais durante a evolução. A probabilidade de ocorrência de diferentes eventos mutacionais deve ser levada em conta na computação de alinhamentos, quais sejam: substituição, inserção e remoção.

Pode-se medir a qualidade de um alinhamento atribuindo-se uma pontuação a cada coluna deste, conforme, por exemplo, os seguintes critérios:

se a coluna tiver duas bases iguais (*match*), receberá 1 ponto;

se a coluna tiver duas bases diferentes (*mismatch*), será penalizada com –1 ponto;

se a coluna tiver uma base e um *gap*, será penalizada com –2 pontos.

A **similaridade** entre as seqüências será dada pela pontuação total obtida com a soma dos pontos de cada coluna do alinhamento. Um **alinhamento ótimo** será aquele que possuir a pontuação máxima e, portanto, é possível haver mais de um alinhamento ótimo. No exemplo da Figura 2. 2, o alinhamento apresentado é ótimo e existe outra possibilidade:

	P	K	M	A	A	G	W	T		P	K	M	A	A	G	W	T
	P	K	M	–	A	G	S	T		P	K	M	A	–	G	S	T
pontuação:	1	1	1	-2	1	1	-1	1		1	1	1	1	-2	1	-1	1
pont. total:					3									3			

Figura 2. 3 Duas possibilidades de alinhamento ótimo

Há que se observar que a expressão ‘alinhamento ótimo’ que está sendo usada aqui, refere-se ao sentido matemático, isto é, aquele que possui pontuação máxima. No entanto, este pode não ser o melhor alinhamento no sentido biológico, isto é, pode não ser o alinhamento que melhor representa o cenário evolucionário.

A noção de **distância** entre duas seqüências, conforme será visto adiante, também deriva do conceito de mutações e da atribuição de pesos para estas mutações.

Um algoritmo para determinar um alinhamento ótimo de duas seqüências dadas $s=s_1, s_2, ..., s_m$ e $t=t_1, t_2, ..., t_n$, com $|s| = m$ e $|t| = n$, baseia-se em olhar para a última coluna de cada uma das seqüências e avaliar as três possibilidades:

- alinhar a última base de *s* com a última base de *t*:

$$\begin{matrix} s_1 & s_2 & \ldots & s_{m-1} & s_m \\ t_1 & t_2 & \ldots & t_{n-1} & t_n \end{matrix}$$

- alinhar a última base de *s* com um *gap* na última posição de *t*;

$$\begin{matrix} s_1 & s_2 & \ldots & s_{m-1} & s_m \\ t_1 & t_2 & \ldots & t_n & - \end{matrix}$$

- alinhar um gap na última posição de s com a última base de *t*;

$$\begin{matrix} s_1 & s_2 & \ldots & s_m & - \\ t_1 & t_2 & \ldots & t_{n-1} & t_n \end{matrix}$$

Uma vez que o alinhamento ótimo é aquele que traduz uma maior similaridade entre as seqüências, o problema reduz-se então a alinhar, recursivamente, as bases restantes de ambas seqüências, buscando a maior similaridade possível.

Seja $sim(s; t)$ a similaridade entre *s* e *t*. Tem-se:

Escolher o maior entre os três valores:

$$\begin{cases} \text{sim}(s_1, s_2, ..., s_{m-1}; t_1, t_2, ..., t_{n-1}) \pm p \\ \text{sim}(s_1, s_2, ..., s_m; t_1, t_2, ..., t_{n-1}) - g \\ \text{sim}(s_1, s_2, ..., s_{m-1}; t_1, t_2, ..., t_n) - g \end{cases}$$

Este procedimento gera um número exponencial de chamadas recursivas. Considerando-se que cada chamada gera outras três e que em duas delas o número de bases envolvidas diminui apenas de um, haverá, potencialmente, 3^{m+n} chamadas.

Setubal e Meidanis [5] mostram detalhadamente como o problema do alinhamento de duas seqüências pode ser resolvido através de programação dinâmica. Como exemplo, a Figura 2. 4 mostra uma matriz bidimensional A[*m+1, n+1*] usada para comparar as seqüências *s* = TGGA e *t* = TGA, considerando os pesos para *matches*, substituições e *gaps* utilizados acima. O algoritmo básico para preenchimento da matriz consome tempo e espaço O(*mn*).

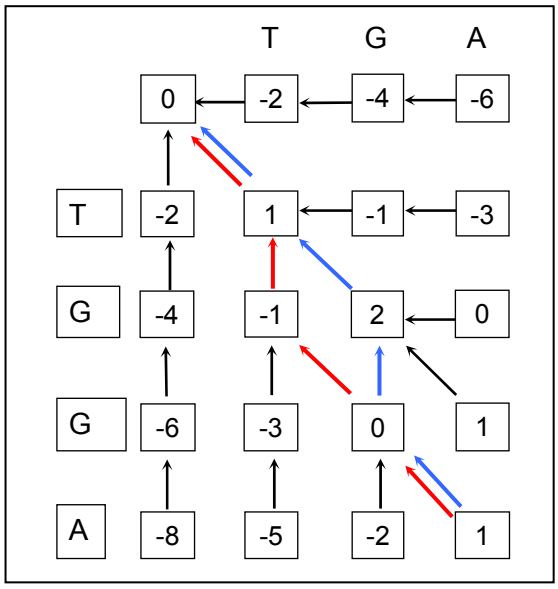


Figura 2. 4 Matriz para alinhamento das seqüências TGA e TGGA. *sim*(*s* ; *t*) = 1.

Seja *s*[*1..i*] o prefixo de *s* de tamanho *i* e *t*[*1..j*] o prefixo de *t* de tamanho *j*, então cada célula (*i, j*) da matriz contém o resultado da comparação de *s*[*1..i*] e *t*[*1..j*]. Inicialmente, são preenchidas a primeira linha e a primeira coluna da matriz, o que corresponde, respectivamente, a alinhar o prefixo vazio de *s* com cada prefixo de *t* (1ª linha) e a alinhar o prefixo vazio de *t* com cada prefixo de *s* (1ª coluna). Esta inicialização corresponde ao caso base da recursão. Cada uma das demais células é calculada da seguinte maneira:

$$a(i, j) = \text{máx}\begin{cases} a(i, j-1) + g \\ a(i-1, j-1) + p \\ a(i-1, j) + g \end{cases}$$

onde: g é o peso de um *gap* e p é o peso do alinhamento $s(i)$ com $t(j)$. No exemplo da Figura 2. 4, $p = +1$ se $s(i) = t(j)$, $p = -1$ se $s(i) \neq t(j)$, e $g = -2$. Na célula (m, n) é encontrado o valor da similaridade entre as seqüências completas.

Para se obter o alinhamento ótimo, são colocadas setas (ponteiros) indicando o lugar de onde veio o valor máximo utilizado em cada célula (em algumas situações é possível atingir o máximo por mais de um caminho). Tem-se, portanto, mais de um alinhamento ótimo. Seguindo-se a partir da célula (m, n), na direção indicada pelas setas até alcançar (0, 0), tem-se as seguintes possibilidades:

(1) se a seta com origem em (*i, j*) é horizontal, corresponde ao alinhamento de um *gap* em *s* com *t*(*j*);

(2) se a seta com origem em (*i, j*) é vertical, corresponde ao alinhamento de *s*(*i*) com um *gap* em *t*;

(3) se a seta com origem em (*i, j*) é inclinada, corresponde ao alinhamento de *s*(*i*) com *t*(*j*).

Um dos alinhamentos ótimos encontrados na Figura 2. 4, está indicado pelas setas em vermelho, o outro pelas setas em azul, os quais são mostrados na Figura 2. 5, como Alinhamento Ótimo 1 e Alinhamento Ótimo 2, respectivamente:

Alinhamento Ótimo 1					Alinhamento Ótimo 2				
s:	T	G	G	A	s:	T	G	G	A
t:	T	–	G	A	t:	T	G	–	A
pontuação:	1	-2	1	1	pontuação:	1	1	-2	1
pont. total:		1			pont. total:		1		

Figura 2. 5 Duas opções de alinhamento ótimo para as seqüências TGGA e TGA

Um algoritmo para obter o alinhamento a partir da matriz A, também é mostrado por Setubal e Meidanis [5] e consome tempo O(*l*), onde *l* é o comprimento do alinhamento retornado. Ali os autores citam algoritmos mais eficientes que, para determinadas escolhas de pesos, conseguem reduzir o tempo para $O(n^2/\log n)$ [6]. Apresentam também algumas modificações no algoritmo básico para preenchimento da matriz de modo que a complexidade de espaço diminui consideravelmente, passando a ser proporcional à soma dos tamanhos das seqüências de entrada, embora isto possa custar o aumento do tempo para até o dobro do original.

Extensões do Algoritmo Básico

Em algumas situações reais, pode ser necessário dar tratamento diferenciado a ***gaps* na extremidade**[1], bem como penalizar mais a abertura de um *gap* do que sua extensão. Em outras, pode-se desejar alinhar apenas uma subcadeia de *s* com uma cadeia *t*. É possível encontrar alinhamentos como estes, fazendo-se algumas alterações no algoritmo básico como segue.

Alinhamento Semiglobal

Um caso típico é o do problema da montagem de fragmentos de seqüências de DNA. O processo de seqüenciamento em larga escala (seqüenciamento de longas moléculas) de DNA envolve a múltipla replicação da molécula a ser seqüenciada, isto é, são fabricadas várias cópias da molécula, e a posterior quebra de todas estas em pedaços de tamanhos aleatórios. O objetivo é

[1] **gaps na extremidade** são aqueles que aparecem sempre antes do primeiro ou depois do último caractere de uma seqüência e recebem, via de regra, tratamento especial na análise de biosseqüências.

que ao final desta etapa do processo possam ser encontrados fragmentos como mostrado na Figura 2. 6:

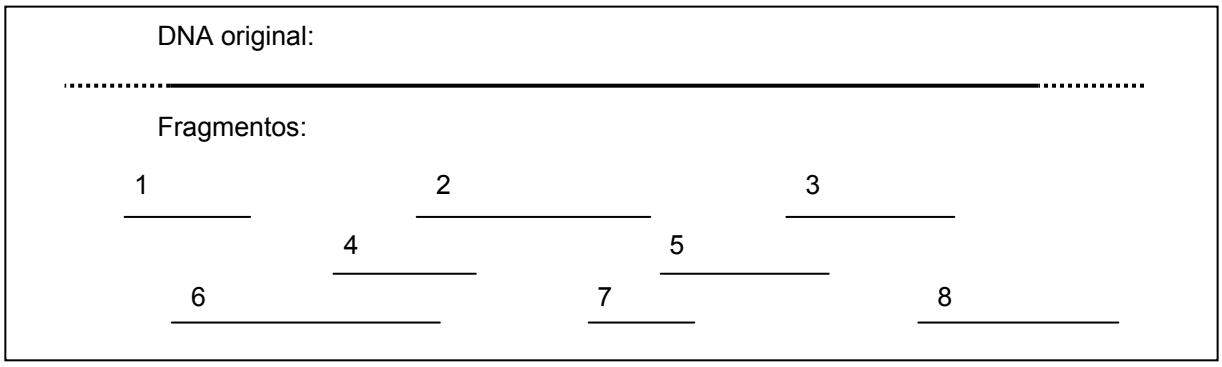


Figura 2. 6 Montagem de Fragmentos de DNA.

Os fragmentos devem existir em quantidade e tamanho suficientes para cobrir toda a seqüência original – de uma extremidade à outra – e possuir sobreposições (entre os fragmentos) de comprimento suficientes para evitar que coincidências aleatórias provoquem uma montagem equivocada. A montagem será feita buscando-se casar extremidades de fragmentos de modo a formar estruturas chamadas **contigs**. Cada novo fragmento é comparado aos contigs existentes em busca de semelhanças, na tentativa de anexá-lo a algum deles, o que irá modificar a família de contigs. Se não for detectada semelhança entre o fragmento e qualquer contig, um novo contig é criado. Com a adição de um novo fragmento, um contig existente pode ser estendido e mesmo dois contigs podem ser fundidos em um.

Um alinhamento como o descrito acima, em que os *gaps* de extremidade não são penalizados é chamado **alinhamento semiglobal**. São possíveis dois casos de alinhamento semiglobal entre duas seqüências *s* e *t*, dadas:

CASO A: são ignoradas:

Remoções no início de *t* e

Inserções no final de *t*, ou

CASO B: são ignoradas:

Remoções no início de *s* e

Inserções no final de *s*;

Para realizar um alinhamento desconsiderando *gaps* nas extremidades das seqüências dadas, existem 4 possibilidades:

(1) Para desconsiderar *gaps* no final de *s*, desconsidera-se o correspondente sufixo de *t*, isto é, procede-se ao alinhamento buscando encontrar a melhor pontuação entre *s* e um prefixo de *t*; esta melhor pontuação será encontrada entre os valores da última linha da matriz. O alinhamento ótimo será obtido seguindo as setas, partindo desta célula até a célula (0, 0).

(2) Para desconsiderar *gaps* no final de t, o procedimento é análogo ao anterior, só que a maior pontuação será encontrada na última coluna da matriz.

Combinar os dois procedimentos acima corresponde a não cobrar por *gaps* no final de qualquer uma das seqüências e, considerando que nunca ocorrem *gaps* na mesma coluna em ambas seqüências simultaneamente, é suficiente escolher o maior valor dentre todas as células da última linha e da última coluna.

(3) Para desconsiderar gaps no início de s, o algoritmo será o mesmo com uma modificação apenas na inicialização da matriz, que terá todos os valores da primeira linha iguais a 0.

(4) Para desconsiderar gaps no início de t, o algoritmo será o mesmo, mas agora os valores da primeira coluna é que serão iguais a 0.

É possível combinar estes quatro procedimentos, pois nenhum interfere no outro.

Sejam, por exemplo, os fragmentos (seqüências) 1 e 6 da Figura 2. 6, respectivamente: ATTCGGAGTCATGTAC e CATGACTATGACCCTG. O objetivo será alinhar estas seqüências, sem penalizar, contudo, as remoções no início de 6 nem as inserções no seu final de (CASO A), como mostra a Figura 2. 7.

A T T C G G A G T **C A T G T A C** – – – – – – – – – –
– – – – – – – – – **C A T G** – **A C** T A T G A C C C T G

Figura 2. 7 Alinhamento Semiglobal.

Serão alinhados, portanto, um sufixo de 1 com um prefixo de 6. A matriz de comparação deverá ser inicializada com todas as células da primeira coluna iguais a zero e ao final deverá ser procurado o maior valor na última linha da matriz.

Assim, para alinhar um prefixo de *s* com um sufixo de *t* (CASO A), deve-se inicializar a primeira linha com zeros e procurar pelo maior valor na última coluna da matriz. Para alinhar um prefixo de *t* com um sufixo de *s* (CASO B), deve-se inicializar a primeira coluna com zeros e procurar pelo maior valor na última linhada matriz.

Alinhamentos locais

Embora duas seqüências não sejam muito similares, podem conter regiões que apresentem alta similaridade, de modo que seja de interesse descobrir tais regiões. Este é o caso quando se deseja identificar proteínas de uma mesma família, cujas seqüências completas podem não apresentar grande similaridade, porém possuir regiões altamente conservadas, as quais podem ser identificadas através de um alinhamento local. Estas regiões podem representar sítios ativos, motivos, estruturas com funções equivalentes, etc.

```
S N H R D V V V D L Q G W V T G N G K G L I Y L T D P Q I H S V D . . . . . Q K V F T T N F G K R G I F Y
S R G R F L V C D L Q G V G . . . . . K T M T D P A I H T L D P . . . Y R F S L S Q T N L G A E G F M
S N K Q M I V V D I Q G V D . . . . . D L Y T D P Q I H T P D . . . . G K G F G L G N L G K A G I N K
S N H E L L I V D I Q G V N . . . . . D F Y T D P Q I H T K S . . . . G E G F G E G N L G E T G F H K
S N H Q L L I I D I Q G V G . . . . . D H Y T D P Q I H T Y D . . . . G V G F G I G N L G Q K G F E K
S G H Q L I V V D I Q G V G . . . . . D L Y T D P Q I H T E K . . . . G T D F G D G N L G V R G M A L
T R G E L L V L D L Q G V G . . . . . E N L T D P Q I H T E V K Q S R G M V F G P A N L G E D A I R N
```

Figura 2. 8 Domínios em seqüências de proteínas.

Um alinhamento local entre duas seqüências *s* e *t* é um alinhamento de um fator de *s* com um fator de *t*. Nesse caso, outros alinhamentos, além do ótimo, também podem ser de interesse, desde que possuam pontuação próxima da ótima. Um algoritmo para determinar esses alinhamentos é também uma extensão do básico, com as seguintes ressalvas:

(1) cada célula a(*i,* j) da matriz A[(*m*+1), (*n*+1)] guardará a pontuação do alinhamento ótimo entre um sufixo de *s* e um sufixo de *t*.

(2) a primeira linha e primeira coluna são inicializadas com zeros.

(3) como haverá sempre, no mínimo, o alinhamento dos sufixos vazios de *s*[1..*i*] e *t*[1..*j*], que dá uma pontuação zero, nunca haverá uma célula com valor negativo. Por isso, o cálculo das células será feito por:

$$a(i, j) = \text{máx} \begin{cases} 0 \\ a(i, j-1) + g \\ a(i-1, j-1) + p \\ a(i-1, j) + g \end{cases}$$

(4) o valor do alinhamento ótimo será dado pelo maior valor da matriz inteira.

(5) para se obter o alinhamento ótimo, o procedimento será: seguir as setas, a partir do maior valor da matriz, até encontrar uma célula cujo valor seja zero.

(6) para encontrar outros bons alinhamentos locais, pode-se escolher outras células com valores apropriados (próximos do máximo) e proceder como no item anterior.

2.1.2 Escolha de parâmetros

Para levar em conta os três tipos de eventos mutacionais que são considerados em alinhamentos, existem diferentes possibilidades com relação aos pesos que serão aplicados a cada tipo de operação – substituição, inserção ou remoção. Estes pesos influenciam fortemente os alinhamentos encontrados e para sua determinação leva-se em conta a probabilidade de ocorrência de cada evento mutacional pois, idealmente, esses valores deveriam refletir o fenômeno biológico que o alinhamento tenta modelar.

Em seqüências de DNA, um esquema que tem sido utilizado para pontuar alinhamentos é:

$2g < m < M$, onde $M \geq 0$ para o casamento de bases iguais, *m* para casamento de bases desiguais e $g \leq 0$, para *gaps*.

Substituições

No caso de seqüências de DNA, as probabilidades de substituição variam de acordo com as bases. Por exemplo, substituições entre purinas ou entre pirimidinas (transições) são mais freqüentes do que entre purinas e pirimidinas e vice-e-versa (transversões). Por isso é comum impor pesos maiores para transversões do que para transições.

No caso de seqüências de proteínas, a probabilidade de substituição de um aminoácido por outro depende basicamente do efeito fenotípico da substituição e da estrutura do código genético. As substituições envolvendo aminoácidos que possuam características bioquímicas semelhantes não afetam muito a estrutura da proteína e conseqüentemente ocorrem com mais freqüência durante a evolução. Observa-se que para distâncias evolucionárias pequenas, as probabilidades de substituição refletem, principalmente, na estrutura do código genético. Para distâncias evolucionárias maiores, as probabilidades dependem, essencialmente, de similaridades bioquímicas entre aminoácidos.

A probabilidade de substituição também varia de acordo com a localização do aminoácido na proteína dobrada. Por isso, algumas matrizes de substituição com parâmetros específicos para cada ambiente (α-hélices ou β-folhas) têm sido desenvolvidas.

Vários métodos têm sido propostos para construir matrizes que reflitam as probabilidades de substituição. Algumas matrizes simples como a matriz unitária:

$$M_{ij} = \begin{cases} 1 & se\ i = j \\ 0 & caso\ contrário \end{cases},$$

ou como a matriz código genético, onde M_{ij} é igual ao número mínimo de substituições de bases necessárias para converter um *codon* de aminoácidos *i* em um *codon j*, não empregam fenômenos biológicos. No caso de seqüências de proteínas, matrizes como as das famílias PAM e BLOSUM, derivam da observação empírica de seqüências ancestrais e seus descendentes atuais.

Matrizes de Substituição PAM

A metodologia das unidades PAM (*Point Accepted Mutations*), assim como a primeira matriz da família, foram desenvolvidas por Margaret Dayhoff e colaboradores [7, 8]. Eles observaram 1572 mutações aceitas em 71 superfamílias de seqüências de proteínas proximamente relacionadas e concluíram que as substituições não ocorreram aleatoriamente. Algumas substituições de aminoácidos aceitáveis ocorreram mais prontamente que outras, provavelmente porque elas não têm grande efeito sobre a função da proteína. Isto significa que seqüências de uma mesma família não devem, necessariamente, ter os mesmos aminoácidos, mas aminoácidos 'comparáveis'.

As **unidades PAM** são utilizadas para medir a distância evolucionária entre duas seqüências de aminoácidos. Duas seqüências s_1 e s_2 divergem de uma unidade PAM se uma série de substituições aceitas, isto é, que foram incorporadas e repassadas para os descendentes, transforma s_1 em s_2 com uma média de um evento de mutação pontual por 100 aminoácidos. Isto não quer dizer que a diferença entre os aminoácidos das seqüências é de 1% porque uma posição pode sofrer mais de uma mutação.

Cada matriz é desenhada para comparar duas seqüências cuja distância evolucionária, refletida no número de unidades PAM, é indicada no próprio nome da matriz (Ex.: PAM_{80}, PAM_{120}, PAM_{250}). Quanto maior o índice, maior a distância evolucionária. Uma célula (a_i, a_j) de uma matriz PAM_k representa a probabilidade (log da probabilidade) de que um aminoácido *i* seja substituído pelo aminoácido *j* em duas seqüências que são *k* unidades PAM divergentes.

Os maiores problemas apontados por Shamir [9], com relação à noção de unidades PAM são: primeiro, muitas vezes não se conhece seqüências ancestrais da que se quer estudar; assim, certas suposições devem ser assumidas como verdadeiras, acarretando em erros no alinhamento. Segundo, apenas substituições são consideradas, inserções e remoções são ignoradas, de modo que não se pode estabelecer a relação exata entre as seqüências.

Matrizes de Substituição BLOSUM

Um outro conjunto de tabelas de substituição foi desenvolvido baseado em alinhamentos de pequenas subcadeias – **blocos conservados** – que casam em algum nível definido de

similaridade [9]. Como as matrizes BLOSUM incorporam muito mais dados, são, presumivelmente, mais precisas. Quanto à nomenclatura utilizada, para as matrizes BLOSUM, quanto maior seu número, menor a distância evolucionária.

Algumas comparações entre a performance das matrizes BLOSUM e PAM são mostradas por Shamir [9], bem como a tentativa de se estabelecer uma correspondência entre uma determinada PAM_k com uma $BLOSUM_l$, mas como se trata de matrizes de naturezas diferentes estas comparações não são necessariamente corretas. Verifica-se entretanto que as matrizes $BLOSUM_{62}$ são altamente recomendadas para alinhamento de seqüências e buscas em bancos de dados.

Funções de Penalização por *Gaps*

Quando não se faz distinção entre abertura ou extensão de *gaps*, a penalização de *gaps* pode ser expressa por uma função linear como:

$w(k) = bk$, onde $w(k)$ é a penalidade cobrada por uma série de k *gaps* consecutivos e

$b = -g$ é o valor absoluto da penalidade associada a um *gap*.

Entretanto, considerando que a criação de um *gap* pontual, bem como um *gap* com extensão maior do que 1, seja devida a um evento mutacional, ao passo que k *gaps* pontuais são criados a partir de k eventos mutacionais, a criação de k *gaps* consecutivos (ou um *gap* com extensão k) é mais provável do que a criação de k *gaps* isolados. Por esse motivo, existe uma tendência de penalizar mais a abertura de um *gap* do que a extensão de um *gap* já aberto.

Para levar em conta esta diferença de probabilidades, muitas funções diferentes são utilizadas para expressar o peso de *gaps*. Pode-se, por exemplo, utilizar uma função sub-aditiva:

$w(k) \leq k\, w(1)$, isto é, $w(k_1 + k_2 + ... + k_n) \leq w(k_1) + w(k_2) + ... + w(k_n)$

Uma função comumente utilizada é a função linear:

$w = a + bk$, onde k é o comprimento do *gap*, a é a penalidade para abertura do *gap* e b é a penalidade para extensão do *gap*.

Entretanto, têm-se mostrado que essa função subestima as probabilidades para longos *gaps*. Um modelo mais realístico considera a função:

$w = a + b \log(k)$.

Needleman e Wunsch [10] propuseram um algoritmo para calcular o valor do alinhamento ótimo que suporta qualquer função w de penalização de *gaps* que é, em sua essência, semelhante

ao algoritmo básico visto anteriormente. A complexidade do algoritmo proposto, entretanto, é cúbica. Algoritmos quadráticos no entanto, foram desenvolvidos, para alguns tipos específicos de função de penalização. Alguns destes são mostrados por Setubal e Meidanis [5].

2.1.3 Distância de Edição

Alinhamentos de seqüências podem ser avaliados sob o ponto de vista de **distância de edição**, ao invés de similaridade. A noção de distância entre seqüências de caracteres sobre um alfabeto Σ foi introduzida na década de 1950. Um dos métodos mais utilizados define **distância** entre duas seqüências *s* e *t* como sendo o menor número possível de certas operações básicas necessárias para transformar *s* em *t*.

As operações básicas normalmente utilizadas são aquelas já vistas:

- substituição de um caracter por outro numa posição qualquer da seqüência;
- remoção de um caracter numa posição qualquer da seqüência;
- inserção de um caracter numa posição qualquer da seqüência;

Por exemplo, para transformar a seqüência AACGAK em ATCGGA são necessárias, pelo menos, as seguintes operações:

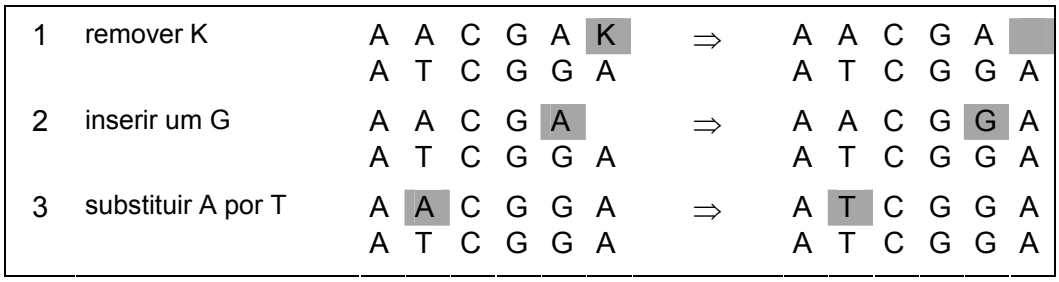


Figura 2. 9 A seqüência AACGAK é transformada na seqüência ATCGGA. A distância entre elas é 3.

Para transformar AACGAK em ATCGGA são necessárias, no mínimo, 3 operações, então diz-se que a **distância de edição**, ou simplesmente **distância** entre as seqüências dadas é 3.

Uma vez que cada operação corresponde a uma mutação ocorrida durante a evolução, grosseiramente, estas edições servem para medir a distância evolucionária entre duas seqüências.

Assim como podem ser atribuídos pesos diferenciados para *gaps*, igualdades e desigualdades no cálculo da similaridade entre seqüências, o mesmo pode ser feito no cálculo da distância de edição. Setubal e Meidanis [5] mostram que os mesmos algoritmos utilizados para cálculo de similaridade podem ser utilizados para cálculo de distâncias. Na análise de seqüências,

buscar por seqüências similares, tanto no sentido global quanto local, pode traduzir-se em buscar por seqüências de pequena distância de edição.

2.1.4 Alinhamento Múltiplos

Um alinhamento múltiplo é um alinhamento de mais de duas seqüências e, assim como nos alinhamentos de duas seqüências, *gaps* são inseridos nas seqüências de modo a que elas fiquem com o mesmo tamanho. Em seguida, as seqüências são alinhadas formando o casamento dos caracteres que ocupam a mesma coluna (bases ou *gaps*). Não são permitidas colunas compostas exclusivamente por *gaps*.

O **consenso** de um alinhamento é uma seqüência composta dos caracteres mais comuns em cada coluna do alinhamento. Existem diversas formas de se medir a divergência de um conjunto de seqüências alinhadas (a distância entre as seqüências). Uma delas – distância do consenso, é considerar a distância total entre as seqüências como sendo a soma do número de caracteres das seqüências em cada coluna que diferem dos caracteres do consenso. Outra – soma de pares, é a soma das distâncias entre todos os pares de seqüências, isto é, a pontuação de cada coluna será dada por:

$\sum_{i<j} p(a_i, a_j)$, onde p é uma função de pontuação como a vista anteriormente, para alinhamentos de duas seqüências.

Uma outra alternativa, usada por J. Kececioglu [11], é através de uma função que toma como argumentos todos os elementos de uma coluna e retorna sua pontuação.

Métodos Heurísticos para Alinhamentos Múltiplos

Os métodos para calcular similaridade (ou distância) e alinhamento ótimo, no sentido matemático, como já foi visto, possuem complexidade quadrática, o que inviabiliza, na maioria das vezes, sua utilização, especialmente quando há muitas seqüências de grande comprimento envolvidas. Nesses casos, métodos heurísticos têm sido utilizados que, se por um lado não garantem o melhor resultado, por outro fornecem resultados razoáveis e ainda são factíveis.

Um método heurístico que tem sido utilizado para alinhar múltiplas seqüências – **heurística star** – consiste em alinhar, inicialmente, duas das seqüências mais próximas e sucessivamente ir escolhendo dentre as outras seqüências, a que mais se aproxima de uma das já escolhidas e agregando-a ao alinhamento. Este método requer, para k seqüências, $O(k^2)$ comparações de pares de seqüências e $O(k)$ inclusões de uma nova seqüência no alinhamento.

Outro método heurístico, semelhante a este, é o **alinhamento global progressivo**, descrito por Duret e Abdeddaim [4] em três passos (ver Figura 2. 10):

(1) Computar os escores dos alinhamentos (ou distâncias) entre todos os pares de seqüências;

(2) Construir uma árvore guia que reflita as similaridades entre seqüências, usando as distâncias dos alinhamentos de pares;

(3) Alinhar as seqüências seguindo a árvore guia. Correspondendo a cada nó na árvore, o algoritmo alinha as duas seqüências ou alinhamentos que estão associados com seus dois nós vizinhos. O processo é repetido a partir da árvore dada (as seqüências) e terminando na raiz.

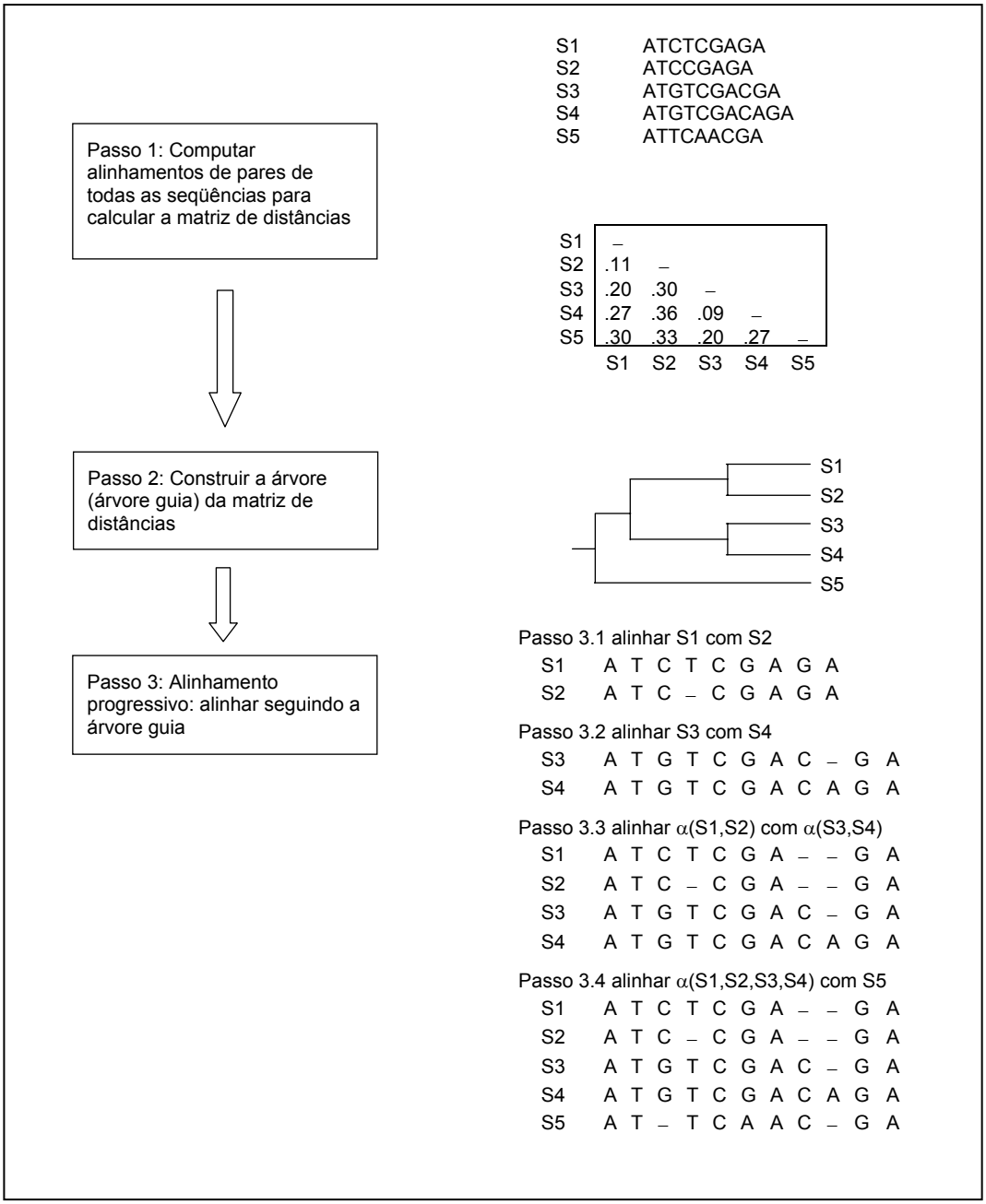


Figura 2. 10 Processo de Alinhamento Progressivo.

Pelo menos dois problemas podem ocorrer com essa abordagem. Um deles é que, como o método exige que uma árvore guia seja construída a partir de distâncias entre pares de seqüências, e pode haver seqüências homólogas que incluam muitos segmentos não sobrepostos, a árvore construída será falsa, pois a distância calculada entre tais segmentos será grande. Outro problema é o conhecido como 'uma vez um *gap*, sempre um *gap*', isto é, *gaps* que são introduzidos em alinhamentos anteriores não podem ser modificados em passos posteriores.

Alinhamento global baseado em blocos

Algumas vezes as seqüências a serem comparadas compartilham **blocos** separados por regiões não conservadas. Quando isto ocorre, uma abordagem mais adequada para análise das seqüências consiste na busca por esses blocos conservados. Blocos são alinhamentos de fragmentos de seqüências que podem ser **exatos** (compostos de segmentos idênticos) ou **não exatos** e podem ser **uniformes** (existentes em todas as seqüências) ou **não uniformes**. Na prática, blocos não uniformes são mais comuns. O conjunto de blocos selecionados pode ser **consistente**, isto é, os blocos ocorrem juntos em um alinhamento global múltiplo. Depois que os blocos já estão alinhados, uma abordagem clássica pode ser utilizada para alinhar regiões entre blocos (ver Figura 2. 11).

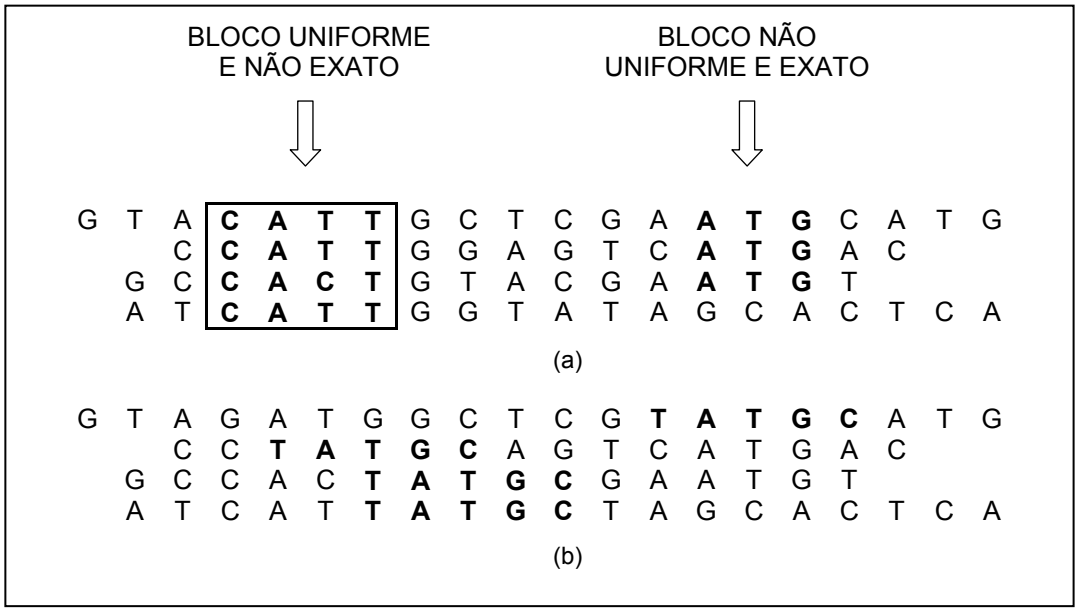


Figura 2. 11 Blocos (a) blocos consistentes; (b) blocos não consistentes

Vários programas têm sido progressivamente propostos para alinhamento de blocos múltiplos. O primeiro utilizava um algoritmo de ordenação, depois surgiram outros baseados em árvores de sufixo, utilizados para computar blocos exatos. Posteriormente surgiu o programa ASSEMBLE que analisa as árvores de todos os pares de bases para encontrar blocos

uniformes não necessariamente exatos. Um maior avanço traduz-se no programa DIALIGN, que permite blocos que não sejam necessariamente uniformes.

Algoritmos quadráticos foram propostos para encontrar blocos consistentes, reduzindo o problema a uma busca de caminho ótimo em um grafo, e ainda algoritmos sub-quadráticos têm sido propostos para este mesmo caso, no entanto, o problema de encontrar blocos não consistentes é intratável.

Alinhamentos baseados em Motivos

Algumas seqüências possuem módulos homólogos que podem ocorrer em diferentes posições relativas nas seqüências e ainda ser repetidas numa mesma seqüência. Para mais de duas seqüências este problema é difícil e é necessária a utilização de heurísticas.

Uma abordagem consiste em, para cada palavra w, de comprimento k, buscar na vizinhança de w, dado um certo ponto de corte, palavras que confiram um bom escore com w. O escore de w será a soma de todos estes escores. O resultado da computação será dado pelas palavras de comprimento k de maior escore. O maior problema com esta abordagem é o requerimento de espaço (proporcional a 20^k, para proteínas).

Outra abordagem combina comparações de pares para computar alinhamentos locais múltiplos. Programas de alinhamento local mais recentes são baseados em métodos estatísticos que usam heurísticas para resolver problemas de otimização. Outros métodos heurísticos para alinhamento múltiplo podem ser encontrados em Shamir.

Perfis

Dado um alinhamento S de comprimento l, um **perfil** é uma matriz $l \times |\Sigma \cup \{-\}|$, cujas colunas são vetores de probabilidades denotando a freqüência de cada símbolo na coluna correspondente do alinhamento. A partir de seqüências de uma mesma família, pode ser construído um perfil da família e a partir desse perfil, é possível avaliar se uma dada seqüência pertence ou não àquela família. Esta avaliação é mais sensível quando é feita por um perfil do que se for feita a partir da comparação com as seqüências separadamente. Ao invés de olhar para as seqüências, o foco volta-se para toda a família de seqüências.

Sejam a seqüência S_1 e o perfil P, ambos de comprimento l . O alinhamento entre S_1 e P pode ser avaliado por $\sum_{j=1}^{l} w(s_j, p_j)$, onde $w(s_j, p_j)$ é a probabilidade do caracter s_j pertencer à coluna j. Deve ser previsto também um valor para $w(s_j, -)$. Obviamente esta avaliação pode ser feita por programação dinâmica.

Apenas a título de ilustração, a Figura 2.12 mostra um exemplo de perfil construído a partir de trechos de 5 seqüências da família das globinas:

Trechos de 5 seqüências de globinas:

Posição:		1	2	3	4	5	
HBB_ALLMI	...	W	K	R	R	Y	...
HBB1_XENBO	...	W	T	Q	R	Y	...
HBB0_MOUSE	...	W	T	Q	R	F	...
HBB_LEPPA	...	W	T	K	R	Y	...
HBB1_CYGMA	...	W	T	Q	R	H	...

Perfil

Aminoácido:	1	2	3	4	5
F	–	–	–	–	0,2
H	–	–	–	–	0,2
K	–	0,2	0,2	–	–
Q	–	–	0,6	–	–
R	–	–	0,2	1,0	–
T	–	0,8	–	–	–
V	–	–	–	–	–
W	1,0	–	–	–	–
Y	–	–	–	–	0,6

Figura 2.12 Um modelo estatístico – perfil – ilustrativo para a família das globinas, construído a partir das seqüências dadas

Modelos Escondidos de Markov

Na prática, um perfil estatístico deve levar em consideração alguns fatores como, por exemplo, o fato de que as seqüências, geralmente, apresentam comprimentos diferentes, o que requer a atribuição de pontos para inserções e remoções. Além disso, pesos diferenciados devem ser dados às bases, a depender da sua posição nas seqüências. Estes refinamentos, embora sejam necessários para a criação de bons perfis, introduzem muitos parâmetros livres que só podem ser calculados através de tentativa-erro.

Um Modelo Escondido de Markov– HMM (do inglês: *Hidden Markov Model*) para uma família de seqüências é um modelo estatístico do consenso da estrutura primária desta família. É um tipo de perfil dinâmico, que possui uma topologia mais elaborada do que o perfil abordado anteriormente. Um HMM pode ser visto como uma máquina de estados finita cujos estados podem emitir os símbolos do alfabeto em questão (ácidos nucléicos ou aminoácidos). Os parâmetros de um HMM são estimados a partir de um conjunto de seqüências de treinamento, através de uma abordagem sistemática, e envolvem: o número de estados, probabilidades de emissão dos símbolos em cada estado (ou de não emissão de qualquer símbolo), probabilidades de transição entre estados.

HMMs foram introduzidos entre os anos 60 e 70 e na década de 80 foram aplicados em sistemas de reconhecimento de fala [12]. Atualmente, o uso de HMMs tem-se intensificado na análise comparativa de seqüências biológicas e será o foco deste trabalho, pelo que será abordado com mais detalhes posteriormente.

2.2 Identificação de Genes

Com o crescimento do número de projetos genômicos vem crescendo também o interesse por métodos automatizados de identificação de genes em regiões recentemente seqüenciadas. É de grande importância para a biologia molecular, dada uma seqüência de DNA, localizar os genes, bem como identificar seus elementos funcionais. Na realidade, diversas abordagens têm sido desenvolvidas que propõem a identificação de genes a partir da identificação de suas unidades funcionais.

Genes em Procariotos

Em organismos procariotos, a maior parte de uma seqüência de DNA é codificante de proteína; cada gene é uma seqüência contínua de bases, sem trechos de não-gênicos. A Figura 2. 13 mostra a estrutura de um gene típico de um organismo procarioto. Algumas bases após a região promotora segue-se o sinal de início da transcrição (do inglês: *transcription start site*). Antes e depois do gene existem as regiões que não serão traduzidas em proteína – 5' UTR (*unstranslated region*) e 3' UTR.

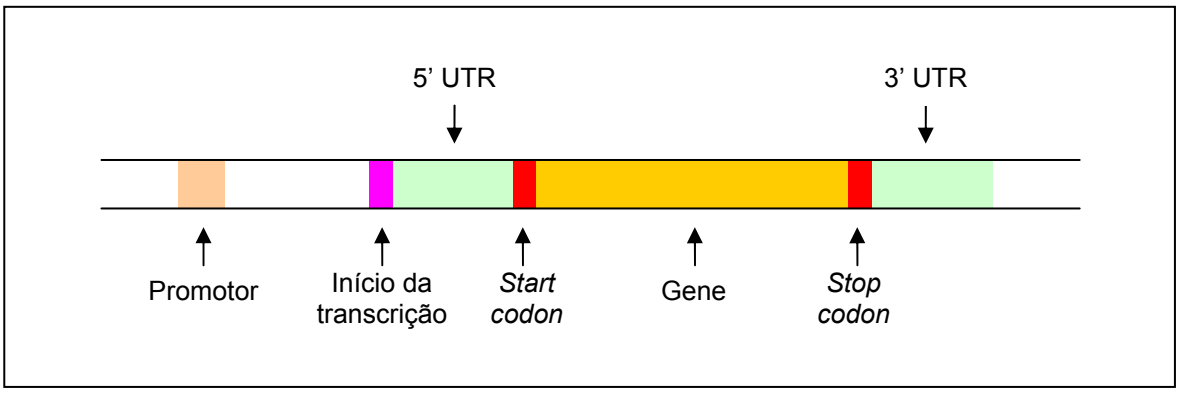


Figura 2. 13 Estrutura de um gene típico de procariotos no nível do DNA.

Na etapa de tradução, cada *codon*, será traduzido em um aminoácido. O ribossomo percorre o mRNA no sentido *downstream* e quando encontra o *start codon* (AUG) começa a gerar uma seqüência de aminoácidos de acordo com a seqüência do mRNA, segundo o código

genético universal (Figura 1. 4). O processo pára quando um *stop codon* (UAA, UAG ou UGA) é encontrado. O modo como a seqüência de mRNA é lida para codificar a proteína varia a depender de qual seja o primeiro nucleotídeo lido, o que conseqüentemente, determinará qual será o primeiro codon lido. A seqüência ACGAGUCGCCAAACUAUUCGU, por exemplo, pode ser lida de três formas diferentes (Figura 2.14). Cada uma das três formas possíveis representa um **quadro de leitura**. Dessa forma, a localização do início do gene não é exatamente determinada.

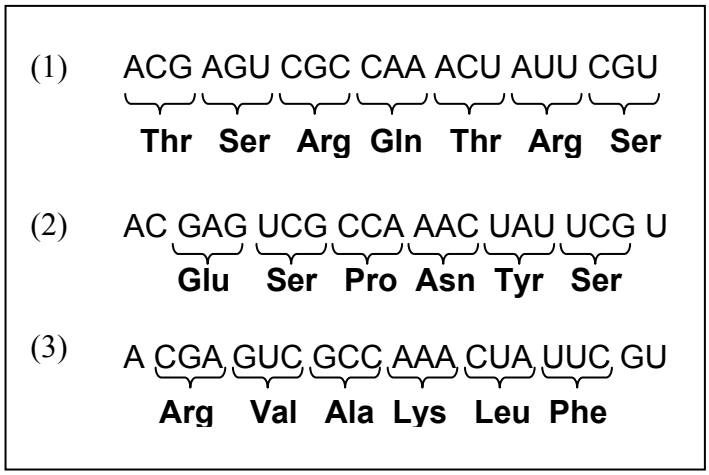


Figura 2.14 3 quadros de leitura possíveis numa seqüência de DNA

ORFs e NORFs

Um quadro de leitura aberto – **ORF** (*open reading frame*) é uma seqüência de *codons* sem *stop codon*. Sabe-se que cada região codificante tem apenas um *stop codon*. Uma **NORF** – (*Non coding open reading frame*) é uma pequena ORF (geralmente com menos de 100 codons). Uma maneira de identificar regiões codificantes é olhar para grandes extensões de ORFs, ou ainda, buscar na seqüência de DNA, considerando os três quadros possíveis de leitura, um *stop codon* e voltar na seqüência até encontrar um *start codon*. Esta abordagem, entretanto, é falha para detectar genes muito pequenos, além do que existem muito mais ORFs do que genes.

Outras abordagens são utilizadas para identificar regiões codificantes em procariotos que envolvem a freqüência dos codons (ou dos aminoácidos nos quais estes são traduzidos) numa seqüência de DNA. Por exemplo, sabe-se que os aminoácidos Leucina, Alanina e Triptofan ocorrem nas proteínas a uma taxa de 6,9:6,5:1,0. Sabe-se também que os nucleotídeos A ou T ocorrem com uma freqüência maior do que 90% na terceira posição do codon (a depender da espécie).

Em métodos que envolvem estas freqüências, os codons podem ser considerados independentemente, entre si, como também pode haver uma dependência entre os codons e seus antecessores.

Cadeias de Markov

Para discriminar regiões codificantes de não codificantes, pode-se utilizar um modelo de cadeia de Markov onde cada nucleotídeo é um estado e são computadas as probabilidades das seqüências dadas estarem em trechos codificantes ou não, a partir das probabilidades de cada nucleotídeo seguir outro nucleotídeo dentro e fora de uma região codificante. Outra abordagem utilizada, assumindo-se que se conhece todas as ORFs de uma seqüência, é traduzir estas ORFs em seqüências de codons e utilizar uma cadeia de Markov com 64 estados. As probabilidades de transição de estados serão as probabilidades de cada codon seguir qualquer outro codon em uma região codificante. Ao final, pode-se computar a probabilidade de que uma dada ORF seja codificante. Em ambos os casos, as cadeias podem ser de primeira ordem ou superior.

Regiões Promotoras

As regiões promotoras em seqüências de DNA funcionam como pontos de âncora para a RNA-polimerase, indicando não só o local de início da transcrição como também a taxa de transcrição. Estas regiões, ainda que curtas, ocorrem com alto grau de conservação na maioria dos organismos. Como exemplo, Shamir [9] apresenta um consenso da região próxima ao local de início da transcrição da *E. Coli*:

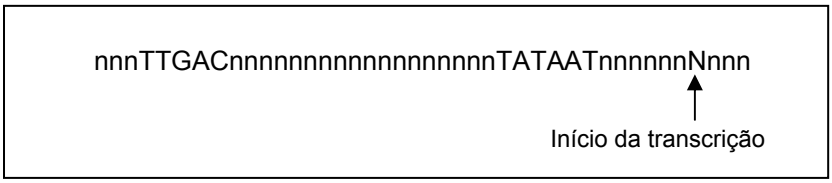


Figura 2. 15 Consenso da região de início da transcrição da *E. Coli*.

TTGAC ocorre 35 bases antes do ponto de início da transcrição – N, e TATAAT (também conhecido como *TATA-box* ou *Pribnow box*), 12 bases antes de N. Alguns métodos têm sido utilizados para identificação de regiões promotoras, baseados na freqüência das bases em cada posição de regiões promotoras conhecidas. Uma matriz posicional simples de pesos pode ser construída com os valores de $f_{b,i}$, onde $f_{b,i}$ é a freqüência da base b na posição i de sufixos de regiões promotoras conhecidas, assumindo-se que as posições são independentes. Com esta matriz, pode-se calcular a probabilidade de uma seqüência dada ser uma região promotora.

Genes em Eucariotos

A estrutura e o mecanismo de expressão de genes em eucariotos é mais complexa do que em procariotos. A Figura 2. 16 ilustra a estrutura de um gene típico de organismos eucariotos. Além do fato de que a seqüência de DNA, na região do gene, possui *exons* e *introns* alternados, logo após a região 3' UTR apresenta uma seqüência de várias Adeninas, chamada de poly-A. O limite entre cada *exon-intron* é chamado *splice-site* e é sinalizado por pequenas seqüências específicas de 2bp (pares de bases). A extremidade 5' de um *intron*, que corresponde à extremidade 3' de um *exon*, é chamada *donnor-site* e a extremidade 3' de um *intron*, ou 5' de um *exon*, é chamada *acceptor-site*. Todos estes sinais, *donnor-site*, *acceptor-site*, poly-A, e mais, promotor, 5' UTR, 3' UTR são fundamentais para a identificação de genes em eucariotos.

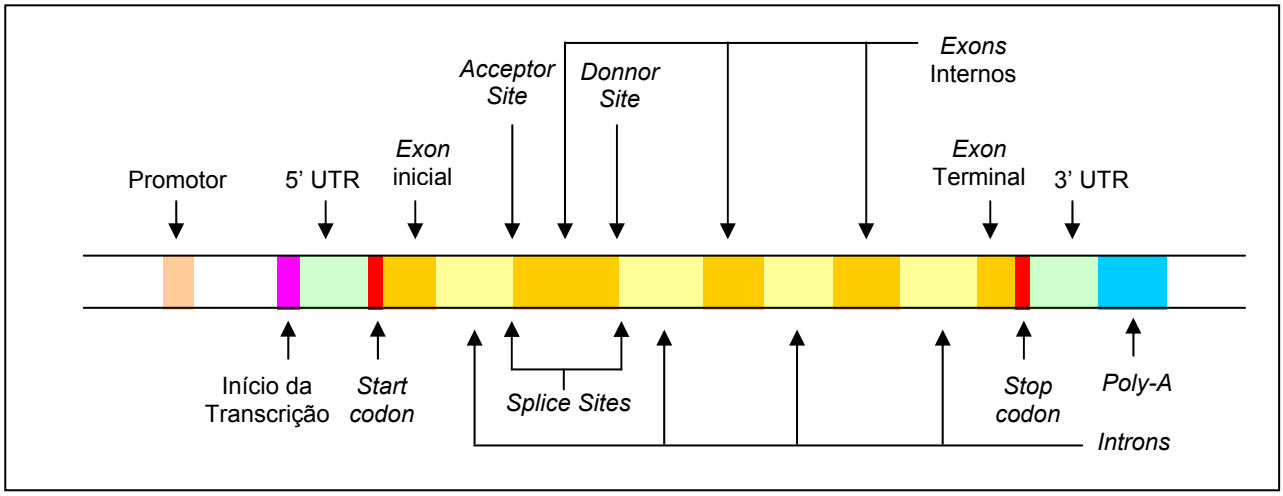


Figura 2. 16 Estrutura de um gene típico de eucariotos no nível do DNA.

Splice-sites

A remoção dos *introns* e conseqüente união dos *exons* de uma cadeia, chamada *splicing* (do inglês – junção), é executada por enzimas que reconhecem as regiões de junção *exon-intron*, as quais, em geral, apresentam alto grau de conservação [9]. Dentro dos *introns* freqüentemente aparece um ponto de âncora, chamado *branch point*. Entre o *branch point* e o *acceptor site* aparece também com freqüência uma região rica em pirimidinas, o que auxilia a identificação de genes em eucariotos. Uma vez que muitos genes têm *alternative splicing*, em algumas variações de um mesmo gene alguns *exons* não são utilizados (mais de 50% dos genes).

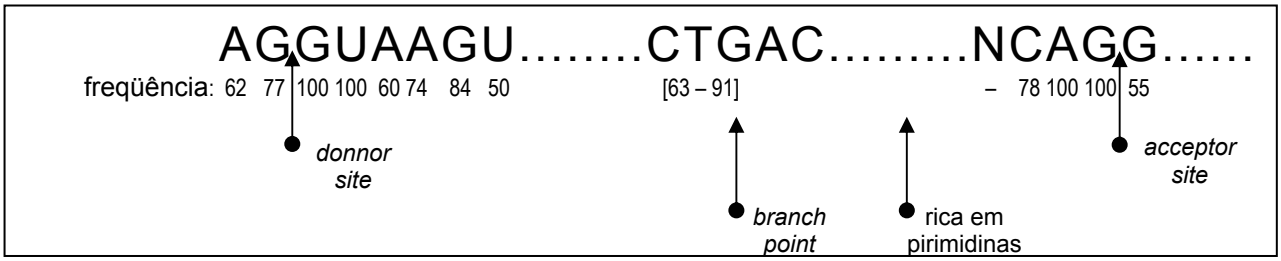


Figura 2. 17 Junções *exon-intron* em um gene eucarioto típico. A similaridade com o consenso em regiões de junção chega a atingir 100%.

Modelos de Markov Escondidos

Existem diversas abordagens utilizadas para identificação de genes, baseadas em cadeias de Markov. Uma bastante comum, por exemplo, é baseada no modelo de Markov de 5ª ordem, que considera uma janela de 6 bases consecutivas na seqüência de DNA. Duas tabelas de probabilidades são utilizadas, uma para regiões codificantes e outra para regiões não codificantes. Para cada 6-tupla as tabelas dão a probabilidade de ocorrer a sexta base, dadas as 5 bases anteriores. Assim, dada uma seqüência, é calculada a sua probabilidade, para cada uma das tabelas, isto é, a probabilidade da seqüência dada ser uma região codificante ou não. Este modelo não leva em consideração informações a respeito do quadro de leitura, e por isso é chamado **homogêneo**. Um modelo que considera os três possíveis quadros de leitura é chamado **não homogêneo**.

Capítulo 3

Aprendizagem de Máquina

A quantidade de seqüências biológicas disponíveis em bancos de dados públicos tem crescido exponencialmente. A base de dados do SWISSPROT, versão 40.28, por exemplo, conta com mais de 160.000 seqüências de proteínas. Embora exista uma massa de dados tão grande disponível, o conhecimento teórico a respeito desses dados, ainda é insuficiente para permitir a obtenção de informações importantes a respeito deles, em um nível satisfatório.

A despeito desta abundância de dados disponíveis nas bases públicas, existe um certo grau de incerteza que envolve estes dados devido a diversos fatores. Uma vez que são obtidos experimentalmente, anotados e submetidos por vários grupos distintos de pesquisadores, sempre lhes são incorporados erros. A margem de erros cresce ainda mais, considerando que essas bases de dados, normalmente, também são mantidas por grupos diversos. Além disso, a própria natureza dos dados sugere a existência de ruídos, pois existem diferenças de seqüências entre organismos, entre indivíduos, entre idades evolucionárias diferentes, etc.

Um problema que surge aí, portanto, é o de extrair informações úteis, confiáveis, desses dados, sem ter o conhecimento teórico formal subjacente a eles, mesmo considerando a presença de incerteza. Esta tarefa pode ser bem desempenhada através de **aprendizagem de máquina**.

A aprendizagem de máquina é um campo de estudo relativamente novo, que ganhou impulso na década de 70, com o paradigma – aprendizagem de conhecimento intensivo. Baseados neste paradigma, surgiram vários métodos de aprendizagem fundamentados em conhecimento rico. Estes métodos pretendem extrair informações essenciais de exemplos individuais, descartando suas características indesejáveis e podem trazer à tona relações e correlações

escondidas em uma grande massa de dados. Isso é possível com a construção de um modelo probabilístico para o conjunto de dados disponível.

O estudo da aprendizagem de máquina está presente em várias áreas como estatística, modelagem cerebral, teoria de controle adaptativo, modelagem psicológica, inteligência artificial e modelagem evolucionária. Neste trabalho, foram construídos modelos escondidos de Markov para classificação de proteínas. Esta técnica de aprendizagem, como outras tantas, é baseada na teoria Bayesiana, a qual será brevemente apresentada neste capítulo.

3.1 O Teorema de Bayes

A Teoria de Bayes, originada no trabalho de Thomas Bayes, no século XVII, foi desenvolvida por Pierre Laplace, em 1812, assumindo a forma que ela tem hoje. Proposta originalmente para aplicação em Economia, apenas recentemente começou ser aplicada sistematicamente em outras áreas como a ciência da computação.

A única maneira de lidar-se com a incerteza é através da probabilidade. Em termos gerais, pode-se dizer que a probabilidade de um evento A, denotada por P(A), é uma medida da plausibilidade atribuída a esse evento. O método Bayesiano, difere da probabilidade clássica por incluir um elemento de subjetividade. Segundo Baldi e Brunak [3], pode-se provar matematicamente, através dos Axiomas de Cox e Jaynes, que a abordagem Bayesiana é a única maneira consistente de se desenvolver métodos em presença de incerteza. Os axiomas serão brevemente apresentados na subseção seguinte. As provas correspondentes podem ser encontradas em [13, 14, 15].

3.1.1 Os Axiomas de Cox e Jaynes

Os objeto dessa discussão serão proposições a respeito do mundo. Seja a proposição X, que pode ser falsa ou verdadeira e $\overline{X}$ o complemento de X. Uma hipótese H sobre o mundo é uma proposição ou uma conjunção de proposições sobre o mundo. Um modelo M é também uma hipótese que pode ser bem mais complexa, envolvendo muitos parâmetros. Essa apresentação axiomática é normalmente atribuída a Cox e Jaynes [16, 17].

Primeiro Axioma

Para incluir a idéia da incerteza, dada uma certa quantidade de informação I, será associado um **grau de plausibilidade** ou **grau de confiança** a cada hipótese, denotado por

$\pi(X \mid I)$. Esta informação I corresponde a uma conjunção de informações disponíveis e pode representar o conhecimento que se tem do fenômeno estudado, bem como dados experimentais. Quando se quer focar um conjunto específico de dados, é possível denotar $I = (I, D)$. Enfim, I não é necessariamente fixa e em algumas situações em que ela for bem definida e fixa, pode ser desconsiderada nas equações.

Pode-se comparar os graus de confiança em duas hipóteses *X* e *Y*, respectivamente, $\pi(X \mid I)$ e $\pi(Y \mid I)$. Assim, pode-se dizer que *X* é mais plausível do que *Y*, denotando-se por $\pi(X \mid I) > \pi(Y \mid I)$. É possível ainda considerar a transitividade da desigualdade, que é o primeiro axioma:

$$\pi(X \mid I) > \pi(Y \mid I) \quad e \quad \pi(Y \mid I) > \pi(Z \mid I) \quad \text{implica em} \quad \pi(X \mid I) > \pi(Z \mid I). \tag{3.1}$$

Deve-se observar aqui que se existe uma relação de ordem entre graus de plausibilidades, então eles podem ser expressos por números reais.

Segundo Axioma

Sejam a proposição X e seu complemento $\overline{X}$. Intuitivamente, sabe-se que quanto mais confiança tem-se em X, menos confiança tem-se em $\overline{X}$. Assim, deve haver um relacionamento entre $\pi(X \mid I)$ e $\pi(\overline{X} \mid I)$, independente da proposição. Portanto, existe uma função *F* tal que:

$$\pi(\overline{X} \mid I) = F\left[\pi(X \mid I)\right], \tag{3.2}$$

que é o segundo axioma.

Terceiro Axioma

Sejam duas proposições *X* e *Y*. O grau de crença de que, por exemplo, *X seja verdadeiro e Y falso*, irá depender do grau de crença de que *X seja verdadeiro* e do grau de crença de que *Y seja falso, dado que X é verdadeiro*. O terceiro axioma afirma que, independente das proposições sendo consideradas, bem como da informação de fundo, existe uma função G, tal que:

$$\pi(X, Y \mid I) = G\left[\pi(X \mid I), \pi(Y \mid X, I)\right]. \tag{3.3}$$

3.1.2 Substituindo graus de confiança por probabilidades

Conforme mostrado por Cox e Jaynes [15, 16], há sempre um reescalonamento *k*, de graus de crença, tal que:

(1) $P(X \mid I) = k[\pi(X \mid I)]$ está no intervalo[0, 1], e

(2) P é única e satisfaz a todas as regras da probabilidade.

Desta forma, tem-se que $F(x) = 1 - x$ e $G(x, y) = xy$ e o segundo e o terceiro axiomas podem ser reescritos da seguinte forma:

$$P(X \mid I) + P(\overline{X} \mid I) = 1 \tag{3.4}$$

e $$P(X, Y \mid I) = P(X \mid I) \,.\, P(Y \mid X, I) \tag{3.5}$$

É possível, portanto, substituir os graus de crença por probabilidades.

Do terceiro axioma (3.5), usando a simetria $P(X, Y \mid I) = P(Y, X \mid I)$, obtém-se o **Teorema de Bayes**:

$$P(Y, X \mid I) = P(Y \mid I) \,.\, P(X \mid Y, I)$$

$$\therefore \quad P(X \mid Y, I) = \frac{P(Y, X \mid I)}{P(Y \mid I)} = \frac{P(X, Y \mid I)}{P(Y \mid I)}$$

$$\therefore \quad P(X \mid Y, I) = \frac{P(X \mid I) \,.\, P(Y \mid X, I)}{P(Y \mid I)} \tag{3.6}$$

O Teorema de Bayes confere a possibilidade de reavaliar ou ampliar o conhecimento frente a uma nova informação adicionada. Está claro que é vantajoso poder reavaliar as probabilidades quando se dispõe de informação adicional. A probabilidade que se conhece antes de obter-se a nova informação Y, $P(X \mid I)$, é chamada probabilidade ***a priori*** e a nova probabilidade $P(X \mid Y, I)$, calculada após a ocorrência de Y, é chamada probabilidade ***a posteriori***, em relação a Y.

3.2 Derivação de modelos com Inferência Bayesiana

Com base no Teorema de Bayes é possível construir modelos parametrizados a partir de um conjunto de dados. Como ele permite que a probabilidade seja reavaliada cada vez que nova informação é apresentada, pode-se dizer que, de uma certa forma, ele carrega em si um processo de aprendizagem. Baseado no Teorema de Bayes, pode-se derivar modelos que vão sendo melhorados iterativamente, como será visto a seguir.

Seja $P(\lambda)$, a estimativa da probabilidade *a priori* de que o modelo λ esteja correto, antes que se tenha obtido qualquer dado (observado) e $P(\lambda \mid O)$, a estimativa atualizada da

probabilidade de que o modelo λ esteja correto, dado que a seqüência de dados O foi observada (probabilidade *a posteriori*). $P(O|\lambda)$ é a probabilidade dos dados, condicionada ao modelo. Por simplicidade, a informação I será retirada das equações seguintes. Do Teorema de Bayes, tem-se:

$$P(\lambda|O) = \frac{P(\lambda)\ P(O|\lambda)}{P(O)} = P(\lambda)\frac{P(O|\lambda)}{P(O)} \tag{3.7}$$

Considerando que O seja uma seqüência de observações $O_1\ O_2\ ...\ O_T$, tem-se que:

$$P(\lambda | O_1\ O_2\ ...\ O_T) = P(\lambda | O_1\ O_2\ ...\ O_{T-1})\frac{P(O_T | \lambda\ , O_1\ O_2\ ...\ O_{T-1})}{P(O_T | O_1\ O_2\ ...\ O_{T-1})}\ , \tag{3.8}$$

isto é, a última *a posteriori* obtida, $P(\lambda | O_1\ O_2\ ...\ O_{T-1})$, passa a ser a probabilidade *a priori*, para o cálculo da nova *a* posterior, $P(\lambda | O_1\ O_2\ ...\ O_T)$ i. Tem-se aí um modelo parametrizado a partir do conjunto de dados observados O.

Considerando que os valores de probabilidades estão no intervalo [0, 1], o produto de vários desses valores pode levar a resultados muito pequenos, que podem, inclusive, cair fora dos limites admitidos pelo computador. Por isso, normalmente é melhor trabalhar com o valor de $\log P$:

$$\log P(\lambda|O) = \log P(\lambda) + \log P(O|\lambda) - \log P(O) \tag{3.9}$$

3.2.1 Probabilidades *a priori*

Como foi visto acima, o processo de modelagem Bayesiana envolve a utilização de probabilidades *a priori*. Existem alguns princípios que podem nortear a definição destas probabilidades; um destes é o da máxima entropia. Alguns estatísticos defendem, inclusive, que o princípio da máxima entropia deva ser um princípio geral que norteie a determinação de probabilidades *a priori*, em qualquer situação. Segundo Baldi e Brunak [3], não existe um tal princípio geral, mas este da entropia máxima é útil em alguns casos.

Quando um evento $X = x_k$, que tem a probabilidade $p_k = 1$, vem a ocorrer, nenhuma informação é ganha. Quando, no entanto, um evento $X = x_k$, com $p_k < 1$ ocorre, alguma informação foi obtida. Esta quantidade de informação ganha após a observação de um evento $X = x_k$, com probabilidade p_k é definida por: $I(x_k) = \log\left(\frac{1}{p_k}\right) = -\log p_k$, onde a base do logaritmo é arbitrária. Quando o logaritmo de base 2 é usado, as unidades de informação são bits.

A entropia de uma distribuição de probabilidade $P = (p_1\ p_2\ \ldots\ p_n)$ corresponde a uma medida da quantidade de informação média ganha quando o resultado é observado e é definida por $H(P) = E(-\log P) = -\sum_{i=1}^{n} p_i \log p_i$. Segundo o princípio da máxima entropia, quando se tem que escolher uma distribuição de probabilidades *a priori* para derivação de um modelo, esta deve ser a que possua a máxima entropia. Este princípio foi defendido por Jaynes [17]: "Quando uma inferência é feita com base em informação incompleta, ela deve ser retirada da distribuição de probabilidade que maximiza a entropia, sujeita a restrições sobre a distribuição."

Existem vários outros critérios utilizados para definição de probabilidades *a priori*, os quais poder ser vistos em [3]. No caso da modelagem de famílias de proteínas através de modelos escondidos de Markov, alguma informação que se tenha dessas famílias pode ser utilizada como conhecimento *a priori* e incorporada ao processo.

3.2.2 Probabilidade dos dados em relação ao modelo

No caso de seqüências biológicas, é necessário pensar em como quantificar a adequação ou o ajuste dos dados ao modelo. Seria como definir 'o quanto' ou 'quão provavelmente' o modelo λ teria dado origem aos dados D, ao invés de analisar deterministicamente se o modelo teria ou não dado origem aos dados. Analisando desta forma, pode-se ter a certeza de que um modelo para seqüências biológicas deve ser probabilístico, de modo a poder levar em conta os aspectos ruidosos dos dados. Deve-se pensar, portanto, na probabilidade dos dados terem sido gerados pelo modelo – $P(O \mid \lambda)$. Obviamente, esta probabilidade depende do modelo e não pode ser discutida isoladamente.

3.2.3 Estimativa de parâmetros

Quando se tem dois modelos para um mesmo conjunto de dados, pode-se comparar suas probabilidades $P(\lambda_1 \mid O)$ e $P(\lambda_2 \mid O)$ com o intuito de determinar qual o melhor modelo, isto é, encontrar o conjunto de parâmetros que maximiza a probabilidade *a posteriori* $P(\lambda \mid O)$, ou $\log P(\lambda \mid O)$, e a correspondente margem de erro. Isto é o mesmo que minimizar $-\log P(\lambda \mid O)$:

$$\mathrm{E} = -\log P(\lambda \mid O) = -\log P(O \mid \lambda) - \log P(\lambda) + \log P(O). \tag{3.10}$$

O valor de $\log P(O)$ não depende dos parâmetros do modelo, sendo, portanto, indiferente para a comparação dos dois modelos. Dessa forma, o problema reduz-se a minimizar

$$\mathrm{E} = -\log P(\lambda \mid O) = -\log P(O \mid \lambda) - \log P(\lambda). \tag{3.11}$$

Além disso, o valor de $\log P(\lambda)$ funciona como um regulador, isto é, uma penalidade adicional que pode ser usada para aplicar restrições adicionais como *smoothness*. Se o valor da probabilidade *a priori*, $\log P(\lambda)$, for retirado da equação acima, o problema fica reduzido a maximizar $P(O|\lambda)$ ou $\log P(O|\lambda)$, que é o que corresponde à estimativa **ML** (*maximization likelihood*). Se a probabilidade *a priori* for mantida, a equação 3.11, corresponderá à **MAP**.

Deve-se observar aqui que o método Bayesiano é iterativo e não existe uma garantia de que um modelo ótimo vai ser encontrado. Assim, a atenção deve estar voltada à função $P(M \mid D)$ em todo espaço do modelo assim como à avaliação das expectativas da $P(M \mid D)$.

Nos próximos capítulos será vista a abordagem baseada na Teoria de Bayes – Modelos de Markov Escondidos e sua aplicação na construção de modelos para famílias de proteínas.

Capítulo 4

Modelos Escondidos de Markov

Modelos Escondidos de Markov pertencem a uma classe de modelos gráficos probabilístico que vêm sendo utilizados em diversas áreas da ciência. Fundamentados na teoria de Bayes, são especialmente adequados a situações em que dados observáveis são fartos, sem que haja uma teoria formal conhecida a respeito desses dados. Para a resolução de muitos problemas que envolvem a manipulação de um conjunto de dados visando a extração de informações úteis desses dados, modelar um sistema pode ser muito útil porque possibilita a aprendizagem sobre a fonte dos dados sem que se tenha essa fonte. Um modelo escondido de Markov é um modelo parametrizado do objeto de estudo, construído a partir das características estatísticas desse objeto, cujos parâmetros podem ser determinados de uma maneira bem definida e precisa.

Entre 1966 e 1972 foram publicados diversos artigos por Baum e colaboradores [18, 19, 20, 21, 22], os quais expunham a base teórica para modelos escondidos de Markov. Posteriormente, vários outros pesquisadores dedicaram-se ao estudo e aperfeiçoamento da técnica [23, 24, 25, 26, 27, 28, 29, 30]. Desde então, HMM's têm sido aplicados a diversos tipos de séries temporais, como nos problemas de reconhecimento de fala [12, 31, 32] e de reconhecimento de caracteres ópticos [33]. Na biologia computacional, a aplicação de HMM's vem se expandindo desde o final da década de 80 com as mais variadas aplicações, inclusive com uso de técnicas de aprendizagem.

Neste capítulo serão apresentados aspectos de modelos escondidos de Markov como arquitetura básica e algoritmos de aprendizagem utilizados.

Conforme posto por Jack Ferguson, citado por Rabiner [12], há três problemas fundamentais envolvidos no projeto de um HMM:

(1) avaliação da probabilidade de uma seqüência de observações, dado um HMM específico;

(2) determinação da melhor seqüência de estados do modelo;

(3) ajuste dos parâmetros do modelo, de modo a considerar, da melhor maneira possível, a seqüência observada.

4.1 Processos Discretos de Markov

Uma cadeia de Markov pode ser descrita como um conjunto de N estados distintos $S_1, S_2, ..., S_N$. Em instantes discretos, isto é, para $t = 1, 2, ..., T$, o sistema sofre uma mudança de estado, de acordo com a probabilidade de transição do estado atual para o estado seguinte (Figura 4. 1). O estado corrente para o tempo t é denotado por q_t.

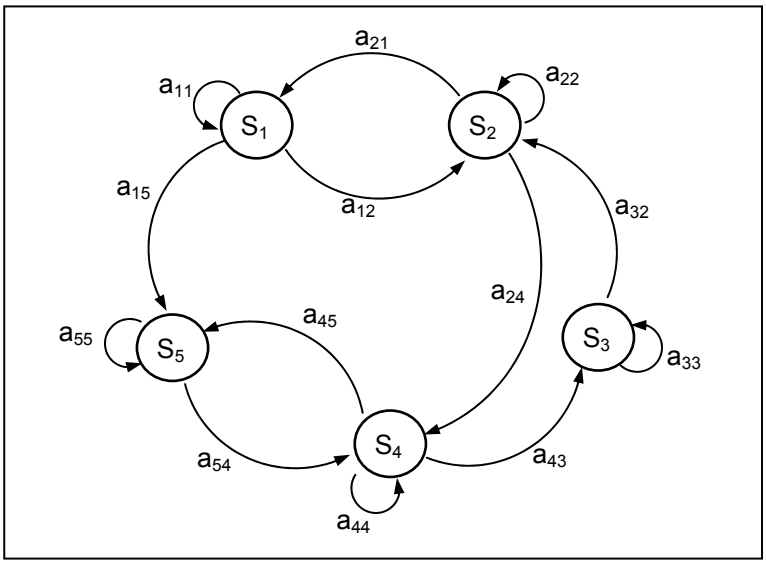


Figura 4. 1 Um HMM com 5 estados

Para uma descrição completa de um sistema como esse, em geral, seria necessária a especificação do estado corrente no tempo t, bem como dos estados predecessores. Para o caso especial de cadeia de Markov de primeira ordem, discreta, são requeridas as especificações apenas do estado corrente e seu predecessor:

$$P\left[q_t = S_k \,\middle|\, q_{t-1} = S_j \,, q_{t-2} = S_i \,, ... \right] = P\left[q_t = S_k \,\middle|\, q_{t-1} = S_j \right], \qquad (4.1)$$

onde $P\left[q_t = S_k \middle| q_{t-1} = S_j \right] = a_{jk}$, $1 \le j,k \le N$ é a probabilidade de transição do estado S_j para o estado S_k, independentemente do tempo.

Como um modelo estabelecido sobre a teoria de Bayes, tem-se que estas probabilidades obedecem a uma restrição estocástica padrão, isto é,

$$a_{jk} \ge 0 \quad \text{e} \quad \sum_{k=1}^{N} a_{jk} = 1 . \tag{4.2}$$

Exemplo 1 – Modelo cara-ou-coroa

Considere-se uma série de experimentos que conta com duas pessoas. Uma delas vai jogar uma ou mais moedas, que podem ser moedas 'honestas' ou 'viciadas', para tirar 'cara' ou 'coroa', sem que a segunda pessoa veja o que está acontecendo. Após algumas jogadas a primeira pessoa irá apenas informar à segunda o resultado das jogadas. Desta seqüência de experimentos cara-ou-coroa escondidos, o resultado será uma seqüência de 'caras' e 'coroas' do tipo

$O = O_1\ O_2\ O_3\ ...\ O_T$ = coroa coroa cara coroa cara cara cara ... coroa

Nesse caso, o número de símbolos possíveis na seqüência, denotado por M, é igual a dois, isto é: M = 2.

Para construir um modelo para este problema é necessário pensar nos seguintes aspectos:

(1) quantos e quais serão os estados do HMM?

(2) quais as probabilidades de se mudar de um estado para outro?

(3) quais as probabilidades de serem observados 'cara' ou 'coroa' em cada estado?

Solução 1. *Considere-se um modelo conforme mostrado na Figura 4. 2, com uma única moeda viciada, isto é, a probabilidade de dar 'cara' é diferente da probabilidade de dar 'coroa'.*

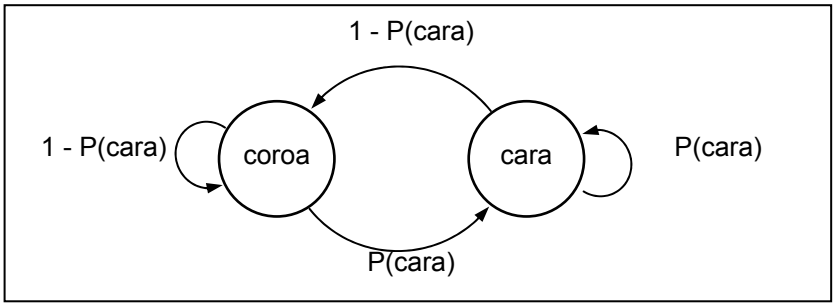


Figura 4. 2 Modelo cara-ou-coroa com uma moeda. O único parâmetro a definir é P(cara)

Uma seqüência de observações pode ser: ***cara cara coroa cara coroa ... cara***

tempo:	t=1	t=2	t=3	t=4	t=5	...	t=T
seqüência:	**cara**	**cara**	**coroa**	**cara**	**coroa**	**...**	**cara**

A probabilidade de se jogar a moeda e tirar 'cara' é denotada por P(cara) e uma vez que P(cara) + P(coroa) = 1, da equação (4.2) *temos, P(coroa) = 1 - P(cara). Definir a probabilidade P(cara), portanto, seria a única questão a resolver para a completa especificação do modelo.*

Solução 2*. Uma outra maneira de modelar o problema seria considerar 2 moedas, cada uma correspondendo a 1 estado do modelo, a partir do qual podem ser observadas as faces das moedas: 'cara' ou 'coroa', conforme a Figura 4. 3.*

Para este HMM, considere que tanto a moeda nº 1 quanto a moeda nº 2 podem ser jogadas, uma de cada vez. Uma seqüência de observações poderia ser:

tempo:	t=1	t=2	t=3	t=4	t=5	t=6	t=7	t=8
moeda:	1	1	2	2	2	1	2	1
seqüência:	**cara**	**coroa**	**coroa**	**coroa**	**cara**	**coroa**	**cara**	**cara**

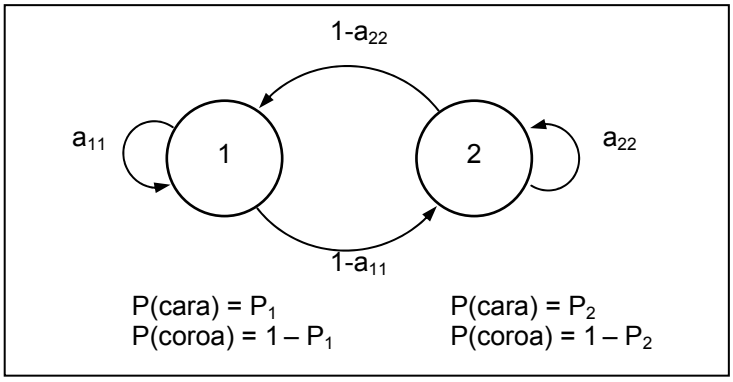


Figura 4. 3 Modelo cara-ou-coroa com duas moedas. Os parâmetros a definir são: a_{11}, a_{22}, P_1 e P_2.

As probabilidades de mudar de moeda ou permanecer com a mesma moeda de uma jogada para outra são caracterizadas por uma matriz de transição de estados $A = \{ a_{jk} \} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$*, onde* a_{11} *é a probabilidade de ter jogado a moeda nº 1 e escolher mesma moeda nº 1 para a próxima jogada;* a_{12} *é a probabilidade de ter jogado a moeda nº 1 e escolher a moeda nº 2 para a próxima jogada,* a_{21} *é a probabilidade de ter jogado a moeda nº 2 e escolher a moeda nº 1 para a próxima jogada e finalmente* a_{22} *é a probabilidade de ter jogado a moeda nº 2 e escolher a mesma moeda nº 2 para a próxima jogada.*

Cada estado é caracterizado por uma distribuição de probabilidades de emissão dos símbolos 'cara' e 'coroa', de modo que, para o estado 1 (a moeda 1), temos P(cara)=P_1 *e para o estado 2 (a moeda 2), temos P(cara)=*P_2*. Isto pode retratar a situação em que a tendenciosidade da moeda 1 é diferente daquela da moeda 2.*

Mais uma vez, devido à propriedade estocástica do modelo, tem-se:

$$
\begin{aligned}
a_{12} &= 1 - a_{11} \\
a_{21} &= 1 - a_{22} \\
P_1(coroa) &= 1 - P_1(cara) \\
P_2(coroa) &= 1 - P_2(cara)
\end{aligned}
\quad \Rightarrow \quad \text{Para se especificar este modelo, tem-se, portanto, 4 parâmetros a serem definidos}
$$

Deve-se observar que a 2ª pessoa só toma conhecimento da seqüência da saída, ela não sabe qual a moeda que foi utilizada em cada instante t, ou seja, ela não conhece a seqüência de estados q1, q2, ..., qt. Esse é o motivo do termo 'escondido' para a modelagem de Markov.

Outros modelos são possíveis, com 3 moedas, 4, etc., que possivelmente estariam mais aptos a modelar uma seqüência de observações, mas para os quais teríamos mais parâmetros a definir. Uma questão chave no projeto de HMMs é portanto definir o número ideal de estados, ou seja, o comprimento do modelo mais adequado ao problema em questão.

4.2 Definição de um HMM

Um HMM discreto de primeira ordem é um modelo gerador de séries temporais, definido por:

(1) *Um conjunto S de N estados.*

Normalmente, o conjunto de estados de um modelo possui algum significado físico. Em um modelo, todos os estados podem estar conectados entre si ou não. O conjunto de estados de um modelo será denotado por $S = \{S_1, S_2, \ldots, S_N\}$. Uma seqüência específica de estados de *S*, será denotada por $Q = q_1 q_2 \ldots q_T$. O estado no tempo t será denotado por q_t. No segundo modelo apresentado para o problema cara-ou-coroa, havia dois estados possíveis, que eram as duas moedas.

(2) *Um alfabeto discreto de M símbolos observáveis em cada estado.*

No caso do modelo cara-ou-coroa, os símbolos eram apenas dois: *cara* e *coroa*. No caso de seqüências de proteínas, os símbolos são os 20 aminoácidos possíveis. Os símbolos individuais são denotados por $\Sigma = \{ v_1, v_2, \ldots, v_M \}$.

(3) *Uma matriz de probabilidades de transição entre estados, denotada por* $A = \{a_{jk}\}$, *onde*

$$a_{jk} = P[q_{t+1} = S_k \mid q_t = S_j], \qquad 1 \le j, k \le N \tag{4.3}$$

(4) *Uma matriz de probabilidades dos símbolos observados em cada estado j,* $B = \{b_j(n)\}$, *onde*

$$b_j(n) \quad ou \quad b_{q_t}(n) = P[n \quad no\,tempo\,t \mid q_t = S_j], 1 \le j \le N \quad \forall\, n \in \sum \tag{4.4}$$

(5) *A distribuição de estados inicial* $\Pi = \{\pi_j\}$, *onde*

$$\pi_j = P[q_1 = S_j], 1 \le j \le N \tag{4.5}$$

Em resumo, a especificação de um HMM requer:

$S \rightarrow$ um conjunto de estados;

$\Sigma \rightarrow$ um alfabeto;

$\lambda \rightarrow$ distribuições de probabilidades (*A, B,* π), onde:

$A = \{a_{jk}\}$ – transição de estados,

$B = \{b_j(n)\}$ – emissão de símbolos,

$\pi = \{\pi_j\}$ – estados iniciais,

sendo satisfeitas as seguintes propriedades:

$a_{jk} \ge 0, \forall\, j,k$; $\quad \sum_k a_{jk} = 1, \forall\, j$;

$b_j(n) \ge 0$; $\quad \sum_n b_j(n) = 1, \forall\, j$;

$\pi_j \ge 0$; $\quad \sum_j \pi_j = 1, \forall\, j$.

Pode-se dizer que, dados os valores apropriados de *N, M, A, B* e π, o HMM pode gerar uma seqüência $O = O_1\ O_2\ \ldots\ O_T$, onde cada observação O_t é um dos símbolos de Σ e T é o número de observações na seqüência, da seguinte maneira:

(1) Escolha um estado inicial $q_1 = S_j$, de acordo com a distribuição de estados inicial π;

(2) Faça $t = 1$;

(3) Escolha $O_t = v_m$ de acordo com a distribuição de probabilidade de símbolos no estado S_j, i. e., $b_j(n)$.

(4) Escolha o próximo estado q_{t+1} de acordo com a distribuição de probabilidades de transição entre estados para o estado S_k, i.e., a_{jk}.

(5) Faça $t=t+1$ e volte para o passo 3, caso $t < T$; caso contrário, termine o procedimento.

Pode-se olhar para este procedimento, tanto como um gerador de seqüências como um verificador da probabilidade de que uma dada seqüência de símbolos tenha sido gerada pelo modelo dado.

No intuito de simplificar a nomenclatura, daqui em diante, um modelo será referido apenas pelo seu conjunto de distribuições de probabilidades λ.

4.3 Os problemas fundamentais para HMMs

Como já foi dito na introdução deste capítulo, três problemas básicos devem ser resolvidos para que um HMM seja útil em aplicações reais. Veremos a seguir a solução matemática, formal, para cada um dos problemas básicos.

4.3.1 Problema 1 - Avaliação

Dados uma seqüência de T símbolos observados $O = O_1\ O_2\ \ldots\ O_T$ e um modelo $\lambda = (A, B, \pi)$, qual a probabilidade de que a seqüência tenha sido produzida pelo modelo – $P(O \mid \lambda)$?

Para se fazer uma tal avaliação, pode-se pontuar (calcular um escore de) o casamento da seqüência de interesse com o modelo, o que é bastante útil, especialmente nos seguintes casos:

(a) quando existe mais de um modelo possível, esta avaliação permite decidir por aquele que melhor casa com as observações;

(b) quando existe um modelo que descreve determinada situação real e deseja-se determinar se uma dada seqüência de observações também compõe aquela situação; este é o caso quando se tem o modelo de uma família de proteínas e deseja-se saber se uma dada seqüência pertence àquela família.

Um modo de computar esta probabilidade de que uma seqüência de T observações tenha sido gerada por um modelo dado λ, é a partir da enumeração de todas seqüências de estados (caminhos) possíveis, de comprimento T.

Seja a seqüência de estados $Q = q_1\ q_2\ \ldots\ q_T$, onde q_1 é o estado inicial. A probabilidade de que esta seqüência de estados seja percorrida no modelo é dada por:

$$P(Q \mid \lambda) = \pi_{q_1} a_{q_1 q_2} a_{q_2 q_3} \ldots a_{q_{T-1} q_T} . \qquad (4.6)$$

Assumindo que as observações são independentes entre si, a probabilidade de que a seqüência de observações *O* seja gerada pelo modelo λ, passando pelo caminho Q, ou seja, a probabilidade de *O*, dados *Q* e λ , é dada por

$$P(O|Q,\lambda) = \prod_{t=1}^{T} P(O_t|q_t,\lambda) \quad (4.7)$$

$$= b_{q_1}(O_1) \,.\, b_{q_2}(O_2) \,.\,.\,.\, .b_{q_T}(O_T) \quad (4.8)$$

Assim, a probabilidade de que, dado o modelo λ, o caminho *Q* seja tomado e a seqüência *O* seja gerada pode ser dada por (terceiro Axioma de Cox Jaynes):

$$P(O,Q|\lambda) = P(O|Q,\lambda).P(Q|\lambda) \quad (4.9)$$

$$= \pi_{q_1} b_{q_1}(O_1)\; a_{q_1 q_2} b_{q_2}(O_2) \;\; \ldots \;\; a_{q_{T-1} q_T} b_{q_T}(O_T) \quad (4.10)$$

e a probabilidade de que a seqüência *O* seja gerada, dado um modelo λ será então a soma das probabilidades da seqüência ser gerada para todos os caminhos:

$$P(O|\lambda) = \sum_{todos Q} P(O|Q,\lambda).P(Q|\lambda) \quad (4.11)$$

Complexidade

Da equação (4.11), podemos ver que, para cada caminho de estados Q, são necessárias (*2T-1*) multiplicações. Como para cada *t = 1, 2, ..., T* existem *N* estados possíveis, então existem N^T caminhos possíveis. Assim, para a computação da equação (4.11) são necessárias $(2T-1)N^T$ multiplicações e N^T-1 adições, o que torna este cálculo infactível, mesmo para pequenos valores de *N* e *T*.

Uma técnica de programação dinâmica conhecida como procedimento ou algoritmo *forward-backward* permite efetuar o cálculo dessa probabilidade de modo mais eficiente [12].

Algoritmo Forward-Backward

Parte 1 - *Forward*

Seja a variável *forward* $\alpha_t(j) = P(O_1 O_2 \ldots O_t, q_t = S_j|\lambda)$, a probabilidade de que a seqüência $O_1\; O_2 \ldots O_t$ seja observada até o tempo *t*, considerando que O_t foi emitido no estado S_j isto é, $q_t = S_j$, dado o modelo λ. Pode-se visualizar melhor a variável *forward* pensando-se nos estados e transições através do tempo como uma treliça. Em cada tempo *t*, cada estado recebe as transições

de todos os estados, independente da seqüência de observações. Na Figura 4. 4 está ilustrada a seqüência de operações necessárias para o cálculo da variável $\alpha_{t+1}(k)$.

A parte *forward* do algoritmo dedica-se ao cálculo de $P(O|\lambda)$ e pode ser descrita em três passos:

(1) Incialização:

$$\alpha_1(j) = \pi_j b_j(O_1), \quad 1 \le j \le N \tag{4.12}$$

(2) Indução:

$$\alpha_{t+1}(k) = \left[\sum_{j=1}^{N} \alpha_t(j)\ a_{jk}\right] b_k(O_{t+1}) \quad , \quad \begin{matrix} 1 \le t \le T-1 \\ 1 \le k \le N \end{matrix} \tag{4.13}$$

(3) Finalização:

$$P(O|\lambda) = \sum_{j=1}^{N} \alpha_T(j) \quad . \tag{4.14}$$

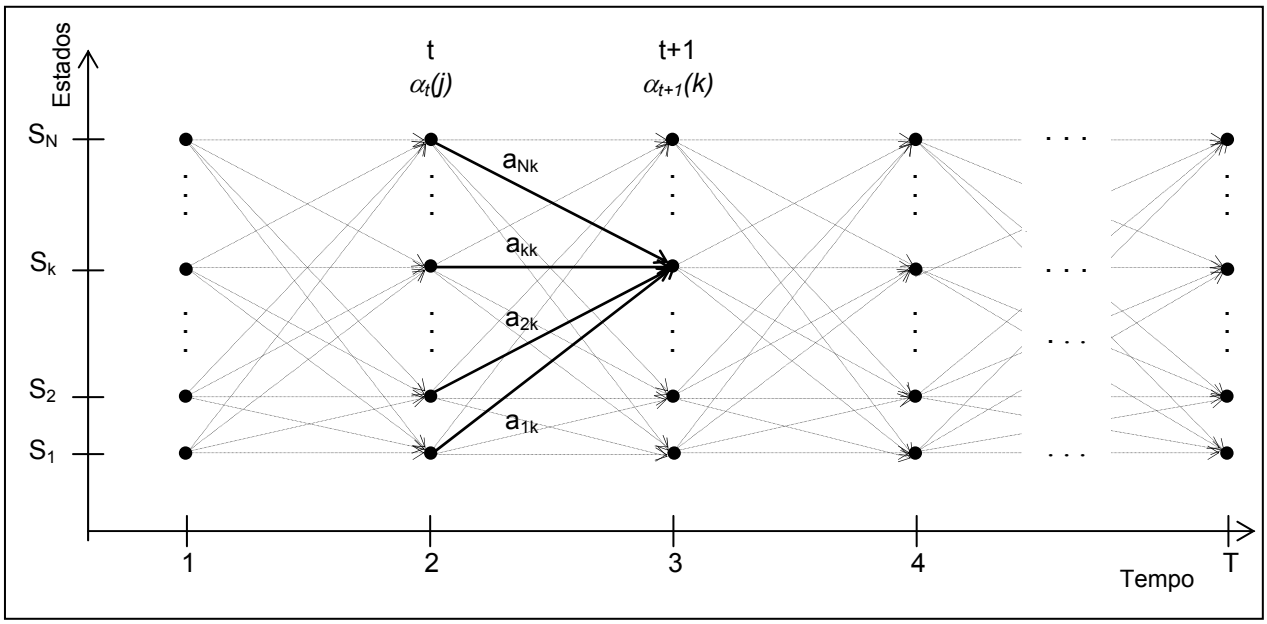


Figura 4. 4 Modelo ilustrativo do cálculo das variáveis *forward*.

O passo (1) inicializa as probabilidades *forward* para todos os estados S_j, como sendo o produto da probabilidade inicial para cada estado S_j pela probabilidade de emissão do símbolo O_1 neste estado. No passo (2), o produto $\alpha_t(j)\ a_{jk}$ denota a probabilidade de que (i) a seqüência $O_1\ O_2$ ... O_t seja observada; (ii) o estado $q_t = S_j$ e (iii) o estado S_k seja atingido no tempo $t+1$, a partir de

S_j. Assim, o somatório deste produto sobre todos os N estados possíveis S_j, $1 \le j \le N$, $\sum_{j=1}^{N} \alpha_t(j)\, a_{jk}$, corresponde à probabilidade de atingir S_k no tempo $t+1$, considerando todas as observações parciais anteriores. O valor de $\alpha_{t+1}(k)$ é então obtido levando-se em conta a probabilidade $b_k(O_{t+1})$. Deve-se notar que para cada t (de 1 a T-1), é calculado $\alpha_{t+1}(k)$, para todos os estados S_k possíveis ($1 \le k \le N$). O passo (3) dá o valor de $P(O \mid \lambda)$ como sendo a soma das variáveis $\alpha_T(j)$ (para todos os estados S_j possíveis). Uma vez que, por definição, $\alpha_t(j) = P\left(O_1 O_2 ... O_t\ ,\ q_t = S_j \middle| \lambda\right)$, para $t = T$ temos: $\alpha_T(j) = P\left(O_1 O_2 ... O_T\ ,\ q_T = S_j \middle| \lambda\right)$, e portanto $P(O \mid \lambda)$ é a soma de todo $\alpha_T(j)$, com $1 \le j \le N$.

Complexidade

No passo (2) serão computadas, para cada k e cada t: ($N+1$) multiplicações e (N-1) adições. Uma vez que o passo se repete para $1 \le t \le T-1$ e $1 \le k \le N$, serão $N(T-1)(N+1)$ multiplicações e $N(T-1)(N-1)$ adições, isto é, $O(N^2T)$, contra $O(2TN^T)$, do algoritmo anterior.

Com a parte *forward* do algoritmo, que faz o cálculo de $P(O \mid \lambda) = \sum_{j=1}^{N} \alpha_T(j)$, está resolvido o problema de avaliação (Problema 1), entretanto, será apresentada logo a parte *backward* do algoritmo, a qual será útil mais adiante.

Parte 2 - *Backward*

De maneira análoga à variável *forward*, seja a variável *backward* definida como

$$\beta_t(j) = P(O_{t+1} O_{t+2} ... O_T \mid q_t = S_j\ , \lambda)\,, \tag{4.15}$$

a probabilidade da seqüência parcial de observações $O_{t+1}\, O_{t+2} \ldots O_T$, do tempo $t+1$ até o fim, dados o estado S_j no tempo t e o modelo λ. O algoritmo é o seguinte:

(1) Inicialização:

$$\beta_T(j) = 1\ ,\quad 1 \le j \le N \tag{4.16}$$

(2) Indução:

$$\beta_t(j) = \sum_{k=1}^{N} a_{jk}\, b_k(O_{t+1})\, \beta_{t+1}(k)\ ,\quad t = T-1, T-2, \ldots, 1 \qquad 1 \le j \le N \tag{4.17}$$

(3) Finalização:

$$P(O \mid \lambda) = \sum_{j=1}^{N} \beta_1(j) \tag{4.18}$$

O passo (1) define arbitrariamente $\beta_T(j) = 1$ para todos os j, $1 \leq j \leq N$. No passo (2) $\beta_t(j)$ é calculado tendo em vista que para se ter estado em S_j no tempo t, considerando a seqüência observada a partir de $t+1$, devem ser considerados todos os estados possíveis S_k no tempo $t+1$, levando-se em conta a transição do estado S_j para o estado S_k , (a_{jk}) e o símbolo observado (O_{t+1}) em cada um desses S_k , $(b_k(O_{t+1}))$, e ainda a probabilidade da seqüência parcial observada remanescente a partir do estado S_k , $(\beta_{t+1}(k))$. A complexidade desta parte do algoritmo pode ser dada, analogamente à da parte *forward*, por O(N^2T). Na finalização, $P(O \mid \lambda)$ é calculado como a soma dos $\beta_1(j)$, para todo j, $1 \leq j \leq N$.

4.3.2 Problema 2 – Seqüência de estados ótima

O problema 2 consiste em tentar descobrir a parte escondida do modelo, isto é, encontrar a seqüência de estados 'correta'. Ocorre que, em muitas situações, não é possível determinar a seqüência 'correta' de estados. Muitas vezes, o próprio sentido de 'correta' não é bem definido. Assim, dada uma seqüência de observações $O_1\ O_2\ ...\ O_T$ e um modelo $\lambda = (A, B, \pi)$, o objetivo é determinar uma seqüência de estados que seja ótima em algum sentido, por exemplo, que melhor explique a seqüência de observações. É preciso definir critérios de otimalidade para a situação em questão. Como, na realidade, existem diversos critérios que podem ser utilizados, a tarefa de definir estes critérios é, via de regra, muito árdua.

Um critério possível é determinar os estados q_t que são, individualmente, mais prováveis e buscar maximizar o número destes estados. Para tanto, é definida a variável probabilidade *a posteriori* $\gamma_t(k) = P(q_t = S_k | O, \lambda)$, como a probabilidade de estar no estado S_k no tempo t, dada a seqüência de observações O e o modelo λ. Esta probabilidade pode ser expressa em termos das variáveis *forward-backward*:

$$\gamma_t(j) = P(q_t = j \mid O, \lambda)$$

$$\gamma_t(j) = \frac{\alpha_t(j)\beta_t(j)}{\sum_{j=1}^{N} \alpha_t(j)\beta_t(j)}\ , \qquad (4.19)$$

onde $\alpha_t(j)$ é a probabilidade da seqüência $O = O_1\ O_2\ ...\ O_t$ estando no estado S_j no tempo t, $\beta_t(j)$ é a probabilidade da seqüência $O = O_{t+1}\ O_{t+2}\ ...\ O_T$, estando no mesmo S_j no tempo t, e o fator de normalização $\sum_{j=1}^{N} \alpha_t(j)\beta_t(j)$ é utilizado para forçar $\sum_{j=1}^{N} \gamma_t(j) = 1$.

Para determinar o estado q_t mais provável em cada tempo t, faz-se:

$$q_t = \arg\max_{1 \le j \le N} [\gamma_t(j)] \,,\ 1 \le t \le T \,. \tag{4.20}$$

Um problema com a utilização deste critério para encontrar a melhor seqüência de estados, pode surgir, por exemplo, quando alguma transição entre dois estados consecutivamente selecionados como ótimos tem probabilidade zero, o que significa que a seqüência de fato não é válida. Uma alternativa para reduzir este problema seria, ao invés de trabalhar com estados individualmente, maximizar o número esperado de pares corretos de estados (q_t , q_{t+1}), ou triplas, ou quádruplas, etc. O critério mais utilizado, para a maioria das aplicações, é encontrar a única melhor seqüência de estados (caminho) de modo a maximizar P($Q \mid O,\ \lambda$), o que é equivalente a maximizar $P(Q,\ O \mid \lambda)$. Para encontrar um tal melhor caminho, o **Algoritmo de Viterbi** apresenta-se como uma boa alternativa.

Dada a seqüência de observações $O = \{O_1\ O_2\ \ldots O_T\}$ e o modelo $\lambda = (A\ ,\ B\ ,\ \pi)$, o objetivo é encontrar uma única melhor seqüência de estados $Q = \{q_1\ q_2\ \ldots q_T\}$.

Seja a variável

$$\delta_t(j) = \max_{q_1, q_2, \ldots, q_{t-1}} P(q_1\ q_2\ \ldots q_{t-1},\ q_t = S_j\ ,\ O_1\ O_2\ \ldots O_t \mid \lambda)\,, \tag{4.21}$$

a probabilidade do caminho mais provável, no tempo *t*, considerando as *t* primeiras observações, terminando no estado S_j. Os passos do algoritmo são dados a seguir. Para recuperar a seqüência de estados, será armazenado o estado *j* onde $\delta_t(j)$ é maximizado para cada *t* e *j*, no vetor $\Psi_t(j)$.

Algoritmo de Viterbi

(1) Inicialização:

$$\delta_1(j) = \pi_j\, b_j(O_1)\,, \quad 1 \le j \le N \tag{4.22}$$

$$\Psi_1(j) = 0 \tag{4.23}$$

(2) Recursão:

$$\delta_t(k) = \max_{1 \le j \le N} \left[\delta_{t-1}(j)\, a_{jk}\right] b_k(O_t)\,, \quad \begin{matrix} 2 \le t \le T \\ 1 \le k \le N \end{matrix} \tag{4.24}$$

$$\Psi_t(k) = \arg\max_{1 \le j \le N} \left[\delta_{t-1}(j)\, a_{jk}\right]\,, \quad \begin{matrix} 2 \le t \le T \\ 1 \le k \le N \end{matrix} \tag{4.25}$$

(3) Finalização

$$P^* = \max_{1 \le j \le N} [\delta_T(j)] \tag{4.26}$$

$$q_T^* = \arg\max_{1 \le j \le N} [\delta_T(j)] \tag{4.27}$$

(4) Caminho mais provável

$$q_t^* = \Psi_{t+1}(q_{t+1}^*), \quad t = T-1, T-2, ..., 1 \tag{4.28}$$

Deve-se notar que a variável $\delta_t(j)$ difere de $\alpha_t(j)$ na medida em que $\alpha_{t+1}(j)$ é calculada a partir do somatório $\sum_{j=1}^{N} (\alpha_t(j)\, a_{jk})$, enquanto que $\delta_{t+1}(j)$ considera apenas o valor máximo de $\delta_t(j)$ a_{jk}, para cada j de *1* a *N*.

4.3.3 Problema 3 – Estimativa de parâmetros

O problema mais difícil num HMM é determinar um método para ajustar seus parâmetros (*A, B,* π) de modo a maximizar a probabilidade de uma seqüência de observações, dado o modelo. Não existe uma maneira analítica de resolver este problema, isto é, de determinar os parâmetros de um HMM, daí a necessidade de lançar mão de métodos de aprendizagem de máquina. Será abordado aqui o método iterativo de **Baum-Welch** (método EM – Expectation-Maximization), que é especialmente adequado para problemas onde existem variáveis escondidas. Com este algoritmo é possível determinar $\lambda = (A, B, \pi)$ de modo que $P(O|\lambda)$ seja localmente maximizada.

Algoritmo Expectation-Maximization (EM)

O algoritmo EM é um algoritmo iterativo que executa basicamente dois passos:

(1) Expectation

Computar as emissões e transições em cada estado considerando os parâmetros atuais do modelo.

(2) Maximization

Computar os novos valores para parâmetros, considerando as expectativas correntes de emissões e transições.

Para apresentação deste método, inicialmente serão mostradas as equações para cálculo dos parâmetros de emissão e transição com estimativa ML, considerando apenas uma única seqüência de observações *O*.

Em seguida, serão feitas pequenas alterações nas equações para suportar múltiplas seqüências. Essas seqüências serão consideradas independentes e portanto a probabilidade total do conjunto de dados será dada pelo produto das probabilidades individuais.

Seqüência única de observações

Seja a variável $\xi(j,k) = P(q_t = S_j, q_{t+1} = S_k \mid O, \lambda)$ a probabilidade de estar no estado S_j, no tempo t, e no estado S_k, no tempo $t+1$, dados O e λ. Esta probabilidade pode ser calculada a partir das variáveis *forward-bacward*:

Probabilidade da seqüência $O_1\ O_2 \ldots O_t$ terminando no estado S_j no tempo t

Probabilidade de ir de S_j para S_k e emitir O_{t+1} em S_k

Probabilidade da seqüência $O_{t+1}\ O_{t+2} \ldots O_T$ começando no estado S_k no tempo $t+1$

$$\xi_t(j,k) = \frac{\overbrace{\alpha_t(j)}\ \overbrace{a_{jk}\ b_k(O_{t+1})}\ \overbrace{\beta_{t+1}(k)}}{\sum_{j=1}^{N}\sum_{k=1}^{N}\alpha_t(j)\,a_{jk}\,b_k(O_{t+1})\,\beta_{t+1}(k)} \qquad (4.29)$$

Novamente, o denominador é um fator de normalização.

Uma vez que $\gamma_t(j)$ foi definida como a probabilidade de estar no estado S_j, no tempo t, pode ser expressa em função de $\xi_t(j\ ,\ k)$:

$$\gamma_t(j) = \sum_{k=1}^{N}\xi_t(j,k) \qquad (4.30)$$

Se este valor for calculado para cada t, o somatório de todos esses resultados, exceto para o tempo T, será o número de vezes que se espera que o estado S_j seja visitado, ou seja, $\sum_{t=1}^{T-1}\gamma_t(j)$ será o número de transições que se espera que ocorram a partir de S_j.

Da mesma forma, o somatório $\sum_{t=1}^{T-1}\xi_t(j,k)$, será o número esperado de transições de S_j para S_k.

A partir daí, podem ser derivadas as fórmulas de reestimativa para π, A *e* B, o que corresponde ao passo E (expectation) do algoritmo EM:

$$\bar{\pi}_j = \gamma_1(j) \quad \Rightarrow \text{nr. de vezes que se espera estar no estado } S_j, \text{ no tempo } t = 1 \tag{4.31}$$

$$\bar{a}_{jk} = \frac{\sum_{t=1}^{T-1} \xi_t(j,k)}{\sum_{t=1}^{T-1} \gamma_t(j)} \quad \Rightarrow \frac{\text{número esperado de transições de } S_j \text{ para } S_k}{\text{número esperado de transições a partir de } S_j} \tag{4.32}$$

$$\bar{b}_k(n) = \frac{\sum_{t=1,\, s.t.\, O_t = v_m}^{T} \gamma_t(k)}{\sum_{t=1}^{T} \gamma_t(k)} \quad \Rightarrow \frac{\text{número esperado de vezes no estado } S_k \text{ e observado o símbolo } v_m}{\text{número esperado de vezes no estado } S_k} \tag{4.33}$$

Assim, a partir do modelo atual, pode-se encontrar cada valor do lado direito das equações acima obtendo-se os novos parâmetros do modelo. Conforme demonstrado por Baum [20, 22], o novo modelo encontrado terá, no mínimo, a mesma qualidade do anterior, isto é, $\bar{\lambda} \geq \lambda$.

Se o processo continua, iterativamente tomando o novo modelo $\bar{\lambda}$ e substituindo os valores correspondentes no lado direito das equações, um novo $\bar{\lambda}$ pode ser encontrado, melhor que o anterior, e prosseguir-se substituindo até que um limite seja atingido. O resultado final encontrado será uma estimativa ML do HMM. Deve-se observar, entretanto, que esta MLE é local, isto é, não há uma garantia de que o modelo encontrado será o ótimo. Quando existe um grau de incerteza e a quantidade de dados disponíveis é pequena, a distribuição $P(\lambda \mid D)$ é irregular, com muitos máximos locais. Assim, a atenção deve estar voltada para a função $P(\lambda \mid D)$ sobre todo o espaço do modelo, ao invés de estar voltada apenas para seu valor máximo.

Deve-se observar que as restrições estocásticas dos parâmetros do HMM são satisfeitos a cada iteração:

$$\sum_{j=1}^{N} \bar{\pi}_j = 1$$

$$\sum_{k=1}^{N} \bar{a}_{jk} = 1 \quad , \quad 1 \leq j \leq N$$

$$\sum_{n \in \Sigma} \bar{b}_k(n) = 1 \quad , \quad 1 \leq k \leq N$$

4.3.4 Múltiplas Seqüências de Observações

As fórmulas de reestimativa apresentadas até agora consideram apenas uma seqüência de observações $O = O_1\ O_2\ ...\ O_T$. Entretanto, para que todos os parâmetros do modelo possam ser estimados adequadamente, é necessário que lhe sejam apresentadas várias seqüências de observações. Desta forma, serão introduzidas algumas alterações nas equações já apresentadas de modo a considerar múltiplas seqüências de observações. Considerando ainda que para o objeto deste trabalho – classificação de proteínas, a arquitetura adotada para o modelo possui um único estado inicial, conforme será visto no próximo capítulo, não haverá necessidade da reestimativa para as probabilidades iniciais, pois $\pi_{S_{inicial}} = 1$ e $\pi_j = 0$, $j \neq S_{inicial}$.

Seja o conjunto de K seqüências de observações: $O = [O^{(1)}\ O^{(2)}\ ...\ O^{(K)}]$, onde $O^{(k)} = \left[O_1{}^{(k)}\ O_2{}^{(k)}\ ...\ O_{T_k}{}^{(k)} \right]$ é a k-ésima seqüência de observações, assumindo que as seqüências são independentes entre si. Os parâmetros do modelo λ, devem ser ajustados de modo a maximizar

$$P(O \mid \lambda) \quad = \prod_{k=1}^{K} P(O^{(k)} \mid \lambda) = \prod_{k=1}^{K} P_k \tag{4.34}$$

As novas fórmulas de reestimativa, considerando K seqüências de observações, devem ser baseadas nas estimativas individuais de cada seqüência. Assim, fazendo os devidos ajustes, as fórmulas passarão a [12]:

$$\bar{a}_{ij} = \frac{\sum_{k=1}^{K} \left[\frac{1}{P_k} \sum_{t=1}^{T_k - 1} \alpha_t^k(i)\ a_{ij}\ b_j(O_{t+1}^{(k)})\ \beta_{t+1}^k(j) \right]}{\sum_{k=1}^{K} \left[\frac{1}{P_k} \sum_{t=1}^{T_k - 1} \alpha_t^k(i)\ \beta_t^k(j) \right]} \tag{4.36}$$

$$\text{e} \quad \bar{b}_j(n) = \frac{\sum_{k=1}^{K} \left[\frac{1}{P_k} \sum_{t=1}^{T_k - 1} \alpha_t^k(j)\ \beta_t^k(j) \right]}{\sum_{k=1}^{K} \left[\frac{1}{P_k} \sum_{t=1}^{T_k - 1} \alpha_t^k(j)\ \beta_t^k(j) \right]} \quad s.t.\ O_t = n \tag{4.37}$$

A seguir, o algoritmo EM para estimativa dos parâmetros do HMM.

Algoritmo Baum-Welch

(1) Inicialização
    Atribuir valores a λ
(2) Expectativa

(2.1) Computar o número esperado de transições do estado *i* para o estado *j*, para cada par de estados *i*-*j* :

$$A_{ij} = \sum_{k=1}^{K}\Big[\frac{1}{P(O^{(k)})}\sum_{t=1}^{T_k-1}\alpha_t^k(i)\,a_{ij}\,b_j\big(O_{t+1}^{(k)}\big)\,\beta_{t+1}^k(j)\Big]$$

(2.2) Computar o número esperado de emissões de cada símbolo *n* ocorridas no estado *j* :

$$B_j(n) = \sum_{k=1}^{K}\Big[\frac{1}{P(O^{(k)})}\sum_{\substack{t=1\\ s.t.\ O_t=n}}^{T_k-1}\alpha_t^k(j)\;\beta_t^k(j)\Big]$$

(3) Maximização

(3.1) Computar novos valores para $\{a_{ij}\}$:

$$\overline{a}_{ij} = \frac{\sum_{k=1}^{K}\frac{1}{P_k}\sum_{t=1}^{T_k-1}\alpha_t^k(i)\,a_{ij}\,b_j\big(O_{t+1}^{(k)}\big)\,\beta_{t+1}^k(j)}{\sum_{k=1}^{K}\frac{1}{P_k}\sum_{t=1}^{T_k-1}\alpha_t^k(i)\;\beta_t^k(i)}$$

(3.2) Computar novos valores para $\{b_j(n)\}$:

$$\overline{b}_j(n) = \frac{\sum_{k=1}^{K}\frac{1}{P_k}\sum_{\substack{t=1\\ s.t.\ O_t=n}}^{T_k-1}\alpha_t^k(j)\;\beta_t^k(j)}{\sum_{k=1}^{K}\frac{1}{P_k}\sum_{t=1}^{T_k-1}\alpha_t^k(j)\;\beta_t^k(j)}$$

(4) Repetir os passos (2) e (3) até que a melhoria do escore da probabilidade para o conjunto de seqüências de treinamento seja menor do que um dado parâmetro ε.

Capítulo 5

Aplicação de Modelos de Markov Escondidos a Seqüências de Proteínas

Apesar de milhares e milhares de seqüências biológicas virem sendo armazenadas em bancos de dados públicos ao redor do mundo, apenas uma pequena fração das informações que jazem nesses dados tem sido sistematicamente explorada. Nesta perspectiva, muito esforço tem sido envidado no estudo de técnicas que possam extrair informações úteis desses dados com mais velocidade e sensitividade.

Na biologia computacional, a aplicação de HMM's vem se expandindo desde o final da década de 80, com as mais variadas aplicações, inclusive com uso de técnicas de aprendizagem: alinhamento de seqüências [34], detecção de padrões em seqüências de proteínas [35], detecção de sítios de ligação em DNA [36], modelagem de famílias de seqüências (proteínas e DNA) [37, 38, 39] , predição de genes [40, 41], modelagem de regiões codificantes e não codificantes em DNA [42]. Em 1992 foi liberada a primeira versão do HMMER – sistema disponível na Internet para construção de HMM's e em 1997, outros bancos de dados de HMMs foram disponibilizados: Pfam – de famílias de proteínas [43], e FORESST – de estruturas secundárias de proteínas [44]. Neste mesmo ano, Burge [45, 46] propôs um HMM para identificação de genes e o disponibilizou juntamente com a ferramenta GENSCAN.

Comparações de pares de seqüências, muitas vezes não explicitam as relações e correlações que podem haver entre essas seqüências, que extrapolam a simples similaridade entre as bases propriamente dita. Várias técnicas têm sido desenvolvidas buscando explorar essas informações subjacentes. Parte dessas técnicas envolve a construção de modelos do consenso a

partir de alinhamentos múltiplos de famílias de proteínas, tais como perfis, padrões flexíveis e blocos, os quais podem ser vistos como casos especiais da abordagem modelo de Markov escondido.

Neste capítulo aplica-se a técnica HMM à modelagem de famílias de proteínas. Os modelos construídos possuem arquitetura *left-right* e o alfabeto considerado será composto pelas 20 letras que representam os aminoácidos. As observações serão as seqüências de aminoácidos que representam a estrutura primária de uma proteína. Um bom modelo para uma família de proteínas será aquele que atribuir alta probabilidade a seqüências que pertençam à família modelada, e baixa probabilidade àquelas não pertençam à família.

Krogh e colaboradores [39] construíram um HMM que identifica os elementos centrais de proteínas homólogas, a partir da identificação de trechos relativamente conservados de suas estruturas primárias. Como eles observam, os alinhamentos produzidos através do HMM construído levam em consideração penalidades *position-dependent*, diferentemente dos métodos tradicionais de alinhamento, que consideram penalidades para inserções, remoções e substituições idênticas, em quaisquer que sejam as regiões das seqüências.

5.1 Arquitetura do HMM

A definição da arquitetura de um HMM é altamente dependente do problema. No caso de seqüências biológicas, o aspecto linear das seqüências é freqüentemente bem capturado pelas arquiteturas *left-right*. Uma arquitetura é dita *left-right* quando ela não admite, uma vez que tenha ocorrido uma transição de um estado q_i para o estado q_j, retornar para o estado q_i. A arquitetura adotada neste trabalho é a **arquitetura *left-right* linear padrão**, proposta por Krogh et al [39]. Para explicar a arquitetura padrão adotada, pode-se imaginar, primeiro, uma arquitetura composta por um conjunto de estados m_j, em que cada estado representa uma coluna da seqüência linear da proteína (ou da família) que se quer modelar, como mostra a Figura 5. 1.

A cada estado está associada uma distribuição de probabilidades de emissão $B = \{b_{m_j}(n)\}$, de acordo com a composição da família na coluna correspondente. A probabilidade de transição, de qualquer estado para o seguinte, nesse caso, seria igual a 1.

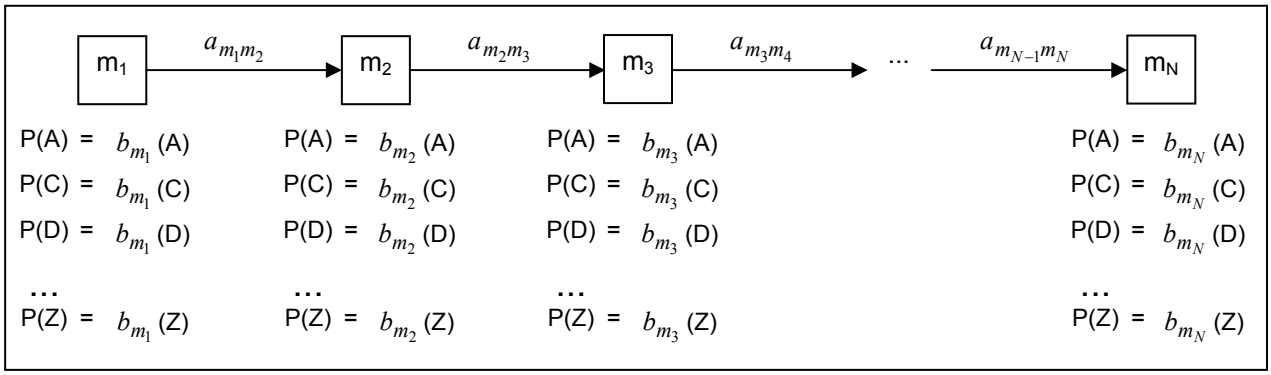


Figura 5. 1 Modelo simplificado para uma família de proteínas

Acontece que um HMM como este não poderia modelar eventos de inserção e remoção que, via de regra, ocorrem durante a evolução, alterando, inclusive, o comprimento das seqüências. Assim, os estados do HMM acima serão chamados de estados ***match*** e serão adicionados estados ***insert*** e ***delete*** em todas as posições possíveis (Figura 5. 2). Embora não seja necessário, serão considerados aqui, apenas por conveniência, dois estados adicionais: o estado inicial $\boldsymbol{m_0}$ e o estado final $\boldsymbol{m_{N+1}}$. No tempo 0, o sistema estará sempre no estado inicial m_0.

A arquitetura padrão terá, portanto, o seguinte conjunto de estados: $Q = \{ m_0, m_1, ..., m_N, i_0, ..., i_N, d_1, ..., d_N, m_{N+1} \}$, onde os estados *match*: $m_1, m_2, ..., m_N$ (representados pelos quadrados na Figura 5. 2) e os estados *insert*: $i_0, i_1, ..., i_N$ (representados pelos losangos) possuem, associadas a eles, as distribuições de probabilidades de emissão $B = \{b_{m_j}(v_n)\}$ e $B = \{b_{i_j}(v_n)\}$, respectivamente, enquanto que os estados *delete*: $d_1, d_2, ..., d_N$ (representados pelos círculos na Figura 5. 2) são estados **mudos**, isto é, que não emitem símbolos, e por essa razão são chamados também estados ***skip***. A utilização de um estado *delete* equivale a adicionar um símbolo espaço ('–') ao alfabeto e forçar a emissão desse símbolo. Os estados m_0 e m_{N+1} também são mudos.

A seqüência linear de estados $m_0 \rightarrow m_1 \rightarrow m_2 \rightarrow ... \rightarrow m_N \rightarrow m_{N+1}$ é o ***backbone*** do modelo. Os *loops* nos estados *insert* permitem que inserções múltiplas ocorram num determinado local da seqüência. As transições de um estado ***j*** para um estado ***k*** ocorrem de acordo com a distribuição de probabilidades de transição $A = \{a_{jk}\}$. A Figura 5. 2 mostra a arquitetura padrão proposta onde, para simplificar, apresenta-se um HMM de comprimento N = 3.

Uma vez que existe uma distribuição de probabilidades distinta associada a cada posição, que avalia a probabilidade de emissão de cada um dos 20 aminoácidos na posição correspondente, bem como a probabilidade de que não haja aminoácidos naquela posição, isto é,

que uma seqüência pertencente à família modelada possua um *gap* naquela posição, pode-se dizer que um HMM que modela uma família de proteínas é uma espécie de **perfil generalizado**. Entretanto, ao invés de descrever o HMM em termos da probabilidade que ele atribui a cada seqüência, pode ser mais conveniente enxergá-lo como uma máquina capaz de gerar seqüências por um processo aleatório.

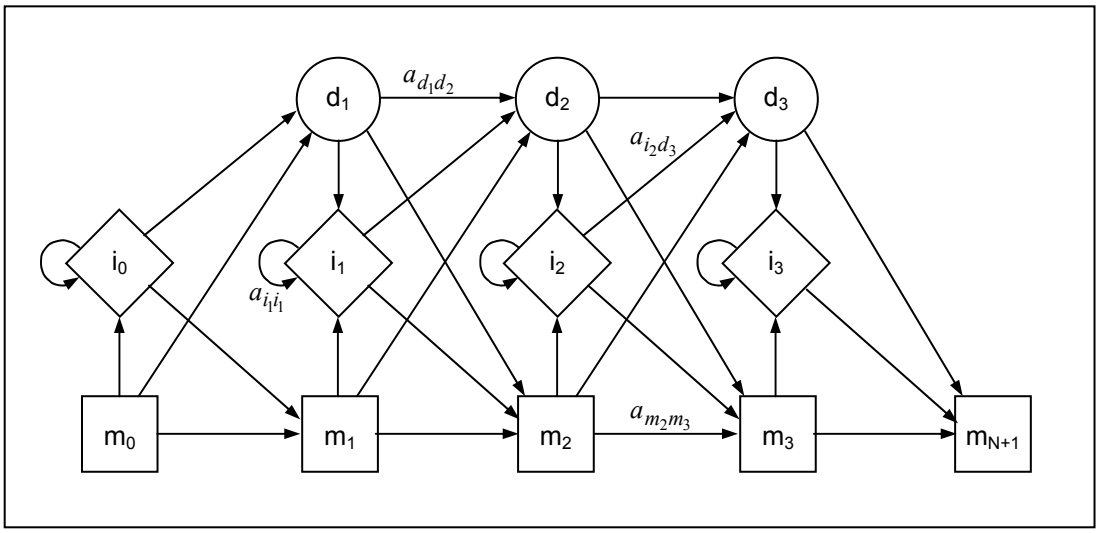


Figura 5. 2 Um HMM com backbone de tamanho 3.

A geração de uma seqüência ocorre a partir de uma 'caminhada' aleatória pelo modelo da seguinte maneira: considere estar, no tempo t=0, no estado m_0 (*inicial*). De acordo com as probabilidades de transição $a_{m_0m_1}$, $a_{m_0i_0}$ e $a_{m_0d_1}$, escolha aleatoriamente o próximo estado[2]. Se m_1 é escolhido, caminhe para m_1 no tempo t=1 e gere o aminoácido O_1, de acordo com as probabilidades de emissão $b_{m_1}(v_m)$. Escolha, aleatoriamente, o próximo estado, de acordo com as probabilidades $a_{m_1m_2}$, $a_{m_1i_1}$ e $a_{m_1d_2}$. Mude para este próximo estado em t=2 e emita o símbolo O_2, de acordo com a probabilidade de emissão do estado ou simplesmente não emita qualquer símbolo, se o estado escolhido for um *delete*[3]. Continue caminhando pelo modelo até atingir o estado m_{N+1}. Como resultado, através do caminho $Q = \{q_0, q_1, ..., q_{N+1}\}$, onde $q_0 = m_0$ e $q_{N+1} = m_{N+1}$, terá sido gerada uma seqüência de aminoácidos $O = O_1 O_2 ... O_L$.

---

[2] Estas probabilidades substituem a distribuição de estados inicial $\Pi = \{\pi_j\}$

[3] Esta é outra diferença entre este modelo aqui apresentado e o modelo geral apresentado no Capítulo anterior (Figura 3.5).

Conforme visto no capítulo anterior, para o caso geral de HMMs, a probabilidade de que um caminho *Q* seja tomado e a seqüência *O* seja gerada, dado o modelo, é:

$$P(O, Q|\lambda) = \pi_{q_1} . b_{q_1}(O_1) . a_{q_1 q_2} . b_{q_2}(O_2) . a_{q_2 q_3} \ldots a_{q_{T-1} q_T} . b_{q_T}(O_T) \qquad (5.1)$$

Para este HMM, temos que considerar que:

(1) Existe um único estado inicial m_0, o qual não emite símbolos. A partir deste estado inicial, o estado seguinte é escolhido de acordo com as probabilidades de transição. Portanto, a distribuição de probabilidades inicial π pode ser vista como a distribuição de probabilidades de transição a partir do estado inicial, isto é, π_{q_1} em (5.1) será $a_{m_0 q_1}$, onde q_1 será m_1, i_0 ou d_1.

(2) Nem todos os estados emitem símbolos, portanto, não há, nesta arquitetura, necessariamente, um símbolo emitido em cada *t*, como no caso geral. Aqui, tanto é possível mudar de estado em um determinado tempo *t* e não emitir qualquer símbolo (quando se está em um estado *delete*) como pode-se mudar o tempo e emitir um símbolo sem ter que mudar de estado (como quando ocorre a transição $i_j \rightarrow i_j$). Por esta razão, serão utilizados os índices ***j*** ou ***k*** para indicar um estado específico do modelo, o índice ***t*** para indicar o tempo e o índice ***s*** para indicar um símbolo específico da seqüência de observações.

Portanto, nesta arquitetura, a probabilidade de que um caminho *Q* seja tomado e a seqüência $O = O_1 O_2 \ldots O_L$ seja gerada, dado o modelo λ, de uma família de proteínas, será

$$P(O, Q|\lambda) = a_{q_N m_{N+1}} \prod_{j=1}^{N} a_{q_{j-1} q_j} . b_{q_j}(O_s), \qquad (5.2)$$

onde $b_{q_j}(O_s)$ é a probabilidade do símbolo O_s ser emitido no estado q_j e $b_{q_j}(O_s) = 1$ sempre que q_j for um estado *delete*.

A probabilidade de uma seqüência de aminoácidos *O*, dado o modelo, será a soma das probabilidades para todos os caminhos possíveis:

$$P(O|\lambda) = \sum_{\text{caminhos}} P(O, Q|\lambda). \qquad (5.3)$$

Esta arquitetura adotada possui a vantagem de, mesmo sendo simples, capturar as características estruturais de uma proteína:

(1) Uma seqüência de símbolos, na qual cada posição pode ter uma distribuição distinta sobre os aminoácidos;

(2) Admite a existência de remoções e de inserções em qualquer posição da seqüência, inclusive consecutivas, retratando o cenário evolucionário que envolve as famílias de proteínas;

(3) A possibilidade de considerar que continuar numa remoção ou numa inserção é mais provável do que começar uma nova.

5.2 Construindo um HMM a partir de um alinhamento de seqüências

Um perfil $\wp$ de comprimento L é um conjunto das probabilidades $b_j(v)$ de se observar o símbolo v na j-ésima posição. A probabilidade de uma seqüência $O = \{O_1\ O_2\ ...\ O_L\}$, dado o perfil $\wp$, será:

$$P(O \mid \wp) = \prod_{j=1}^{L} b_j(O_j) \tag{5.4}$$

Pode-se calcular o escore da probabilidade para um alinhamento sem gaps de O com o perfil $\wp$:

$$Score(O \mid \wp) = \sum_{j=1}^{L} \log \frac{b_j(O_j)}{p(O_j)}, \tag{5.5}$$

onde $p(O_j)$ é a freqüência conhecida de ocorrências do símbolo O_j, independente da posição.

O HMM apresentado, como já foi dito, é um tipo de perfil generalizado, que possui estados *insert* e *delete*, formando várias camadas de 3 estados. A atribuição de probabilidades para as transições que entram ou que saem de estados *insert* corresponde à aplicação de uma função afim de penalidades de *gap*. Por exemplo, um *gap* de comprimento *h*, considerando o escore logarítmico da probabilidade, teria o seguinte custo (Figura 5. 3):

$$\underbrace{\log(a_{m_j i_j}) + \log(a_{i_j m_{j+1}})}_{\text{criação do gap}} + \underbrace{(h-1)\,.\quad \log(a_{i_j i_j})}_{\text{extensão do gap}}$$

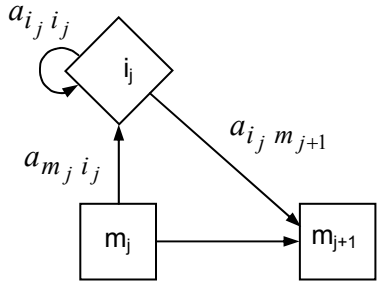


Figura 5. 3 Custo de inserções num HMM.

Já para o caso de uma remoção de comprimento h, a função custo não é afim, uma vez que a remoção ocorre caminhando-se, através do modelo, pelos estados *delete*, cujas probabilidades de transição não são, necessariamente, iguais.

Os parâmetros de um HMM podem ser estimados manualmente, a partir de um alinhamento múltiplo de seqüências, chamado **alinhamento de treinamento**. Cada probabilidade a_{jk} de transição de um estado j para um estado k será igual ao número de vezes, A_{jk}, que a transição $j \rightarrow k$ ocorre, dividido pelo número total de transições, A_{jq}, que ocorrem a partir do estado j:

$$a_{jk} = \frac{A_{jk}}{\sum_{q \in Q} A_{jq}} \tag{5.6}$$

Da mesma forma, as probabilidades de emissão podem ser estimadas da seguinte maneira:

$$b_k(v) = \frac{E_k(v)}{\sum_{x \in \Sigma} E_k(x)}, \tag{5.7}$$

onde $E_k(v)$ é o número de vezes que o símbolo v aparece na coluna k do alinhamento, e $\sum_{x \in \Sigma} E_k(x)$ é o número total de símbolos que aparecem na coluna.

1. Probabilidades de emissão (coluna 1):

v	$E_1(v)$	correção de Laplace	$E_1(v)$ corrigido	$b_1(v)$
A	0	+1	1	1/27
C	0	+1	1	1/27
D	0	+1	1	1/27
E	0	+1	1	1/27
F	1	+1	2	2/27
G	0	+1	1	1/27
H	0	+1	1	1/27
I	1	+1	2	2/27
K	0	+1	1	1/27
L	0	+1	1	1/27
M	0	+1	1	1/27
N	0	+1	1	1/27
P	0	+1	1	1/27
Q	0	+1	1	1/27
R	0	+1	1	1/27
S	0	+1	1	1/27
T	0	+1	1	1/27
V	5	+1	6	6/27
W	0	+1	1	1/27
Y	0	+1	1	1/27
Total	7	20	27	27/27

Alinhamento

	1 2 3	4 5 6 7 8
HBA_HUMAN	. . . V G A	- - H A G E Y . . .
HBB_HUMAN	. . . V	- - - - N V D E V . . .
MYG_PHYCA	. . . V E A	- - D V A G H . . .
GLB3_CHITP	. . . V K G	- - - - - - D . . .
GLB5_PETMA	. . . V Y S	- - N I P K H . . .
LGB2_LUPLU	. . . F N A	- - N I P K H . . .
GLB1_GLYDI	. . . I A G	A D N G A G V . . .
	* * *	* * * * *

2. Probabilidades de transição (col. 1 para col. 2):

$m_1 \rightarrow$ l	$A_{m_1 l}$	correção de Laplace	$A_{m_1 l}$ corrigido	$a_{m_1 l}$
$m_1 \rightarrow m_2$	6	+1	7	7/10
$m_1 \rightarrow i_1$	0	+1	1	1/10
$m_1 \rightarrow d2$	1	+1	2	2/10
Total	7	3	10	10/10

Figura 5. 4 Parte de um alinhamento múltiplo de sete seqüências de globina. As colunas marcadas com asterisco são tratadas como '*matches*' num perfil HMM.

Como exemplo, apresenta-se um alinhamento na Figura 5. 4 com as estimativas de probabilidades de emissão para a coluna 1 e de transição da coluna 1 para a coluna 2.

Um dos problemas que podem surgir, quando o número de seqüências no alinhamento de treinamento é muito pequeno, é que algumas transições ou emissões podem não ocorrer em sequer uma seqüência deste alinhamento, fazendo com que elas sejam estimadas em zero. Neste caso, pode-se usar algum tipo de correção para evitar o zero, como por exemplo a correção de Laplace, segundo a qual soma-se um a cada freqüência no alinhamento.

Neste trabalho, ao invés de se alinhar as seqüências de treinamento para, a partir do alinhamento, estimarem-se os parâmetros do HMM, a abordagem utilizad7a foi a aprendizagem automática dos parâmetros a partir de um conjunto de seqüências não alinhadas.

5.3 Os Algoritmos

Os algoritmos apresentados no capítulo anterior, para HMMs gerais, serão novamente mostrados, focalizando, agora, o caso particular do modelo. Será perseguida a resolução dos 3 problemas básicos: (i) determinar a probabilidade de que uma seqüência tenha sido gerada pelo modelo, ou melhor, de que a seqüência pertença à família modelada pelo HMM; (ii) encontrar o caminho mais provável pelo qual a seqüência é gerada pelo modelo; isto equivale a alinhar uma seqüência ao HMM, determinando portanto a posição de cada símbolo da seqüência em relação ao modelo; (iii) estimar os parâmetros mais adequados para o HMM.

5.3.1 Algoritmo *forward-backward*

No capítulo anterior, a variável *forward* foi definida como sendo a probabilidade $\alpha_t(j)$, do prefixo $O_1\, O_2 \ldots O_t$, atingindo o estado $q_t = j$. Considerando que agora os índices da seqüência $O = O_1\, O_2 \ldots O_S \ldots O_L$ não são mais coincidentes com o tempo t=1, 2, ..., T, ao invés de se calcular os valores de α para cada t, eles serão calculados para cada símbolo observado:

$$\alpha_s(j) = P\left(O_1\, O_2 \ldots O_s\, , q_s = j \,|\, \lambda\right) \;, \tag{5.8}$$

como sendo a probabilidade de emitir o prefixo $O_1\, O_2 \ldots O_s$, atingindo o estado $q_s = j$. Uma vez que o processo começa sempre no estado inicial m_0, e que este não emite símbolo, a inicialização do algoritmo seria:

$$\begin{aligned} \alpha_0(m_0) &= 1 \\ \forall j \neq m_0\, , \alpha_0(j) &= 0 \end{aligned} \tag{5.9}$$

O passo da indução seria, então:

$$\alpha_{s+1}(k) = \left[\sum_{j \in Q} \alpha_s(j)\ a_{jk}\right] b_k(O_{s+1}) \tag{5.10}$$

e o término do processo, com o estado *final* = m_{N+1} :

$$P(O) = \sum_{j \in Q} \alpha_L(j)\ .\ a_{j m_{N+1}} \tag{5.11}$$

Particularizando ainda mais, serão considerados em separado os estados que emitem símbolos (*match* e *insert*) dos que não emitem (*delete*). Neste modelo, cada estado recebe, no máximo, 3 transições, a partir de 3 estados, de modo que os estados predecessores de um estado *match* m_j serão m_{j-1}, i_{j-1} e d_{j-1}; os predecessores de um estado *insert* i_j são os três estados da mesma camada m_j, d_j e o próprio i_j; finalmente, os predecessores de um estado *delete* d_j, serão os três estados da camada anterior m_{j-1}, i_{j-1} e d_{j-1}. Exceções a estas regras acontecem nas pontas do modelo, as quais serão tratadas separadamente.

Assim, dada uma seqüência de observações $O_1\ O_2\ ...\ O_L$, as probabilidades *forward* serão dadas por:

$$\begin{aligned}
&\alpha_s(m_j) = P(O_1\ O_2\ ...\ O_s \quad \text{com } O_s \text{ sendo emitido no estado match } m_j) \\
&\alpha_s(i_j) = P(O_1\ O_2\ ...\ O_s \quad \text{com } O_s \text{ sendo emitido no estado insert } i_j) \\
&\alpha_s(d_j) = P(O_1\ O_2\ ...\ O_s \quad \text{terminando no estado } d_j \text{ sem emitir qualquer símbolo})
\end{aligned} \quad ,$$

e o algoritmo:

Parte 1 – Forward

(1) Inicialização:

(1.1) caso base:

$$\begin{aligned}
&\alpha_0(m_0) = 1 \\
\forall j \neq 0, \quad &\alpha_0(m_j) = 0 \\
\forall j, \quad &\alpha_0(i_j) = 0 \\
\forall s \neq 0, \quad &\alpha_s(m_0) = 0
\end{aligned}$$

(1.2) os estados delete são casos especiais, os quais são calculados para cada j :

$$\alpha_0(d_1) = \alpha_0(m_0)\ .\ a_{m_0 d_1} \quad \Rightarrow \quad \alpha_0(d_1) = a_{m_0 d_1}$$

para $2 \leq j \leq N$ calcular: $\alpha_0(d_j) = \alpha_0(d_{j-1})\ .\ a_{d_{j-1} d_j}$

(1.3) na ponta anterior do modelo também existe um caso especial, que é o do estado i_0:

Para cada $1 \leq s \leq L$, calcular:

$$\alpha_s(i_0) = b_{i_0}(O_s)\left[\alpha_{s-1}(m_0) \,.\, a_{m_0 i_0} + \alpha_{s-1}(i_0) \,.\, a_{i_0 i_0}\right]$$

(2) Indução:

(2.1) Para $j = 1$

para cada $1 \leq s \leq L$:

$$\begin{cases} \alpha_s(m_1) = b_{m_1}(O_s) \,.\, [\alpha_{s-1}(m_0) \,.\, a_{m_0 m_1} + \alpha_{s-1}(i_0) \,.\, a_{i_0 m_1}] \\ \alpha_s(i_1) = b_{i_1}(O_s) \,.\, [\alpha_{s-1}(m_1) \,.\, a_{m_1 i_1} + \alpha_{s-1}(i_1) \,.\, a_{i_1 i_1} + \alpha_{s-1}(d_1) \,.\, a_{d_1 i_1}\ ] \\ \alpha_s(d_1) = [\alpha_s(m_0) \,.\, a_{m_0 d_1} + \alpha_s(i_0) \,.\, a_{i_0 d_1}\ ] \end{cases}$$

(2.2) Para cada $1 < j \leq N$, calcular, recursivamente,

para cada $1 \leq s \leq L$:

$$\begin{cases} \alpha_s(m_j) = b_{m_j}(O_s) \,.\, [\alpha_{s-1}(m_{j-1}) \,.\, a_{m_{j-1} m_j} + \alpha_{s-1}(i_{j-1}) \,.\, a_{i_{j-1} m_j} + \alpha_{s-1}(d_{j-1}) \,.\, a_{d_{j-1} m_j}] \\ \alpha_s(i_j) = b_{i_j}(O_s) \,.\, [\alpha_{s-1}(m_j) \,.\, a_{m_j i_j} + \alpha_{s-1}(i_j) \,.\, a_{i_j i_j} + \alpha_{s-1}(d_j) \,.\, a_{d_j i_j}\ ] \\ \alpha_s(d_j) = [\alpha_s(m_{j-1}) \,.\, a_{m_{j-1} d_j} + \alpha_s(i_{j-1}) \,.\, a_{i_{j-1} d_j} + \alpha_s(d_{j-1}) \,.\, a_{d_{j-1} d_j}] \end{cases}$$

(3) Finalização – probabilidade forward da seqüência:

$$P(O) = \alpha_L(m_N) \,.\, a_{m_N m_{N+1}} + \alpha_L(i_N) \,.\, a_{i_N m_{N+1}} + \alpha_L(d_N) \,.\, a_{d_N m_{N+1}}$$

Para evitar problemas de *underflow*, pode-se trabalhar com escores logarítmicos, transformando os produtos em somas. Desta maneira, ao invés de estar calculando o valor de cada variável α, calcula-se $\log\alpha$ e portanto, para que o valor seja reutilizado, aplica-se a função exponencial.

O logaritmo de uma soma de probabilidades calculado a partir dos logaritmos das probabilidades pode ser calculado da seguinte maneira:

$$\log(p+q) = \log(\exp(\log p) + \exp(\log q)), \qquad (5.12)$$

Assim, $\alpha_s(j)$ será definida formalmente como o logaritmo da probabilidade de emitir o prefixo $(O_1\ O_2\ \ldots O_s)$, dado que $q_s = j$. A parte *forward* do algoritmo ficará:

Probabilidade *Forward* Logarítmica

(1) Inicialização:

(1.1) caso base:

$$\alpha_0(m) = 0$$
$$\forall j \neq 0, \quad \alpha_0(m_j) = -\infty$$
$$\forall j, \quad \alpha_0(i_j) = -\infty$$
$$\forall s \neq 0, \quad \alpha_s(m_0) = -\infty$$

(1.2) estados delete:

$$\alpha_0(d_1) = \log(a_{m_0 d_1})$$

para $2 \leq j \leq N$ calcular: $\alpha_0(d_j) = \log\left(\exp(\alpha_0(d_{j-1})) . a_{d_{j-1} d_j}\right)$

(1.3) i_0:

Para cada $1 \leq s < L$, calcular:

$$\alpha_s(i_0) = \log\left[b_{i_0}(O_s) . \left[\exp(\alpha_{s-1}(m_0)) . a_{m_0 i_0} + \exp(\alpha_{s-1}(i_0)) . a_{i_0 i_0}\right]\right]$$

(2) Indução:

(2.1) Para $j = 1$

para cada $1 \leq s \leq L$:

$$\begin{cases} \alpha_s(m_1) = \log(b_{m_1}(O_s)) + \log\left[\exp(\alpha_{s-1}(m_0)) . a_{m_0 m_1} + \exp(\alpha_{s-1}(i_0)) . a_{i_0 m_1}\right] \\ \alpha_s(i_1) = \log(b_{i_1}(O_s)) + \log\left[\exp(\alpha_{s-1}(m_1)) . a_{m_1 i_1} + \exp(\alpha_{s-1}(i_1)) . a_{i_1 i_1} + \exp(\alpha_{s-1}(d_1)) . a_{d_1 i_1}\right] \\ \alpha_s(d_1) = \log\left[\exp(\alpha_s(m_0)) . a_{m_0 d_1} + \exp(\alpha_s(i_0)) . a_{i_0 d_1}\right] \end{cases}$$

(2.2) Para cada $1 < j \leq N$,

para cada $1 \leq s \leq L$, calcular, recursivamente,:

$$\begin{cases} \alpha_s(m_j) = \log(b_{m_j}(O_s)) + \log\left[\exp(\alpha_{s-1}(m_{j-1})) . a_{m_{j-1} m_j} + \exp(\alpha_{s-1}(i_{j-1})) . a_{i_{j-1} m_j} + \exp(\alpha_{s-1}(d_{j-1})) . a_{d_{j-1} m_j}\right] \\ \alpha_s(i_j) = \log(b_{i_j}(O_s)) + \log\left[\exp(\alpha_{s-1}(m_j)) . a_{m_j i_j} + \exp(\alpha_{s-1}(i_j)) . a_{i_j i_j} + \exp(\alpha_{s-1}(d_j)) . a_{d_j i_j}\right] \\ \alpha_s(d_j) = \log\left[\exp(\alpha_s(m_{j-1})) . a_{m_{j-1} d_j} + \exp(\alpha_s(i_{j-1})) . a_{i_{j-1} d_j} + \exp(\alpha_s(d_{j-1})) . a_{d_{j-1} d_j}\right] \end{cases}$$

(3) Finalização:

$$P(O) = \log\left[\exp(\alpha_L(m_N)) . a_{m_N m_{N+1}} + \exp(\alpha_L(i_N)) . a_{i_N m_{N+1}} + \exp(\alpha_L(d_N)) . a_{d_N m_{N+1}}\right]$$

O mesmo raciocínio é válido para as variáveis *backward*, onde $\beta_s(j) = P(O_{s+1} \ldots O_L, q_s = j)$ é a probabilidade da seqüência parcial de observações de $s+1$ até L, considerando que O_1 até O_s foi emitida até o estado j. Desta forma, a parte *backward* do algoritmo ficará:

Parte 2 - *Backward*

(1) Inicialização:

(1.1) última camada do modelo, para o último símbolo da cadeia:

$$\begin{cases} \beta_L(m_N) = a_{m_N m_{N+1}} \\ \beta_L(i_N) = a_{i_N m_{N+1}} \\ \beta_L(d_N) = a_{d_N m_{N+1}} \end{cases}$$

(1.2) última camada do modelo, para $L > s \geq 0$:

$$\begin{cases} \beta_s(m_N) = \beta_{s+1}(i_N) \,.\, a_{m_N i_N} \,.\, b_{i_N}(O_{s+1}) \\ \beta_s(i_N) = \beta_{s+1}(i_N) \,.\, a_{i_N i_N} \,.\, b_{i_N}(O_{s+1}) \\ \beta_s(d_N) = \beta_{s+1}(i_N) \,.\, a_{d_N i_N} \,.\, b_{i_N}(O_{s+1}) \end{cases}$$

(1.3) último símbolo da cadeia, para $N-1 \geq j \geq 0$:

$$\begin{cases} \beta_L(m_j) = \beta_L(d_{j+1}) \,.\, a_{m_j d_{j+1}} \\ \beta_L(i_j) = \beta_L(d_{j+1}) \,.\, a_{i_j d_{j+1}} \\ \beta_L(d_j) = \beta_L(d_{j+1}) \,.\, a_{d_j d_{j+1}} \end{cases}$$

(2) Indução:

Para cada estado $N > j \geq 1$

para cada $L > s \geq 0$ calcular, recursivamente:

$$\begin{cases} \beta_s(m_j) = \beta_{s+1}(m_{j+1}).a_{m_j m_{j+1}} .b_{m_{j+1}}(O_{s+1}) + \beta_{s+1}(i_j).a_{m_j i_j} .b_{i_j}(O_{s+1}) + \beta_s(d_{j+1}).a_{m_j d_{j+1}} \\ \beta_s(i_j) = \beta_{s+1}(m_{j+1}).a_{i_j m_{j+1}} .b_{m_{j+1}}(O_{s+1}) + \beta_{s+1}(i_j).a_{i_j i_j} .b_{i_j}(O_{s+1}) + \beta_s(d_{j+1}).a_{i_j d_{j+1}} \\ \beta_s(d_j) = \beta_{s+1}(m_{j+1}).a_{d_j m_{j+1}} .b_{m_{j+1}}(O_{s+1}) + \beta_{s+1}(i_j).a_{d_j i_j} .b_{i_j}(O_{s+1}) + \beta_s(d_{j+1}).a_{d_j d_{j+1}} \end{cases}$$

(3) Finalização:

(3.1) estado i_0 para $L > s \geq 0$

$$\beta_s(i_0) = \beta_{s+1}(m_1).a_{i_0 m_1} .b_{m_1}(O_{s+1}) + \beta_{s+1}(i_0).a_{i_0 i_0} .b_{i_0}(O_{s+1}) + \beta_s(d_1).a_{i_0 d_1}$$

(3.2) probabilidade backward da seqüência

$$P(O) = \beta_1(m_1) \,.\, a_{m_0 m_1}.b_{m_1}(O_1) + \beta_1(i_0) \,.\, a_{m_0 i_0}.b_{i_0}(O_1) + \beta_0(d_1) \,.\, a_{m_0 d_1}$$

Da mesma maneira que com as variáveis *forward*, as fórmulas acima serão transformadas para o espaço logarítmico:

Probabilidade *Backward* Logarítmica

(1) Inicialização:

(1.1) última camada do modelo, para o último símbolo da cadeia:

$$\begin{cases} \beta_L(m_N) = \log(a_{m_N m_{N+1}}) \\ \beta_L(i_N) = \log(a_{i_N m_{N+1}}) \\ \beta_L(d_N) = \log(a_{d_N m_{N+1}}) \end{cases}$$

(1.2) última camada do modelo, para $L > s \geq 0$:

$$\begin{cases} \beta_s(m_N) = \log[\exp(\beta_{s+1}(i_N)) . a_{m_N i_N} . b_{i_N}(O_{s+1})] \\ \beta_s(i_N) = \log[\exp(\beta_{s+1}(i_N)) . a_{i_N i_N} . b_{i_N}(O_{s+1})] \\ \beta_s(d_N) = \log[\exp(\beta_{s+1}(i_N)) . a_{d_N i_N} . b_{i_N}(O_{s+1})] \end{cases}$$

(1.3) último símbolo da cadeia, para $N - 1 \geq j \geq 0$:

$$\begin{cases} \beta_L(m_j) = \log[\exp(\beta_L(d_{j+1}) . a_{m_j d_{j+1}})] \\ \beta_L(i_j) = \log[\exp(\beta_L(d_{j+1}) . a_{i_j d_{j+1}}] \\ \beta_L(d_j) = \log[\exp(\beta_L(d_{j+1}) . a_{d_j d_{j+1}}] \end{cases}$$

(2) Indução:

Para cada estado $N > j \geq 1$

para cada $L > s \geq 0$ calcular recursivamente:

$$\begin{cases} \beta_s(m_j) = \log[\exp(\beta_{s+1}(m_{j+1})) . a_{m_j m_{j+1}} . b_{m_{j+1}}(O_{s+1}) + \exp(\beta_{s+1}(i_j)) . a_{m_j i_j} . b_{i_j}(O_{s+1}) + \exp(\beta_s(d_{j+1})) . a_{m_j d_{j+1}}] \\ \beta_s(i_j) = \log[\exp(\beta_{s+1}(m_{j+1})) . a_{i_j m_{j+1}} . b_{m_{j+1}}(O_{s+1}) + \exp(\beta_{s+1}(i_j)) . a_{i_j i_j} . b_{i_j}(O_{s+1}) + \exp(\beta_s(d_{j+1})) . a_{i_j d_{j+1}}] \\ \beta_s(d_j) = \log[\exp(\beta_{s+1}(m_{j+1})) . a_{d_j m_{j+1}} . b_{m_{j+1}}(O_{s+1}) + \exp(\beta_{s+1}(i_j)) . a_{d_j i_j} . b_{i_j}(O_{s+1}) + \exp(\beta_s(d_{j+1})) . a_{d_j d_{j+1}}] \end{cases}$$

(3) Finalização:

(3.1) estado i_0 para $L > s \geq 0$

$$\beta_s(i_0) = \log[\exp(\beta_{s+1}(m_1)) . a_{i_0 m_1} . b_{m_1}(O_{s+1}) + \exp(\beta_{s+1}(i_0)) . a_{i_0 i_0} . b_{i_0}(O_{s+1}) + \exp(\beta_s(d_1)) . a_{i_0 d_1}]$$

(3.2) probabilidade backward da seqüência

$$P(O) = \log[\exp(\beta_1(m_1)) . a_{m_0 m_1} . b_{m_1}(O_1) + \exp(\beta_1(i_0)) . a_{m_0 i_0} . b_{i_0}(O_1) + \exp(\beta_0(d_1)) . a_{m_0 d_1}]$$

Alinhando uma seqüência a um perfil HMM

De modo semelhante às variáveis *forward-backward*, será considerado aqui o caso especial da arquitetura adotada. Seja a seqüência de observações $O = \{O_1 O_2 \ldots O_L\}$ e um HMM = λ. Para cada $1 \leq j \leq N$ e $1 \leq s \leq L$, sejam as seguintes definições:

(1) Seja $\delta_s(m_j)$ a probabilidade do caminho mais provável para a seqüência $O = \{O_1\ O_2 \ldots O_s\}$, com O_s sendo emitido pelo estado m_j.

(2) Seja $\delta_s(i_j)$ a probabilidade do caminho mais provável para a seqüência, $O = \{O_1\ O_2 \ldots O_s\}$, com O_s sendo emitido pelo estado i_j.

(3) Seja $\delta_s(d_j)$ a probabilidade do caminho mais provável para a seqüência $O = \{O_1\ O_2 \ldots O_s\}$, terminando no estado d_j, sem emitir qualquer símbolo.

Um *array* – ψ – será utilizado para marcar o caminho ótimo.

Algoritmo de Viterbi

(1) Inicialização:

(1.1) caso base:

$$\begin{aligned} & \delta_0(m_0) = 1 \\ \forall j \neq 0, \quad & \delta_0(m_j) = 0 \\ \forall j, \quad & \delta_0(i_j) = 0 \\ \forall s \neq 0, \quad & \delta_s(m_0) = 0 \end{aligned}$$

(1.2) os estados delete são casos especiais, os quais calculam-se para cada j :

$$\delta_0(d_1) = \delta_0(m_0).\, a_{m_0 d_1} \ \Rightarrow\ \delta_0(d_1) = a_{m_0 d_1}$$

para $2 \le j \le N$ calcular : $\quad \delta_0(d_j) = \delta_0(d_{j-1}).\, a_{d_{j-1} d_j}$

(1.3) estado i_0

Para cada $1 \le s \le L$, calcular:

$$\delta_s(i_0) = b_{i_0}(O_s).\max\begin{cases}\delta_{s-1}(m_0).\, a_{m_0 i_0} \\ \delta_{s-1}(i_0).\, a_{i_0 i_0}\end{cases} \qquad \psi_s(i_0) = \arg\max\begin{cases}\delta_{s-1}(m_0).\, a_{m_0 i_0} \\ \delta_{s-1}(i_0).\, a_{i_0 i_0}\end{cases}$$

(2) Indução – Calcular, recursivamente:

(2.1) para $j = 1$

para cada $1 \le s \le L$:

$\delta_s(m_1) =$	$b_{m_1}(O_s).$	$\max\begin{cases}\delta_{s-1}(m_0).\, a_{m_0 m_1} \\ \delta_{s-1}(i_0).\, a_{i_0 m_1}\end{cases}$	$\psi_s(m_1) =$	$\arg\max\begin{cases}\delta_{s-1}(m_0).\, a_{m_0 m_1} \\ \delta_{s-1}(i_0).\, a_{i_0 m_1}\end{cases}$
$\delta_s(i_1) =$	$b_{i_1}(O_s).$	$\max\begin{cases}\delta_{s-1}(m_1).\, a_{m_1 i_1} \\ \delta_{s-1}(i_1).\, a_{i_1 i_1} \\ \delta_{s-1}(d_1).\, a_{d_1 i_1}\end{cases}$	$\psi_s(i_1) =$	$\arg\max\begin{cases}\delta_{s-1}(m_1).\, a_{m_1 i_1} \\ \delta_{s-1}(i_1).\, a_{i_1 i_1} \\ \delta_{s-1}(d_1).\, a_{d_1 i_1}\end{cases}$
$\delta_s(d_1) =$		$\max\begin{cases}\delta_s(m_0).\, a_{m_0 d_1} \\ \delta_s(i_0).\, a_{i_0 d_1}\end{cases}$	$\psi_s(d_1) =$	$\arg\max\begin{cases}\delta_s(m_0).\, a_{m_0 d_1} \\ \delta_s(i_0).\, a_{i_0 d_1}\end{cases}$

(2.2) para cada $1 < j \leq N$

para cada $1 \leq s \leq L$:

$$\delta_s(m_j) = b_{m_j}(O_s) \cdot \max \begin{cases} \delta_{s-1}(m_{j-1}) \cdot a_{m_{j-1}m_j} \\ \delta_{s-1}(i_{j-1}) \cdot a_{i_{j-1}m_j} \\ \delta_{s-1}(d_{j-1}) \cdot a_{d_{j-1}m_j} \end{cases} \qquad \psi_s(m_j) = \arg\max \begin{cases} \delta_{s-1}(m_{j-1}) \cdot a_{m_{j-1}m_j} \\ \delta_{s-1}(i_{j-1}) \cdot a_{i_{j-1}m_j} \\ \delta_{s-1}(d_{j-1}) \cdot a_{d_{j-1}m_j} \end{cases}$$

$$\delta_s(i_j) = b_{i_j}(O_s) \cdot \max \begin{cases} \delta_{s-1}(m_j) \cdot a_{m_j i_j} \\ \delta_{s-1}(i_j) \cdot a_{i_j i_j} \\ \delta_{s-1}(d_j) \cdot a_{d_j i_j} \end{cases} \qquad \psi_s(i_j) = \arg\max \begin{cases} \delta_{s-1}(m_j) \cdot a_{m_j i_j} \\ \delta_{s-1}(i_j) \cdot a_{i_j i_j} \\ \delta_{s-1}(d_j) \cdot a_{d_j i_j} \end{cases}$$

$$\delta_s(d_j) = \max \begin{cases} \delta_s(m_{j-1}) \cdot a_{m_{j-1}d_j} \\ \delta_s(i_{j-1}) \cdot a_{i_{j-1}d_j} \\ \delta_s(d_{j-1}) \cdot a_{d_{j-1}d_j} \end{cases} \qquad \psi_s(d_j) = \arg\max \begin{cases} \delta_s(m_{j-1}) \cdot a_{m_{j-1}d_j} \\ \delta_s(i_{j-1}) \cdot a_{i_{j-1}d_j} \\ \delta_s(d_{j-1}) \cdot a_{d_{j-1}d_j} \end{cases}$$

(3) Finalização – probabilidade do caminho mais provável:

$$P(O|Q^*) = \max \begin{cases} \delta_L(m_N) \cdot a_{m_N m_{N+1}} \\ \delta_L(i_N) \cdot a_{i_N m_{N+1}} \\ \delta_L(d_N) \cdot a_{d_N m_{N+1}} \end{cases} \qquad q_L^* = \arg\max \begin{cases} \delta_L(m_N) \cdot a_{m_N m_{N+1}} \\ \delta_L(i_N) \cdot a_{i_N m_{N+1}} \\ \delta_L(d_N) \cdot a_{d_N m_{N+1}} \end{cases}$$

(4) Caminho ótimo: Para cada $L > s \geq 1$:

$$q_s^* = \psi_{s+1}(q_{s+1}^*)$$

Mais uma vez, para evitar *underflow*, $\delta_s(j)$ será o escore logarítmico do caminho mais provável para o prefixo $(O_1\ O_2\ \ldots O_s)$ que termina no estado *j*. O algoritmo será então:

Algoritmo de Viterbi – escore logarítmico do caminho mais provável

(1) Inicialização:

(1.1) caso base:

$$\begin{aligned} \delta_0(m_0) &= 0 \\ \forall j \neq 0, \quad \delta_0(m_j) &= -\infty \\ \forall j, \quad \delta_0(i_j) &= -\infty \\ \forall s \neq 0, \quad \delta_s(m_0) &= -\infty \end{aligned}$$

(1.2) estados delete :

$$\delta_0(d_1) = \delta_0(m_0) + \log(a_{m_0 d_1}) \implies \delta_0(d_1) = \log(a_{m_0 d_1})$$

para $2 \leq j \leq N$ calcular: $\delta_0(d_j) = \delta_0(d_{j-1}) + \log(a_{d_{j-1}d_j})$

(1.3) estado i_0.

Para cada $1 \le s \le L$, calcular:

$$\delta_s(i_0) = \log\left[b_{i_0}(O_s)\right] + \max\begin{cases}\delta_{s-1}(m_0) + \log(a_{m_0 i_0}) \\ \delta_{s-1}(i_0) + \log(a_{i_0 i_0})\end{cases}$$

(2) Indução – Calcular, recursivamente:

(2.1) para $j = 1$

para cada $1 \le s \le L$:

$$\begin{array}{lll}
\delta_s(m_1) = & \log(b_{m_1}(O_s)). & \max\begin{cases}\delta_{s-1}(m_0) + \log(a_{m_0 m_1}) \\ \delta_{s-1}(i_0) + \log(a_{i_0 m_1})\end{cases} \\
\hline
\delta_s(i_1) = & \log(b_{i_1}(O_s)). & \max\begin{cases}\delta_{s-1}(m_1) + \log(a_{m_1 i_1}) \\ \delta_{s-1}(i_1) + \log(a_{i_1 i_1}) \\ \delta_{s-1}(d_1) + \log(a_{d_1 i_1})\end{cases} \\
\hline
\delta_s(d_1) = & & \max\begin{cases}\delta_s(m_0) + \log(a_{m_0 d_1}) \\ \delta_s(i_0) + \log(a_{i_0 d_1})\end{cases}
\end{array}$$

(2.2) para cada $1 \le j \le N$

para cada $1 \le s \le L$:

$$\begin{array}{lll}
\delta_s(m_j) = & \log(b_{m_j}(O_s)). & \max\begin{cases}\delta_{s-1}(m_{j-1}) + \log(a_{m_{j-1} m_j}) \\ \delta_{s-1}(i_{j-1}) + \log(a_{i_{j-1} m_j}) \\ \delta_{s-1}(d_{j-1}) + \log(a_{d_{j-1} m_j})\end{cases} \\
\hline
\delta_s(i_j) = & \log(b_{i_j}(O_s)). & \max\begin{cases}\delta_{s-1}(m_j) + \log(a_{m_j i_j}) \\ \delta_{s-1}(i_j) + \log(a_{i_j i_j}) \\ \delta_{s-1}(d_j) + \log(a_{d_j i_j})\end{cases} \\
\hline
\delta_s(d_j) = & & \max\begin{cases}\delta_s(m_{j-1}) + \log(a_{m_{j-1} d_j}) \\ \delta_s(i_{j-1}) + \log(a_{i_{j-1} d_j}) \\ \delta_s(d_{j-1}) + \log(a_{d_{j-1} d_j})\end{cases}
\end{array}$$

(3) Finalização:

$$P(O|Q^*) = \max\begin{cases}\delta_L(m_N) + \log(a_{m_N m_{N+1}}) \\ \delta_L(i_N) + \log(a_{i_N m_{N+1}}) \\ \delta_L(d_N) + \log(a_{d_N m_{N+1}})\end{cases}$$

5.3.2 Algoritmo Baum-Welch – Equações de reestimativa

Para modelar uma família de proteínas é necessário que o HMM seja treinado a partir de várias seqüências que sabidamente pertençam à família. Desta maneira, devem ser utilizadas as

fórmulas 3.36 e 3.37. Fazendo-se as devidas conformações à arquitetura proposta, o algoritmo ficará como segue:

Algoritmo Baum-Welch

(1) Inicialização

(1.1) Atribuir valores iniciais para as probabilidades de emissão $\{b_j(n)\}$.

(1.2) Atribuir valores iniciais para as probabilidades de transição $\{a_{ij}\}$.

(2) Expectativa

(2.1) Calcular o número de vezes esperado que cada símbolo *s* da seqüência apareça em cada coluna *j* do modelo, considerando a distribuição atual $\{b_j(n)\}$.

Para $e = m$; $e = i$

Para cada $1 \le j \le N$

$$E_{e_j}(n) = \sum_{k=1}^{K} [\frac{1}{P(O^k)} \sum_{\substack{s=1 \\ O_s = n}}^{L} \alpha_s(e_j).\beta_s(e_j)]$$

Obs.: Devem ser calculadas as expectativas também para o estado i_0:

$$E_{i_0}(n) = \sum_{k=1}^{K} [\frac{1}{P(O^k)} \sum_{\substack{s=1 \\ O_s = n}}^{L} \alpha_s(i_0).\beta_s(i_0)]$$

(2.2) Calcular o número de vezes esperado que ocorra a transição a_{ij}, para cada par *i-j* , considerando a distribuição atual $\{a_{ij}\}$.

Para $e = m$; $e = i$ e $e = d$

Para cada $0 \le j \le N-1$, calcular:

$$A_{e_j m_{j+1}} = \sum_{k=1}^{K} [\frac{1}{P(O^{(k)})} \sum_{s=1}^{L} \alpha_s(e_j) \,.\, a_{e_j m_{j+1}} \,.\, b_{s+1}(m_{j+1}) \,.\, \beta_{s+1}(m_{j+1})]$$

$$A_{e_j i_j} = \sum_{k=1}^{K} [\frac{1}{P(O^{(k)})} \sum_{s=1}^{L} \alpha_s(e_j) \,.\, a_{e_j i_j} \,.\, b_{s+1}(i_j) \,.\, \beta_{s+1}(i_j)]$$

$$A_{e_j d_{j+1}} = \sum_{k=1}^{K} [\frac{1}{P(O^{(k)})} \sum_{s=1}^{L} \alpha_s(e_j) \,.\, a_{e_j d_{j+1}} \,.\, \beta_s(d_{j+1})]$$

Obs.: Alguns aspectos específicos da arquitetura proposta devem ser considerados:

1. Não existe o estado d_0;

2. Devem ser acrescidos aos somatórios, os valores calculados para s=0, para as transições a partir de m_0 , e para as transições a partir dos estados delete, pois apenas nestes é possível ainda não ter emitido nenhum símbolo da seqüência ($\alpha_0 \neq 0$).
3. Devem ser calculados os valores relativos às transições que começam na última camada do modelo (N), observando que só existem aí transições para i_N e para o estado final m_{N+1}.
4. No caso das transições para o estado final, observar que este último estado é mudo, e toda seqüência já deverá ter sido gerada antes que o estado final seja atingido, de modo que as equações terão a forma:

$$A_{e_N m_{N+1}} = \sum_{k=1}^{K}\Big[\frac{1}{P(O^{(k)})}\sum_{s=1}^{L}\alpha_s(e_N)\,.\,a_{e_N m_{N+1}}\Big]$$

5. Após o último símbolo da cadeia haver sido emitido, os próximos estados a serem percorridos até que o estado final seja atingido terão que ser estados *delete*.

(2.3) Calcular o fator de normalização *G*

Para $e = m$; $e = i$; $e = d$

Para cada $0 \le j \le N$ (se $e_j \neq d_0$)

$$G_{e_j} = \sum_{k=1}^{K}\Big[\frac{1}{P(O^k)}\sum_{s=1}^{L}\alpha_s(e_j).\beta_s(e_j)\Big]$$

(3) Maximização

(3.1) Calcular os novos valores (melhorados) para as probabilidades $\{b_j(n)\}$

Para $e = m$; $e = i$

Para cada $1 \le j \le N$

$$\bar{b}_{e_j}(n) = \frac{\sum_{k=1}^{K}\Big[\frac{1}{P(O^k)}\sum_{\substack{s=1\\ O_s=n}}^{L}\alpha_s(e_j).\beta_s(e_j)\Big]}{\sum_{k=1}^{K}\Big[\frac{1}{P(O^k)}\sum_{s=1}^{L}\alpha_s(e_j).\beta_s(e_j)\Big]}$$

Obs.: Devem ser calculadas as expectativas também para o estado i_0:

$$\bar{b}_{i_0}(n) = \frac{\sum_{k=1}^{K}\Big[\frac{1}{P(O^k)}\sum_{\substack{s=1\\ O_s=n}}^{L}\alpha_s(i_0).\beta_s(i_0)\Big]}{\sum_{k=1}^{K}\Big[\frac{1}{P(O^k)}\sum_{s=1}^{L}\alpha_s(i_0).\beta_s(i_0)\Big]}$$

(3.2) Calcular os novos valores (melhorados) para as probabilidades $\{a_{ij}\}$

(3.2.1) Para $e = m$; $e = i$; $e = j$

Para cada $0 \le j \le N-1$, calcular:

$$\overline{a}_{e_j m_{j+1}} = \frac{\sum_{k=1}^{K} [\frac{1}{P(O^{(k)})} \sum_{s=1}^{L} \alpha_s(e_j) \cdot a_{e_j m_{j+1}} \cdot b_{s+1}(m_{j+1}) \cdot \beta_{s+1}(m_{j+1})]}{\sum_{k=1}^{K} [\frac{1}{P(O^k)} \sum_{s=1}^{L} \alpha_s(e_j) . \beta_s(e_j)]}$$

$$\overline{a}_{e_j i_j} = \frac{\sum_{k=1}^{K} [\frac{1}{P(O^{(k)})} \sum_{s=1}^{L} \alpha_s(e_j) \cdot a_{e_j i_j} \cdot b_{s+1}(i_j) \cdot \beta_{s+1}(i_j)]}{\sum_{k=1}^{K} [\frac{1}{P(O^k)} \sum_{s=1}^{L} \alpha_s(e_j) . \beta_s(e_j)]}$$

$$\overline{a}_{e_j d_{j+1}} = \frac{\sum_{k=1}^{K} [\frac{1}{P(O^{(k)})} \sum_{s=1}^{L} \alpha_s(e_j) \cdot a_{e_j d_{j+1}} \cdot \beta_s(d_{j+1})]}{\sum_{k=1}^{K} [\frac{1}{P(O^k)} \sum_{s=1}^{L} \alpha_s(e_j) . \beta_s(e_j)]}$$

Os mesmos cuidados citados no passo (2) devem ser tomados aqui, observando-se as particularidades da arquitetura em questão, mais especificamente, das pontas do modelo.

(3.2.2) As transições para o estado final serão:

Para $e = m$; $e = i$; $e = d$

$$\overline{a}_{e_N m_{N+1}} = \frac{\sum_{k=1}^{K} [\frac{1}{P(O^{(k)})} \sum_{s=1}^{L} \alpha_s(e_N) \cdot a_{e_N m_{N+1}}]}{\sum_{k=1}^{K} [\frac{1}{P(O^k)} \sum_{s=1}^{L} \alpha_s(e_j) . \beta_s(e_j)]}$$

(4) Verificação

Repetir os passos (2) e (3) até que a melhoria da probabilidade do modelo, para as seqüências de treinamento seja menor que um parâmetro ε:

Voltar para (2) enquanto $P(O \mid \lambda) = \prod_{k=1}^{K} P(O^{(k)} \mid \lambda) > \varepsilon$

Capítulo 6

Experimentos com Famílias de Proteínas

Para aplicar a técnica apresentada nos capítulos anteriores, foram realizados experimentos com três famílias reais de proteínas, a saber, globinas, proteinoquinases e GTPases. Para cada uma das famílias, um HMM foi treinado a partir de seqüências da família, não alinhadas, encontradas no banco de dados SWISSPROT [47]. Foi utilizado um modelo inicial homogêneo que não continha conhecimento inicial sobre as famílias.

Após o treinamento, outro conjunto de seqüências, composto tanto de seqüências também pertencentes, quanto de não pertencentes à família, foi apresentado ao respectivo modelo (globina, proteinoquinase, GTP). Os resultados encontrados mostram que os HMMs construídos são capazes de classificar as seqüências apresentadas aos modelos como membros ou não-membros das famílias.

Os experimentos foram realizados em um microcomputador com processador AMD K7 de 800 Mhz, e 128 MB de RAM.

6.1 Experimentos com globinas

A primeira família utilizada para a construção de um HMM foi a das globinas. As globinas compõem uma família de proteínas envolvidas no armazenamento e transporte de oxigênio que possuem estados oligométricos e arquiteturais gerais diferentes. Algumas globinas, como as Hemoglobinas, são compostas de duas cadeias α e outras duas subunidades (geralmente β, γ, δ ou θ), outras, como as Mioglobinas, são cadeias simples.

Cada cadeia simples de globina possui um sítio de ligação de oxigênio. Assim, enquanto em muitos vermes marinhos, insetos e peixes primitivos, uma cadeia polipeptídica de globina é a responsável pelo carreamento do oxigênio, nos vertebrados maiores a oxigenação é feita por moléculas similares à hemoglobina, que com suas quatro cadeias consegue transportar quatro moléculas de oxigênio.

A família das globinas consiste de um conjunto de proteínas que possuem a mesma estrutura tridimensional global, mas seqüências altamente divergentes. O comprimento das seqüências varia entre 130 e 170 resíduos (com poucas exceções).

6.1.1 O treinamento do modelo

O HMM para as globinas foi treinado com um conjunto de seqüências de globinas não alinhadas, extraídas do SWISSPROT. Foram realizados diversos treinamentos, considerando-se modelos de tamanhos diferentes e com diferentes números de iterações. Cada iteração durou, em média, 6,2 minutos.

O conjunto de treinamento

Inicialmente, foram retiradas do diretório *special selections*, da base de dados do SWISSPROT, versão 40.28, as seqüências que continham a palavra-chave '*globin*' ou '*Globin*' no campo de descrição (`DES`), formando o **conjunto das globinas**. Neste procedimento foram encontradas 878 seqüências, das quais foram eliminados os falsos positivos, as seqüências repetidas, os fragmentos de seqüências, as seqüências com caracteres não pertencentes ao alfabeto e as seqüências com comprimento muito diferente do comprimento típico, restando 810 seqüências de globinas. Destas, foram escolhidas, aleatoriamente, 539 seqüências para compor o conjunto de treinamento. As demais foram reservadas para testar o modelo com seqüências de globinas que não tivessem sido utilizadas no processo de treinamento.

As seqüências foram apresentadas ao modelo desalinhadas, o que significa que um perfil estatístico foi construído durante o alinhamento das seqüências, de modo que as penalidades foram aprendidas dos próprios dados, ao contrário do que ocorre com métodos convencionais em que as penalidades são definidas antes da construção propriamente dita do alinhamento.

Parâmetros iniciais

Não obstante os parâmetros serem aprendidos dos dados, é necessário que sejam propostos parâmetros iniciais para o processo de aprendizagem. Para as probabilidades de emissão e transição, foram considerados valores iguais, respeitando-se a restrição estocástica. Assim, para as probabilidades de emissão, foi utilizado o valor $b_j(v) = \frac{1}{20} = 0.05$ para quaisquer estados *match* ou *insert*. Da mesma forma, para as probabilidades de transição foi utilizado o valor $a_{jk} = 0{,}33\ldots$, para todas as transições, exceto nas extremidades do modelo, onde de cada estado só saem duas transições, as quais foram iniciadas com $0{,}5$. As probabilidades iniciais resumem-se, portanto, em:

$$\begin{cases} b_{m_j}(v) = 0.05 & 1 \le j \le N\ ;\ \forall\ v \\ b_{i_j}(v) = 0.05 & 0 \le j \le N\ ;\ \forall\ v \\ a_{X_0 i_0} = 0.33 & X = m \text{ ou } i \\ a_{X_{j-1} Y_j} = 0.33 & 1 \le j \le N\ ;\ X = m,\ i \text{ ou } d\ ;\ Y = m,\ i \text{ ou } d \\ a_{X_{N-1} i_N} = 0.33 & X = m,\ i \text{ ou } d \\ a_{X_N m_{N+1}} = 0.33 & X = m,\ i \text{ ou } d \end{cases}$$

Os modelos encontrados

Foram realizados cinco treinamentos utilizando-se tamanhos de modelo diferentes, conforme a Tabela 6. 1. Em cada treinamento foi construído um modelo com um escore logarítmico, encontrado da seguinte forma (considerando que P é o logaritmo negativo da probabilidade do conjunto de seqüências O, dado o modelo λ):

$P(O \mid \lambda) = \sum_{k=1}^{K} P(O^{(k)} \mid \lambda)$, com o número de seqüências K = 539.

A Tabela 6. 1 mostra o escore encontrado para cada modelo construído cujo treinamento atingiu 500 iterações.

tamanho (N)	escore
143	105775,30
144	105105,37
145	105196,44
146	105737,25
147	106343.03

Tabela 6. 1 Resultados dos treinamentos de HMM's para globinas.

6.1.2 Testes de classificação realizados

Para testar a qualidade dos modelos encontrados, foram-lhes apresentados conjuntos de seqüências membros e não-membros da família. O conjunto de seqüências não membros, ou **conjunto de não_globinas,** foi inicialmente formado pelo complemento do conjunto das globinas, extraídas do diretório *special_selections* do SWISSPROT. O **conjunto de testes** foi composto das globinas que não compunham o conjunto de treinamento. Do conjunto de não_globinas, foram desprezadas as seqüências que continham caracteres não pertencentes ao alfabeto, seqüências muito longas e repetições. Após o tratamento das seqüências, os conjuntos ficaram com os seguintes tamanhos.

- **conjunto de globinas de teste** – 271 seqüências;
- **conjunto de não globinas** – aproximadamente 40000 seqüências;
- **conjunto de treinamento** – o mesmo conjunto usado no treinamento também foi apresentado ao modelo com o objetivo de comparar resultados com os outros conjuntos – 539 seqüências.

Foram gerados gráficos de dispersão do comprimento das seqüências *versus* escores logarítmicos das probabilidades para todos os modelos, os quais podem ser encontrados no Apêndice A, com exceção daquele do melhor modelo (N=144), cujo gráfico é mostrado na Figura 6. 1.

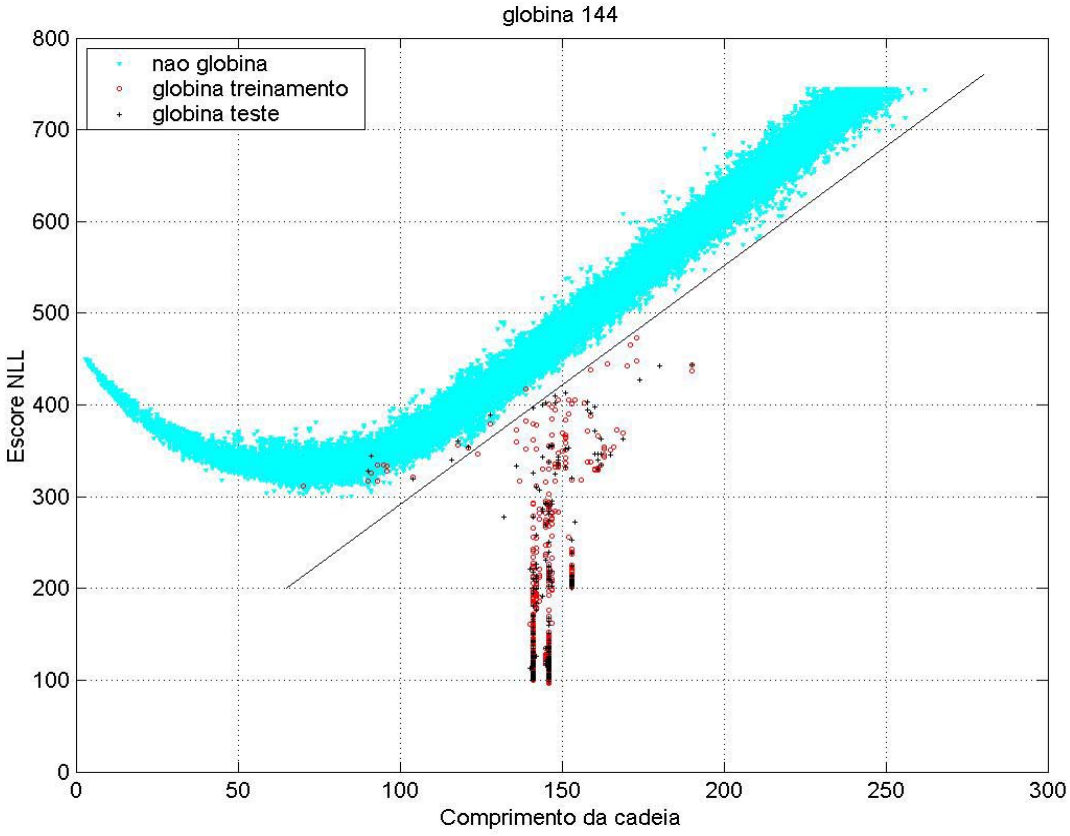


Figura 6. 1 Gráfico de dispersão *Escores NLL* x *Comprimento* para o HMM N=144 das globinas

Os gráficos gerados são semelhantes àquele encontrado por Krogh [39] para a família das globinas, mostrado na Figura 6. 2. A maioria das seqüências tende a posicionar-se ao longo de uma curva com um trecho inicial semelhante a um pedaço de parábola, seguido de um trecho linear, indicando que a partir de um certo limiar, os escores para essas seqüências são proporcionais aos seus comprimentos. Estas são as seqüências que não pertencem à família e portanto parecem aleatórias ao modelo da família das globinas.

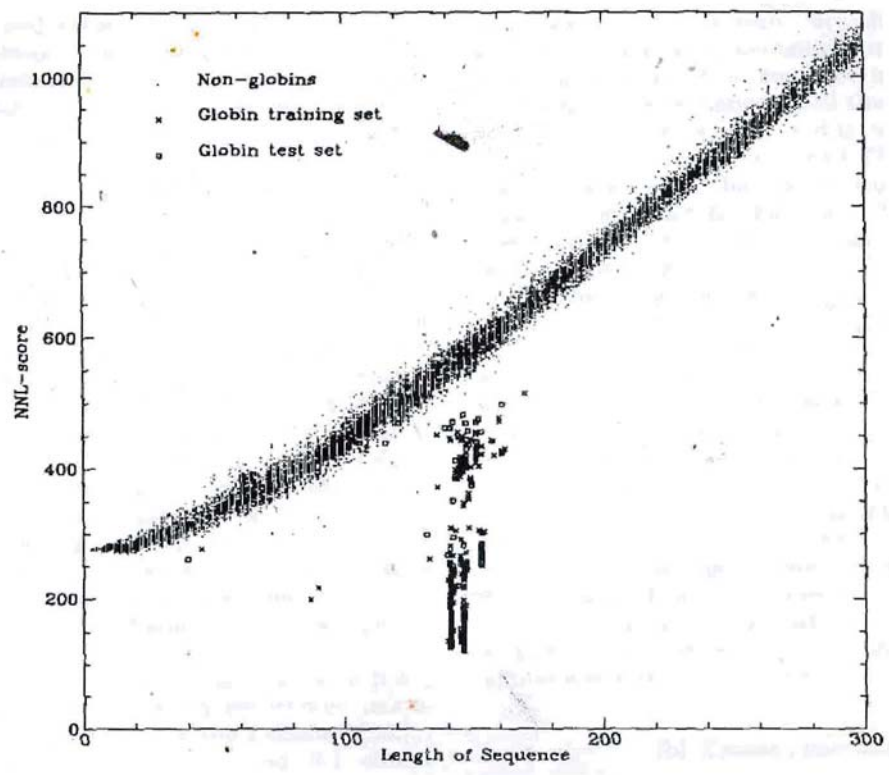


Figura 6. 2 Gráfico de dispersão dos escores NLL *versus* comprimento de seqüências para globinas e não-globinas, extraído de [39].

Foi traçada uma reta aproximadamente paralela ao trecho linear do gráfico plotado, com o intuito de dividi-lo em duas regiões distintas, separando assim seqüências de membros da família de globinas de seqüências de não-membros. Avaliando desta maneira os resultados encontrados, pode-se quantificar em:

- 0 % de seqüências não-globinas, classificadas pelo modelo como globinas (falsos positivos), e
- 3,3210 % de seqüências do conjunto de testes de globinas, classificadas como não-globinas (falsos negativos).

Para as seqüências do conjunto de treinamento, o percentual de falsos negativos foi de 2,4119 %. Nas tabelas 6.2 e 6.3, aparecem, respectivamente, as seqüências de globinas do conjunto de treinamento e de testes que não foram devidamente classificadas. Pode-se perceber que a maioria delas é descrita como precursor e/ou secretoglobin ou ainda secretory protein. Uma análise mais

apurada sobre a estrutura e funcionalidade destas seqüências não classificadas (falsos negativos) pode ajudar a esclarecer os motivos pelos quais elas não foram devidamente classificadas.

GLOBINAS - CONJUNTO DE TREINAMENTO					
Identificação				**DE**scrição	
ID	GLBN_NOSSN	STANDARD;	PRT; 118 AA.	DE	Cyanoglobin.
ID	GLBO_MYCTU	STANDARD;	PRT; 128 AA.	DE	Hemoglobin-like protein HbO.
ID	LPPA_HUMAN	STANDARD;	PRT; 90 AA.	DE	Lipophilin A precursor (Secretoglobin family 1D member 1).
ID	MGBA_HUMAN	STANDARD;	PRT; 93 AA.	DE	Mammaglobin A precursor (Mammaglobin 1) (Secretoglobin family 2A DE member 2).
ID	MGBB_HUMAN	STANDARD;	PRT; 95 AA.	DE	Mammaglobin B precursor (Mammaglobin 2) (Lipophilin C) (Lacryglobin). DE (Secretoglobin family 2A member 1).
ID	UGR1_HUMAN	STANDARD;	PRT; 93 AA.	DE	Uteroglobin-related protein 1 precursor (Secretoglobin family 3A DE member 2).
ID	UGR1_MOUSE	STANDARD;	PRT; 139 AA.	DE	Uteroglobin-related protein 1 precursor (Secretoglobin family 3A DE member 2).
ID	UGR2_MOUSE	STANDARD;	PRT; 104 AA.	DE	Uteroglobin-related protein 2 precursor (Cytokine HIN-1) (High in DE normal-1) (Secretoglobin family 3A member 1).
ID	UTER_HUMAN	STANDARD;	PRT; 91 AA.	DE	Clara cell phospholipid-binding protein precursor (CCPBP) (Clara cells DE 10 kDa secretory protein) (CC10) (Uteroglobin) (Urine protein 1) DE (UP1).
ID	UTER_MACFU	STANDARD;	PRT; 70 AA.	DE	Clara cell phospholipid-binding protein (CCPBP) (Clara cells 10 kDa DE secretory protein) (CC10) (Uteroglobin).
ID	UTER_MOUSE	STANDARD;	PRT; 96 AA.	DE	Clara cell phospholipid-binding protein precursor (CCPBP) (Clara cells DE 10 kDa secretory protein) (CC10) (Uteroglobin) (PCB-binding protein) DE (Clara cell 17 kDa protein).
ID	UTER_RAT	STANDARD;	PRT; 96 AA.	DE	Clara cell phospholipid-binding protein precursor (CCPBP) (Clara cells DE 10 kDa secretory protein) (CC10) (Uteroglobin) (PCB-binding protein).
ID	GLB_TETPY	STANDARD;	PRT; 121 AA.	DE	Myoglobin (Hemoglobin).

Tabela 6. 2 Seqüências de globinas do conjunto de treinamento que não foram classificadas corretamente (falsos negativos)

GLOBINAS - CONJUNTO DE TESTE					
Identificação				**DE**scrição	
ID	HBAD_PASMO	STANDARD;	PRT; 141 AA.	DE	Hemoglobin alpha-D chain.
ID	GLBN_NOSCO	STANDARD;	PRT; 118 AA.	DE	Cyanoglobin.
ID	HBAD_TURME	STANDARD;	PRT; 141 AA.	DE	Hemoglobin alpha-D chain.
ID	GLBO_MYCLE	STANDARD;	PRT; 128 AA.	DE	Hemoglobin-like protein HbO.
ID	LPPB_HUMAN	STANDARD;	PRT; 90 AA.	DE	Lipophilin B precursor (Secretoglobin family 1D member 2).
ID	UGR2_HUMAN	STANDARD;	PRT; 104 AA.	DE	Uteroglobin-related protein 2 precursor (Cytokine HIN-1) (High in DE normal-1) (Secretoglobin family 3A member 1).
ID	UTER_LEPCA	STANDARD;	PRT; 91 AA.	DE	Uteroglobin precursor (Blastokinin).
ID	UTER_RABIT	STANDARD;	PRT; 91 AA.	DE	Uteroglobin precursor (Blastokinin).

Tabela 6. 3 Seqüências de globina do conjunto de teste que não foram classificadas corretamente (falsos negativos)

6.2 Experimentos com Proteinoquinases

A atividade de algumas proteínas é regulada, principalmente, por uma infinidade de pequenas moléculas que quando se ligam a essas proteínas provocam uma alteração em sua conformação, acoplando dois sítios de ligação distantes. Este tipo de acoplamento é chamado de **alosteria** e por isso, diz-se que uma proteína que é regulada desta forma sofre uma **transição alostérica**. Em outras proteínas, o controle das atividades é desempenhado pela ligação de um grupo fosfato a uma cadeia lateral de aminoácido que provoca a alteração estrutural da proteína. Este fenômeno, chamado de **fosforilação**, é o tipo de controle predominante das atividades de células eucarióticas. Os fosfatos são transferidos de moléculas de ATP por **proteinoquinases**.

As proteinoquinases compõem uma vasta família de enzimas que contém o domínio catalítico **quinase**. Para os experimentos com esta família, visando a simplificação dos trabalhos, foram selecionadas algumas proteínas, com comprimentos em torno de 190 aminoácidos, conforme se vê mais adiante. Nas diversas seqüências da família, trechos anteriores e posteriores ao domínio são, geralmente, não conservados. Freqüentemente, existem também pequenas seqüências de aminoácidos distintas inseridas em alças dentro do domínio. Apesar destes trechos não conservados, o HMM construído mostrou trabalhar bem na classificação de proteinoquinases.

6.2.1 O treinamento do modelo

O treinamento do modelo para as proteinoquinases foi feito de modo semelhante ao das globinas. As seqüências para o treinamento foram apanhadas no SWISSPROT e apresentadas ao modelo desalinhadas. Foram treinados cinco modelos com comprimentos variando de 189 a 193. Cada treinamento foi realizado com 500 iterações que duraram cerca de 6,4 minutos cada.

O conjunto de treinamento

Inicialmente, foram retiradas do diretório *special selections*, da base de dados do SWISSPROT, versão 40.28, as seqüências que continham a palavra-chave '*kinase*' no campo de descrição (`DES`), formando o **conjunto das proteinoquinases**. Deste procedimento resultou um arquivo com 3576 seqüências, o qual recebeu o mesmo tratamento dado ao arquivo das globinas. Para restringir mais o conjunto das proteinoquinases, foram selecionadas especificamente as proteínas:

2-amino-4-hydroxy-6-hydroxymethyldihydropteridine pyrophosphokinase
Adenylate kinase
Adenylylsulfate kinase

Cyclin-dependent kinase
Cytidylate kinase
Dephospho-CoA kinase
Guanylate kinase
Nucleoside diphosphate kinase
Shikimate kinase
Thymidylate kinase
Uridine kinase
Uridylate kinase

O conjunto das proteinoquinases ficou então com 515 seqüências, das quais foram selecionadas, aleatoriamente, 344 seqüências para comporem o conjunto de treinamento. As 271 restantes foram reservadas para testar o modelo.

Parâmetros iniciais

Foram utilizados os mesmos parâmetros iniciais utilizados para o treinamento das globinas.

Os modelos encontrados

Os escores logarítmicos encontrados para os modelos estão colocados na tabela abaixo:

tamanho (N)	Escore
189	142522,00
190	142510,51
191	141694,91
192	142252,81
193	141722,60

Tabela 6. 4 Resultados dos treinamentos de HMM's para proteinoquinases.

6.2.2 Testes de classificação realizados

Após o treinamento dos modelos foi aplicado o procedimento de classificação aos conjuntos de seqüências:

- **conjunto de proteinoquinases de teste** – 171 seqüências;
- **conjunto de não globinas** – aproximadamente 40000 seqüências;
- **conjunto de treinamento** – 344 seqüências.

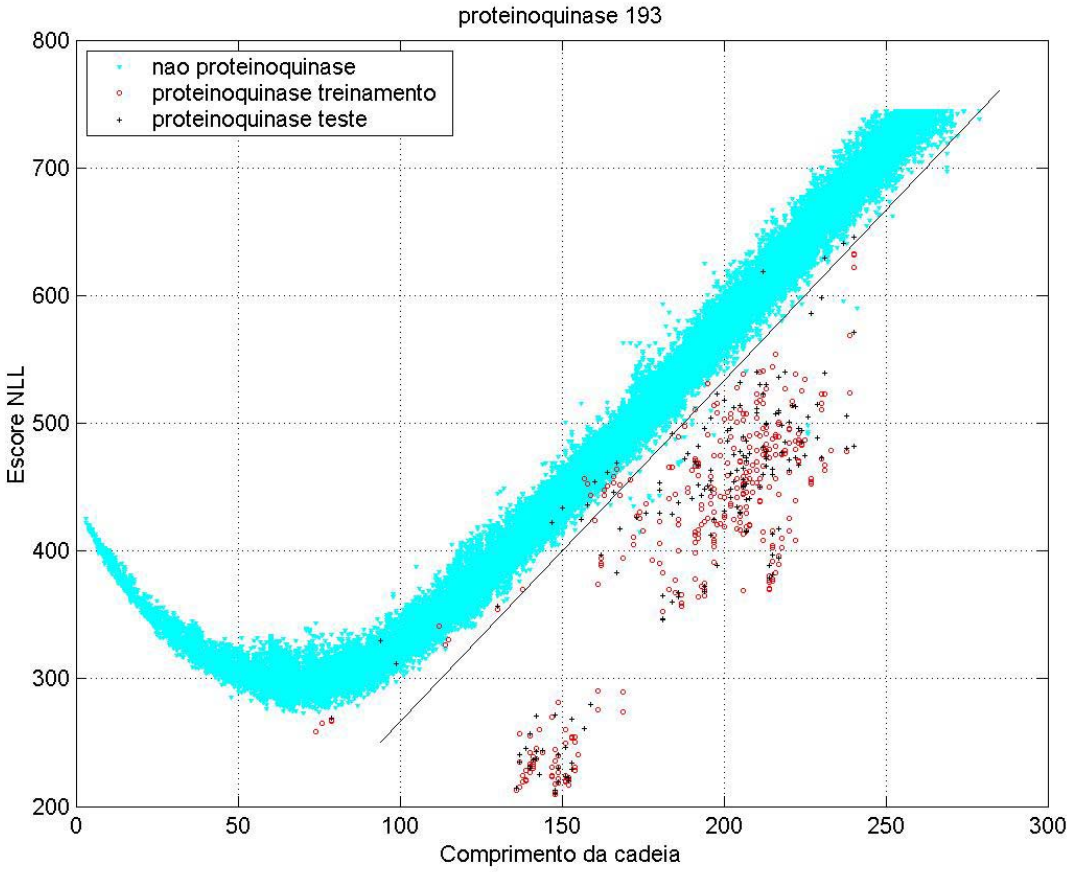


Figura 6. 3 Gráfico de dispersão *Escores NLL* x *Comprimento* para o HMM de comprimento 193 das proteinoquinases

Os resultados encontrados para os vários modelos podem ser vistos nos gráficos de dispersão do Apêndice A e no gráfico da Figura 6. 3 (gráfico para o modelo N=193). Para este último modelo, obteve-se:

- 0,1828 % de seqüências não-proteinoquinases, classificadas pelo modelo como proteinoquinases (falsos positivos), e
- 8,7719 % de seqüências de proteinoquinases do conjunto de testes, classificadas como não-proteinoquinases (falsos negativos).
- 4,9419 % de seqüências de proteinoquinases do conjunto de treinamento, classificadas como não-proteinoquinases (falsos negativos).

De todas as seqüências de proteinoquinases dos conjuntos de treinamento e testes, as que não foram devidamente classificadas são aquelas descritas como ‘pyrophosphokinase’, ‘Cyclin dependent’ ou ‘Uridylate kinase’.

Os resultados para esta família foram inferiores aos encontrados para a família das globinas. Pode-se atribuir isto ao tamanho muito maior desta família (uma variedade muito maior

de seqüências) ou mesmo a particularidades da estrutura das proteínas que a compõem, mas seria necessário mais investigação a respeito. Pode ser mais adequado para uma família tão grande como a das proteinoquinases construir um modelo apenas para o domínio catalítico quinase, ao invés de se trabalhar com as seqüências inteiras. Isso será possível com pequenas alterações na arquitetura do modelo utilizado como será apontado no Capítulo 7.

6.3 Experimentos com GTPases

A outra família utilizada para a construção de um HMM foi a das GTPases – proteínas ligadoras de GTP. O GTP é o principal nucleotídeo trifosfato utilizado na síntese de RNA e em algumas reações de transferência de energia e possui uma função especial na síntese de proteínas e sinalização celular. Como foi visto na seção anterior, a adição de grupos fosfato a uma proteína, ou sua remoção pode ser usada pela célula para controlar a atividade desta proteína. No caso das proteinoquinases, um grupo fosfato de moléculas de ATP liga-se a uma cadeia lateral de aminoácidos. Em outros casos, o fosfato pode não estar ligado diretamente à proteína, antes ele é parte do nucleotídeo de guanina GTP que se liga fortemente à proteína.

Proteínas ligadoras de GTP, ou **GTPases**, constituem uma ampla família de proteínas que possuem um domínio similar de ligação de GTP.

6.3.1 O treinamento do modelo

Os procedimentos para o treinamento foram semelhantes aos adotados para as famílias anteriores. Foram treinados modelos com comprimentos de 196 a 200 AA, conforme a Tabela 6. 5. Cada iteração durou, em média, 2,9 minutos.

O conjunto de treinamento

Inicialmente, foram retiradas do mesmo diretório *special selections*, (SWISSPROT 40.28), as seqüências que continham as palavras-chave (‘*GTP*’ **e** ‘*binding*’) **ou** (‘*GTP*’ **e** ‘*cyclohidrolase*’) **ou** (‘ADP-ribosylation’) no campo de descrição (`DES`), formando o **conjunto das GTPases**. Neste procedimento foram encontradas 433 seqüências. Foram dados a este arquivo os mesmos tratamentos aplicados para as famílias anteriores (eliminação de falso-positivos, repetições, etc.), restando 214 seqüências, das quais 143 ficaram no conjunto de treinamento e 71 no conjunto de testes.

Parâmetros iniciais

Os parâmetros iniciais foram os mesmos utilizados no treinamento das famílias anteriores.

Os modelos encontrados

Após cada treinamento foram encontrados os seguintes modelos e respectivos escores.

tamanho (N)	escore
196	46110,58
197	46974,87
198	46839,59
199	47113,87
200	45626,35

Tabela 6. 5 Resultados dos treinamentos de HMM's para GTPases.

6.3.2 Testes de classificação realizados

Os testes foram realizados com os seguintes conjuntos de seqüências:

- **conjunto de GTPases de teste** – 71 seqüências;
- **conjunto de não GTPases** – aproximadamente 47000 seqüências;
- **conjunto de treinamento** – 143 seqüências.

Foram gerados gráficos de dispersão do comprimento das seqüências *versus* escores logarítmicos das probabilidades, os quais podem ser encontrados no Apêndice A. Na Figura 6. 4 é mostrado o gráfico do melhor modelo encontrado (N=200). Considerando a reta traçada para separar as seqüências de membros das de não-membros, observou-se que as seqüências constantes do conjunto de não-GTPases classificadas como membros da família (as quais seriam falso-positivos), podem ser, de fato, GTPases que não haviam sido detectadas quando da formação do conjunto de não-GTPases. A quantidade de cada uma destas proteínas é mostrada na Tabela 6. 6.

Considerando-se os resultados como estão colocados no gráfico GTPase 200, foram obtidos os seguintes percentuais:

- 0,5703 % de seqüências não-GTPases, classificadas pelo modelo como GTPases (falsos positivos), e
- 1,4084 % de seqüências de GTPases do conjunto de testes, classificadas como não-GTPases (falsos negativos).
- 0 % de seqüências do conjunto de treinamento classificadas indevidamente.

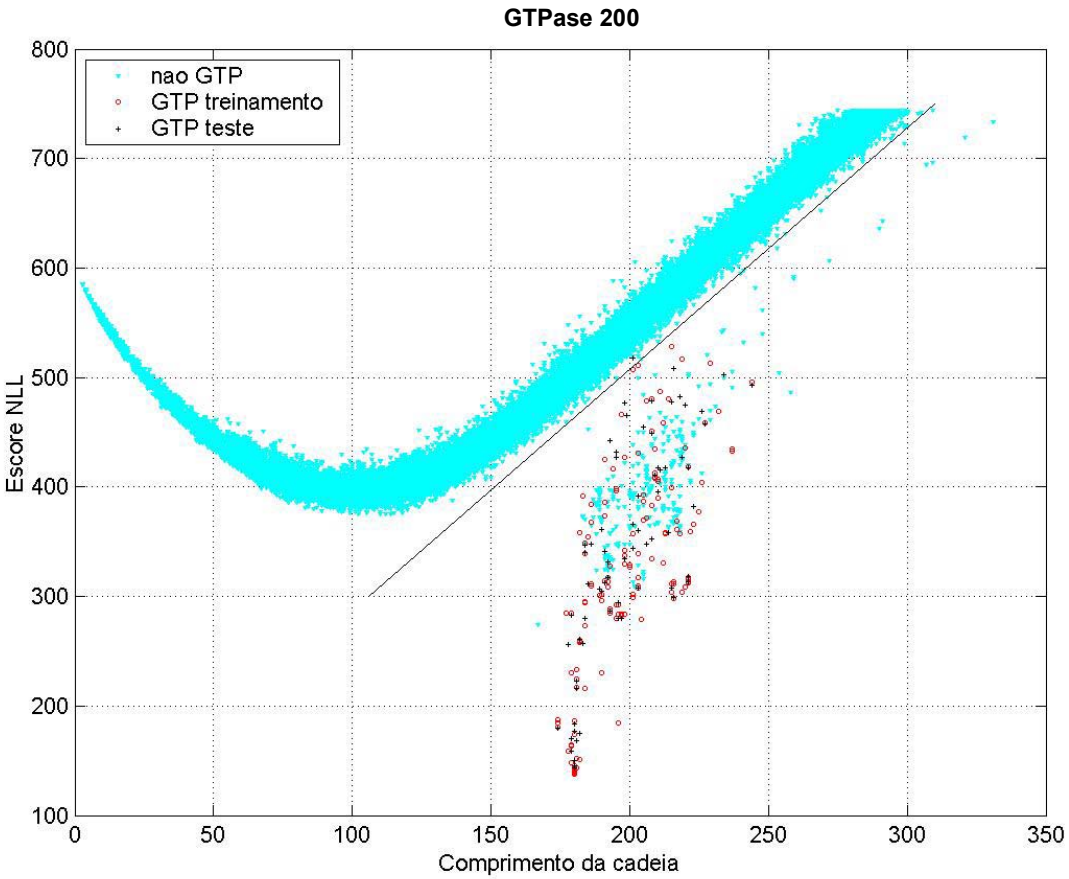


Figura 6. 4 Gráfico de dispersão *Escores NLL* x *Comprimento* para o HMM de comprimento 200 das GTPases

NÃO_GTPASES – CONUNTO DE TESTES	
DE 24 kDa RAS-like protein.	1
DE 30 kDa salivary gland allergen Aed a 3 precursor.	1
DE 60S ribosomal protein L22.	1
DE ARF-related protein (ARP).	2
DE Cdc42 homolog.	1
DE Cell division control protein 42 homolog (CDC42CE).	5
DE High mobility group-T protein (HMG-T) (HMG-T1) (HMG-1).	1
DE Hypothetical protein MJ0920.	2
DE Protamine	3
DE Putative ras-related protein	3
DE Rab-like protein	2
DE Ras-protein.	2
DE Ras-like protein	27
DE Ras-related C3 botulinum toxin substrate	7
DE Ras-related protein	184
DE RHO protein.	7
DE Signal recognition particle receptor beta subunit (SR-beta)	2
DE Spermatid-specific protein T2 [Contains: Sperm protamine SP2].	1
DE Spiralin.	2
DE TEM1 protein.	1
DE Transcription antitermination protein nusG.	1
DE Transforming protein Rho	6
DE Transforming protein Ras	23

Tabela 6. 6 Seqüências constantes do conjunto de não-GTPases classificadas como membros da família (falsos positivos).

No conjunto de testes, apenas uma seqüência de 201 AA não foi classificada corretamente:

ID RANG_HUMAN STANDARD; PRT; 201 AA.
DE Ran-specific GTPase-activating protein (Ran binding protein 1)(RanBP1).

Os gráficos referentes aos demais modelos podem ser vistos no Apêndice A. Os resultados encontrados para essa família foram muito bons, na medida em que apenas uma seqüência do conjunto dos membros não foi classificada como tal. Dentre as seqüências do conjunto de não_GTPases, a maioria é, de alguma forma, relacionada com elas, haja visto que trazem em sua descrição termos como **ras**, **rho**, **rab**. Uma seleção mais cuidadosa dessas seqüências (falsos positivos e falsos negativos) pode ajudar na compreensão desses resultados e até mesmo numa melhoria do modelo, com a ampliação ou redução do conjunto de treinamento.

Capítulo 7

Conclusões e Trabalhos Futuros

A técnica HMM possui um embasamento teórico bem fundamentado matematicamente e permite a extração de informações estatísticas contidas em uma grande quantidade de seqüências de proteínas de uma mesma família. Como foi mostrado neste trabalho, é possível construir modelos de famílias de proteínas a partir de seqüências desalinhadas e sem nenhum conhecimento prévio a respeito delas. Os parâmetros destes modelos são estimados na medida em que eles vão sendo construídos, ao contrário de outras técnicas baseadas em perfis, cujos parâmetros são estimados a partir do conhecimento já existente. Além disso, a técnica permite que penalidades diferentes sejam aplicadas a regiões diferentes da seqüência, o que permite retratar a realidade das proteínas, as quais possuem, via de regra, regiões altamente conservadas, bem como regiões bastante variáveis.

Uma vez que um modelo é construído para uma família, é possível encontrar outros membros desta família em uma base de dados. Para isso, as diversas seqüências são apresentadas ao modelo e a probabilidade, ou melhor, o logaritmo negativo da probabilidade, é calculado para cada seqüência, o qual, de acordo com o comprimento da seqüência, poderá definir se ela é membro ou não da família. A utilização desses modelos pode auxiliar os profissionais de biologia, acelerando o processo de classificação de proteínas.

Os resultados alcançados com os modelos de globinas e GTPases podem ser bem visualizados com os gráficos de dispersão do comprimento *versus* escore do logaritmo negativo da probabilidade. No caso específico do HMM de comprimento 141, para as globinas, o

percentual de acerto para as seqüências dos conjuntos de teste e treinamento é bastante alto, especialmente se forem descartadas as seqüências que estão incompletas, o que deve de fato ser feito, uma vez que o escore calculado para elas é irreal. As demais seqüências dos conjuntos de teste e treinamento que não foram classificadas como globinas são do tipo *Hemoglobin-like*, ou *Uteroglobin precursor*, ou *Mammaglobin precursor*. Merece maior investigação o fato de que há dois tipos predominantes de seqüências não-membros que foram caracterizadas como membros, a saber *50S ribosomal protein L7/L12* e *Histone H1*.

Para a família das GTPases, o conjunto de seqüências utilizado no treinamento foi bem menor do que o utilizado na família das globinas. Para o HMM de comprimento 209, o percentual de acerto, para as seqüências da família, foi ligeiramente maior que o das globinas, enquanto que o percentual de erro para as seqüências não-membros foi bem maior (cerca de 4,5 vezes). Possivelmente esta diferença tenha relação com o menor número de seqüências utilizado no treinamento da GTPases, o que pode ter provocado *overfitting*. Uma solução apontada por Krogh e colaboradores [39] para superar o problema de um número reduzido de seqüências no conjunto de treinamento é adicionar algum conhecimento inicial ao modelo, o que pode ser possível alinhando-se um número pequeno de seqüências e transferindo-se os parâmetros obtidos para o modelo inicial (conforme seção 5.2).

Um outro motivo ao qual pode-se atribuir os erros apresentados pelos modelos construídos é o fato de que esses modelos são sub-ótimos, uma vez que o algoritmo EM não garante encontrar o modelo ótimo global para um dado conjunto de treinamento. Um importante problema aberto é encontrar um modo confiável de prevenir que o EM emperre em um máximo local e retorne uma solução sub-ótima.

Embora os modelos construídos tenham sido capazes de classificar seqüências das famílias modeladas, outra função importante de um HMM não foi aqui explorada. Conforme visto nos capítulos 4 e 5, é possível implementar o Algoritmo de Viterbi para construção de um alinhamento das seqüências da família apresentadas ao modelo. Na ferramenta apresentada, este alinhamento ainda não é possível, mas, uma vez que o cálculo das probabilidades *forward* e *backward* já está implementado, pouco falta para a implementação do algoritmo citado acima e, conseqüentemente, construção do alinhamento.

Uma limitação que se apresenta na ferramenta, por ora, é relativa ao tamanho das seqüências que podem ser processadas e, conseqüentemente, ao tamanho do modelo que pode ser construído (uma vez que o tamanho do modelo é função do tamanho médio das seqüências de treinamento). Da mesma forma, para a classificação de seqüências, quando as seqüências são muito grandes (algo em torno de 250, a depender do tamanho do modelo), o valor do escore cresce muito ao

ponto de não poder ser calculado pela máquina. É preciso trabalhar no sentido de normalizar os resultados parciais e finais, a fim de permitir a apresentação de seqüências maiores ao modelo.

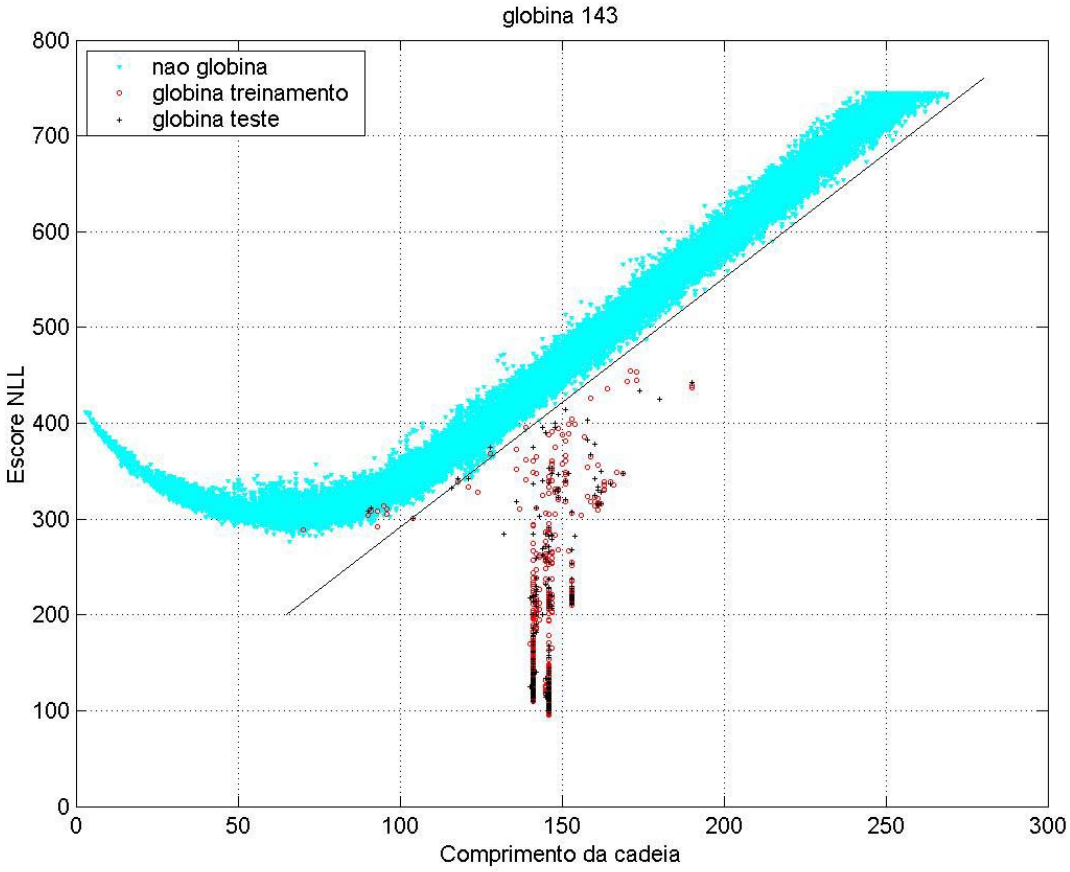
Vários desafios se apresentam para a melhoria da ferramenta. Um deles é introduzir um método para que o comprimento ideal para o modelo também seja aprendido automaticamente dos dados. Outro é incrementar a arquitetura atual com módulos que permitam agrupar as seqüências dadas em subfamílias, ou modelar domínios de proteínas a partir das seqüências dadas. É necessário ainda automatizar a classificação das seqüências no sentido de se estabelecer uma função entre o logaritmo negativo da probabilidade e o comprimento da seqüência. Isso é possível através de um método estatístico simples – uma técnica de janela, mostrada por Krogh [39].

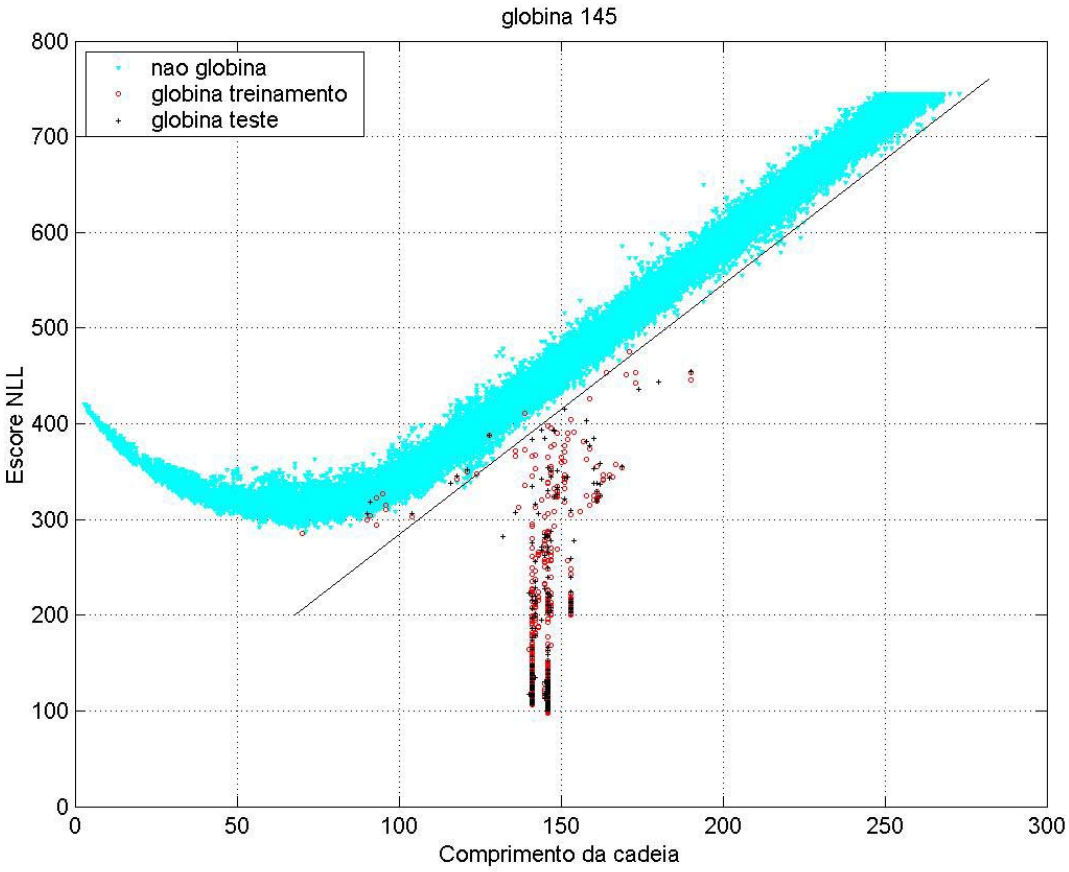
Embora ainda haja muito caminho a ser percorrido, pelo que existe na literatura hoje, Modelos Escondidos de Markov possuem um grande potencial pra modelagem estatística de seqüências de proteínas. Propostas como a de modelos híbridos – HMM-Rede Neural 3 apresentam-se promissoras e com muito ainda a ser explorado.

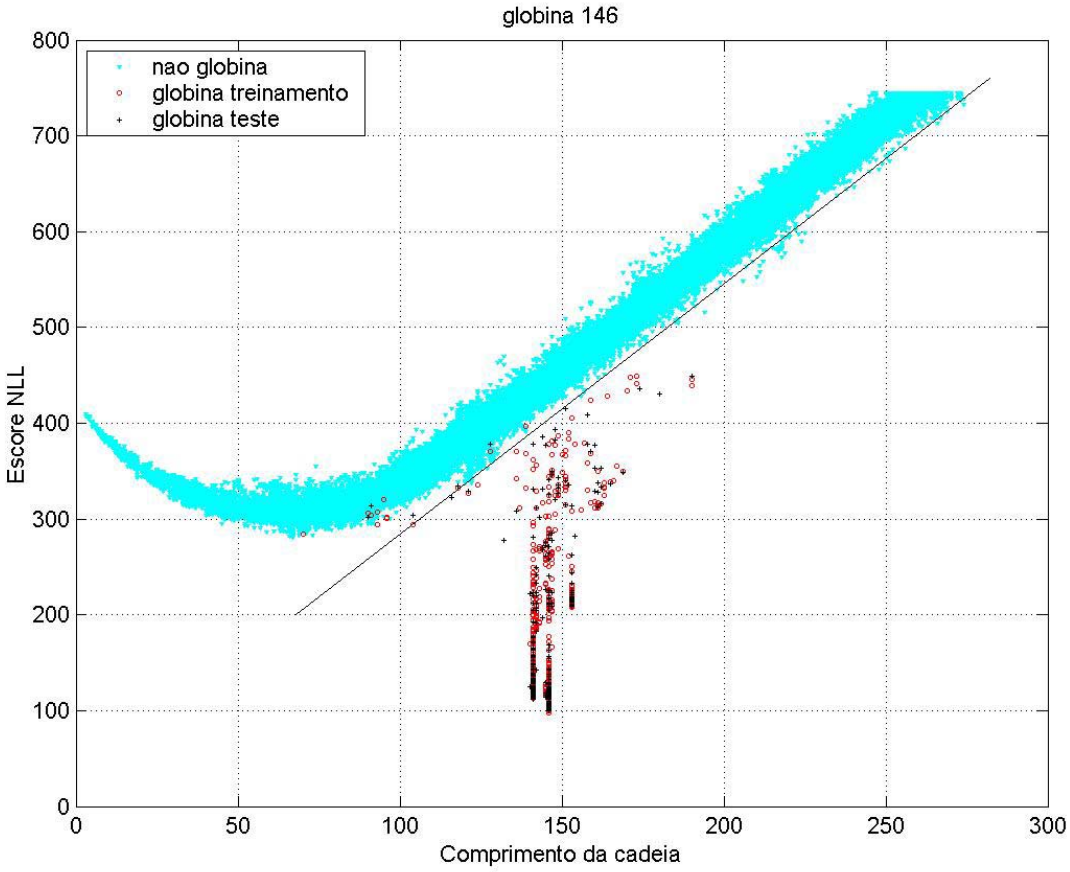
Apêndice A

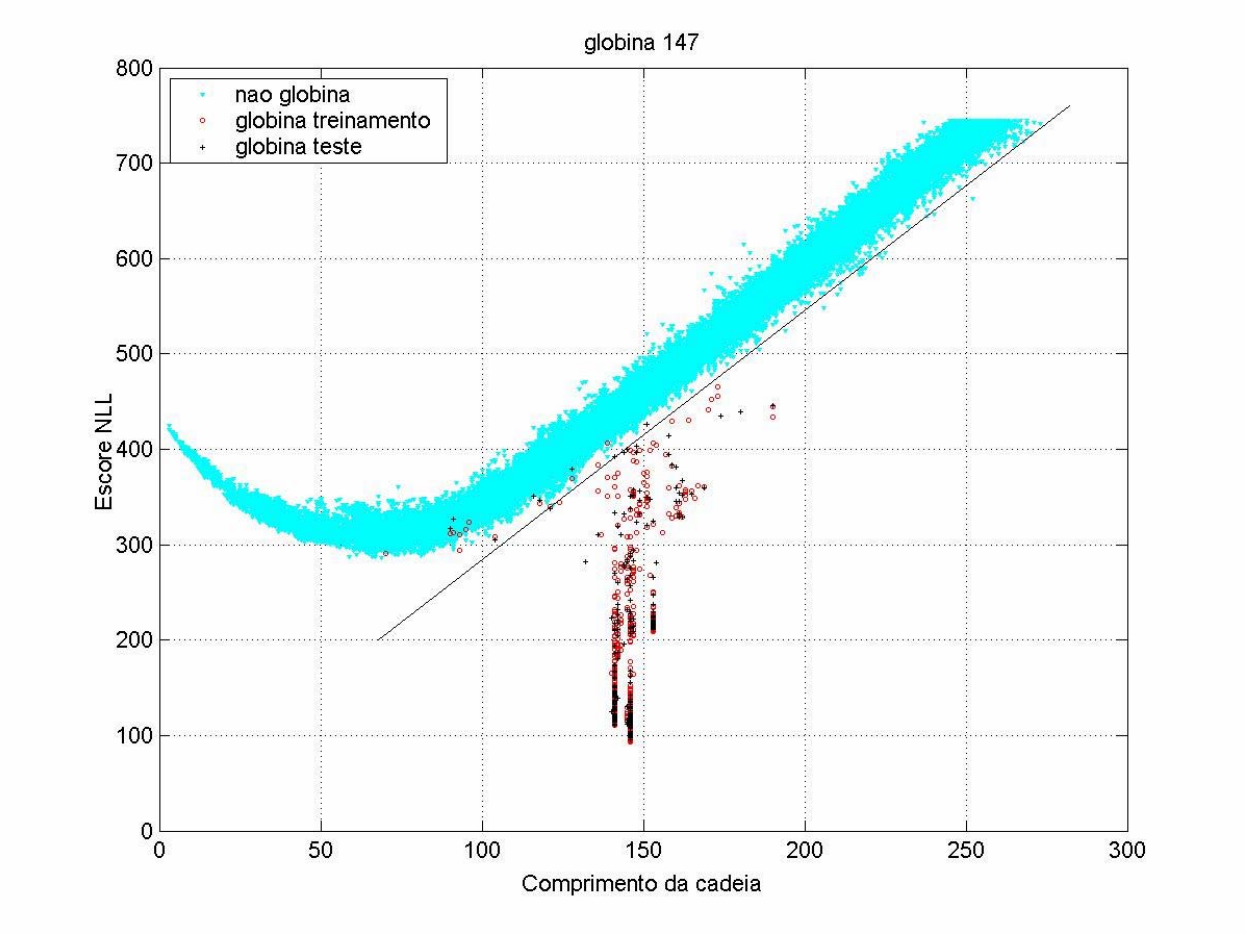
Testes de Discriminação – Gráficos

1. Modelos para Globinas

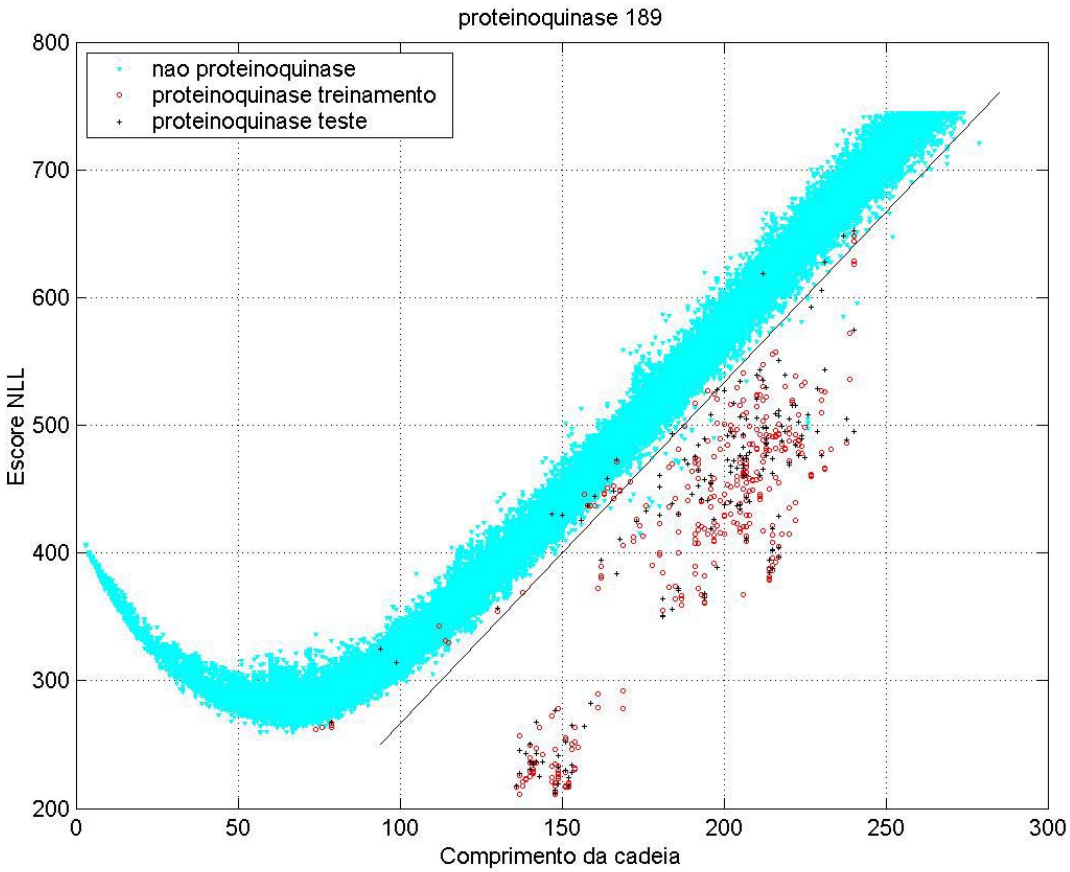


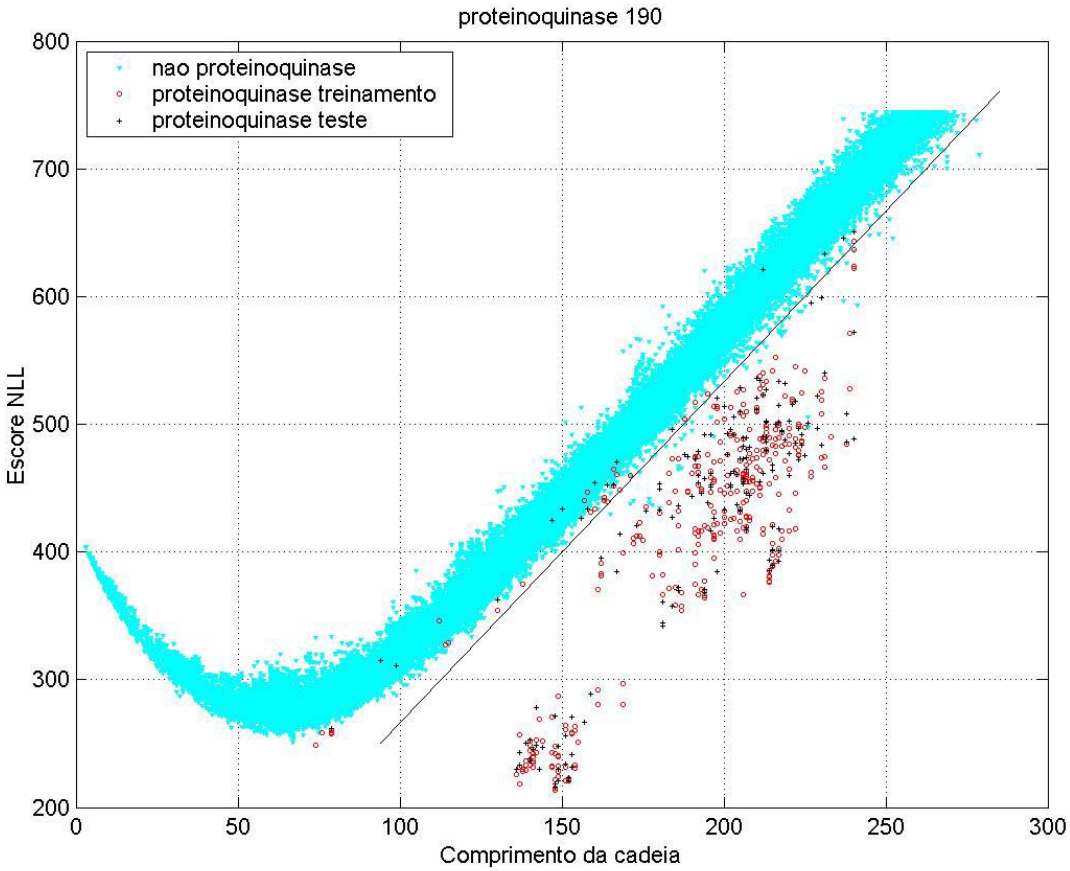

globina 145
nao globina
globina treinamento
globina teste
Escore NLL
Comprimento da cadeia
0
100
200
300
400
500
600
700
800
0
50
100
150
200
250
300

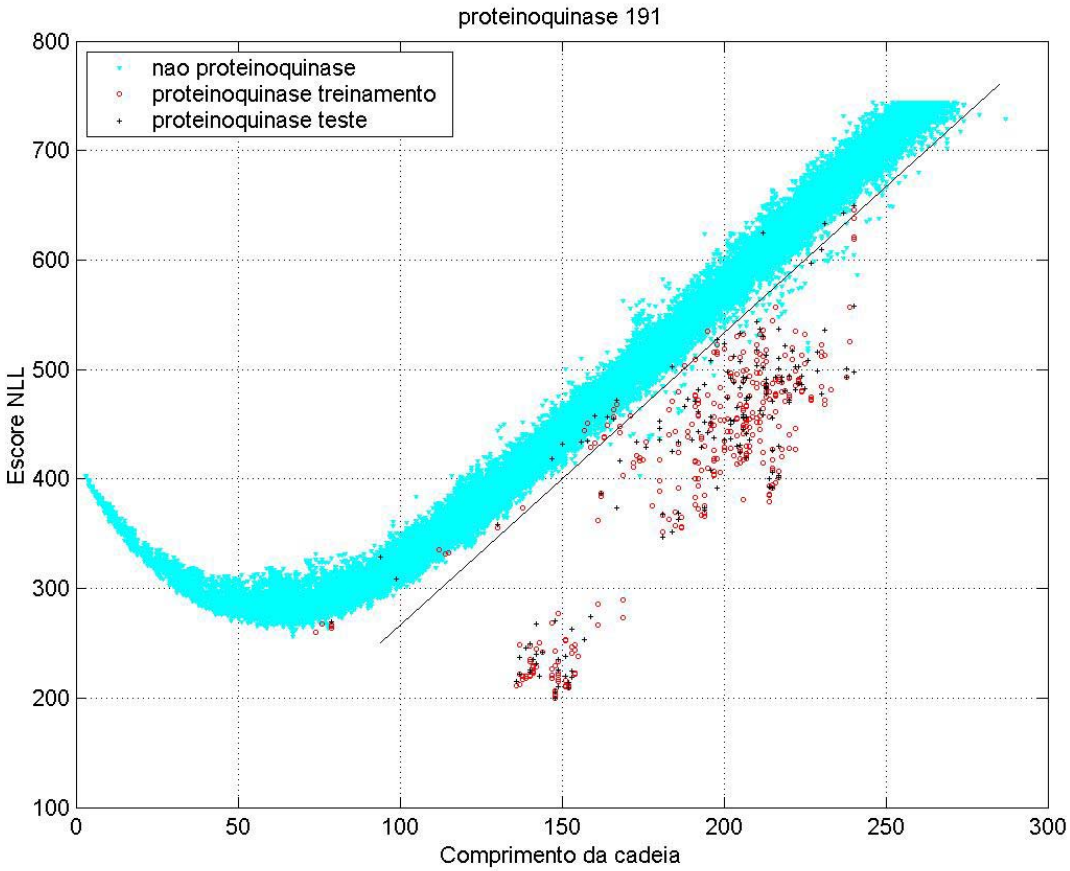

globina 146
nao globina
globina treinamento
globina teste
Escore NLL
Comprimento da cadeia
0
100
200
300
400
500
600
700
800
0
50
100
150
200
250
300

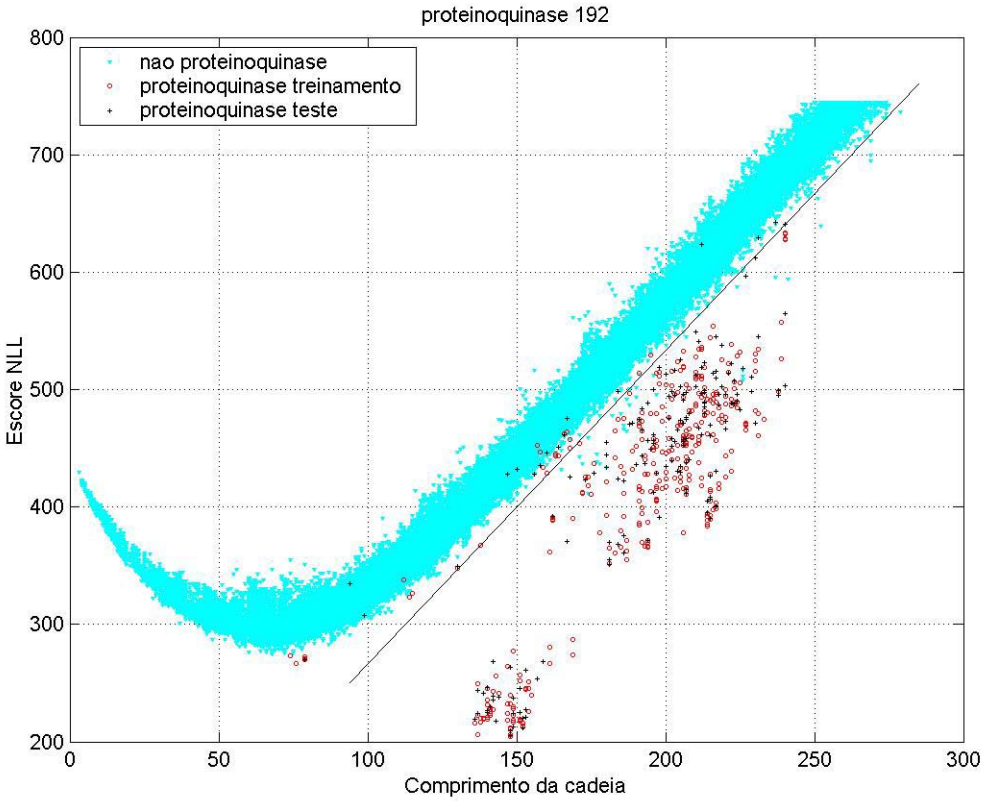


2. Modelos para Proteinoquinases

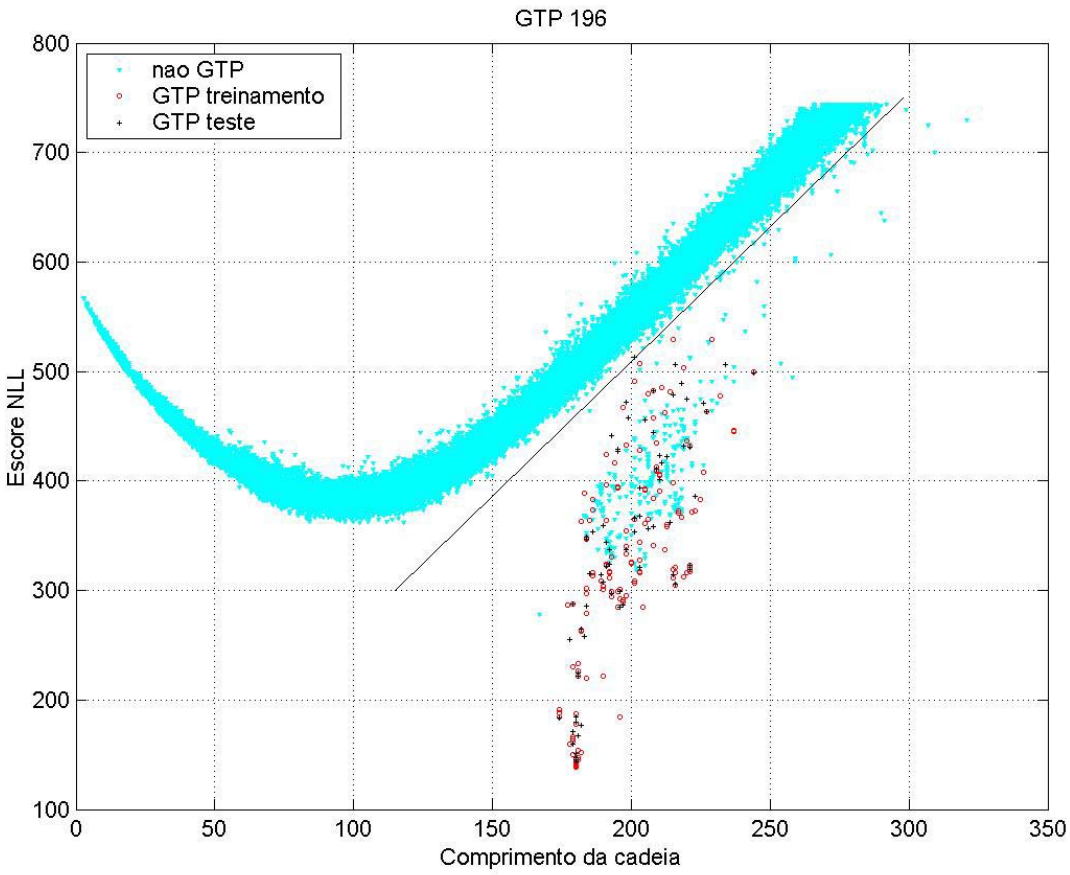


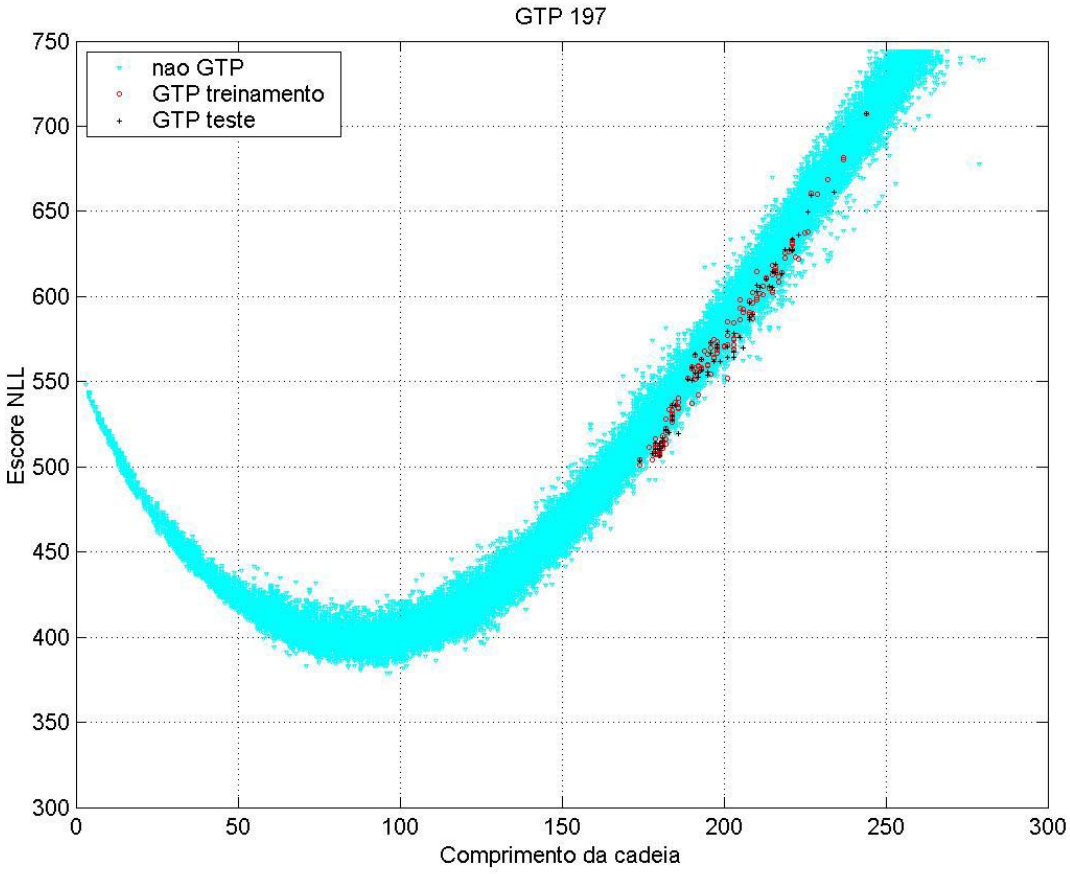

proteinoquinase 190
nao proteinoquinase
proteinoquinase treinamento
proteinoquinase teste
Escore NLL
Comprimento da cadeia
800
700
600
500
400
300
200
0
50
100
150
200
250
300

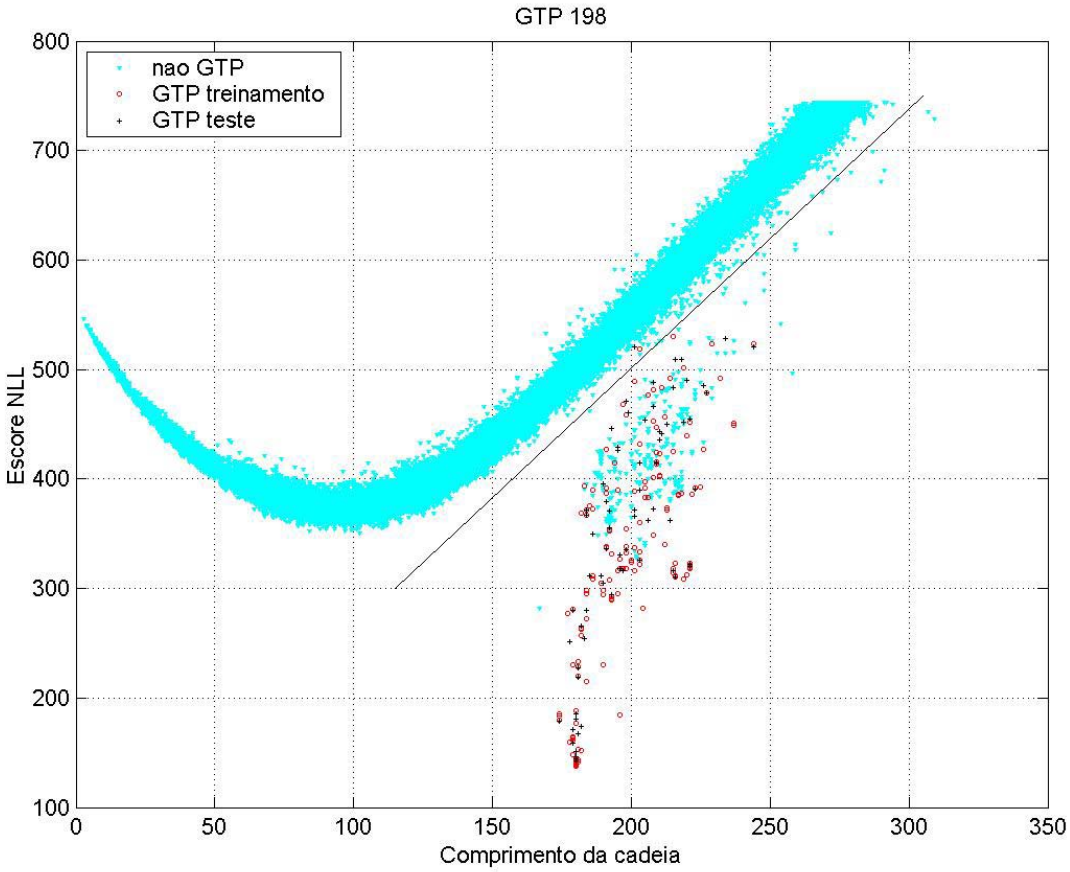

proteinoquinase 191
nao proteinoquinase
proteinoquinase treinamento
proteinoquinase teste
Escore NLL
Comprimento da cadeia
800
700
600
500
400
300
200
100
0
50
100
150
200
250
300

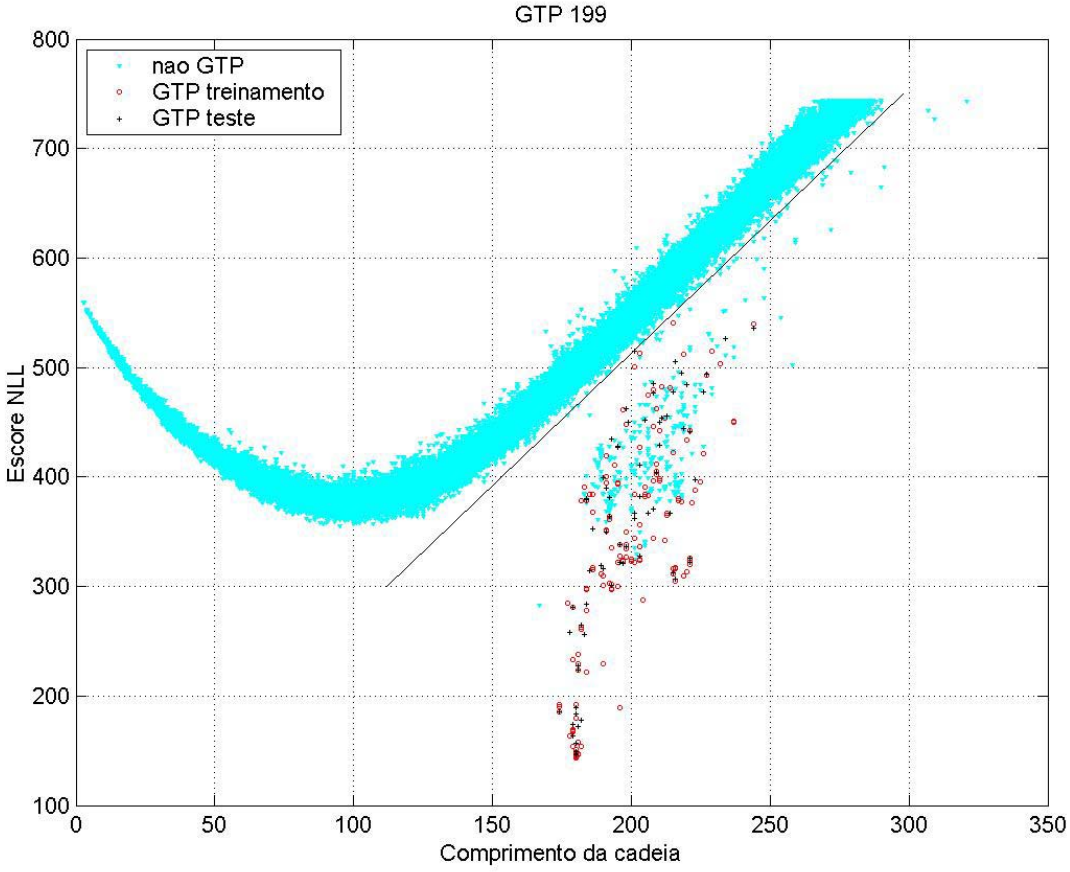

proteinoquinase 192
nao proteinoquinase
proteinoquinase treinamento
proteinoquinase teste
Escore NLL
Comprimento da cadeia
800
700
600
500
400
300
200
0
50
100
150
200
250
300

Modelos para GTPases






GTP 198
nao GTP
GTP treinamento
GTP teste
Escore NLL
Comprimento da cadeia
800
700
600
500
400
300
200
100
0
50
100
150
200
250
300
350


GTP 199
nao GTP
GTP treinamento
GTP teste
Escore NLL
Comprimento da cadeia
800
700
600
500
400
300
200
100
0
50
100
150
200
250
300
350

Bibliografia

1 ALBERTS, Bruce et all. *Molecular Biology of the Cell*. 3 ed. Garland Publishing Inc., 1994.

2 WATSON, J. D.; CRICK, F. H. C. *Molecular structure of nucleic acids. A structure for desoxyribose nucleic acid*. [S.l.]: Nature, 171:737-738.,1953.

3 BALDI, Pierre F.; BRUNAK, Soren. *Bioinformatics: the machine learning approach*. 2nd ed. Massachussets: MIT Press, 2001.

4 HIGGINS, D.G.; TAYLOR, W. *Bioinformatics – sequence structure and databanks.* (Org.). DURET, L. & ABDEDDAIM, S. *Multiple alignements for structural, functional, or phylogenetic analyses of homologous sequences*. New York: Oxford University Press, 2000.

5 SETUBAL, J. C.; MEIDANIS, J. *Introduction to Computational Molecular Biology*. Boston: PWS Pub, 1997.

6 SIMON, I. *Sequence comparison: some theory and some pratice.* Proc. LIPT Spring School on Theoretical Computer Science, vol. 377 of Lecture Notes in Computer Science, pp 77-92, 1987.

7 SCHWARTZ, R.; PEARSON, W. & ORCUTT, B. *A model of evolutionary change in proteins*. Atlas of Protein Sequence and Structure, 5:345-352, 1978.

8 GONNET, G.H. *A Tutorial Introduction to Computational Biochemistry Using Darwin.* Zürich: E.T.H., 1992.

9 SHAMIR, R. *Algorithms for Molecular Biology* – Lectures. Tel Aviv: Tel Aviv Univ., 2001.

10 NEEDLEMAN, Saul B. e WUNSH, Christian D. *A general method applicable to the search for similarities in the amino acid sequence of two proteins*. J. of Molecular Biology, 48:443-453, 1970.

11 KECECIOGLU, J. *The maximum weight trace problem in multiple sequence alignment.* CPM93 Proceedings, vol. 684 of Lecture Notes in Computer Science, pp. 106-119. Springer Verlag, 1993.

12 RABINER, L.R. *A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition*. Proceedings of the IEEE, 77:257-286, 1989.

13 BARNETT, V. *Comparative Statistical Inference*. New York: John Wiley, 1982

14 BERGER, J. O. *Statistical Decision Theory and Bayesian Analysis*. New York: Springer-Verlag, 1985.

15 JAYNES, E.T. *Probability Theory: The logic of science.* Unpublished. 1994.

16 COX, R.T. *Probability, frequency and reasonable expectation*. Am. J. Phys., 14:1-13, 1964.

17 JAYNES, E.T. *Bayesian Methods: General background.* In J. H. Justice, editor, *Maximum entropy and Bayesian methods in statistics.* pp. 1-25. Cambridge: Cambridge University Press, 1986.

18 BAUM, L.E.; PETRIE, T. *Statistical inference for probabilistic functions of finite state Markov chains*. Ann. Math. Stat., vol. 37, pp. 1554-1563, 1966.

19 BAUM, L.E.; EGON, J.A. *An inequality with applications to statistical estimation for probabilistic functions of a Markov process and to a model for ecology*. Bull. Amer. Meteorol. Soc., vol. 73, pp. 360-363, 1967.

20 BAUM, L. E.; SELL, G.R. *Growth functions for transformations on manifolds*. Pac. J. Math., vol. 27, nº 2, pp. 211-227, 1968.

21 BAUM, L.E.; PETRIE, T.; SOULES, G.; WEISS, N. *A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains*. Ann. Math. Stat., vol.41, nº 1, pp. 164-171, 1970.

22 BAUM, L.E. *An inequality and associated maximization technique in statistical estimation for probabilistic functions of Markov processes*. Inequalities, vol. 3, pp. 1-8, 1972.

23 DEMPSTER, A.P.; LAIRD, N.M.; RUBIN, D.B. *Maximum likelihood from incomplete data via the EM algorithm*. J. Roy. Statist. Soc., B, 39, 1-38, 1977.

24 ABE, N.; WARMUTH, M. *On the computational complexity of approximating distributions by probabilistic automata*. Proceedings of the 3rd Workshop on Computational Learning Theory, pp. 52-66. New York: Morgan Kaufmann Rochester, 1990.

25 BERGER, J. *Statistical Decision Theory and Bayesian Analysis*. New York: Springer-Verlag, 1985.

26 RABINER, L.R.; JUANG, B.H. *An introduction to hidden Markov models*. IEEE ASSP Mag., vol 3, nº 1, pp. 4-16, 1986.

27 DRAKE, A.W. *Fundamentals of Applied Probability Theory*. New York: McGraw-Hill, 1967.

28 FORNEY, G.D. *The Viterbi algorithm*. Proc. IEEE, vol. 61, pp. 268-278, 1973.

29 LIPORACE, L.A. *Maximum likelihood estimation for multivariate observations of Markov sources*. IEEE Trans. Informt. Theory, vol. IT-28, nº 5, pp. 729-734, 1982.

30 JUANG, B.H. *Maximum likelihood estimation for mixture multivariate stochastic observations of Markov chains*. AT&T Tech. J., vol. 63, nº 6, pp. 1235-1249, 1985.

31 JUANG, B.H. *On the hidden Markov model and dynamic time warping for speech recognition – A unified view*, AT&T Tech. J. vol. 63, nº 7, pp. 1213-1243, 1984.

32 PORITZ, A.B. *Linear predictive hidden Markov models and the speech signal*. Proc. ICASSP'82, pp. 1291-1294, Paris, 1982.

33 LEVIN, E.; PIERACCINI, R. *Planar hidden Markov modeling: From speech to optical caracter recognition*. Em HANSON, S.J.; COWAN, J.D.; GILLES, C.L. (Ed.) Advances in Neural Information Processing System, vol. 5, pp. 731-738, San Mateo: Morgan Kaufmann, 1993.

34 ALLISON, L.; WALLACE, C.S.; YEE, C.N. *Finite-state model in the alignment of macromoleulas*. J. Mol. Evol. 35, 77-89, 1992.

35 BAIROCH, A. *Prosite: a dictionary of sites and patterns in proteins*. Nucl. Acids Res. 20, 2013-2018, 1992.

36 CARDON, L.R.; STORMO, G.D. *Expectation maximization algorithm for identifying protein-binding sites with variable lengths from unaligned DNA fragments*. J. Mol. Biol. 223, 159-170, 1992.

37 HAUSSLER, D.; KROGH, A., MIAN, I.SI.; SJÖLANDER, K. *Protein* m*odeling using hidden Markov models: analysis of globins. Thechnical Report UCSC-CRL-92-23*. Santa Cruz: University of California at Santa Cruz, 1992.

38 DURBIN, R.; EDDY, S.; KROGH, A.; MITCHISON, G. *Biological Sequence Analysis: Probabilistics Models of Proteins and Nucleic Acids*. Cambridge: Cambridge University Press, 1998.

39 KROGH, A.; BROWN, M.; MIAN, I. S.; SJÖLANDER, K.; HAUSSLER, D. *Hidden Markov Models in Computational Biology – Applications to Protein Modeling*. J. Mol. Biol. 235:1501-1531, 1994.

40 KROGH, A.; MIAN, I.S.; HAUSSLER, D. *A hidden Markov model that finds genes in E. coli. DNA*. Technical Report UCSC-CRL-93-33. Santa Cruz: University of California at Santa Cruz, 1993.

41 KULP, D.; HAUSSLER, D.; REESE, M.G.; EECKMAN, F.H. *A generalized hidden Markov model for recognition of human genes in DNA*. Proc. Conference On Intelligent Systems in Molecular Biology. St. Louis: AAAI/MIT Press, 1996.

42 CHURCHILL, G.A. *Stochastic models for heterogeneous DNA sequences*, Bull. Mathem. Biol., 51:79-94, 1989.

43 SONNHAMMER, E.L.L.; EDDY, S.R.; DURBIN, R. *Pfam: a comprehensive database of protein domain families based on seed alignments*. Proteins: Structure, Function and Genetics, 28:405-420, 1997.

44 Di FRANCESSO, V.; GARNIER, J.; MUNSON, P.J. *Protein topology recognition from secondary structure sequences-applications of hidden Markov models to the alpha class proteins*. J. Mol. Biol., 267:446-463, 1997.

45 BURGE, C. *Identification of complete gene structures in human genomic DNA*. Phd thesis. Stanford: Stanford University, 1997.

46 BURGE, C.; KARLIN, S. *Prediction of Complete Gene Structures in Human Genomic DNA*. J. Mol. Biol, 268:78-94, 1997.

47 SWISSPROT – http://www.expasy.org