Arcabouço probabilístico para análise de sequências de RNA

Rafael Mathias Ferreira

Dissertação apresentada
ao
Instituto de Matemática e Estatística
da
Universidade de São Paulo
para
obtenção do título
de
Mestre em Ciências

Programa de Pós-Graduação em Ciência da Computação
Orientador: Prof. Dr. Alan Mitchell Durham

Durante parte do desenvolvimento deste trabalho
o autor recebeu auxílio financeiro da CAPES

28 de agosto de 2015

Arcabouço probabilístico para análise de sequências de RNA

Esta é a versão original da dissertação elaborada pelo candidato Rafael Mathias Ferreira, tal como submetida à Comissão Julgadora.

Resumo

Arcabouço probabilístico para análise de sequências de RNA.

RNA é um polímero formado por quatro tipos de ácidos nucleicos denotados por A, C, G e U que representam Adenina, Citosina, Guanina e Uracila respectivamente. Os nucleotídeos G-C e A-U se ligam formando pontes de hidrogênio e são ditos complementares, contudo, outros tipos de ligações podem ocorrer. RNAs são moléculas de fita única que dobram-se formando pareamentos entre bases complementares. A estrutura formada por esses pareamentos de bases complementares é chamada de estrutura secundária. Estudos recentes mostram que uma grande quantidade de RNAs não codificantes desempenham papéis importantes em uma variedade de processos biológicos, como silenciamento gênico, regulação da expressão gênica, processamento de RNA, modificação de RNA, controle da tradução e transcrição entre outros. Essas moléculas estão associadas também a diversos tipos de doenças como o câncer, doenças neurológicas como Alzheimer e Parkinson, doenças cardiovasculares e muitas outras. Dessa forma, torna-se importante descobrir novos RNAnc e suas respectivas estruturas secundárias, visto a estrita relação existente entre a estrutura secundária e a função biológica dessas moléculas. Neste trabalho desenvolvemos um arcabouço probabilístico utilizando modelo de Markov de estados ocultos sensível ao contexto para caracterização de sequências e perfil de sequências com distância arbitrária entre símbolos, como as que encontramos em sequências de RNA e em alinhamentos de RNA. Nossa implementação foi desenvolvida como uma extensão do arcabouço probabilístico ToPS e conta com algoritmos de inferência otimizados a fim de obtermos tempos de execução eficientes. Comparamos nossa implementação com outras ferramentas que possuem o mesmo propósito e pudemos constatar que nosso arcabouço se mostra bastante competitivo além de de oferecer ao usuário maior liberdade na definição de modelos.

Palavras-chave: RNAs não codificantes, csHMM, *Profile*-csHMM.

Abstract

Probabilístic Framework for RNA sequence analysis.

RNA is a four nucleotides polymer denoted by A, C, G, U which represent, respectively, Anenine, Cytosine, Guanine and Uracil. The bases A and U form hydrogen bonds, as well as the bases C and G, and these kinds of base paring are called canonical. Nevertheless, other kinds of base paring can be formed. RNAs are molecules of a single string that can fold into themselves by base pairing interactions. The structure resulted from those interactions is called RNA's secondary structure. Recent studies have shown that non-coding RNAs act upon a variety of biological processes such as gene silencing, gene expression, transcription and translation control. They are also associated with various types of diseases such as cancer, neurological diseases - as alzheimer and parkinson -, cardiovascular diseases, among others. It is therefore of fundamental importance to find new non-coding RNAs and its respective secondary structure due to the close relationship between the secondary structure and the biological function of these molecules. In this work we developed a probabilistic framework using context sensitive hidden Makov models to characterize sequences and profile of sequences with arbitrary distance between symbols, such as those found in RNA sequences and RNA alignments. Our development was made as an extension of the probabilistic framework ToPS and includes optimized versions of the inference algorithms in order to achieve efficient runtimes. We compared our approach with other framworks with similar purposes and noticed that our framework proves itself quite competitive, in addition to offering increased freedom in model definition.

Keyword: non-coding RNA, csHMM, *Profile*-csHMM.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Durante muito tempo, acreditou-se que os RNAs eram apenas intermediários passivos entre moléculas de DNA e proteínas, com exceção dos RNAs estruturais - como RNA transportador e RNA ribossômico [YV04b]. Estudos recentes mostram que essa perspectiva a respeito das moléculas de RNAs tem sido muito limitada e incompleta para explicar os processos biológicos em organismos complexos [YV08b]. Uma grande quantidade de RNAs não codificantes[1] (ncRNA, do inglês *non coding RNA*) desempenham papéis importantes em uma variedade de processos biológicos, como silenciamento gênico [SZB02], regulação da expressão gênica [LQRLT01], processamento de RNA [Edd01], controle da tradução [WZS99] e transcrição [YZLZ01, WZS99], entre outros. Os ncRNA estão associados também a diversos tipos de doenças como o câncer [Est11], doenças neurológicas como Alzheimer [KKD+03, Est11] e Parkinson [DMY+03, Est11], doenças cardiovasculares [Est11], entre muitas outras.

As descobertas acima mencionadas ilustram a necessidade de compreendermos os papéis que os ncRNA exercem nos processos biológicos. Estima-se que existem muitos ncRNA que ainda não foram descobertos [Pla02] e, infelizmente, o processo de anotação de genes que codificam ncRNA ainda é imaturo se comparado à anotação de genes codificantes de proteínas [YV08b].

A identificação de regiões de ncRNA é um problema em aberto. Móleculas de RNA não possuem sinais estatísticos fortes em sua estrutura primária [Sak03] e tendem a conservar mais sua estrutura secundária [DEKM98]. Desse modo, os métodos tradicionais, usados para identificação de genes codificantes de proteínas, não podem ser aplicados de forma direta para identificação de ncRNA. Pois, em sua maioria, esses métodos baseiam-se em estatísticas sobre a composição de bases da estrutura primária [YV04a].

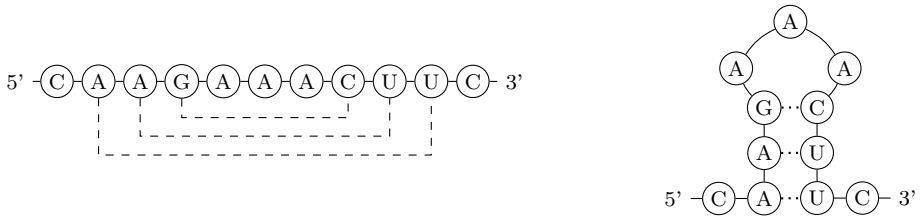
Nesse contexto, é de fundamental importância descobrir novos RNAnc e suas respectivas estruturas secundárias, visto que a estrutura secundária de um RNA e sua função biológica estão relacionadas. Porém, a quantidade de dados disponíveis para análise é muito grande e encontrar novos ncRNA por meios experimentais torna-se impraticável [YV08a]. Faz-se necessário, então, a utilização de um modelo computacional confiável que leve em consideração tanto a estrutura primária quanto a estrutura secundária dos RNAs.

[1] São moléculas de RNA que não são traduzidas em proteínas.

1.1 Estrutura do RNA

RNA é um polímero formado por quatro tipos de ácidos nucleicos denotados por A, C, G e U que representam Adenina, Citosina, Guanina e Uracila respectivamente. Os nucleotídeos G-C e A-U se ligam formando pontes de hidrogênio e são ditos complementares. Além das ligações canônicas formadas pelos pares G-C e A-U, existem as ligações não canônicas, sendo que a mais comum é a ligação G-U.

RNAs são moléculas de fita única que dobram-se formando pareamentos entre bases complementares. A estrutura formada por esses pareamentos de bases complementares é chamada de estrutura secundária do RNA. A fita única que contém os nucleotídeos arranjados sequencialmente é chamada de estrutura primária do RNA. Podemos ver na Figura 1.1 exemplos da estrutura primária e secundária de uma molécula de RNA fictícia.



(a) *Estrutura primária.* **(b)** *Estrutura secundária.*

Figura 1.1: *Exemplos de estrutura primária* ***(a)*** *e secundária* ***(b)*** *de RNA.*

Componentes estruturais são formados a partir dos pareamentos que ocorrem entre as bases complementares. Podemos identificar alguns desses componentes através da Figura 1.2.

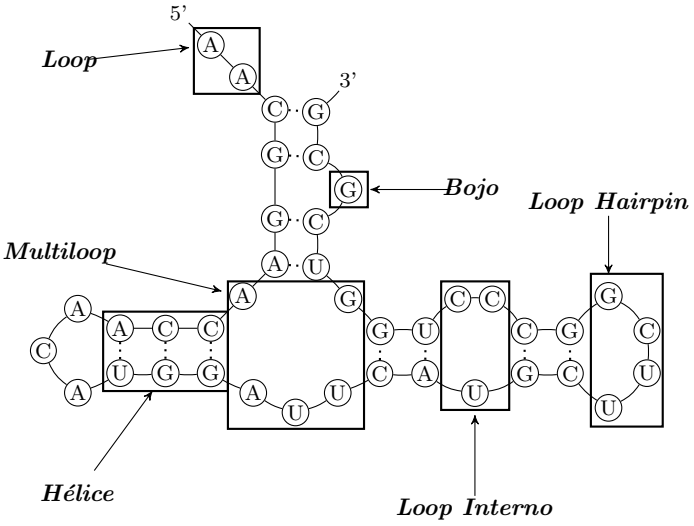


Figura 1.2: *Principais componentes da estrutura secundária de um RNA fictício.*

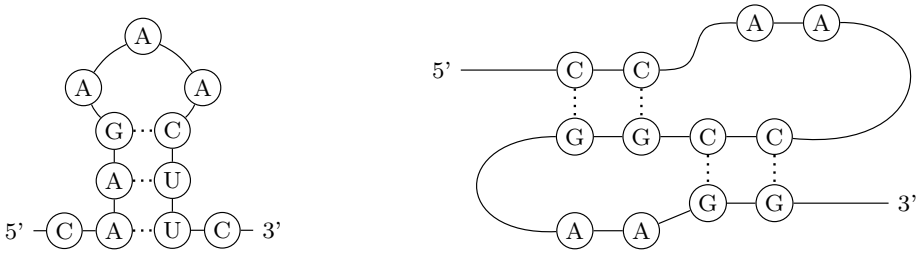
A maioria dos RNAs possui estrutura secundária aninhada, isto é, se considerarmos um pareamento entre bases das posições i e j $(i < j)$ e outro pareamento entre bases nas posições k e l $(k < l)$, teremos a seguinte condição satisfeita

$$i < k < l < j \text{ ou } k < i < j < l. \tag{1.1}$$

Porém, existem casos em que a condição acima não é satisfeita, mas a seguinte condição é

$$i < k < j < l \text{ ou } k < i < l < j. \tag{1.2}$$

Nesse caso, dizemos que os pares de base se cruzam. Estruturas secundárias que apresentam pares de bases que se cruzam são chamadas de pseudo-nós. Esse tipo de estrutura adiciona complicações na análise das sequências de RNA, de forma que algumas ferramentas não conseguem lidar com esses componentes estruturais. Na Figura 1.3 podemos ver exemplos de RNA com estrutura aninhada **(a)** e com cruzamentos entre pares de bases **(b)**.



(a) *Estrutura aninhada.* **(b)** *Estrutura com cruzamentos entre pares de bases.*

Figura 1.3: *Estruturas secundárias com pareamentos aninhados e cruzados.*

1.2 Análise de RNAs não codificantes

Como dito anteriormente, a estrutura secundária do RNA possui papel fundamental na realização de sua função biológica. Como resultado, muitas famílias de RNA possuem estrutura secundária característica compartilhada entre seus membros. Além disso, RNAs tendem a conservar sua estrutura secundária mais que a estrutura primária [DEKM98]. Portanto, ao analisá-las, é importante levar em consideração tanto a estrutura primária quanto secundária dessas sequências.

A estrutura secundária de uma molécula de RNA é definida basicamente pelos pareamentos de bases da estrutura primária [YV07, NPGK78]. Em consequência disso, o número de possíveis estruturas secundárias cresce exponencialmente em relação ao tamanho da sequência.

Torna-se necessário, então, identificar a estrutura correta do ponto de vista biológico. Uma variedade de métodos que utilizam diferentes abordagens foram propostos para predizer a estrutura secundária de RNAs, bem como para a caracterização de famílias de RNAs. A seguir, faremos uma breve descrição de algumas abordagens e métodos utilizados descritos em [MLDPD08].

Abordagem Termodinâmica: Essa abordagem assume que a estrutura secundária correta é aquela com menor energia livre de equilíbrio (ΔG), e é baseada no valor da energia livre de Gibbs para estruturas de RNA. O cálculo da energia livre é proveniente das subestruturas formadas pela estrutura secundária como: *loops*, pareamento de bases, hairpins, etc. Os parâmetros de ΔG para predição da estrutura vêm sendo aperfeiçoados, porém ainda carregam erros experimentais e de precisão que limitam a exatidão da predição estrutural [MLDPD08].

Exemplos de ferramentas que utilizam a abordagem termodinâmica para predição da estrutura secundária são Mfold [ZS81], RNASTRUCTURE [MDC+04], ViennaRNA [Hof03], entre outros.

Abordagem Probabilística: Nesta abordagem são criados modelos probabilísticos a partir de um conjunto de amostras, os parâmetros do modelo são calculados com base nas características encontradas na amostra. A vantagem da utilização dessa abordagem é que a mesma possui uma fundamentação teórica bem definida e é possível caracterizar diferentes tipos de amostras. A desvantagem reside no fato de que sua confiabilidade depende da confiabilidade da amostra a ser caracterizada; outra desvantagem é que por vezes a quantidade de parâmetros a ser estimado cresce de acordo com a sofisticação do modelo ou até mesmo com o tamanho das amostras. Dentre os modelos que baseiam-se nessa abordagem temos os Modelos de Covariância [ED94], HMM sensível ao contexto [YV04b], *Profile*-csHMM [YV08b], PHMMTS [Sak03], PSTAG [MSS04], entre outros.

Capítulo 2

Modelos probabilísticos

Abordaremos neste capítulo o modelo de Markov de estados ocultos sensível ao contexto (csHMM, do inglês *context sensitive Hidden Markov Model*) que é uma extensão do modelo de Markov de estados ocultos (HMM, do inglês *Hidden Markov Model*) tradicional. Iremos mostrar também uma variante do csHMM chamada *Profile*-csHMM para realização de perfil de sequências de ncRNA. A seguir, faremos uma fundamentação inicial de dois modelos Markovianos para melhor entendermos os modelos que serão objeto de estudo deste trabalho.

2.1 Processos Markovianos

Um processo Markoviano é um processo estocástico no qual, dada uma variável aleatória X_t, os valores de X_s, para $s > t$, não são influenciados pelos valores de X_u, para $u < t$. Em outras palavras, o comportamento probabilístico do presente depende somente de seu passado imediato [TK98]. Isso define a propriedade Markoviana, formalmente descrita em (2.1), onde a probabilidade de um estado $S_{n+1} = s$ depende apenas do estado S_n .

$$Pr(S_{n+1} = s|S_0, S_1, S_2, ..., S_n) = Pr(S_{n+1} = s|S_n) \tag{2.1}$$

2.2 Cadeias de Markov

2.2.1 Definição Formal

Uma cadeia de Markov é uma trinca $\lambda = (Q, a, \pi)$, onde:

- $Q = \{s_1, s_2, ..., s_N\}$ é um conjunto finito de estados;
- a é a matriz $N \times N$ de probabilidades de transição, onde a_{kl} denota a probabilidade de transição do estado s_k para s_l, com s_k e $s_l \in Q$;
- π é o vetor de tamanho $|Q|$ de probabilidades iniciais, onde $\pi(i)$ denota a probabilidade do modelo iniciar no estado $s_i \in Q$.

Uma cadeia de Markov pode ser vista como um conjunto de vértices conectados através de arestas direcionadas. Os vértices representam os estados e as arestas representam as probabilidades de transição entre os estados, como pode ser visto na Figura 2.1 que representa um HMM com dois estados.

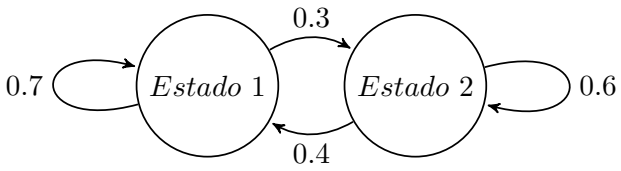


Figura 2.1: *Representação de uma cadeia de Markov de dois estados.*

Vamos utilizar o exemplo mostrado por Rabiner [Rab89] para ilustrar o funcionamento de uma cadeia de Markov.

Seja uma variável estocástica X, que representa o tempo em suas realizações definidas no conjunto $Q = \{s_1 = \textit{chuvoso},\ s_2 = \textit{nublado},\ s_3 = \textit{ensolarado}\}$. As observações são feitas diariamente e sem possibilidade de combinação entre os estados em um mesmo dia. As probabilidades de transição entre os estados são dadas pela matriz a, onde a_{ij} representa a probabilidade de transição do estado s_i para o estado s_j, com i e j assumindo valores de 1 a 3. Definimos também que a probabilidade π de iniciarmos em qualquer estado é igual para todos os estados.

$$a = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} = \begin{pmatrix} 0.4 & 0.3 & 0.3 \\ 0.2 & 0.6 & 0.2 \\ 0.1 & 0.1 & 0.8 \end{pmatrix}$$

A Figura 2.2 representa graficamente a cadeia de Markov descrita acima.

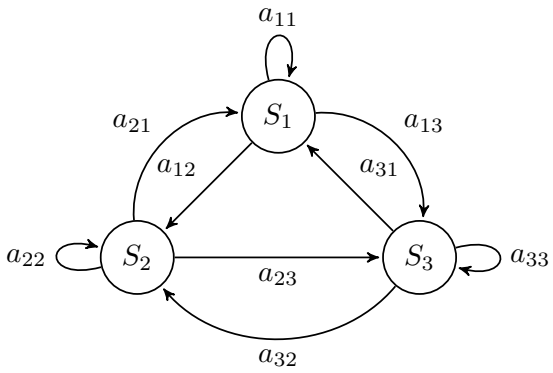


Figura 2.2: *Cadeia de Markov da modelagem do tempo.*

Para sabermos o valor da probabilidade de que o tempo para os próximos sete dias seja "ensolarado, ensolarado, chuvoso, chuvoso, ensolarado, nublado e ensolarado", dado que o tempo no primeiro dia é "ensolarado", devemos definir a sequência de observação $O = \{X_0 = s_3,\ X_1 = s_3,\ X_2 = s_3,\ X_3 = s_1,\ X_4 = s_1,\ X_5 = s_3,\ X_6 = s_2,\ X_7 = s_3\}$. Podemos, então, calcular a probabilidade de O, dado o modelo:

$$\begin{aligned} P(O|\lambda) &= P(s_3, s_3, s_3, s_1, s_1, s_3, s_2, s_3|\lambda) \\ &= P(s_3)P(s_3|s_3)P(s_3|s_3)P(s_1|s_3)P(s_1|s_1)P(s_3|s_1)P(s_2|s_3)P(s_3|s_2) \\ &= \pi(3) \cdot a_{33} \cdot a_{33} \cdot a_{31} \cdot a_{11} \cdot a_{13} \cdot a_{32} \cdot a_{23} \\ &= (0.33)(0.8)(0.8)(0.1)(0.4)(0.3)(0.1)(0.2) \\ &= 5.069 \times 10^{-4} \end{aligned}$$

2.3 Modelo de Markov de Estados Ocultos

O modelo de Markov de estados ocultos é uma generalização das cadeias de Markov. Este modelo consiste em um processo invisível de estados ocultos que satisfazem (2.1) e um processo visível de observações.

2.3.1 Definição Formal

Um HMM é uma quíntupla $\lambda = (Q, \Sigma, a, e, \pi)$, onde:

- $Q = \{s_1, s_2, ..., s_N\}$ é um conjunto finito de estados;
- $\Sigma = \{p_1, p_2, ..., p_M\}$ é o alfabeto finito de símbolos observáveis;
- $a = \{a_{kl}\}$ é o cojunto que define as probabilidades de transição de um estado s_k para s_l, com s_k e $s_l \in Q$, podemos denotar a como uma matriz $N \times N$ de probabilidades de transição. A equação 2.2 define essas probabilidades, onde $y = y_1y_2...y_L$ é sequência de estados e t o instante atual.

$$a_{kl} = P(y_t = s_l|y_{t-i} = s_k) \tag{2.2}$$

- $e = \{e_k(i)\}$ é o cojunto que define as probabilidades de emissão do símbolo $p_i \in \Sigma$ pelo estado $s_k \in Q$, podemos denotar e como uma matriz $N \times M$ de probabilidades de emissão. A equação 2.3 define essas probabilidades, onde $x = x_1x_2...x_L$ é a sequência de observações, $y = y_1y_2...y_L$ a sequência de estados e t o instante atual.

$$e_k(i) = P(x_t = p_i|y_t = s_k) \tag{2.3}$$

- $\pi = \{\pi(i)\}$ é um conjunto de probabilidades iniciais. Podemos denotar π como um vetor de tamanho $|Q|$ de probabilidades iniciais. A equação 2.4 define essas probabilidades, onde y_1 representa o primeiro rótulo;

$$\pi(i) = P(y_1 = s_i), 1 \leq i \leq N \tag{2.4}$$

Note que as equações em (2.5) devem ser satisfeitas:

$$\sum_{j=1}^{N} a_{kl} = 1, 1 \leq k \leq N, \sum_{i=1}^{M} e_k(i) = 1, 1 \leq k \leq N. \tag{2.5}$$

Assim como as cadeias de Markov, um HMM pode ser representado como um grafo dirigido. Para cada estado, deve ser especificada uma função $e_k(i)$. A sequência de observações $x = x_1x_2...x_L$ é gerada por uma sequência de estados $y = y_1y_2...y_L$ que descreve um passeio realizado no grafo.

Podemos calcular a probabilidade conjunta $P(x, y)$ da observação x ter sido gerada pela sequência de estados y através da equação abaixo:

$$P(x, y) = \pi(y_1)P(x_1|y_1)\prod_{t=2}^{L} P(y_t|y_{t-1})P(x_t|y_t) \tag{2.6}$$

2.3.2 Exemplo do Casino Desonesto

Ilustraremos nesta seção como podemos utilizar o modelo de Markov de estados ocultos para modelar o problema do cassino desonesto. Suponha que um determinado cassino possui 2 tipos de dados, um honesto e um viciado. O cassino usa, na maior parte do tempo o dado honesto, mas ocasionalmente troca para o dado viciado. As probabilidades iniciais, de transição e de emissão podem ser vistas na Figura 2.3.

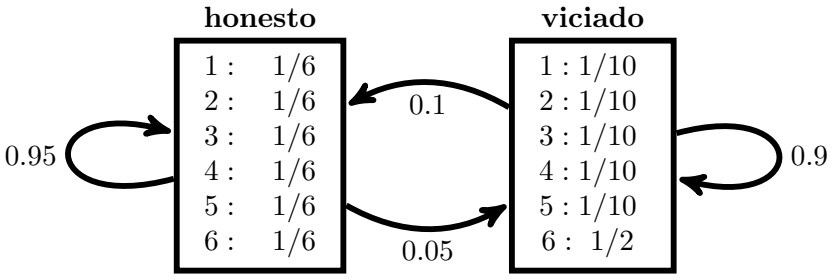


Figura 2.3: *HMM do cassino desonesto.*

Suponha que desejamos saber a sequência de estados que deu origem a observação $O = \{1, 6, 6, 6\}$. Podemos notar que existe mais de uma sequência de estados que gera a observação O e cada sequência possui uma probabilidade associada, que pode ser calculada utilizando-se a equação 2.6;

- $y = \{y_1 = honesto, y_2 = honesto, y_3 = honesto, y_4 = honesto\}; P(O, s) = 0.00067$;
- $y = \{y_1 = honesto, y_2 = honesto, y_3 = viciado, y_4 = viciado\}; P(O, s) = 0.00017$;
- $y = \{y_1 = honesto, y_2 = viciado, y_3 = viciado, y_4 = viciado\}; P(O, s) = 0.00084$

Como podemos perceber pelo exemplo acima, a sequência de estados que deu origem à observação é desconhecida pelo usuário do cassino, que somente conhece as observações, o que torna a equação 2.6 não muito útil na prática. O usuário deseja saber se o dado desonesto foi utilizado nos lançamentos. Uma abordagem satisfatória seria selecionar a sequência de estados que possui a maior probabilidade.

Achar a sequência de estados mais provável, dado um HMM e uma observação, se encaixa no problema da decodificação. O problema da decodificação representa um dos três problemas básicos de HMM que serão definidos na próxima seção.

2.3.3 Problemas básicos de HMM

Rabiner [Rab89] definiu três problemas básicos de um HMM. São eles:

Problema da Decodificação: *Dada uma sequência de observação $x = x_1x_2...x_n$ e um HMM λ, encontrar a sequência de estados $y^* = y_1^*y_2^*...y_n^*$ que otimize a geração de x segundo algum critério e calcular a probabilidade de y^*.*

Problema da Avaliação: *Dada uma sequência de observação $x = x_1x_2...x_n$ e um HMM λ, calcular a probabilidade $P(x|\lambda)$ dessa sequência ter sido gerada pelo modelo.*

Problema do Treinamento: *Dado um conjunto de treinamento $x^1, ..., x^n$, como ajustar os parâmetros de um HMM λ de forma a maximizar $P(x^1, ..., x^n|\lambda)$.*

No decorrer desta seção, utilizaremos a notação de Durbin [DEKM98].

2.3.4 Algoritmo de Viterbi

O algoritmo de Viterbi tem como objetivo resolver o problema da decodificação, ou seja:

$$y^* = \arg\max_y P(x, y), \tag{2.7}$$

onde y^* representa a sequência de estados com maior probabilidade. Note que y^* pode ser obtida calculando-se a probabilidade de todas as sequências de estados que podem dar origem à observação x, e escolher aquela com maior probabilidade. Contudo, a quantidade de sequências de estados que podem dar origem a uma determinada observação cresce exponencialmente em relação ao tamanho da observação, o que torna a utilização de (2.7) inviável computacionalmente.

Podemos calcular y^* em tempo $O(LN^2)$, onde L representa o tamanho da sequência e N a quantidade de estados. Para isso, utilizamos o algoritmo de Viterbi. Este algoritmo utiliza programação dinâmica e baseia-se no seguinte princípio: se a variável $v_k(i)$ representa a probabilidade da sequência de estados mais provável até o estado y_k com observação x_i, para todo o estado y_k. Podemos, então, calcular a probabilidade para a observação x_{i+1} pela recorrência:

$$v_l(i+1) = e_l(x_{i+1}) \max_k(v_k(i)a_{kl}). \tag{2.8}$$

Para encontrarmos a sequência de estados mais provável, devemos utilizar ponteiros que irão indicar o estado y_{i-1} que antecedeu o estado y_i. Após isso, uma rotina de *traceback* pode encontrar a sequência de estados mais provável.

2.3.5 Algoritmo *Forward*

O algoritmo *Forward* tem como objetivo solucionar o problema da avaliação calculando a probabilidade $P(x)$ de uma determinada observação ter sido gerada pelo modelo. Essa probabilidade é calculada através da soma das probabilidades de todas as possíveis sequências de estados que geram essa observação

$$P(x) = \sum_y P(x, y). \tag{2.9}$$

Cada $P(x, y)$ pode ser calculado utilizando-se a equação (2.6). Contudo, como no problema da decodificação, as possíveis sequências de estados y crescem exponencialmente, o que torna a utilização da equação (2.9) inviável. Como alternativa, podemos calcular $P(x|\lambda)$ de forma similar ao algoritmo de Viterbi. Para isso, alteramos a recorrência substituindo as maximizações por somas. O algoritmo *Forward* também possui complexidade de tempo $O(LN^2)$ para uma sequência de tamanho L e N estados. Definimos, então, $\alpha_l(i)$ como a soma da probabilidade de todas as sequências de estados até o estado $y_i = k$ com observação x_i. $P(x)$ pode, então, ser calculado como abaixo:

$$\alpha_l(i+1) = e_l(x_{i+1}) \sum_k \alpha_k(i) a_{kl}. \tag{2.10}$$

2.3.6 Algoritmo *Backward*

O algoritmo *Backward* tem a mesma finalidade que o algoritmo *Forward*, calcular a probabilidade $P(x)$ de uma determinada observação ser gerada pelo modelo. Contudo, o seu funcionamento é um pouco diferente. No algoritmo *Forward* procedemos com os cálculos da esquerda para a direita da sequência de observação, enquanto que no algoritmo *Backward* procedemos com os cálculos da direita para a esquerda. $P(x)$ pode ser calculado então utilizando-se a seguinte relação de recorrência

$$\beta_k(i) = \sum_l a_{kl} e_l(x_{i+1}) \beta_l(i+1), \tag{2.11}$$

onde, $\beta_k(i)$ representa a soma da probabilidade de todas as sequências de estados que terminam com o símbolo x_i e estado $y_i = k$.

Para resolvermos o problema da decodificação é necessário apenas o algoritmo *Forward* ou o *Backward*, porém os dados intermediários das matrizes de programação dinâmica α e β são úteis na resolução do problema do treinamento.

2.3.7 Estimativa dos Parâmetros do Modelo

Ilustraremos aqui, uma das possíveis abordagens para resolver o problema do treinamento para HMMs. O algoritmo apresentado tem como objetivo resolver o problema do treinamento e utiliza o método de esperança e maximização (EMM, do inglês *Expectation-Maximization Method*). Esse método opera, primeiramente, calculando uma aproximação para os parâmetros do modelo. Após isso, essa aproximação é utilizada como base para calcular novamente os parâmetros. Esse processo se repete até que algum critério de parada seja satisfeito.

Apresentaremos o algoritmo de *Baum-Welch* [DEKM98], que necessita de um conjunto de sequências de exemplo, chamado conjunto de treinamento, para calcular os parâmetros do modelo. Denotaremos o conjunto de treinamento por $x^1, ..., x^n$. O algoritmo funciona calculando o número esperado de vezes que cada emissão e transição são utilizadas com base no conjunto de treinamento e, para isso, as variáveis α e β são utilizadas.

Podemos calcular o número esperado de vezes da probabilidade a_{kl} ser usada no instante t na sequência x por

$$P(y_t = s_k, y_{t+1} = s_l | x, \lambda) = \frac{\alpha_k(i) a_{kl} e_l(x_{i+1}) b_l(i+1)}{P(x)}. \tag{2.12}$$

Podemos então calcular o número esperado de vezes que a_{kl} é usado utilizando a equação (2.12) para todas as posições de x e todas as sequências de treinamento,

$$A_{kl} = \sum_j \frac{1}{P(x^j)} \sum_i \alpha_k^j(i) a_{kl} e_l(x_{i+1}^j) \beta_l^j(i+1). \tag{2.13}$$

De forma similar, podemos encontrar o número esperado de vezes do símbolo σ ser emitido pelo estado k, e armazenamos esse valor na variável $E_k(\sigma)$

$$E_k(\sigma) = \sum_j \frac{1}{P(x)} \sum_{\{i|x_i^j=\sigma\}} \alpha_k^j(i) \beta_k^j(i). \tag{2.14}$$

Após o cálculo esperado das variáveis A_{ij} e $E_i(\sigma)$, podemos usar as seguintes equações (2.15) para reestimar os parâmetros do modelo

$$a_{kl} = \frac{A_{kl}}{\sum_{l'} A_{kl'}}, \; e_k(\sigma) = \frac{E_k(\sigma)}{\sum_{\sigma'} E_k(\sigma')}. \tag{2.15}$$

Repetimos esse processo até que um critério de parada seja satisfeito. É importante ressaltar a possibilidade de algum estado k não ser usado no conjunto de treinamento. Como consequência, o estado k não terá nenhum valor de emissão e transição atribuído. Uma solução satisfatória para esse problema é a utilização de pseudo-contadores.

2.3.8 Modelo de Markov de estados ocultos e gramáticas transformacionais

O modelo de Markov de estados ocultos é um modelo estocástico amplamente utilizado em muitas áreas de estudo, das quais podemos citar: reconhecimento de voz [Rab89], biometria [BA08], biologia computacional [DEKM98, Kro97, Edd98], entre outras.

Embora muito utilizado, HMMs possuem algumas limitações, por exemplo, a incapacidade de modelar sequências que possuem associações com distância arbitrária entre símbolos, como as que encontramos em sequências de RNA.

Noam Chomsky, em seu trabalho [Cho59] a respeito de gramáticas transformacionais, categorizou todas as gramáticas formais em 4 classes, de acordo com as restrições aplicadas às regras de produção. Gramáticas regulares, gramáticas livres de contexto, gramáticas sensíveis ao contexto e gramáticas com estrutura de frase são os nomes das 4 classes que compõem a hierarquia de Chomsky para gramáticas transformacionais. A Figura 2.4 representa a hierarquia disposta conforme as restrições nas regras de produção.

Quanto maior o número de restrições aplicadas às regras de produção, menor é o poder descritivo da classe em questão. De acordo com a hierarquia de Chomsky, HMMs são classificados como gramáticas estocásticas regulares, visto que ambas possuem o mesmo poder descritivo e são intercambiáveis. A diferença entre esses dois modelos consiste na forma de representá-los. HMMs são tradicionalmente representados como uma máquina de estados, onde cada estado pode ou não emitir um símbolo (estados silenciosos); enquanto que gramáticas estocásticas regulares são representadas através de regras de produção, onde os símbolos são emitidos ao se usar uma das regras de produção.

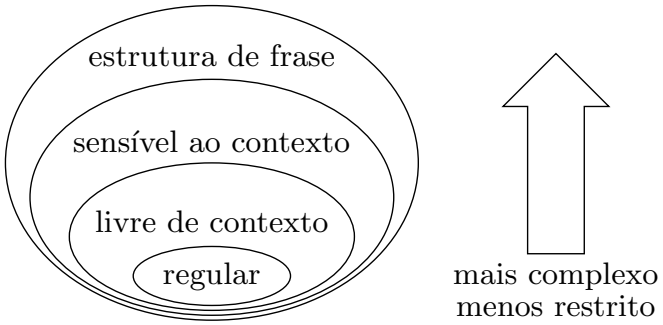


Figura 2.4: *Hierarquia de Chomsky.*

2.4 Modelo de Markov de Estados Ocultos Sensível ao Contexto

Moléculas de RNA podem ser vistas como "palíndromos biológicos" por consequência das interações que ocorrem entre bases complementares. A linguagem dos palíndromos compreende todas as sequências de símbolos que mantém o mesmo sentido quando lidas tanto da esquerda para a direita, quanto da direita para a esquerda. Para exemplificar, considere os palíndromos da Figura 2.5 sobre o alfabeto $\{A, B\}$. Perceba que existem associações entre os símbolos que compõem a sequência (as linhas indicam essas associações). Agora note que também possuímos essas associações entre os símbolos de uma sequência de RNA, como mostra a figura 2.6

(a) **(b)**

Figura 2.5: *Exemplos de palíndomos.*

Figura 2.6: *Exemplo de como uma sequência de RNA pode ser vista como um palíndromo.*

Gramáticas regulares, assim como seus correspondentes HMMs, não conseguem representar de forma eficiente linguagens com essa estrutura. Portanto, não podemos utilizar esses modelos na análise de moléculas de RNA, visto a importância de considerarmos tanto a estrutura primária quanto secundária ao analisarmos essas sequências [Edd02].

A fim de superar essa limitação dos HMMs, Yoon e Vaidynathan [YV04b] propuseram o modelo de Markov de estados ocultos sensível ao contexto (csHMM, do inglês *context-sensitive Hidden Markov Model*). Este modelo é uma extensão do HMM tradicional, onde alguns estados possuem uma memória auxiliar cuja função é criar um "contexto" ao modelo. Determinados estados, ao emitir um símbolo, armazenam esse símbolo na memória auxiliar, o que afetará as probabilidades de emissão e transição de estados futuros.

A principal vantagem do csHMM é a possibilidade de modelar fortes associações entre pares

de símbolos distantes, o que não é possível em HMM, que somente consegue modelar associações entre símbolos adjacentes [YV06]. A característica acima mencionada faz desse modelo uma boa ferramenta a ser utilizada na análise de RNAs.

2.4.1 Definições

O csHMM possui três classes de estados ocultos, que são:

- **Estado de emissão única -** S_n: Idêntica ao estado de um HMM tradicional.
- **Estado de emissão pareada -** P_n: Quase idêntico ao estado de um HMM tradicional, a única diferença consiste no fato de que quando um estado pertencente à essa classe emite um símbolo, esse símbolo é armazenado em uma memória auxiliar associada à esse estado.
- **Estado sensível ao contexto -** C_n: Nesse estado, as probabilidades de transição e emissão não são fixas, elas dependem do contexto do modelo, isto é, as probabilidades dependem dos dados que estão armazenados na memória auxiliar.

A união das três classes de estados representa o conjunto $Q = \mathcal{S} \cup \mathcal{P} \cup \mathcal{C}$ de estados do modelo, onde, $\mathcal{P} = \{P_1, P_1, ...P_{M_1}\}$, $\mathcal{C} = \{C_1, C_1, ...C_{M_1}\}$ e $\mathcal{S} = \{S_1, S_1, ...S_{M_2}\}$ representam, respectivamente, os conjuntos de estados de emissão pareada, sensíveis ao contexto e de emissão única. A quantidade de estados de emissão pareada e sensíveis ao contexto deve ser igual, de forma que esses estados formam pares $(\mathcal{P}_n, \mathcal{C}_n)$ e compartilham uma memória auxiliar Z_n.

Denotamos por $x = x_1x_2...x_L$ a sequência de símbolos de observação, onde x_i representa o símbolo emitido no instante i, e por $y = y_1y_2...y_L$ a sequência de estados que deu origem a observação x.

Probabilidade de Transição

A probabilidade de transição de um estado $s_i = v$ para o estado $s_{i+1} = w$, onde $v \in \mathcal{S} \cup \mathcal{P}$ é definida por

$$P(s_{i+1} = w | s_i = v) = t(v, w). \tag{2.16}$$

Para um estado $s_i = v \in \mathcal{C}$ (para simplificar, adote $v = \mathcal{C}_n$), as probabilidades de transição dependem da memória auxiliar Z_n associada ao estado estar vazia ou não. Caso Z_n esteja vazia, são permitidas transições para um determinado conjunto de estados $\mathcal{E} \subset Q$. Caso contrário, são permitidas transições para o conjunto de estados $\mathcal{F} \subset Q$, onde $\mathcal{E} \cap \mathcal{F} = \emptyset$. Definimos então as probabilidade de transição de um estado $v \in \mathcal{C}$ da seguinte maneira

$$P(s_{i+1} = w | s_i = v, Z) = \begin{cases} t_e(v, w) & \text{se } Z_n \text{ está vazia} \\ t_f(v, w) & \text{se } Z_n \text{ não está vazia.} \end{cases} \tag{2.17}$$

Como $\mathcal{E} \cap \mathcal{F} = \emptyset$, podemos simplificar (2.17) e definir

$$P(s_{i+1} = w | s_i = v, Z_n) = t(v, w). \tag{2.18}$$

Note que $\sum_{w \in \mathcal{E}} t(v, w) = 1$ e $\sum_{w \in \mathcal{F}} t(v, w) = 1$.

Probabilidade de Emissão

A probabilidade de um estado $s_i = v$ emitir o símbolo $x_i = x$, com $v \in \mathcal{S} \cup \mathcal{P}$, é definida por

$$P(x_i = x|s_i = v) = e(x|v). \tag{2.19}$$

Caso $v \in \mathcal{C}$, a probabilidade de emissão depende do contexto Z_n, portanto ela é definida por

$$P(x_i = x|s_i = v, Z_n) = e(x|v, Z_n). \tag{2.20}$$

Podemos simplificar (2.20) se a memória auxiliar for uma pilha. Nesse caso, a probabilidade de emissão dependerá apenas do símbolo x_p que está no topo da pilha. A equação (2.20) será redefinida, então, para

$$P(x|s_i = v, Z_n) = e(x|v, x_p). \tag{2.21}$$

2.4.2 Exemplos de csHMM

A linguagem dos palíndromos é um exemplo de linguagem que possui fortes associações entre símbolos distantes. Iremos ilustrar o funcionamento de um csHMM que gera somente palíndromos baseado no exemplo utilizado por Yoon [YV06]. Como podemos ver na Figura 2.7, os estados P_1 e C_1 formam par e estão associados a uma pilha (Pilha 1). O modelo inicia no estado *Início* e então faz uma transição para o estado de emissão pareada P_1, esse estado irá transitar para si mesmo n vezes e a cada transição o símbolo emitido é armazenado na pilha. Em um determinado momento, P_1 irá transitar ou para S_1 ou para C_1. Caso ele transite para S_1 um símbolo será emitido por S_1, então S_1 faz uma transição para C_1. Uma vez no estado C_1, as probabilidades de emissão e transição serão ajustadas de modo que C_1 irá emitir o mesmo símbolo que se encontra no topo da pilha. C_1 irá transitar para si mesmo enquanto a pilha não estiver vazia, caso contrário C_1 fará uma transição para o estado *Fim*.

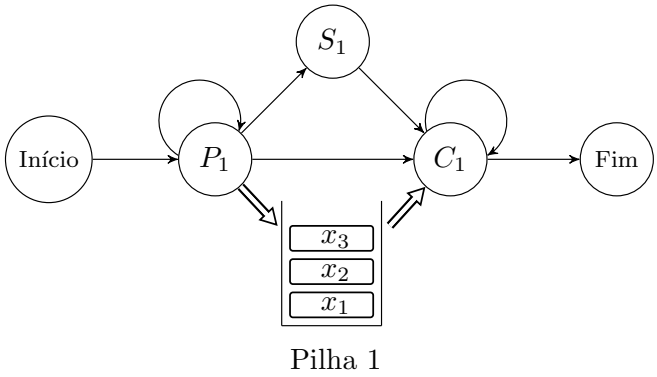
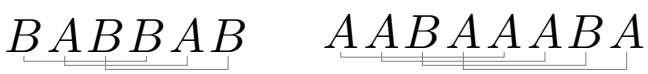


Figura 2.7: *HMM sensível ao contexto que gera somente palíndromos.*

Podemos perceber que, se o modelo passar pelo estado S_1, a observação gerada será do tipo $x_1...x_n x_{n+1} x_n...x_1$ (sequência de comprimento ímpar) e, caso não passe por S_1, a observação será do tipo $x_1...x_n x_n...x_1$ (sequência de comprimento par). Podemos perceber pelo exemplo acima que o csHMM consegue representar de forma eficaz associações entre símbolos arbitrariamente distantes.

Outra linguagem que também possui fortes associações entre seus símbolos é a linguagem de

cópia. A linguagem de cópia é definida por todas as sequências de símbolos que, ao serem divididas na metade resulta em duas subsequências de símbolos idênticas. Veja na Figura 2.8 exemplos de palavras sobre o alfabeto $\{A, B\}$ que pertencem a essa linguagem (as linhas indicam as associações entre os símbolos).



(a) **(b)**

Figura 2.8: *Exemplo de palavras pertencentes a linguagem de cópia.*

Podemos construir um csHMM que gera somente sequências pertencentes à essa linguagem. Na Figura 2.9 temos um exemplo. Perceba que, nesse caso, a memória auxiliar é uma fila, diferente do csHMM dos palíndromos que usa uma pilha. Nesse csHMM os estados P_1 e C_1 formam par e estão associados a uma fila (Fila 1). O modelo inicia no estado *Início* e então faz uma transição para o estado de emissão pareada P_1, esse estado irá transitar para si mesmo n vezes e a cada transição o símbolo emitido é armazenado no final da fila. Em um determinado momento, P_1 fará uma transição para C_1. Uma vez no estado C_1, as probabilidades de emissão e transição serão ajustadas de modo que C_1 irá emitir o mesmo símbolo que se encontra no início da fila. C_1 irá transitar para si mesmo enquanto a fila não estiver vazia. Quando a fila estiver vazia, C_1 fará uma transição para o estado *Fim*.

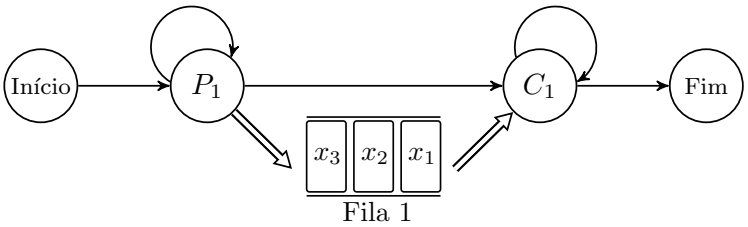


Figura 2.9: *HMM sensível ao contesto que representa a linguagem de cópia.*

No csHMM da linguagem de cópia são geradas somente sequências do tipo $x_1x_2...x_nx_1x_2...x_n$, que é a concatenação de duas sequências idênticas. Uma característica interessante da linguagem de cópia é que a mesma gera símbolos que possuem associações cruzadas.

A princípio, podemos ter qualquer estrutura de dados servindo como memória auxiliar para o csHMM, bastando para isso que essa estrutura possua uma interface que permita operações de leitura, escrita e verificação de vazio.

Podemos perceber que o csHMM possui um alto poder descritivo. Na Figura 2.10 temos um exemplo de como modelar um molécula de RNA que possui relacionamentos cruzados entre pares de bases (pseudo-nó).

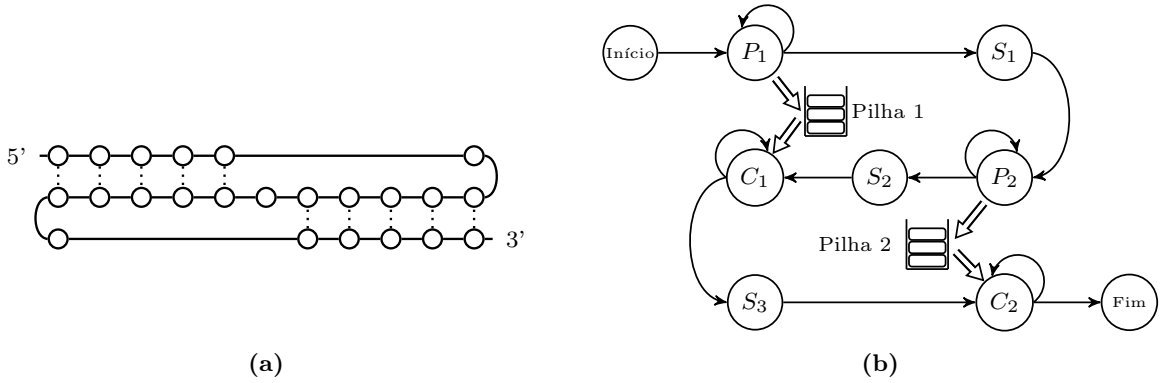


Figura 2.10: *Estrutura de um RNA com pseudo-nó* ***(a)*** *e sua representação em csHMM* ***(b)****.*

Na Figura 2.11 temos um exemplo de como modelar um molécula de RNA no formato de trevo.

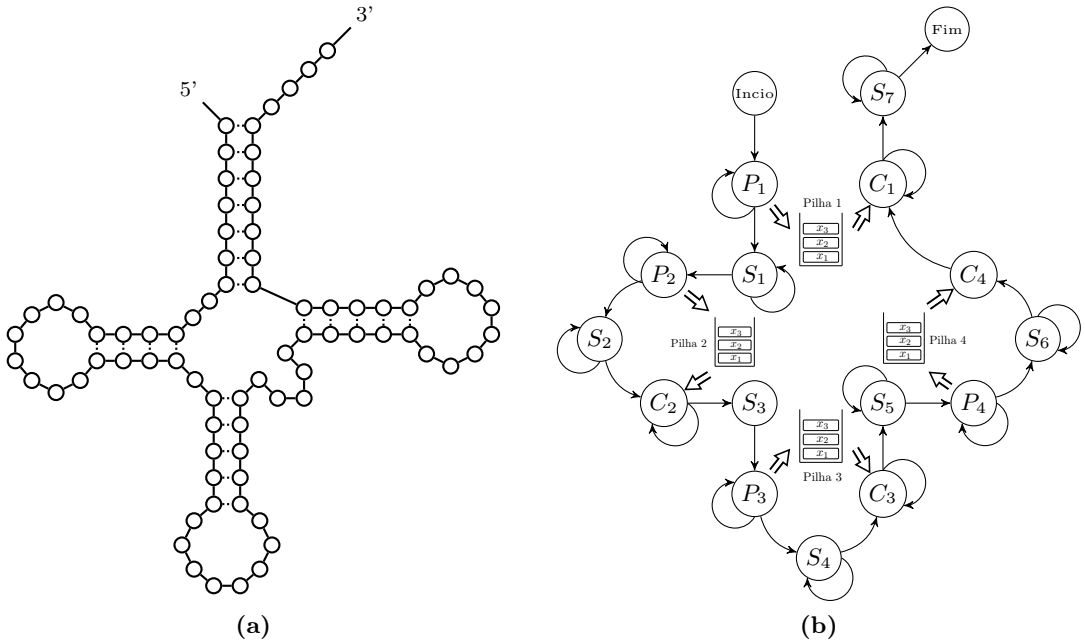


Figura 2.11: *Estrutura de um RNA no formato de trevo* ***(a)*** *e sua representação em csHMM* ***(b)****.*

O csHMM possui os mesmos problemas básicos definidos na Seção 2.3.3, porém as abordagens utilizadas para HMM não podem ser diretamente aplicadas em csHMM. No exemplo a seguir retirado de [YV06], iremos mostrar por que a mesma abordagem não pode ser aplicada.

Seja o HMM sensível ao contexto representado pela Figura 2.12 que opera sobre o alfabeto $\Sigma = \{a, b\}$, as probabilidades de transição e emissão estão ilustradas na figura. Vamos calcular a probabilidade da sequência de estados mais provável para a observação $x = abbba$. Podemos perceber que as sequências de estados $s_1 = P_1S_1S_1S_1C_1$ e $s_2 = P_1P_1S_1C_1C_1$ podem dar origem à observação x. Vamos considerar a sequência de estados para os três primeiros símbolos de x. Sejam $s'_1 = P_1S_1S_1$ e $s'_2 = P_1P_1S_1$ as subsequências de estados de s_1 e s_2 respectivamente para os três

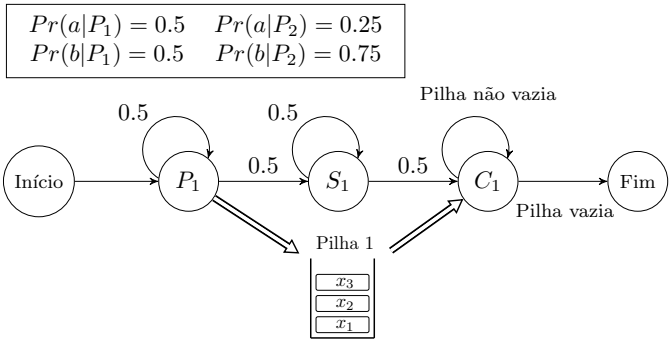


Figura 2.12: *Exemplo de um HMM sensível ao contexto.*

primeiros símbolos de x. As probabilidades de s'_1 e s'_2 são:

$$P(s'_1) = \frac{1}{2} \times \frac{1}{2} \times \frac{3}{4} \times \frac{1}{2} \times \frac{3}{4} = \frac{9}{128}, \tag{2.22}$$

$$P(s'_2) = \frac{1}{2} \times \frac{1}{2} \times \frac{1}{2} \times \frac{1}{2} \times \frac{3}{4} = \frac{3}{64}. \tag{2.23}$$

A sequência de estados mais provável para os três primeiros símbolos de x é s'_1. Porém, ao calcularmos as probabilidades de s_1 e s_2, temos:

$$P(s'_1) = \frac{1}{2} \times \frac{1}{2} \times \frac{3}{4} \times \frac{1}{2} \times \frac{3}{4} \times \frac{1}{2} \times \frac{3}{4} \times \frac{1}{2} \times 1 \times 1 = \frac{27}{2048}, \tag{2.24}$$

$$P(s'_2) = \frac{1}{2} \times \frac{1}{2} \times \frac{1}{2} \times \frac{1}{2} \times \frac{3}{4} \times \frac{1}{2} \times 1 \times 1 \times 1 \times 1 = \frac{3}{128}. \tag{2.25}$$

Esse exemplo nos mostra que a abordagem utilizada no algoritmo de Viterbi não pode ser aplicada em csHMM. Porém existem algoritmos que utilizam técnicas de programação dinâmica para resolver os problemas definidos na Seção 2.3.3. Os algoritmos que serão apresentados a seguir levam em consideração que as associações entre os símbolos ocorrerá de forma aninhada. Segundo o autor, é possível criar algoritmos que conseguem lidar com associações cruzadas, porém no melhor de nosso conhecimento tais algoritmos não foram publicados em artigos.

2.4.3 Cálculo da Probabilidade da Sequência de Estados Mais Provável

O algoritmo que será apresentado tem por objetivo resolver o problema da decodificação e utiliza técnicas de programação dinâmica para encontrar a sequência de estados mais provável em tempo polinomial. O algoritmo é conceitualmente similar ao algoritmo CYK [LY90] - utilizado para fazer *parser* de gramáticas livres de contexto - e inicia sua execução calculando a sequência de estados mais provável para subsequências de tamanho um e, em seguida para subsequências de tamanho dois e repete esse procedimento de forma recursiva para subsequências maiores. Ao final do algoritmo teremos calculado $P(x, y^*|\lambda)$, onde y^* satisfaz

$$y^* = \arg\max_y P(x, y). \tag{2.26}$$

Vamos definir as variáveis que serão utilizadas no algoritmo. Assumiremos que o csHMM possui M estados distintos que serão denotados pelo conjunto $\mathcal{Q} = \{1, 2, ..., M\}$, onde o estado 1 denota o estado inicial e o estado M denota o estado final. $x = x_1x_2...x_L$ é a sequência de observação e $y = y_1y_2...y_L$ a sequência de estados associada. Para um estado $v \in \mathcal{P} \cup \mathcal{C}$, vamos definir $\overline{v}$ como o estado complementar de v, ou seja,

$$v = P_n \rightarrow \overline{v} = C_n,\ v = C_n \rightarrow \overline{v} = P_n.$$

Definimos $\gamma(i, j, v, w)$ como o log da probabilidade da sequência de estados mais provável entre todas as subsequências $y_i...y_j$ com $y_i = v$ e $y_j = w$. Denotamos a probabilidade de emissão do símbolo x pelo estado v por $e(x|v)$ caso $v \in \mathcal{S} \cup \mathcal{P}$ e por $e(x|v, x_p)$ caso $v \in \mathcal{C}$, onde x_p representa o símbolo no topo da pilha. No cálculo de $\gamma(i, j, v, w)$ serão levadas em consideração somente as subsequências onde todos os estados de emissão pareada e seus respectivos estados sensíveis ao contexto estão formando par dentro da subsequência $y_i...y_j$. Por fim definimos as variáveis $\lambda_l(i, j, v, w)$ e $\lambda_r(i, j, v, w)$ necessárias para o procedimento de *trace-back* que irá computar a sequência de estados y^* mais provável.

Inicialização: calculamos o log da probabilidade de todas as subsequências que consistem em um único símbolo. Como consideramos apenas sequências de estados onde os estados de emissão pareada e sensíveis a contexto formam par dentro da sequência, inicializamos $\gamma(i, i, v, v)$ com o log da probabilidade do estado v emitir o símbolo x_i se $v \in \mathcal{S}$. Caso $v \in \mathcal{P} \cup \mathcal{C}$ inicializamos com $-\infty$.

Para $i = 1, ..., L; v = 2, ..., M - 1$.

$$\begin{aligned}
\gamma(i, i, v, v) &= \begin{cases} \log e(x_i|v) & v \in \mathcal{S} \\ -\infty & \text{caso contrário} \end{cases} \\
\lambda_l(i, i, v, v) &= (0, 0, 0, 0) \\
\lambda_r(i, i, v, v) &= (0, 0, 0, 0)
\end{aligned}$$

Iteração

Para $i = 1, ..., L - 1; j = i + 1, ..., L$ e $v = 2, ..., M - 1; w = 2, ..., M - 1$.

(i) $v \in \mathcal{C}$ ou $w \in \mathcal{P}$: como $y_i = v \in \mathcal{C}$ é o estado mais à esquerda de $y_i...y_j$, ele não formará par com seu respectivo estado de emissão pareada, pois tal estado não está dentro da subsequência $y_i...y_j$ (2.13). O mesmo é aplicado para $s_j = w \in \mathcal{P}$. Dessa forma atribuímos $-\infty$ à variável $\gamma(i, j, v, w)$.

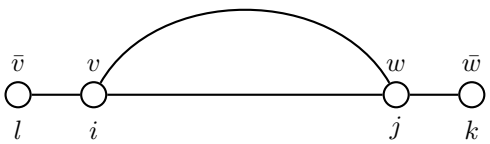


Figura 2.13: *Ilustração do caso (**i**) para o cálculo da sequência de estados mais provável.*

$$\begin{aligned}\gamma(i,j,v,w) &= -\infty \\ \lambda_l(i,j,v,w) &= (0,0,0,0) \\ \lambda_r(i,j,v,w) &= (0,0,0,0)\end{aligned}$$

(ii) $v \in \mathcal{P},\ w \in \mathcal{S}$: calculamos $\gamma(i,j,v,w)$ estendendo $\gamma(i,j-1,v,u)$ em um símbolo para a direita. Como o estado $y_j = w \in \mathcal{S}$ não possui interação com nenhum outro estado, temos que todos os estados de emissão pareada e sensíveis ao contexto formam par dentro de $y_i...y_{j-1}$ (2.14). Calculamos, então, $\gamma(i,j,v,w)$ com a probabilidade da sequência de estados mais provável $y_i...y_{j-1}$ ($\gamma(i,j-1,v,u)$), onde $y_{j-1} = u$. Adicionamos a probabilidade de transição do estado u para w ($t(u,w)$) mais a probabilidade de emissão do símbolo x_i pelo estado w ($e(x_i|w)$). Nesse cálculo, devemos escolher o estado u que irá maximizar o valor de $\gamma(i,j,v,w)$.

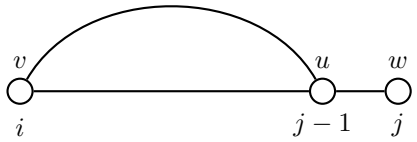


Figura 2.14: *Ilustração do caso (**ii**) para o cálculo da sequência de estados mais provável.*

$$\begin{aligned}\gamma(i,j,v,w) &= \max_u \left[\gamma(i,j-1,v,u) + \log t(u,w) + \log e(x_j|w)\right] \\ u^* &= \arg\max_u \left[\gamma(i,j-1,v,u) + \log t(u,w) + \log e(x_j|w)\right] \\ \lambda_l(i,j,v,w) &= (i,j-1,v,u^*) \\ \lambda_r(i,j,v,w) &= (j,j,w,w)\end{aligned}$$

(iii) $v \in \mathcal{S},\ w \in \mathcal{C}$: Similar ao caso **(ii)**, porém neste caso $\gamma(i,j,v,w)$ será obtido estendendo $\gamma(i+1,j,v,w)$ em um símbolo a esquerda (2.15).

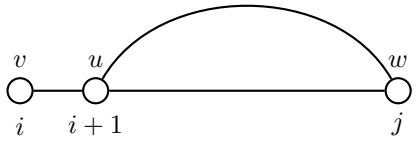


Figura 2.15: *Ilustração do caso (**iii**) para o cálculo da sequência de estados mais provável.*

$$\begin{aligned}\gamma(i,j,v,w) &= \max_u \left[\log e(x_i|v) + \log t(v,u) + \gamma(i+1,j,u,w)\right] \\ u^* &= \arg\max_u \left[\log e(x_i|v) + \log t(v,u) + \gamma(i+1,j,u,w)\right] \\ \lambda_l(i,j,v,w) &= (i,i,v,v) \\ \lambda_r(i,j,v,w) &= (i+1,j,u^*,w)\end{aligned}$$

(iv) $v = \mathcal{P}_n,\ w = \mathcal{C}_m\ (n \neq m),\ j < i + 3$: nesse caso, a sequência de estados ou é do formato $y_i y_{i+1}$ ou $y_i y_{i+1} y_{i+2}$, temos também que y_i e y_j não formam par e não existem estados intermediários suficientes para y_i e y_j formarem pares dentro da subsequência de estados, dessa forma atribuímos $-\infty$ à variável $\gamma(i, j, v, w)$.

$$\begin{aligned} \gamma(i,j,v,w) &= -\infty \\ \lambda_l(i,j,v,w) &= (0,0,0,0) \\ \lambda_r(i,j,v,w) &= (0,0,0,0) \end{aligned}$$

(v) $v = \mathcal{P}_n,\ w = \mathcal{C}_m\ (n \neq m),\ j \geq i + 3$: nesse caso, $v = y_i$ e $w = y_j$ não formam par, porém $y_i = P_n$ deve formar par com algum $y_k = \overline{v} = C_n$, onde $i + 1 \leq k \leq j - 2$. Similarmente, $y_j = C_m$ deve formar par com algum $y_l = \overline{w} = P_m$, onde $k + 1 \leq l \leq j - 1$ (2.16). Consequentemente todos os estados de emissão pareada e os estados sensíveis ao contexto devem existir em pares dentro de $y_i ... y_k$ e $y_{k+1} ... y_j$. Portanto, calculamos $\gamma(i, j, v, w)$ pela adição de $\gamma(i, j, v, \overline{v})$ com a probabilidade de transição $t(\overline{v}, u)$ e $\gamma(k + 1, j, u, w)$.

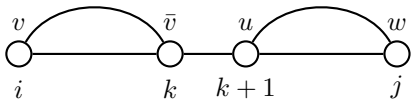


Figura 2.16: *Ilustração do caso (**v**) para o cálculo da sequência de estados mais provável.*

$$\begin{aligned} \gamma(i,j,v,w) &= \max_u \left(\max_{k=i+1,\ldots,j-2} [\gamma(i,k,v,\overline{v}) + \log t(\overline{v},u) + \gamma(k+1,j,u,w)] \right) \\ (k^*,u^*) &= \arg\max_{(u,k),k=i+1,\ldots,j-1} [\gamma(i,k,v,\overline{v}) + \log t(\overline{v},u) + \gamma(k+1,j,u,w)] \\ \lambda_l(i,j,v,w) &= (i,k^*,v,\overline{v}) \\ \lambda_r(i,j,v,w) &= (k^*+1,j,u^*,w) \end{aligned}$$

(vi) $v = \mathcal{P}_n,\ w = \mathcal{C}_n,\ j = i + 1$: nesse caso, $v = y_i$ e $w = y_j$ formam par e estão lado a lado. Dessa forma, calculamos $\gamma(i, j, v, w)$ pela adição da probabilidade de emissão do símbolo x_i pelo estado v ($e(x_i|v)$), com a probabilidade de transição do estado $s_i = v$ para o estado $s_j = w$ ($t(v, w)$), mais a probabilidade de emissão do símbolo x_j pelo estado w, visto que o símbolo armazenado na pilha foi x_p ($e(x_i|w, x_p)$).

$$\begin{aligned} \gamma(i,j,v,w) &= \log e(x_i|v) + \log t(v,w) \log e(x_j|w,x_i) \\ \lambda_l(i,j,v,w) &= (0,0,0,0) \\ \lambda_r(i,j,v,w) &= (0,0,0,0) \end{aligned}$$

(vii) $v = \mathcal{P}_n,\ w = \mathcal{C}_n,\ j > i + 1$: neste caso temos duas possibilidades a considerar. Uma das possibilidades é ilustrada pelo caso (**v**) onde v e w não formam par, a outra possibilidade é quando v e w formam par. Nessa última possibilidade, todos os estados de emissão pareada e os estados sensíveis ao contexto em $s_{i+1}..s_{j-1}$ devem estar pareados(Fig. 2.17). $\gamma(i, j, v, w)$ assumirá o maior valor entre γ_1 e γ_2.

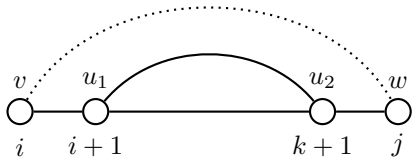


Figura 2.17: *Ilustração do caso (**vii**) para o cálculo da sequência de estados mais provável.*

$$
\begin{aligned}
\gamma_1 &= \max_u \left(\max_{k=i+1,\ldots,j-2} [\gamma(i,k,v,\overline{v}) + \log t(\overline{v},u) + \gamma(k+1,j,u,w)] \right) \\
(k^*,u^*) &= \arg\max_{(u,k),k=i+1,\ldots,j-1} [\gamma(i,k,v,\overline{v}) + \log t(\overline{v},u) + \gamma(k+1,j,u,w)] \\
\gamma_2 &= \max_{u_1,u_2} [\log e(x_i|v) + \log t(v,u_1) + \gamma(i+1,j-1,u_1,u_2) + \log t(u_2,w) + \log e(x_j|w,x_i)] \\
(u_1^*,u_2^*) &= \arg\max_{(u_1,u_2)} [\log e(x_i|v) + \log t(v,u_1) + \gamma(i+1,j-1,u_1,u_2) + \log t(u_2,w) + \log e(x_j|w,x_i)] \\
\gamma(i,j,v,w) &= \max(\gamma_1,\gamma_2)
\end{aligned}
$$

Se $\gamma_1 \geq \gamma_2$,

$$
\begin{aligned}
\lambda_l(i,j,v,w) &= (i,k^*,v,w) \\
\lambda_r(i,j,v,w) &= (k^*+1,j,u^*,w)
\end{aligned}
$$

Senão

$$
\begin{aligned}
\lambda_l(i,j,v,w) &= (i+1,j-1,u_1^*,u_2^*) \\
\lambda_r(i,j,v,w) &= (0,0,0,0)
\end{aligned}
$$

(viii) $v \in \mathcal{S},\ w \in \mathcal{S}$: Nesse caso, $\gamma(i,j,v,w)$ pode ser calculada usando **(ii)** ou **(iii)**.

Após completarmos os passos da iteração, podemos calcular $P(x,s^*|\lambda)$ computando $\gamma(1,L,v,w)$ para todo $v,w = 2,3,...,M-1$, como mostrado na finalização.

Finalização

$$
\begin{aligned}
\log P(\mathbf{x},\mathbf{s}^*|\lambda) &= \max_{v,w} [\log t(1,v) + \gamma(1,L,v,w) + \log t(w,M)] \\
(v^*,w^*) &= \arg\max_{(v,w)} [\log t(1,v) + \gamma(1,L,v,w) + \log t(w,M)] \\
\lambda^*(i,j,v,w) &= (1,L,v^*,w^*)
\end{aligned}
$$

Trace-Back

Após o cálculo da probabilidade do caminho de estados mais provável, podemos obter o caminho de estados y^* que deu origem a essa probabilidade. Para isso, iremos utilizar as variáveis λ_l e λ_r calculadas anteriormente. Usaremos também uma pilha T

Inicialização

$y_i = 0 (i = 1, 2, ..., L)$.

Empilhar λ^* em T.

Iteração

Desempilhar $\lambda_t = (i, j, v, w)$ de T.

Se $\lambda_t \neq (0, 0, 0, 0)$.

Se $y_i = 0$ então $y_i = v$.

Se $y_j = 0$ então $y_i = w$.

Empilhar $\lambda_l(\lambda_t)$ em T.

Empilhar $\lambda_r(\lambda_t)$ em T.

Se T estiver vazia, ir para **Terminação**.

Caso contrário, repetir **Iteração**.

Terminação

A sequência de estados ótima é $y^* = y_1 y_2 ... y_L$.

Análise de Complexidade

Vamos verificar a complexidade computacional do algoritmo. Para isso vamos fazer uma análise de cada caso de forma separada. Primeiramente, o algoritmo itera por $i = 1, ..., L-1, j = i+1, ..., L$ e $v = 2, ..., M-1, w = 2, ..., M-1$. Em cada caso a complexidade depende da classe dos estados v e w. Adote M_1 como a quantidade de estados de emissão pareada ou sensíveis ao contexto, M_2 a quantidade de estados de emissão única e por $M = 2 \times M_1 + M_2$. Abaixo a análise de cada caso:

- **i**: a complexidade para uma iteração é $O(1)$. Dessa forma, a complexidade total é $O(L^2 M_1 M)$, visto que esse caso somente será executado quando $v \in \mathcal{C}$ ou $w \in \mathcal{P}$.
- **ii**: a complexidade para uma iteração é $O(M)$, já que devemos percorrer por todos os estados. Dessa forma, a complexidade total é $O(L^2 M_1 M_2 M)$, visto que esse caso somente será executado quando $v \in \mathcal{P}$ e $w \in \mathcal{S}$.
- **iii**: podemos perceber que a complexidade para este caso é a mesma do caso **ii**.
- **iv**: A complexidade para uma iteração é $O(1)$. Dessa forma, a complexidade total é $O(L^2 M_1^2)$, visto que esse caso será executado somente quando $v \in \mathcal{P}_n$, $w \in \mathcal{C}_m (n \neq m)$ e $j < i + 3$.
- **v**: A complexidade para uma iteração é $O(ML)$, visto que devemos iterar por todos os estados e pela sequência de observações. Dessa forma, a complexidade total é $O(L^3 M_1^2 M)$, visto que esse caso será executado somente quando $v \in \mathcal{P}_n$, $w \in \mathcal{C}_m (n \neq m)$ e $j > i + 3$
- **vi**: A complexidade para uma iteração é $O(1)$. Portanto, a complexidade total é $O(LM_1)$, visto que esse caso somente é executado quando $j = i + 1$ e P_n, $w \in \mathcal{C}_n$.
- **vii**: Como temos dois casos a analisar, a complexidade para uma iteração é a soma da complexidade de cada caso. Temos então que, para uma iteração, a complexidade é $O(ML) + O(M^2)$. Portanto, a complexidade total é $O(L^3 M_1 M) + O(L^2 M_1 M^2)$, visto que esse caso será executado somente quando $j > i + 1$ e P_n, $w \in \mathcal{C}_n$

- **viii**: A complexidade para uma iteração é $O(M)$. Executamos esse caso $O(L^2M_2^2)$. Portanto a complexidade total é $O(L^2M_2^2M)$

A Tabela 2.1 mostra uma síntese da complexidade de cada caso junto com a complexidade total do algoritmo.

Tabela 2.1: *Análise de complexidade para cálculo da sequência de estados mais provável.*

Caso	Complexidade de uma iteração	Número de iterações	Complexidade total
i	$O(1)$	$O(L^2M_1M)$	$O(L^2M_1M)$
ii	$O(M)$	$O(L^2M_1M_2)$	$O(L^2M_1M_2M)$
iii	$O(M)$	$O(L^2M_1M)$	$O(L^2M_1M)$
iv	$O(1)$	$O(L^2M_1^2)$	$O(L^2M_1^2)$
v	$O(ML)$	$O(L^2M_1^2)$	$O(L^3M_1^2M)$
vi	$O(1)$	$O(L^2M_1)$	$O(L^3M_1^2M)$
vii	$O(ML)+O(M^2)$	$O(L^2M_1)$	$O(L^3M_1M)+O(L^2M_1M^2)$
viii	$O(M)$	$O(L^2M_2^2)$	$O(L^3M_2^2M)$
Total			$O(L^3M_1^2M)+O(L^2M_1M^2)+O(L^2M_2^2M)$

Pela analise de cada caso separadamente concluí-se que a complexidade, de forma geral, do algoritmo é $O(L^3M^3)$. Apesar da complexidade ser maior que a do algoritmo de Viterbi, utilizado em HMM, ainda é polinomial em L e M. Porém, devemos lembrar que csHMMs possuem um poder descritivo muito maior que HMMs tradicionais como ilustrado no início da Seção 2.4.

2.4.4 Algoritmo *Inside*

O algoritmo *Inside* tem por finalidade resolver o problema da avaliação. Esse algoritmo pode ser visto como uma variação do algoritmo para cálculo da sequência de estados mais provável, a diferença consiste em substituir as maximizações por somatórios. Iremos utilizar as mesmas notações utilizadas na Seção 2.4.3. Definimos, então, a variável $\alpha(i,j,v,w)$ como a soma das probabilidades de todas as subsequências $y_i...y_j$ com $y_i = v$ e $y_j = w$. Ao final do algoritmo, teremos calculado $P(x|\lambda)$. Abaixo temos o algoritmo completo.

Inicialização:

Para $i = 1,...,L; v = 2,...,M-1$.

$$\alpha(i,i,v,v) \quad = \quad \begin{cases} e(x_i|v) & v \in \mathcal{S} \\ -\infty & \text{caso contrário} \end{cases}$$

Iteração

Para $i = 1,...,L-1; j = i+1,...,L$ e $v = 2,...,M-1; w = 2,...,M-1$.

(i) $v \in \mathcal{C}$ ou $w \in \mathcal{P}$:

$$\alpha(i,j,v,w) \quad = \quad 0$$

(ii) $v \in \mathcal{P},\ w \in \mathcal{S}$:

$$\alpha(i,j,v,w) \quad = \quad \sum_u [\alpha(i,j-1,v,u)t(u,w)e(x_j|w)]$$

(iii) $v \in \mathcal{S},\ w \in \mathcal{C}$:

$$\alpha(i,j,v,w) \quad = \quad \sum_u [e(x_i|v)t(v,u)\alpha(i+1,j,u,w)]$$

(iv) $v = \mathcal{P}_n,\ w = \mathcal{C}_m\ (n \neq m),\ j < i+3$:

$$\alpha(i,j,v,w) \quad = \quad 0$$

(v) $v = \mathcal{P}_n,\ w = \mathcal{C}_m\ (n \neq m),\ j \geq i+3$:

$$\alpha(i,j,v,w) \quad = \quad \sum_u \sum_{k=i+1}^{j-2} \alpha(i,k,v,\overline{v})t(\overline{v},u)\alpha(k+1,j,u,w)$$

(vi) $v = \mathcal{P}_n,\ w = \mathcal{C}_n,\ j = i+1$:

$$\alpha(i,j,v,w) \quad = \quad e(x_i|v)t(v,w)e(x_j|w,x_i)$$

(vii) $v = \mathcal{P}_n,\ w = \mathcal{C}_n,\ j > i+1$:

$$\begin{aligned}\alpha(i,j,v,w) \quad = \quad & \sum_u \sum_{k=i+1}^{j-2} \alpha(i,k,v,w)t(w,u)\alpha(k+1,j,u,w) + \\ & \sum_{u_1} \sum_{u_2} [e(x_i|v)t(v,u_1)\alpha(i+1,j-1,u_1,u_2)t(u_2,w)e(x_j|w,x_i)]\end{aligned}$$

(viii) $v \in \mathcal{S},\ w \in \mathcal{S}$

Nesse caso, $\alpha(i,j,v,w)$ pode ser calculada usando **(ii)** ou **(iii)**.

Finalização

$$P(\mathbf{x}|\lambda) \quad = \quad \sum_v \sum_w t(1,v)\alpha(1,L,v,w)t(w,M)$$

Calculamos, então, a probabilidade $P(\mathbf{x}|\lambda)$ do modelo gerar a sequência de observação $\mathbf{x}$. A complexidade de tempo do algoritmo *Inside* é a mesma do algoritmo para calcular a sequência de estados mais provável.

2.4.5 Algoritmo *Outside*

O algoritmo *Outside* tem o mesmo propósito que o *Inside*. Definimos, aqui, a variável $\beta(i,j,v,w)$ como a soma das probabilidades de todas as subsequências $y_1...y_i y_j...s_L$, com exceção da subsequência $y_{i+1}...y_{j-1}$. O cálculo da variável β torna-se necessário para a resolução do problema do treinamento que será visto adiante.

Serão usadas aqui as mesmas notações e restrições que foram utilizadas na Seção 2.4.4. Para o cálculo da variável β será necessário a utilização da variável α previamente calculada. Segue o algoritmo.

Inicialização: no caso (**i**) da inicialização, temos uma sequência vazia a ser computada. Dessa forma, setamos $\beta(0,L+1,1,M) = 1$. No caso (ii), calculamos a probabilidade de todas as subsequências do formato $x_1...x_i$, onde $y_1 = u$ e $y_i = v$. Como todos os pareamentos devem ocorrer

dentro de $x_1...x_i$ e $\alpha(1,i,u,v)$ representa a probabilidade de todas as sequências de estados $x_1...x_i$, $\beta(i,L+1,v,M)$ é calculada pelo produto da transição do estado 1 para o estado u e $\alpha(1,i,u,v)$ $\forall u$. O caso (**iii**) é similar ao caso (**ii**), porém calculamos a probabilidade de todas as subsequências do formato $x_i...x_L$, onde $s_i = w$ e $s_l = u$.

Para $i = 1,...,L; v = 1,...,M$.

$$\textbf{(i)}\ \beta(0,L+1,v,w) \quad = \quad \begin{cases} 1 & v = 1, w = M \\ 0 & \text{caso contrário} \end{cases}$$

$$\textbf{(ii)}\ \beta(i,L+1,v,w) \quad = \quad \begin{cases} \sum_u t(1,u)\alpha(1,i,u,v) & w = M \\ 0 & \text{caso contrário} \end{cases}$$

$$\textbf{(iii)}\ \beta(0,i,v,w) \quad = \quad \begin{cases} \sum_u \alpha(i,L,w,u)t(u,M) & v = 1 \\ 0 & \text{caso contrário} \end{cases}$$

Iteração

Para $i = 1,...,L-1; j = i+1,...,L$ e $v = 1,...,M; w = 1,...,M$.

(i) $v = 1$ ou $w = M$

$$\beta(i,j,v,w) \quad = \quad 0$$

(ii) $v \in \mathcal{P},\ w \in \mathcal{P}$: $s_j = w$ deve formar par com algum $\overline{w}$ entre $j+1$ e $k-1$ e $s_i = v$ deverá formar par com algum $\overline{v}$ entre k e L, onde $j+2 \leq k \leq L+1$ (2.18). Utilizamos, então, a probabilidade das subsequências $s_j...s_{k-1}$ e $s_1...s_i s_{k+1}...s_L$, que podem ser encontradas nas variáveis $\alpha(j,k-1,w,\overline{w})$ e $\beta(i,k,v,u)$ respectivamente, para calcular $\beta(i,j,v,w)$.

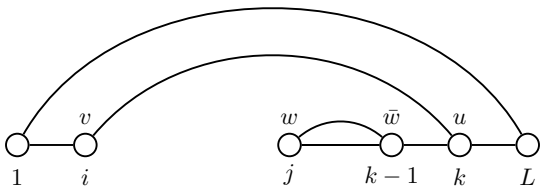


Figura 2.18: *Ilustração do caso (**ii**) do algoritmo Outside.*

$$\beta(i,j,v,w) \quad = \quad \sum_u \sum_{k=j+2}^{L+1} \beta(i,k,v,u)\alpha(j,k-1,w,\overline{w})t(\overline{w},u)$$

(iii) $v \in \mathcal{C},\ w \in \mathcal{C}$: Procedemos aqui de forma similar ao caso (**ii**).

$$\beta(i,j,v,w) \quad = \quad \sum_u \sum_{k=0}^{i-2} \beta(k,j,u,w)\alpha(k+1,i,\overline{v},v)t(u,\overline{v})$$

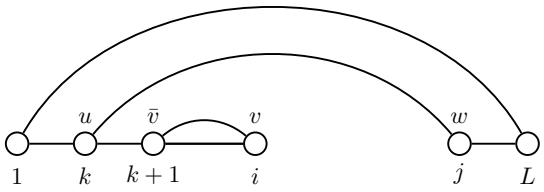


Figura 2.19: *Ilustração do caso (**iii**) do algoritmo Outside.*

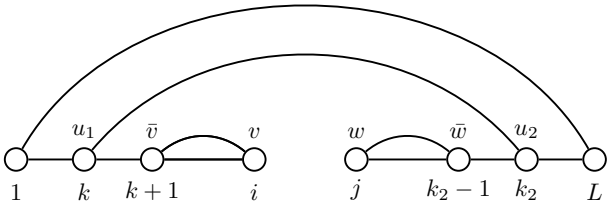


Figura 2.20: *Ilustração do caso (**iv**) do algoritmo Outside.*

(iv) $v \in \mathcal{C},\ w \in \mathcal{P}$: $s_i = v$ deve formar par com algum $\overline{v}$ entre $k_1 + 1$ e $i - 1$, da mesma forma $s_j = w$ deve formar par com algum $\overline{w}$ entre $j + 1$ e $k_2 - 1$, onde $0 \leq k_1 \leq i - 2$ e $j + 2 \leq k_2 \leq L + 1$ (2.20).

$$\begin{aligned}\beta(i,j,v,w) \quad = \quad & \sum_{u_1,u_2} \sum_{k_1=0}^{i-2} \sum_{k_2=j+2}^{L+1} \beta(k_1,k_2,u_1,u_2)\alpha(k_1+1,i,\overline{v},v) \\ & \times\alpha(j,k_2-1,w,\overline{w})t(u_1,\overline{v})t(\overline{w},u_2)\end{aligned}$$

(v) $v \notin \mathcal{S},\ w \in \mathcal{S}$: estendemos $\beta(i, j + 1, v, u)$ em um símbolo (2.21).

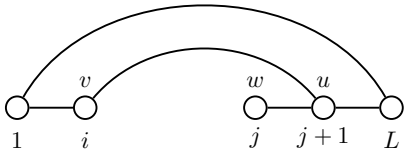


Figura 2.21: *Ilustração do caso (**v**) do algoritmo Outside.*

$$\beta(i,j,v,w) \quad = \quad \sum_{u} \beta(i,j+1,v,u)t(w,u)e(x_j|w)$$

(vi) $v \in \mathcal{S},\ w \notin \mathcal{S}$: similar ao caso **(v)**, também estendemos $\beta(i - 1, j, v, u)$ em um símbolo (2.22).

$$\beta(i,j,v,w) \quad = \quad \sum_{u} \beta(i-1,j,u,w)t(u,v)e(x_i|v)$$

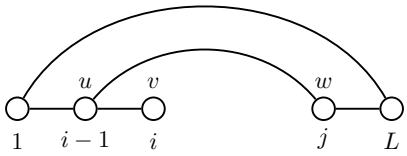


Figura 2.22: *Ilustração do caso (**vi**) do algoritmo Outside.*

(vii) $v = \mathcal{P}_n,\ w = \mathcal{C}_m (n \neq m)$: $s_i = P_n$ e $s_j = C_m$ não formar par, visto $n \neq m$, dessa forma atribuímos o valor 0 variável β.

$$\beta(i, j, v, w) \quad = \quad 0$$

(viii) $v = \mathcal{P}_n,\ w = \mathcal{C}_n$: $s_i = v = P_n$ e $s_j = w = C_n$ formam par (2.23). $\beta(i, j, v, w)$ pode ser calculada estendendo $\beta(i-1, j+1, u_1, u_2)$.

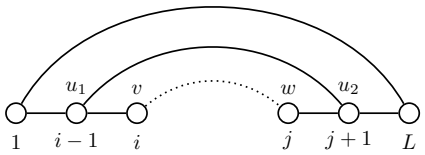


Figura 2.23: *Ilustração do caso (**viii**) do algoritmo Outside.*

$$\beta(i, j, v, w) \quad = \quad \sum_{u_1, u_2} \beta(i-1, j+1, u_1, u_2) t(u_1, v) e(x_i|v) e(x_j|w, x_i) t(w, u_2)$$

(ix) $v \in \mathcal{S},\ w \in \mathcal{S}$: podemos calcular $\beta(i, j, v, w)$ através do caso **(v)** ou do caso **(vi)**.

Finalização: Aqui, precisamos apenas estender os símbolos adjacentes para obtermos a probabilidade total. O calculo é mostrado a seguir.

$$P(\mathbf{x}|\lambda) \quad = \quad \sum_{v,w} \beta(i, i+1, v, w) t(v, w) \qquad \text{para qualquer } i.$$

Análise de Complexidade

Yoon [YV06] não faz uma análise de complexidade do algoritmo *Outside*, visto que ele é usado somente para treinar o modelo e, portanto, sua complexidade não é um problema. De toda forma, fizemos uma análise da complexidade do algoritmo, que pode ser vista na Tabela 2.2.

2.4.6 Estimativa dos Parâmetros do Modelo

Assim como em HMM, devemos ajustar os parâmetros do modelo de forma a maximizar $P(x|\lambda)$. Utilizaremos o método de esperança e maximização para calcular esses parâmetros. Para simplificar, iremos mostrar como os parâmetros podem ser calculados para uma única sequência de observação. Vamos definir as seguintes variáveis:

Tabela 2.2: *Análise de complexidade para o algoritmo Outside.*

Caso	Complexidade de uma iteração	Número de iterações	Complexidade total
i	$O(1)$	$O(L^2M^2)$	$O(L^2M^2)$
ii	$O(ML)$	$O(L^2M_1^2)$	$O(L^3M_1^2M)$
iii	$O(ML)$	$O(L^2M_1^2)$	$O(L^3M_1^2M)$
iv	$O(M^2L^2)$	$O(L^2M_1^2)$	$O(L^4M_1^2M^2)$
v	$O(M)$	$O(L^2M_1M_2)$	$O(L^2M_1M_2M)$
vi	$O(M)$	$O(L^2M_1M_2)$	$O(L^2M_1M_2M)$
vii	$O(1)$	$O(L^2M_1^2)$	$O(L^2M_1^2)$
viii	$O(M^2)$	$O(L^2M_1)$	$O(L^2M_1M^2)$
ix	$O(ML)$	$O(L^2M_2^2)$	$O(L^3M_2^2M)$
Total			$O(L^4M_1^2M^2)$

$$\begin{aligned}\tau_i(v,w) &= \text{A probabilidade de } s_i = v \text{ e } s_j = w \text{ dado } \lambda \text{ e o símbolo da observação } x.\\ \sigma_i(v) &= \text{A probabilidade de } s_i = v \text{ dado } \lambda \text{ e a observação } x.\\ \delta_v(i,j) &= \text{A probabilidade de } s_i = v \text{ e } s_j = \overline{v} \text{ sejam complementares.}\end{aligned}$$

Podemos calcular $\tau_i(v,w)$ da seguinte forma:

$$\tau_i(v,w) = \frac{\beta(i,i+1,v,w)t(v,w)}{P(x|\lambda)}. \tag{2.27}$$

A probabilidade $\sigma_i(v)$ pode ser obtida simplesmente pela adição de $\tau_i(v,w)$ para todo w

$$\sigma_i(v) = \sum_w \tau_i(v,w). \tag{2.28}$$

E por último, a probabilidade $\delta_v(i,j)$ pode ser obtida através de

$$\delta_v(i,j) = \frac{\sum_{u_1,u_2} \alpha(i,j,v,\overline{v})\beta(i-1,j+1,u_1,u_2)t(u_1,v)t(\overline{v},u_2)}{P(x|\lambda)}. \tag{2.29}$$

A partir das probabilidades acima podemos, então, reestimar os parâmetros do modelo.

Probabilidades de Transição

Para estimar a probabilidade de transição de v para w, onde $v \in \mathcal{P} \cup \mathcal{S}$, devemos usar

$$\hat{t}(v,w) = \frac{\text{Número esperado de transições de } v \text{ para } w}{\text{Número esperado de transições a partir de } v} = \frac{\sum_{i=0}^{L}\tau_i(v,w)}{\sum_{i=0}^{L}\sigma_i(v)}. \tag{2.30}$$

Para $v = C_n$, as transições permitidas dependem, de a memória auxiliar associada estar ou não vazia. Dessa forma, se $w \in \mathcal{E}_n$

$$\begin{aligned}\hat{t}(v,w) &= \frac{\text{Número esperado de transições de } v = C_n \text{ para } w \in \mathcal{E}_n}{\text{Número esperado de transições de } v = C_n \text{ para qualquer estado pertencente a } \mathcal{E}_n}\\ &= \frac{\sum_{i=0}^{L}\tau_i(v,w)}{\sum_{i=0}^{L}\sum_{u\in\mathcal{E}_n}\tau_i(v,u)}.\end{aligned} \tag{2.31}$$

Se $w \in \mathcal{F}_n$

$$\begin{aligned}\hat{t}(v,w) &= \frac{\text{Número esperado de transições de } v = C_n \text{ para } w \in \mathcal{F}_n}{\text{Número esperado de transições de } v = C_n \text{ para qualquer estado pertencente a } \mathcal{F}_n} \\ &= \frac{\sum_{i=0}^{L} \tau_i(v,w)}{\sum_{i=0}^{L} \sum_{u \in \mathcal{F}_n} \tau_i(v,u)}. \end{aligned} \tag{2.32}$$

Probabilidade de Emissão

Para $v \in \mathcal{P} \cup \mathcal{S}$, a probabilidade de emissão não depende do contexto. Dessa forma, podemos estimar a probabilidade utilizando a equação a seguir:

$$\begin{aligned}\hat{e}(x|v) &= \frac{\text{Número esperado de vezes que o símbolo } x_i \text{ foi emitido no estado } v}{\text{Número esperado de ocorrências do estado } v} \\ &= \frac{\sum_{i=1|x_i=x}^{L} \sigma_i(v)}{\sum_{i=1}^{L} \sigma_i(v)}. \end{aligned} \tag{2.33}$$

Para $v \in \mathcal{C}$, a probabilidade de emissão depende do contexto, isto é, depende do símbolo x_p que se encontra na memória auxiliar. Sendo assim, devemos estimar tal probabilidade como mostrado a seguir:

$$\begin{aligned}\hat{e}(x|v) &= \frac{\text{Número esperado de emissões de } x \text{ no estado } v \in \mathcal{C} \text{ dado o contexto } x_p}{\text{Número esperado de emissões no estado } v \in \mathcal{C} \text{ dado o contexto } x_p} \\ &= \frac{\sum_{j=2|x_j=x}^{L} \sum_{i=1|x_i=x_p}^{j-1} \delta_v(i,j)}{\sum_{j=2}^{L} \sum_{i=1|x_i=x_p}^{j-1} \delta_v(i,j)}. \end{aligned} \tag{2.34}$$

Com os valores de $\hat{t}(v,w)$, $\hat{e}(x|v)$ e $\hat{e}(x|v,x_p)$, podemos atualizar os parâmetros do modelo e repetir o processo até que um critério de parada seja satisfeito.

2.5 Modelo de Markov de Estados Ocultos Sensível ao Contexto Para Perfil de Sequências

Proteínas, RNAs e outras características do genoma geralmente são classificados em famílias de sequências e estruturas relacionadas [HGP+97]. Sequências de uma mesma família normalmente possuem as mesmas funções e características, de forma que é uma tarefa comum procurar saber as características que os membros de uma família compartilham. Ao encontrarmos tais características estamos fazendo um perfil das sequências, as características extraídas podem então ser usadas para encontrar novos membros pertencentes à família em questão.

O modelo de Markov de estados ocultos sensível ao contexto para perfil de sequências (*Profile* csHMM, do inglês *Profile Context Sensitive Hidden Markov Model*) é uma variação do modelo de Markov de estados ocultos sensível ao contexto. O *Profile*-csHMM foi proposto por Yoon e Vaidynathan [YV08b] e tem por finalidade fazer o perfil de sequências de RNA. O csHMM possui várias semelhanças estruturais com o *Profile* HMM. Porém, este último tem por objetivo fazer o perfil de sequências de proteínas. Ambos diferem quanto ao algoritmo para encontrar a sequência

de estados mais provável, além de que, o *Profile* chHMM usa alguns dos conceitos apresentados na seção 2.4. Existem outros modelos que possuem o objetivo de fazer o perfil de sequências de RNA, entre eles, o mais difundido são os Modelos de Covariância. Tal modelo foi proposto por Eddy et al. [ED94], e é baseado em gramáticas estocásticas livres de contexto. Porém, o mesmo não é capaz de lidar com pseudo-nós, o que cria uma vantagem para o Profile csHMM, que a princípio é capaz de lidar com qualquer tipo de pseudo-nós.

2.5.1 Inferindo um Profile HMM Sensível ao Contexto

Krogh e colegas [KBM+94] elaboraram uma arquitetura baseada em HMM (*Profile* HMM) que consegue representar de forma adequada perfis de sequências de proteínas. Essa arquitetura possui 3 classes de estados ocultos chamados estado de correspondência, de inserção e de deleção No *Profile*-csHMM temos as mesmas classes de estados, com a diferença de que no *Profile* csHMM existe três tipos de estados de correspondência, que são os mesmos que foram definidos na seção 2.4.1. Iremos ilustrar como derivar um *Profile* HMM a partir de um alinhamento múltiplo de RNA. Iremos utilizar o exemplo mostrado em [YV08b].

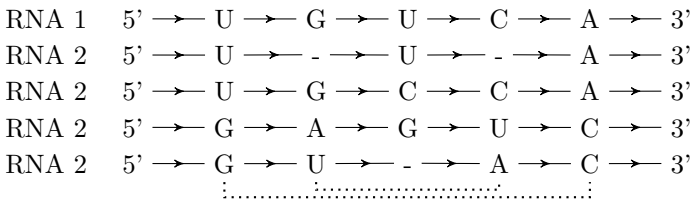


Figura 2.24: *Alinhamento múltiplo de cinco sequências de RNA (as linhas pontilhadas representam interações entre as colunas do alinhamento).*

Suponha que temos o alinhamento múltiplo indicado na figura 2.24. Devemos, primeiramente, definir o tamanho do modelo. Esse valor depende de quais colunas do alinhamento múltiplo serão tratadas como correspondências e quais colunas serão tratadas como inserções. Uma abordagem simples é modelar como inserções as colunas nas quais metade dos resíduos correspondem a lacunas, caso contrário, modelar como correspondências. Dessa forma, no alinhamento da figura 2.24, o modelo correspondente terá 5 estados de correspondência. Salientamos também que as colunas 1-4 e 2-5 formam pareamento de bases. Portanto, temos que modelar os estados M_1 e M_2 como estados de emissão pareada e os estados M_4 e M_5 como estados sensíveis ao contexto. Modelamos também os estados inicial e final. Temos então que o modelo será como mostrado na figura 2.25. Porém, o mesmo não está modelado para tratar inserções e deleções.

O modelo acima descrito tem utilidade um tanto quanto limitada, pois mesmo sequências homólogas podem possuir inserções ou deleções de resíduos. Devemos, então, modelar essas inserções e deleções no modelo da figura 2.25 e, para isso, devemos adicionar estados de deleção e inserção. Consideremos o caso em que uma determinada observação é maior que o consenso do alinhamento. Nesse caso, podemos dizer que a observação sofreu inserções e os estados de inserção irão tratar esse caso. Similarmente temos os estados de deleção para o caso no qual a observação é menor que o consenso do alinhamento. Devemos ressaltar que os estados de deleção não emitem símbolo algum. Temos então que o modelo final será como mostrado na figura 2.26

Devemos, agora, calcular as probabilidades de emissão e transição do modelo. Para isso, podemos

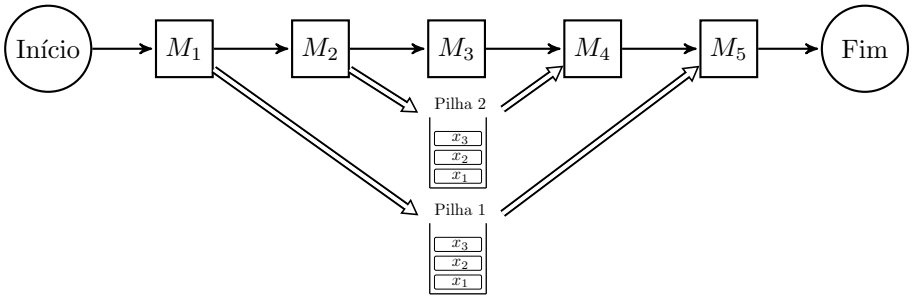


Figura 2.25: *Profile csHMM sem inserções ou deleções.*

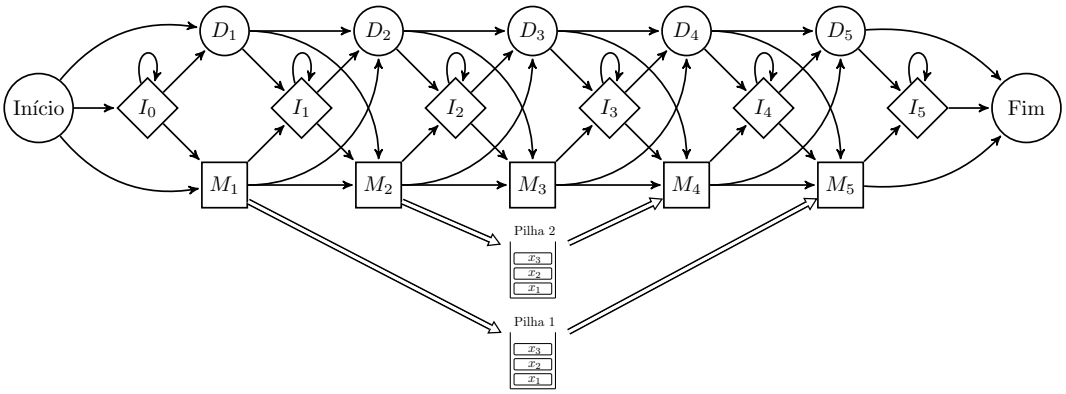


Figura 2.26: *Profile csHMM completo.*

usar as equações

$$a_{kl} = \frac{A_{kl}}{\sum_{l'} A_{kl'}},\ e_k(b) = \frac{E_k(b)}{\sum_{b'} E_k(b')} \tag{2.35}$$

para calcular tais probabilidades. Agora que temos o modelo que representa o perfil das sequências, podemos utilizar esse modelo para encontrar novos RNA que se assemelham ao perfil da família.

Yoon e Vaidynathan propuseram um algoritmo para resolver o problema da decodificação chamado *Sequential Component Adjoining* (SCA). Segundo Yoon o algoritmo proposto é capaz de lidar com qualquer tipo de pseudo-nó. Yoon define duas regras que devem ser aplicadas no passo do algoritmo, porém não mostra como devemos aplicá-las. Entramos em contato com o autor com o intuito de obter o código fonte da implementação. Porém, o autor nos informou que não possuía o código fonte da implementação. Tentamos descompilar os binários disponíveis na página pessoal de Yoon, porém, o máximo que conseguimos foram os nomes das funções, o que revelou a existência de um conjunto de 27 regras de junção. Devido esses problemas, optamos por desenvolver novas versões dos algoritmos de cálculo da sequência de estados mais provável e *Inside* adaptadas especificamente para *Profile*-csHMM.

Capítulo 3

Estendendo o ToPS

3.1 ToPS

ToPS (do inglês *Toolkit of Probabilistic Model od Sequências*) é um arcabouço orientado a objetos implementado utilizando-se a linguagem C++. Inicialmente desenvolvido por Kashiwabara [Kas12] com o propósito de manipular modelos probabilísticos, tal arcabouço facilita a caracterização de sequências heterogêneas, onde vários modelos probabilísticos são necessários para segmentar as diferentes entidades presentes nas sequências. A princípio, Kashiwabara estava particularmente interessado na predição de genes, onde é comum a utilização de um modelo agregador - e vários submodelos. No entanto, o ToPS é um arcabouço genérico capaz de analisar qualquer sequência de símbolos de um alfabeto discreto.

ToPS foi desenvolvido seguindo os padrões de projeto descritos em *Design Patterns: Elements of Reusable Object-Oriented Software* [GHJV94], o que lhe confere facilidade de entendimento e alta flexibilidade para extensão. Outra característica importante do ToPS é o conjunto de ferramentas disponibilizadas que podem ser utilizadas para gerar e rotular sequências, treinar modelos, etc. Para isso, precisamos apenas passar como parâmetro um arquivo de texto que descreve a entrada a ser usada. Esse arquivo é descrito utilizando-se uma linguagem simples e bem próxima da linguagem matemática. Esse conjunto de características acelera a definição de modelos, visto que para modificarmos parâmetros ou arquiteturas são necessárias simples modificações no arquivo de texto. A figura 3.1 representa as ferramentas disponíveis pelo arcabouço, onde as caixas com bordas arredondadas representam as ferramentas que mencionamos e as caixas sem bordas arredondadas representam processos manuais, arquivos de configuração ou arquivos de saída.

3.1.1 Linguagem

A fim de manter um caráter simples e intuitivo, o ToPS possui uma linguagem de definição de arquivos de entrada muito conveniente. Tome como exemplo o csHMM definido abaixo para a linguagem dos palíndromos sobre o alfabeto $\{A, B\}$, cuja representação gráfica pode ser vista através da figura 3.2.

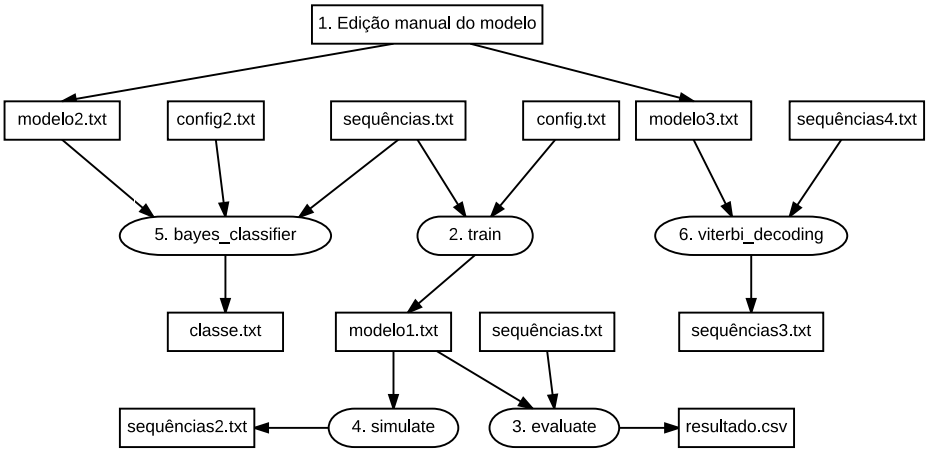


Figura 3.1: *Diagrama de uso do ToPS.*

Palíndromos

```
model_name = "ContextSensitiveHiddenMarkovModel"
state_names = ("INI","P1","S1","C1","FIM")
observation_symbols = ("A","B")
transitions = ("P1" | "INI": 0.5;
               "S1" | "INI": 0.5;
               "P1" | "P1": 0.333333;
               "S1" | "P1": 0.333333;
               "C1" | "P1": 0.333333;
               "C1" | "S1": 0.5;
               "FIM" | "S1": 0.5;
               "FIM" | "C1, empty": 1;
               "C1" | "C1, not_empty": 1)
emission_probabilities = ("A" | "P1": 0.5;
                          "B" | "P1": 0.5;
                          "A" | "S1": 0.5;
                          "B" | "S1": 0.5;
                          "A" | "C1, A": 1;
                          "B" | "C1, A": 0;
                          "A" | "C1, B": 0;
                          "B" | "C1, B": 1)
initial_probabilities = ("INI":  1)
```

Perceba que a definição das probabilidades do modelo é muito parecida com a definição formal, por exemplo $p(P_1|INI) = 0.5$. Essa característica torna a modificação dos parâmetros uma tarefa simples, além de que não é necessário nenhum conhecimento prévio em programação para definir os modelos.

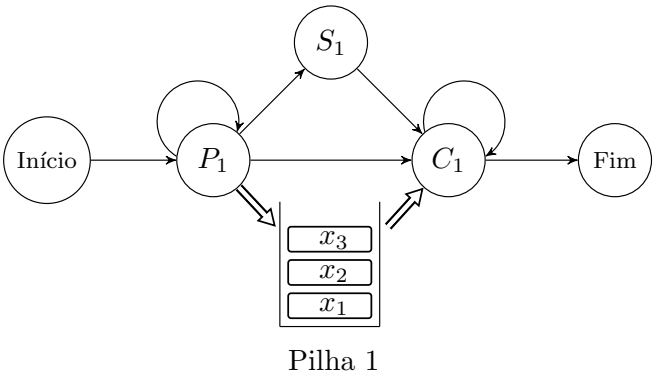


Figura 3.2: *HMM sensível ao contexto que gera somente palíndromos.*

3.1.2 Arquitetura

ProbabilisticModel é a raiz da arquitetura de classes do ToPS. Todos os modelos probabilísticos implementados são subclasses de *ProbabilisticModel*, como podemos ver na figura 3.3. Ressaltamos aqui o papel da classe abstrata *DecodableModel*. Tal classe possui a interface para os algoritmos *viterbi, forward e backward*. Os algoritmos dos modelos implementados não possuem exatamente esses nomes, mas possuem os mesmos objetivos. Dessa forma, temos *HiddenMarkovModel* como subclasse de *Decodable Model*.

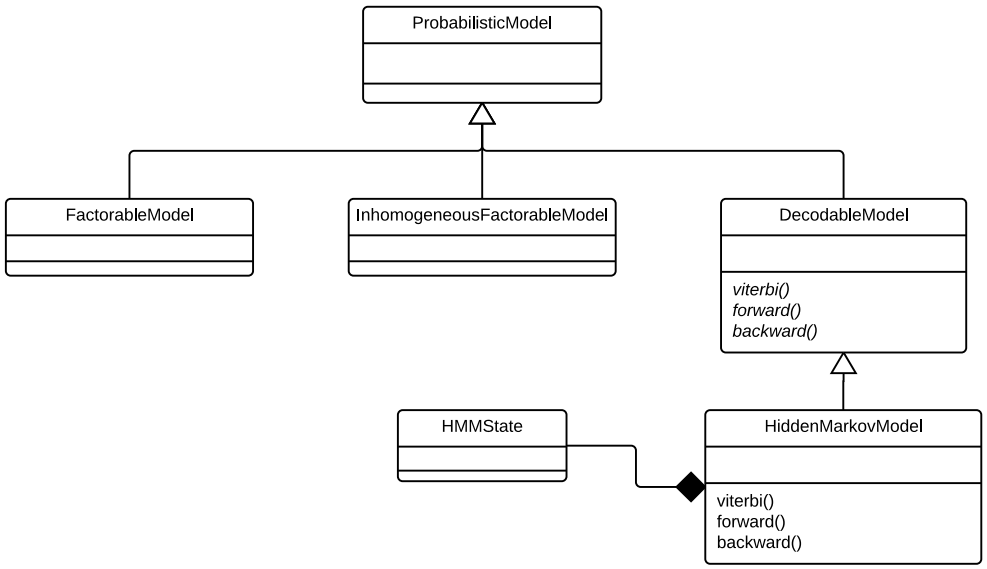


Figura 3.3: *Modelo de classes original do ToPS.*

3.2 Extensões

3.2.1 Modelagem

Como dito anteriormente, o ToPS é um arcabouço orientado a objetos, característica que facilita sua extensão. Desenvolvemos uma modelagem com a intenção de reutilizar ao máximo a arquitetura já existente. Optamos, então por definir as classes *ContextSensitiveHMM* e *ProfileContextSensiti-*

veHMM. *HiddenMarkovModel* é superclasse de *ContextSensitiveHMM*, que por sua vez é superclasse de *ProfileContextSensitiveHMM*.

Os métodos e atributos de *HiddenMarkovModel* foram herdados e reescritos adequadamente para os novos modelos, o que nos deu uma vantagem de modelagem, pois podemos reaproveitar partes da interface de *HiddenMarkovModel*. Criamos, também, duas subclasses da classe *HMMState*, *PairwiseState* e *ContextSensitiveState* para representar os estados de emissão pareada e sensível ao contexto, respectivamente. Por fim, definimos a classe *Stack* que irá compor as classes *PairwiseState* e *ContextSensitiveState*, visto que essas devem compartilhar uma memória auxiliar dedicada. O modelo de classes estendido pode ser visualizado na Figura 3.4.

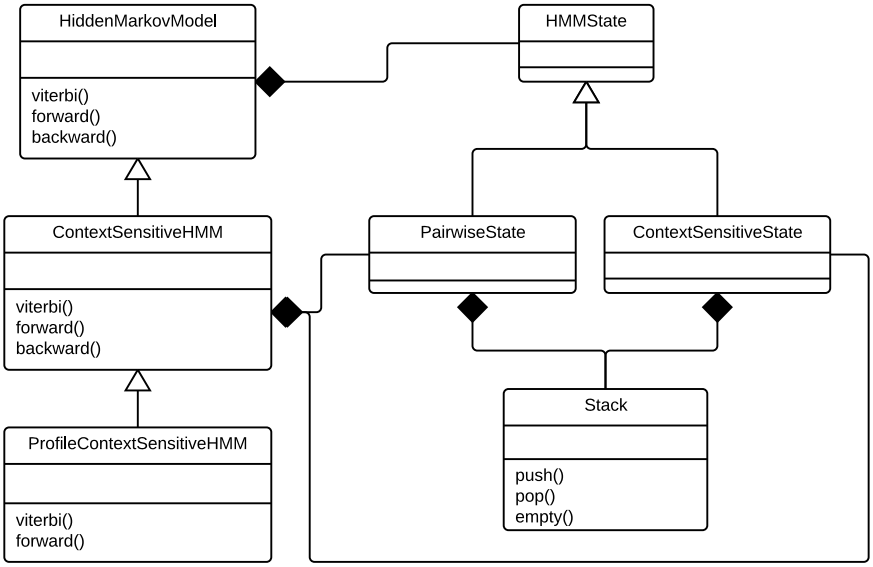


Figura 3.4: *Modelo de classes estendido do ToPS para csHMM e Profile-csHMM.*

3.2.2 Otimizações

Modelo de Markov de Estados Ocultos Sensível ao Contexto

Buscamos, durante nossa implementação, minimizar o tempo de execução dos algoritmos de inferência, visto a alta complexidade computacional exigida pelos mesmos. Podemos perceber que todos os csHMMs descritos na Seção 2.4 possuem estrutura linear. É possível ver outro exemplo através da Figura 3.5, onde temos a representação da estrutura secundária consenso dos elementos de resposta ao ferro (IRE, do inglês *iron response elements*). IREs são moléculas com estrutura secundária no formato de *hairpin*. IREs são encontrados nas regiões não codificantes de vários RNAs mensageiros, essas moléculas ligam-se às proteínas reguladoras de ferro (IRP, do inglês *iron regulatory proteins*) controlando o metabolismo do ferro dentro das células [HK96]. As bases sombreadas da figura 3.5 representam regiões de alta conservação na estrutura consenso.

Podemos representar a estrutura secundária consenso dos IREs através do csHMM da figura 3.6. Perceba que tal representação também possui estrutura linear. Dessa forma, as variáveis $\gamma(i, j, v, w)$, $\alpha(...)$ e $\beta(...)$ não terão valores atribuídos para $w < v$. Assim, propomos uma otimização com base nessa observação. Para tal, alteramos o passo da iteração e alguns dos casos dos algoritmos de inferência de forma a não serem executados nos casos onde temos $w < v$. Essa variante do

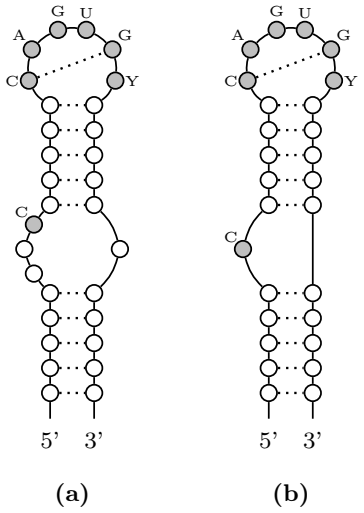


Figura 3.5: *Estrutura secundária consenso dos elementos de resposta ao ferro.*

algoritmo original é utilizada automaticamente na presença de um modelo com estrutura linear. Além disso, diferente dos algoritmos para HMM do ToPS, incluímos em nossa implementação uma lista de adjacência para os estados do modelo, o que minimizou à complexidade computacional do algoritmo que calcula a sequência de estados mais provável e *Inside* de $O(L^3M^3)$ para $O(L^3M^2K)$, onde $K \leq M$ é o grau máximo do grafo que representa o modelo.

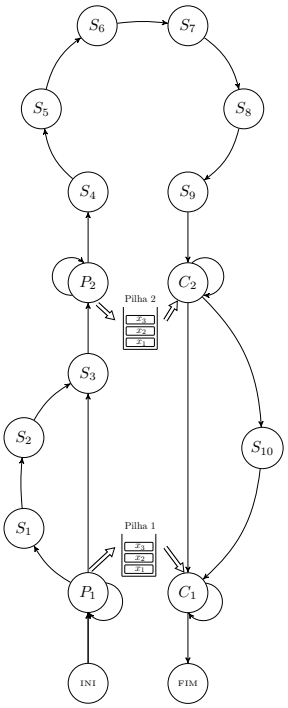


Figura 3.6: *CsHMM para representar a estrutura secundária consenso de IREs.*

Modelo de Markov de Estados Ocultos Sensível ao Contexto para Perfil de Sequências

Como dito na seção 2.5, *Profile*-csHMM podem ser vistos como uma subclasse de csHMM com uma estrutura muito bem definida que consegue representar de forma mais eficiente perfis de famílias de RNAs.

Antes de prosseguirmos com os detalhes a respeito das extensões que implementamos no ToPS para *Profile*-csHMM, é importante ressaltar que um dos objetivos deste trabalho era implementar o algoritmo SCA [YV08b]. Tal algoritmo é utilizado para calcular $P(x, y^*|\lambda)$ e a princípio é capaz de lidar com qualquer tipo de pseudonó. Tínhamos como objetivo também estender a ideia do SCA para calcular $P(X|\lambda)$. Porém, como dito anteriormente na Seção 2.5.1, o algoritmo se mostrou incompleto e ambíguo.

Algoritmos de Inferência

Desenvolvemos versões dos algoritmos de cálculo da sequência de estados mais provável e *Inside* adaptadas especificamente para *Profile*-csHMM. As versões desses algoritmo foram desenvolvidas de forma à lidar mais eficientemente com a estrutura linear do *Profile*, nos permitindo fazer uso da otimização para modelos lineares proposta para csHMM. Perceba que, através dessa otimização, a complexidade de tempo dos algoritmos de inferência sofre uma redução de $O(L^3M^2K)$ para $O(L^3M^2)$, pois o grau máximo do grafo que representa o *Profile* é três como podemos ver na Figura 2.26.

Restrição da Região de Correspondência

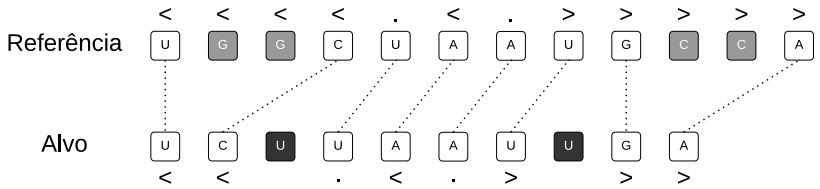
Introduzimos em nossa implementação do *Profile*-csHMM um parâmetro configurável D, que representa a região de correspondência de uma determinada sequência de entrada com o *Profile*. A introdução desse parâmetro é motivada pelo fato de que, ao submetermos uma sequência ao *Profile*, as regiões de correspondência tendem a estar próximas uma da outra. Por exemplo, a Figura 3.7 ilustra dois casos onde a distância entre as bases correspondentes é dois (**a**) e quatro (**b**). É pouco provável que bases muito distantes sejam correspondentes.

A introdução do parâmetro D foi proposto por Yoon [YV08b] para o algoritmo SCA, porém podemos facilmente aplicá-la em nossa implementação. Restringimos a região de busca por determinada base x_i do RNA alvo que faz correspondência com a coluna k do *Profile* considerando apenas as bases entre o intervalo $[k - d_1, k + d_2]$, como podemos observar pela Figura 3.8. Ao limitarmos a região de busca, a complexidade de tempo dos algoritmos de inferência passa a ser $O(LD^2M^2)$.

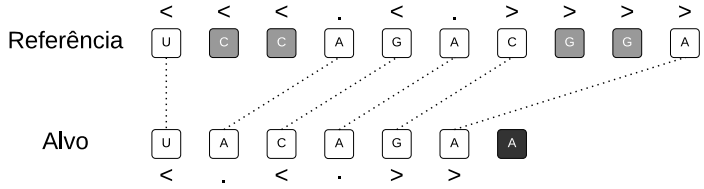
O parâmetro D pode ser estimado encontrando-se a maior distância entre bases correspondentes no alinhamento múltiplo, e então fazer D ser um pouco maior.

Otimização de Memória

Durante o desenvolvimento do *Profile*-csHMM nos deparamos com um problema de memória. Observe que o algoritmo que calcula a sequência de estados mais provável precisa definir as variáveis γ, λ_l e λ_r, que são vetores de quatro dimensões de cardinalidade $|L| * |L| * |M| * |M|$, onde L representa o tamanho da sequência de entrada e M a quantidade de estados do *Profile*. Essas variáveis necessitam de uma quantidade de memória muito grande. Para ilustrar melhor o problema, considere a família AfaR do banco de dados Rfam [NBB+14]. Tal família é composta por duas



(a) *Exemplo de alinhamento onde a distância máxima entre bases correspondentes é dois.*



(b) *Exemplo de alinhamento onde a distância máxima entre bases correspondentes é quatro.*

Figura 3.7: *Exemplos de distâncias entre bases correspondentes. Os quadrados com sombreamento mais claro representam inserções e os mais escuros representam deleções*

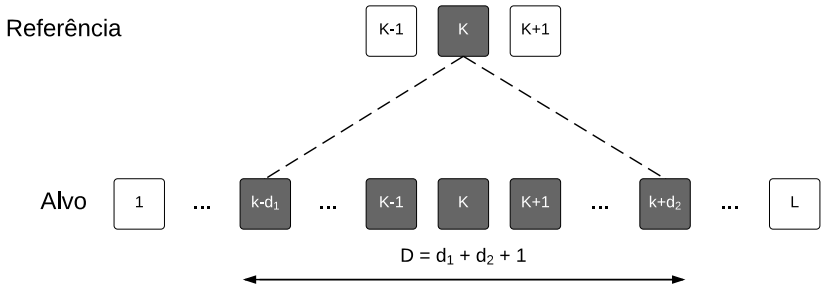


Figura 3.8: *Restrição da região de correspondência.*

sequências com 235 nucleotídeos de comprimento. O *Profile* resultante desse alinhamento terá 708 estados, pois todas as colunas do alinhamento serão tratadas como correspondências. Ao submetermos uma sequência alvo com 235 nucleotídeos ao *Profile* resultante do alinhamento, haverá a necessidade de 618GB de memória[1], tornando o uso do *Profile* impossível na prática.

Buscamos, então, em nossa implementação, minimizar a quantidade de memoria exigida pelo *Profile*. Perceba que as variáveis γ, λ_l e λ_r não terão valores atribuídos para $j < i$ e $w < v$ devido a estrutura linear do *Profile*. Dessa forma, criamos uma estrutura de dados que utiliza um vetor de uma dimensão que aloca memória apenas para as posições de α, λ_l e λ_r que terão valores atribuídos. Se olharmos as variáveis γ, λ_l e λ_r como uma matriz de dimensões $|L| * |L|$ denotada por γ_1, onde cada posição contém uma outra matriz de dimensões $|M| * |M|$ denotada por γ_2, as posições que terão valores atribuídos são representadas pelas matrizes triangulares superiores de γ_1 e γ_2. Através dessa otimização, ao submetermos uma sequência de 235 nucleotídeos ao *Profile* da família AfaR, as variáveis γ, λ_l e λ_r, necessitarão, agora, de 158GB de memória.

[1]Esses valores foram baseados no compilador g++ para OSX 10.10.4

Capítulo 4

Testes e Validações

Neste capítulo apresentamos os testes de validações realizados a fim de verificar o funcionamento de nossa implementação de csHMM e *Profile* csHMM, bem como analisar o impacto das otimizações implementadas. Todos os experimentos foram realizados em um *MacBook Air* com processador *Intel Core i5* de 1,7GHz e 4 GB de memória RAM. Antes de proseguirmos, devemos informar que o alto custo de memória exigido pelo *Profile*-csHMM limitou o espaço possível de testes.

4.1 Modelo de Markov de Estados Ocultos Sensível ao Contexto

4.1.1 Validação da Implementação

Procuramos investigar, primeiramente, a corretude dos algoritmos implementados. Para tal, utilizamos o csHMM representado pela figura 4.1 que modela a estrutura secundária consenso dos IREs. Considere a sequência de símbolos $x = CUGUCACAGUGUUGG$. Perceba que existem várias sequências de estados que podem dar origem a x. Utilizando força bruta, geramos todas a possíveis sequências de estados para a observação x e calculamos a probabilidade dessas sequências. A partir dessas probabilidades, pudemos calcular $P(x|\lambda) = 5.0183 \times 10^{-7}$ e $P(x, y^*|\lambda) = 4.7932 \times 10^{-7}$.

Calculamos, então, a probabilidade da sequência de estados mais provável para a observação x utilizando o algoritmo implementado e obtivemos $y^* = P_1S_1S_2S_3P_2P_2S_4S_5S_6S_7S_8S_9C_2C_2C_1$, onde $P(x, y^*|\lambda) = 4.7932 \times 10^{-7}$, o que coincidiu com o resultado obtido através de força bruta. De forma similar, utilizamos o algoritmo *Inside* para calcular a probabilidade da sequência x ter sido gerada pelo modelo e obtivemos $P(x|\lambda) = 5.0183 \times 10^{-7}$, o mesmo valor obtido através da força bruta.

Para analisar o funcionamento do algoritmo de treinamento, geramos 200 sequências baseadas no csHMM da Figura 4.1 (os valores das probabilidades podem ser vistas no CSHMM 1 do Apêndice A). Inicializamos aleatoriamente os parâmetros do modelo e rodamos o algoritmo de treinamento utilizando as sequências previamente geradas. Os parâmetros convergem após algumas iterações, como podemos perceber através do Gráfico 4.2 que representa a média aritmética e o desvio padrão das probabilidades das sequências utilizadas no treinamento. O modelo com os novos parâmetros estimados se assemelha muito aos parâmetros do modelo utilizado para gerar as sequências[1].

[1] O modelo com os parâmetros reestimados pode ser visto no CSHMM 2 do Apêndice A

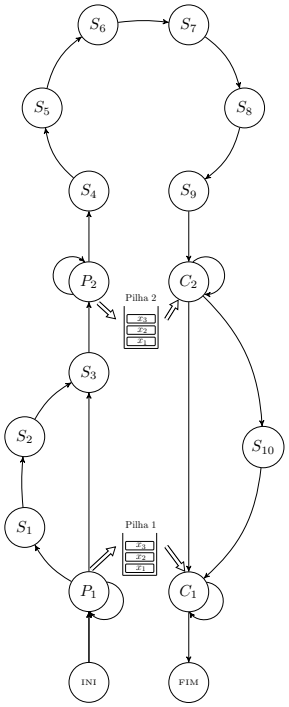


Figura 4.1: *CsHMM para representar a estrutura secundária consenso de IREs.*

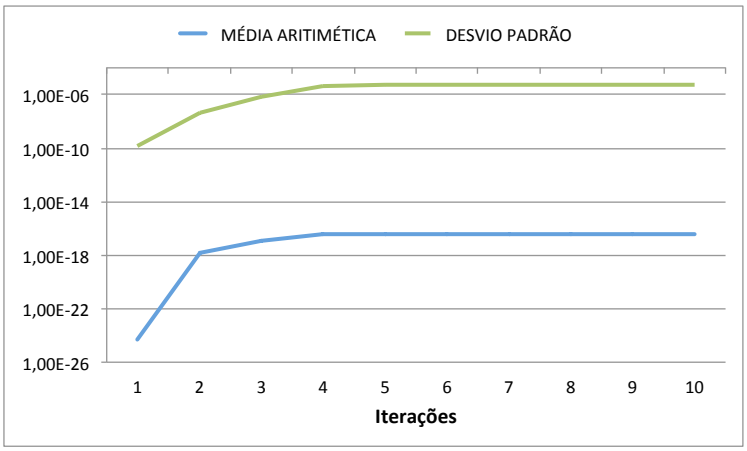


Figura 4.2: *Média aritmética e desvio padrão das probabilidades das sequências de treinamento para cada iteração.*

4.1.2 Otimizações

Nossa segunda investigação foi referente à otimização para modelos lineares. Para isso, utilizamos o csHMM representado pela figura 3.6. Utilizamos 100 sequências agrupadas em 10 conjuntos, cada conjunto contendo 10 sequências de mesmo comprimento, variando de 40 a 400 símbolos. Para cada conjunto, executamos as versões otimizada e não otimizada dos algoritmos de cálculo da sequência de estados mais provável e *Inside*.

Para o cálculo da sequência de estados mais provável, obtivemos os rótulos e os *scores* de ambas versões do algoritmo. Como esperado, tanto as rotulações quanto os *scores* foram idênticos. O Gráfico 4.3 mostra o tempo de execução médio utilizado pelas variantes dos algoritmos. Podemos perceber uma melhora no tempo de execução do algoritmo otimizado que, em alguns dos conjuntos, obteve uma redução entre 20% e 22% do tempo exigido pela versão não otimizada. Investigamos

também o tempo médio de execução das versões otimizada e não otimizada do algoritmo *Inside* que pode ser visto através do Gráfico 4.4

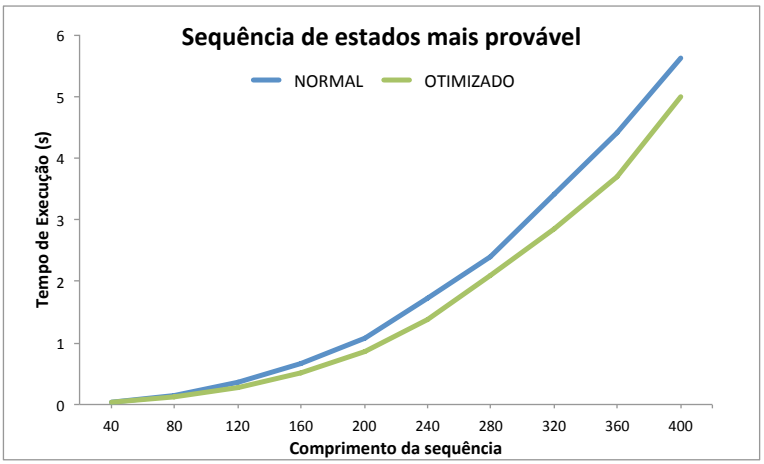


Figura 4.3: *Tempo de execução da versão otimizada e não otimizada do algoritmo para cálculo da sequência de estados mais provável.*

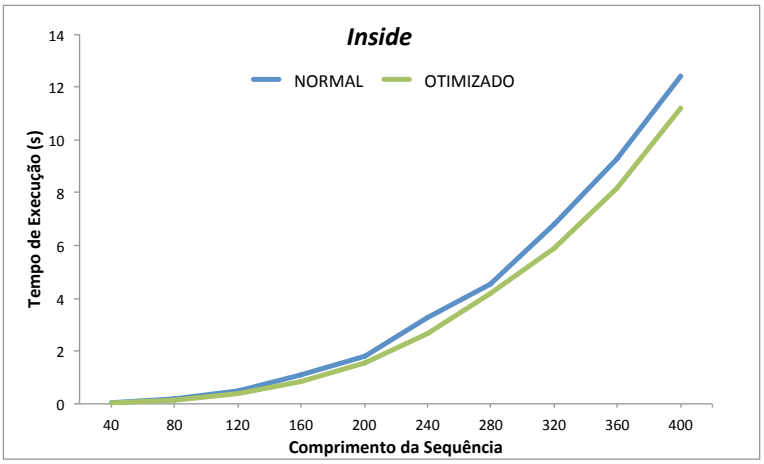


Figura 4.4: *Tempo de execução da versão otimizada e não otimizada do algoritmo Inside.*

Essa otimização é ainda mais evidente na execução do algoritmo *Outside* (4.5), devido sua alta complexidade computacional, o que impactou drasticamente no tempo de execução do algoritmo de treinamento (4.6). Abaixo podemos ver os gráficos com o tempo médio dos mesmos. Devido ao alto custo computacional, utilizamos apenas as sequências dos cinco primeiros conjuntos anteriormente definidos, o que se mostrou suficiente para obtermos uma ideia da taxa de crescimento das funções.

4.1.3 Validação Cruzada e Comparações

A fim de testar a performance preditiva de nossa implementação, criamos um csHMM para classificação de micro RNAs precursores. Essa escolha foi motivada pela possibilidade de comparação com outras duas ferramentas, a implementação de Agarwal S. *et al.* para csHMM [WHS04] e o miPred [NM07], que é um classificador baseado em Máquinas de Vetor de Suporte. Para realizar comparações entre os métodos, utilizamos os mesmos conjuntos de dados desenvolvidos por Ng e Mishra[NM07], que também foram utilizados por Agarwal S. *et al.*.

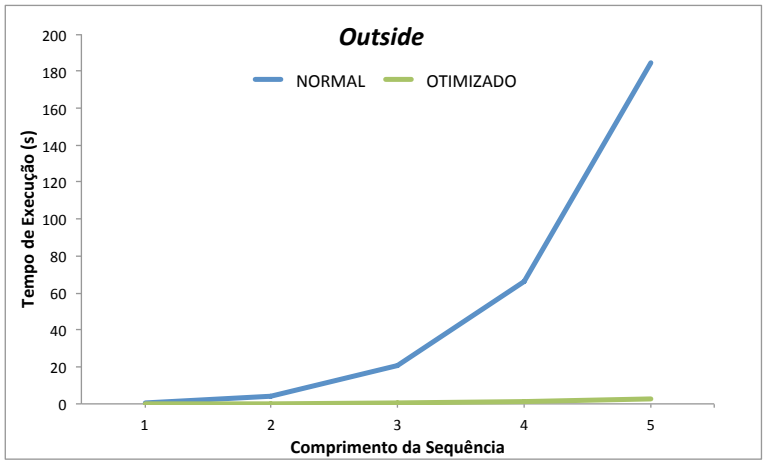


Figura 4.5: *Tempo de execução da versão otimizada e não otimizada do algoritmo de Outside.*

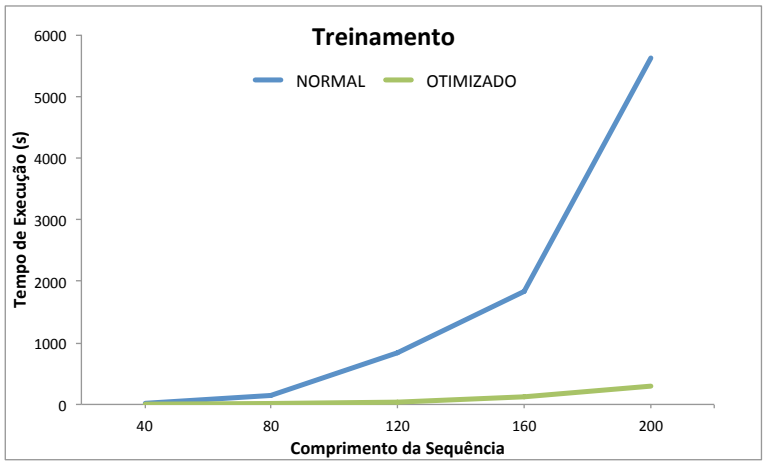


Figura 4.6: *Tempo de execução da versão otimizada e não otimizada do algoritmo de treinamento.*

Micro RNA precursores

Micro RNA (miRNA) é uma molécula de RNA pequena com tamanho entre 20-25 nucleotídeos. Tais moléculas podem ser encontradas em plantas, animais e alguns vírus desempenhando funções como silenciamento de RNA e regulação da expressão gênica [Amb04]. Em animais, micro RNAs desempenham importantes funções no processo de crescimento, hematopoese, apoptose, crescimento celular e várias doenças [IFL+05, EKS06, LZD+08]. Micro RNAs são moléculas codificadas a partir de uma molécula precursora (micro RNA precursor) de tamanho entre 60-120 nucleotídeos que possuem estrutura secundária no formato de *hairpin* que desempenha um papel importante durante o processo de biogênese de um micro RNA maduro.

Representação de micro RNA precursores

Como dito anteriormente, micro RNA precursores possuem estrutura secundária no formato de *hairpin*. Dessa forma, propusemos um csHMM para representar essa estrutura. Definimos um estado de emissão pareada P_1, que irá empilhar o símbolo emitido na memória auxiliar associada, e um estado sensível ao contexto C_1 que irá ler o símbolo previamente emitido pelo estado de emissão pareada e recalcular as probabilidades de transição e emissão. Esse par de estados representa as hélices da estrutura secundária. A fim de representarmos os bojos e loops adicionamos os estados

de emissão única S_1, S_2, S_3, S_4 e S_5. A estrutura do csHMM proposto pode ser vista através da figura 4.7

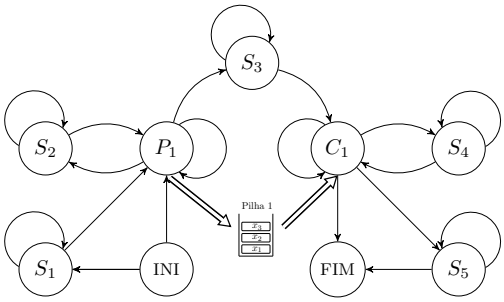


Figura 4.7: *HMM proposto para representar micro RNA precursores.*

Conjuntos de Dados

Ng e Mishra utilizaram os seguintes conjuntos de dados:

- 2241 sequências de micro RNAs precursores de 41 espécies (**D1**);
- 8494 sequências de "pseudo" micro RNAs precursores[2] (**D2**).
- 323 sequências (**D3**) de micro RNAs precursores de humanos retiradas do conjunto **D1**;
- 646 sequências (**D4**) de de "pseudo micro RNAs precursores" retiradas do conjunto **D2**

Esses conjuntos de dados foram os mesmos utilizado por Agarwal S. *et al.* [WHS04] e Mishra S. K. *et al.* [NM07].

Validação Cruzada

A fim de verificar a performance preditiva do nosso modelo, utilizamos *5-fold cross validation* e. Para isso, partes dos conjuntos de dados de **D3** (200 sequências) e **D4** (400 sequências) foram usadas para validação cruzada. Dessa forma, cada *fold* do conjunto de dados **D3** (conjunto que caracteriza micro RNAs precursores) possui um conjunto de treinamento com 160 sequências e um conjunto de testes com 40 sequências. Similarmente, cada *fold* do conjunto de dados **D4** (conjunto que caracteriza pseudo micro RNAs precursores) possui um conjunto de treinamento com 320 sequências e um conjunto de testes com 80 sequências.

Para cada *fold* da validação cruzada foi realizada uma rodada de treinamento. Após essa fase, um classificador Bayesiano foi utilizado para medir a performance preditiva do nosso modelo. Para tal, calculamos as seguintes medidas:

- **TP (*true positive*)**: quantidade de micro RNA precursores corretamente classificados como micro RNA precursores.
- **FN (*false negative*)**: quantidade de micro RNA precursores incorretamente classificados como pseudo micro RNA precursores.

[2]Sequências de regiões gênicas de humanos que podem dobrar-se em uma estrutura *hairpin*, semelhante a dos micro RNAs precursores.

- **TN (*true negative*)**: quantidade de pseudo micro RNA precursores corretamente classificados como pseudo micro RNA precursores.
- **FP (*false positive*)**: quantidade de pseudo micro RNA precursores incorretamente classificados como micro RNA precursores.

A partir das contagens acima mencionadas, podemos então calcular as medidas de sensibilidade (**SN**), especificidade (**SP**) e acurácia (**ACC**) da seguinte maneira:

$$SN = \frac{TP}{TP + FN} \times 100 \tag{4.1}$$

$$SP = \frac{TN}{TN + FP} \times 100 \tag{4.2}$$

$$ACC = \frac{TP + TN}{TP + FN + FP + TN} \times 100 \tag{4.3}$$

Abaixo temos uma comparação dos os resultados obtidos pelo ToPS (com desvio padrão), miPred e pelo csHMM de Agarwal S. *et al.*. É importante ressaltar que os dados do miPred e do csHMM Agarwal S. *et al.* foram retirados diretamente dos artigos.

Tabela 4.1: *Medidas de sensibilidade (**SN**), especificidade (**SP**) e acurácia (**ACC**) para o ToPS, miPred e csHMM de Agarwal S. et al. para validação cruzada.*

	SN	**SP**	**ACC**
ToPS	84,0 (2,85)	94,75 (2,56)	91,33 (2,01)
miPred	84,55	97,97	93,5
csHMM (Agarwal S. *et al.*)	85,0	97,0	93,0

Holdout

Após as rodadas de validação cruzada, utilizamos o restantes das sequências dos conjuntos **D3** e **D4** (123 e 246 sequências, respectivamente) para aplicar o método de validação *holdout*. Esse método consiste em dividir o conjunto de dados em dois conjuntos, um para treinamento e outro para teste. Para o conjunto **D3** foram retirados membros pertencentes à mesma família, o que reduziu a quantidade de sequências em **D3** para 82. Abaixo temos os resultados obtidos.

Tabela 4.2: *Medidas de sensibilidade (**SN**), especificidade (**SP**) e acurácia (**ACC**) para o ToPS, miPred e csHMM de Agarwal S. et al. para o método holdout.*

	SN (%)	**SP (%)**	**ACC (%)**
ToPS	76,83	97,15	92,07
miPred	78,05	97,97	93,0
csHMM (Agarwal S. *et al.*)	76,83	98,37	93,0

Por fim, utilizamos o restante dos conjuntos de dados de **D1** e **D2** em nosso classificador. Para o conjunto de dados **D1**, obtivemos uma sensibilidade de 94% (1803 verdadeiros positivos e 115 falsos negativos) e para **D2** obtivemos uma especificidade de 95,76% (8134 verdadeiros negativos e 360 falsos positivos).

Conclusões

Pela impossibilidade de replicarmos os experimentos de validação cruzada para o csHMM Agarwal S. *et al.* e miPred, não podemos estabelecer qual ferramenta possui melhor performance preditiva, visto que os resultados obtidos são similares e estão dentro do desvio padrão do ToPS. Nossa implementação, no entanto, permite a definição de modelos para representar diferentes formatos de estruturas secundárias, diferente da implementação de Agarwal S. *et al.*, que foi exclusivamente desenvolvida para micro RNAs precursores. Outra vantagem em nossa modelagem é que não precisamos saber de antemão a estrutura secundária das sequências de treinamento, diferentemente da implementação de Agarwal S. *et al.* que necessita do conhecimento prévio da estrutura secundária para realizar o treinamento, visto que o mesmo utiliza o método da Máxima Verossimilhança para estimar os parâmetros de emissão e transição do modelo. Outro fato importante a se considerar é que a implementação de Agarwal S. *et al.* quebra a definição de que devemos ter a quantidade de estados de emissão pareada e sensível ao contexto.

No que diz respeito ao miPred, nossa implementação possui a vantagem conseguir classificar bem micro RNAs precursores de pseudo micro RNAs precursores utilizando apenas o *score* da sequência como característica. O miPred, por outro lado, necessita de um total de 29 características - energia livre, entropia, propensão de pareamento de bases, distância entre pares de bases, entre outros - para classificar micro RNAs precursores de pseudo micro RNAs precursores. Outra vantagem do csHMM é que o mesmo é um modelo gerador, em oposição ao modelo discriminativo utilizado pelo miPred (SVM), o que nos permite predizer a estrutura secundária mais provável para um determinado micro RNA precursor.

4.2 Modelo de Markov de Estados Ocultos Sensível ao Contexto Para Perfil de Sequências

Nesta seção, descrevemos os métodos utilizados para validar nossa implementação de *Profile*-csHMM. É importante ressaltar que, devido à complexidade de tempo e memória exigida pelo *Profile*, se tornou inviável realizar testes para alinhamentos muito extensos.

4.2.1 Validação da Implementação

A primeira de nossas investigações referente ao *Profile* foi quanto a corretude dos algoritmos implementados cálculo da sequência de estados mais provável e *Inside*. Dessa forma, utilizamos o alinhamento múltiplo da figura 2.24 para inferir o *Profile* e treiná-lo utilizando o método da máxima verossimilhança. O modelo com os parâmetros estimados pode ser visto no *Profile*-csHMM do Apêndice B. Considere, então, a sequência de símbolos $x = UGCA$. Perceba que existem várias sequências de estados que podem dar origem a x. Geramos todas a possíveis sequências de estados para a observação x e calculamos a probabilidade dessas sequências. Após isso, calculamos a probabilidade da sequência x utilizando o algoritmo para cálculo da sequência de estados mais provável e obtivemos $y^* = M_1M_2D_3M_4M_5$, onde $P(x, y^*|\lambda) = 1.562 \times 10^{-3}$ que coincidiu com o resultado obtido através de força bruta. O mesmo método foi aplicado para o algoritmo *Inside*. Porém, somamos a probabilidade de todos as possíveis sequências de estados, e em ambos obtivemos $P(x|\lambda) = 1.717 \times 10^{-3}$.

4.2.2 Otimização de Mémoria

A segunda da nossas investigações para o *Profile*, foi quanto à otimização de memória implementada. Para isso, utilizamos as famílias IRE_I e Tymo_tRNA-like do banco de dados Rfam [NBB+14]. Executamos a ferramenta de *Profiling* Xcode para medir o consumo de memória das versões otimizada e não otimizada do algoritmo de cálculo da sequência de estados mais provável. Medimos também o consumo de memória da implementação de Yoon para *Profile*-csHMM e da ferramenta INFERNAL. A Tabela 4.3 mostra os resultados obtidos.

Tabela 4.3: *Consumo médio de memória (em GB) para cálculo da sequência de estados mais provável.*

	IRE_I	**Tymo_tRNA-like**
ToPS (Não Otimizado)	0,348	9,925
ToPS (Otimizado)	0,107	2,635
Profile-CSHMM (Yoon)	0,039	0,16
INFERNAL	0,011	0,086

Podemos perceber uma diminuição considerável de memória alcançada através da otimização. Porém, ainda temos um consumo alto de memória para famílias onde a quantidade de colunas de correspondência é elevado, visto que isso gera um maior número de estados no modelo e consequentemente maior alocação de memória. Não podemos estimar com precisão a quantidade de memória que será otimizada, visto que a mesma depende de quantas colunas do alinhamento serão tratadas como correspondência. No entanto, nossos testes indicam que podemos alcançar até 75% de economia. É importante ressaltar que a alta quantidade de memória exigida afeta a performance do algoritmo de cálculo da sequência de estados mais provável.

4.2.3 Otimização de Restrição da Região de Correspondência

Nossa terceira investigação para o *Profile*-csHMM foi quanto à implementação da restrição da região de busca, mais especificamente quanto à redução de tempo alcançada através da otimização. Dessa forma, utilizamos novamente as famílias IRE_I e Tymo_tRNA-like do banco de dados Rfam [NBB+14] para rodarmos os testes. Comparamos o tempo médio de CPU necessário para calcular a sequência de estados mais provável. A Tabela 4.4 mostra com os resultados obtidos, bem como os resultados obtidos pela implementação de Yoon para *Profile*-csHMM e da ferramenta INFERNAL. É importante ressaltar que utilizamos a versão com otimização de memória do algoritmo que encontra a sequência de estados mais provável a fim de obter, de forma isolada, o efeito gerado pela introdução do parâmetro D.

Tabela 4.4: *Tempo médio de CPU (em segundos) para cálculo da sequência de estados mais provável.*

	IRE_I	**Tymo_tRNA-like**
ToPS (Não Otimizado)	1,116	56,530
ToPS (Otimizado)	0,180	8,924
Profile-CSHMM (Yoon)	0,05	0,14
INFERNAL	0,01	0,02

Como previsto, os rótulos das sequências, bem como suas probabilidades, foram idênticas tanto para o algoritmo otimizado quanto não otimizado. Como o parâmetro D depende da distância entre bases correspondentes do alinhamento o qual estamos avaliando, não é possível calcular uma

estimativa geral a respeito da taxa de redução de tempo que a introdução desse parâmetro acarreta. Porém, como podemos perceber através da Tabela 4.4, houve uma redução significativa do tempo médio de CPU necessário para calcular a sequência de estados mais provável através da utilização do mesmo.

4.2.4 Validações e Comparações

A fim de investigar a performance preditiva do *Profile*-cshMM, utilizamos novamente a técnica *5-fold cross validation*. Selecionamos 66 famílias do banco de dados Rfam e agrupamo-as em dois conjuntos de acordo com o formato da estrutura secundária formada pela família. Dessa forma, o primeiro conjunto **C1** (64 alinhamentos) é caracterizado por famílias que possuem estrutura secundária no formato *hairpin* e o segundo conjunto **C2** (2 alinhamentos) é caracterizado por famílias com estrutura secundária no formato de trevo (tRNA). Os seguintes conjuntos de dados foram utilizados[3]

- **C1**: 64 alinhamentos constituindo um total de 290 sequências;
- **C2**: 2 alinhamentos constituindo 1.063 sequências;

Validação Cruzada Sobre e Conjunto de dados C1

Escolhemos o alinhamento com o maior número de sequência do conjunto **C1** para classificar entre membros pertencentes à família e membros não pertencentes à família. Dessa forma, **C1** foi dividido em dois subconjuntos **C1.1** (1 alinhamento com 33 sequências) e **C1.2** (63 alinhamentos com 257 sequências). Utilizamos um *Profile*-csHMM para representar o conjunto **C1.1** e como modelo alternativo (**C1.2**) utilizamos um modelo nulo com distribuição de probabilidades das sequências de treinamento, visto que essa é a abordagem indicada por Machado-Lima [MLKD10] para diminuir o número de falsos positivos e é uma das técnicas aplicadas pelos INFERNAL [NE13] para classificação. Utilizamos também o conjunto de dados **C1** para estimar os valores de sensibilidade e especificidade da ferramenta INFERNAL. A Tabela 4.5 ilustra os resultados obtidos por ambas as ferramentas:

Tabela 4.5: *Medidas de sensibilidade (**SN**) e especificidade (**SP**) para validação cruzada sobre o conjunto **C1** para o ToPS e INFERNAL*

	SN (%) (desvio padrão)	**SP (%)** (desvio padrão)
ToPS	100,00 (0)	99,62 (0,86)
INFERNAL	100,00 (0)	100,00 (0)

Validação Cruzada Sobre e Conjunto de dados C2

O conjunto de dados **C2** é formado pelos alinhamentos *Selenocysteine transfer RNA* e *tRNA* (números de acesso RF01852 e RF00005 respectivamente). Nosso objetivo é classificar entre membros pertences a família *Selenocysteine transfer RNA* e não membros. Dessa forma, **C2** foi dividido em dois subconjuntos **C1.1** (109 sequências) e **C1.2** (954 sequências). Utilizamos, novamente, um *Profile*-csHMM para representar o conjunto **C1.1** e como modelo alternativo (**C1.2**) utilizamos um

[3]Os números de acesso das famílias utilizadas pode ser encontradas no apêndice B

modelo nulo com distribuição de probabilidades das sequências de treinamento. Similarmente a **C1**, utilizamos o conjunto de dados **C2** para estimar os valores de de sensibilidade e especificidade da ferramenta INFERNAL. A Tabela 4.6 ilustra os resultados obtidos:

Tabela 4.6: *Medidas de sensibilidade (**SN**), especificidade (**SP**) e acurácia (**ACC**) para validação cruzada sobre o conjunto **C2***

	SN (%) (desvio padrão)	**SP (%)** (desvio padrão)
ToPS	100,00 (0)	95,91 (2,59)
INFERNAL	100,00 (0)	100,00 (0,44)

Conclusões

Os testes realizados mostram que nossa implementação de *Profile*-csHMM possui performance de tempo inferior à implementação de *Profile*-csHMM de Yoon e da ferramenta INFERNAL. Porém a implementação de Yoon utiliza o algoritmo SCA para calcular a sequência de estados mais provável, cuja complexidade de tempo não é fixa e pode variar de $O(D^2K)$ a $O(D^6K)$ dependendo das associações existentes entre os símbolos, onde D representa o tamanho da região de correspondência e K o número de estados de correspondência do *Profile*. Dessa forma, era esperado que a implementação de Yoon apresentasse performance de tempo superior. A ferramenta INFERNAL, também se mostrou superior em performance de tempo, como esperado, porém é importante ressaltar que a mesma paraleliza a execução de seus algoritmos. Quanto à performance preditiva, a ferramenta INFERNAL também se mostra superior. Porém, devemos ressaltar que os alinhamentos utilizados nos testes foram todos construídos utilizando o INFERNAL. Além disso, tal ferramenta utiliza uma variedade de filtros com o objetivo de diminuir o tempo de execução e o número de falso positivos.

Capítulo 5

Considerações Finais e Trabalhos Futuros

Desenvolvemos um arcabouço probabilístico orientado a objetos para análise de sequências que possuem associações com distância arbitrária entre símbolos. Para tal, implementamos dois modelos probabilísticos, o csHMM e *Profile*-csHMM como uma extensão do ToPS. Esta extensão, somada aos modelos probabilísticos já existentes, abre um leque de novas possibilidades de utilização do arcabouço. Utilizamos esses modelos especificamente para análise de sequências de RNA não codificantes - classe de moléculas biológicas ainda pouco conhecida, mas que exercem papeis fundamentais em uma variedade de processos biológicos.

Para csHMM, implementamos os algoritmos de cálculo da sequência de estados mais provável, *Inside*, *Outside* e treinamento. Propusemos uma otimização no csHMM para modelos lineares que reduziu consideravelmente o tempo de execução dos algoritmos de inferência. Constatamos que nossa abordagem permite a definição de modelos para representar diversos formatos de estrutura secundária, diferente da implementação de Agarwal S. *et al.* que foi exclusivamente desenvolvida para micro RNAs precursores. Além disso, nosso modelo possui a vantagem de necessitar somente do *score* da sequência para classificação, enquanto o miPred necessita um total de 29 características.

Desenvolvemos novos algoritmos para o cálculo da sequência de estados mais provável, *Inside* [1] para *Profile*-csHMM, bem como propusemos uma otimização de memória e implementamos uma otimização de tempo baseada na restrição da região de correspondência. Essas otimizações reduziram drasticamente o tempo de execução e o consumo de memória desses algoritmos.

Comparamos também, a performance preditiva de nossa implementação com a ferramenta INFERNAL, que é umas das ferramentas de referência para perfil de sequências de RNA. Apesar do INFERNAL se mostrar mais eficiente, nossa implementação alcança valores de predição competitivos. Considerando que o *pipeline* da ferramenta INFERNAL, paraleliza a execução dos algoritmos e utiliza vários filtros para diminuir o número de falsos positivos. O que justifica a performance preditiva superior.

Comparamos a performance de tempo e de memória de nossa implementação com a ferramenta INFERNAL e a implementação de *Profile*-csHMM de Yoon. Os resultados mostram que nossa abordagem possui uma performance inferior, apesar das otimizações.

Como trabalhos futuros, é importante destacar o estudo de novos algoritmos para lidar com

[1] O *Profile*-csHMM de Yoon não apresenta o algoritmo *Inside*

pseudo-nós tanto para csHMM quanto *Profile*-csHMM, o que não pode ser alcançado para modelos baseados em gramáticas livres de contexto, como a ferramenta INFERNAL. Em particular, desenvolver a ideia do algoritmo SCA para *Profile*-csHMM. Outra possibilidade seria investigar o efeito da utilização de uma fila como memória auxiliar compartilhada pelos estados de emissão pareada e sensível ao contexto. Devemos destacar também a necessidade de otimizar ainda mais os recursos de memória exigidos pelo *Profile*-csHMM que ainda é bastante alto, apesar da otimização implementada, o que por sua vez acarretará em uma melhora de desempenho de tempo do mesmo. Uma possível ideia para resolver esse problema é restringir a alocação de memória de acordo com o parâmetro D que restringe a região de busca. Ainda a respeito do *Profile*-csHMM, seria interessante estudar a possibilidade de utilização do método de Esperança e Maximização no treinamento.

Por fim, é importante ressaltar que a ferramenta ToPS vem sendo aplicada com sucesso em predição de genes [Kas12] e alinhamentos múltiplos [Onu12]. Recentemente, Bonadio [Bon13] obteve resultados promissores utilizando Campos Aleatórios Condicionais em segmentação de sequências.

Apêndice A

Modelo de Markov de Estados Ocultos Sensível ao Contexto Para IRE

Apresentamos neste apêndice o csHMM utilizado para gerar as sequências utilizadas no treinamento (CSHMM 1) e o csHMM obtido após inicializarmos os parâmetros aleatoriamente e utilizarmos as sequências geradas (CSHMM 2).

CSHMM 1

```
model_name = "ContextSensitiveHiddenMarkovModel"
state_names = ("INI","P1","S1","S2","S3","P2","S4","S5","S6","S7","S8","S9"
  ,"C2","S10","C1","FIM")
observation_symbols = ("A","C","G","U")
transitions = ("P1" | "INI": 1;
 "P1" | "P1": 0.878286;
 "S1" | "P1": 0.121714;
 "S2" | "S1": 1;
 "S3" | "S2": 1;
 "P2" | "S3": 1;
 "P2" | "P2": 0.778474;
 "S4" | "P2": 0.221526;
 "S5" | "S4": 1;
 "S6" | "S5": 1;
 "S7" | "S6": 1;
 "S8" | "S7": 1;
 "S9" | "S8": 1;
 "C2" | "S9": 1;
 "S10" | "C2, empty": 0.515625;
 "C1" | "C2, empty": 0.484375;
 "C2" | "C2, not_empty": 1;
 "C1" | "S10": 1;
 "FIM" | "C1, empty": 1;
 "C1" | "C1, not_empty": 1)
emission_probabilities = (
 "A" | "P1": 0.0481939;
 "C" | "P1": 0.269133;
 "G" | "P1": 0.24456;
 "U" | "P1": 0.438114;
 "A" | "S1": 0.0152946;
 "C" | "S1": 0.12786;
 "G" | "S1": 0.0308403;
 "U" | "S1": 0.826005;
 "A" | "S2": 0.10838;
 "C" | "S2": 0.0185374;
 "G" | "S2": 0.833668;
```

```
37 | "U" | "S2": 0.0394147;
38 | "A" | "S3": 0.089057;
39 | "C" | "S3": 0.869728;
40 | "G" | "S3": 0.0153987;
41 | "U" | "S3": 0.0258161;
42 | "A" | "P2": 0.394278;
43 | "C" | "P2": 0.2203;
44 | "G" | "P2": 0.00995386;
45 | "U" | "P2": 0.375468;
46 | "A" | "S4": 0.121212;
47 | "C" | "S4": 0.833333;
48 | "G" | "S4": 0.030303;
49 | "U" | "S4": 0.0151515;
50 | "A" | "S5": 0.818182;
51 | "C" | "S5": 0.0606061;
52 | "G" | "S5": 0.0909091;
53 | "U" | "S5": 0.030303;
54 | "A" | "S6": 0.0606061;
55 | "C" | "S6": 0.0151515;
56 | "G" | "S6": 0.833333;
57 | "U" | "S6": 0.0909091;
58 | "A" | "S7": 0.0151515;
59 | "C" | "S7": 0.0151515;
60 | "G" | "S7": 0.106061;
61 | "U" | "S7": 0.863636;
62 | "A" | "S8": 0.0606061;
63 | "C" | "S8": 0.0454545;
64 | "G" | "S8": 0.848485;
65 | "U" | "S8": 0.0454545;
66 | "A" | "S9": 0.106061;
67 | "C" | "S9": 0.318182;
68 | "G" | "S9": 0.0454545;
69 | "U" | "S9": 0.530303;
70 | "A" | "C2, A": 0.0210542;
71 | "C" | "C2, A": 0.0172536;
72 | "G" | "C2, A": 0.0214135;
73 | "U" | "C2, A": 0.940279;
74 | "A" | "C2, C": 0.0346786;
75 | "C" | "C2, C": 0.0152548;
76 | "G" | "C2, C": 0.815894;
77 | "U" | "C2, C": 0.134172;
78 | "A" | "C2, G": 0.316044;
79 | "C" | "C2, G": 0.17092;
80 | "G" | "C2, G": 0.171196;
81 | "U" | "C2, G": 0.341839;
82 | "A" | "C2, U": 0.723018;
83 | "C" | "C2, U": 0.00955774;
84 | "G" | "C2, U": 0.242452;
85 | "U" | "C2, U": 0.0249723;
86 | "A" | "S10": 0.0838554;
87 | "C" | "S10": 0.694055;
88 | "G" | "S10": 0.159883;
89 | "U" | "S10": 0.0622067;
90 | "A" | "C1, A": 0.106959;
91 | "C" | "C1, A": 0.249626;
92 | "G" | "C1, A": 0.108478;
93 | "U" | "C1, A": 0.534938;
94 | "A" | "C1, C": 0.181933;
95 | "C" | "C1, C": 0.132648;
```

```
 "G" | "C1, C": 0.669747;
 "U" | "C1, C": 0.0156716;
 "A" | "C1, G": 0.106903;
 "C" | "C1, G": 0.753409;
 "G" | "C1, G": 0.0781652;
 "U" | "C1, G": 0.0615227;
 "A" | "C1, U": 0.544515;
 "C" | "C1, U": 0.130075;
 "G" | "C1, U": 0.286842;
 "U" | "C1, U": 0.0385683)
initial_probabilities = ("INI":  1)
```

CSHMM 2

```
model_name = "ContextSensitiveHiddenMarkovModel"
state_names = ("INI","P1","S1","S2","S3","P2","S4","S5","S6","S7","S8","S9"
  ,"C2","S10","C1","FIM")
observation_symbols = ("A","C","G","U")
transitions = ("P1" | "INI": 1;
 "P1" | "P1": 0.877537;
 "S1" | "P1": 0.122463;
 "S2" | "S1": 1;
 "S3" | "S2": 1;
 "P2" | "S3": 1;
 "P2" | "P2": 0.779531;
 "S4" | "P2": 0.220469;
 "S5" | "S4": 1;
 "S6" | "S5": 1;
 "S7" | "S6": 1;
 "S8" | "S7": 1;
 "S9" | "S8": 1;
 "C2" | "S9": 1;
 "S10" | "C2, empty": 0.564356;
 "C1" | "C2, empty": 0.435644;
 "C2" | "C2, not_empty": 1;
 "C1" | "S10": 1;
 "FIM" | "C1, empty": 1;
 "C1" | "C1, not_empty": 1)
emission_probabilities = ("A" | "P1": 0.0405737;
 "C" | "P1": 0.292111;
 "G" | "P1": 0.229912;
 "U" | "P1": 0.437403;
 "A" | "S1": 0.020824;
 "C" | "S1": 0.108837;
 "G" | "S1": 0.0306888;
 "U" | "S1": 0.83965;
 "A" | "S2": 0.105379;
 "C" | "S2": 0.0215802;
 "G" | "S2": 0.839898;
 "U" | "S2": 0.0331427;
 "A" | "S3": 0.0954012;
 "C" | "S3": 0.852559;
 "G" | "S3": 0.00681742;
 "U" | "S3": 0.0452225;
 "A" | "P2": 0.370056;
 "C" | "P2": 0.232512;
 "G" | "P2": 0.0177233;
 "U" | "P2": 0.379709;
 "A" | "S4": 0.0784314;
 "C" | "S4": 0.857843;
 "G" | "S4": 0.0441176;
```

```
 "U" | "S4": 0.0196078;
 "A" | "S5": 0.794118;
 "C" | "S5": 0.0686275;
 "G" | "S5": 0.107843;
 "U" | "S5": 0.0294118;
 "A" | "S6": 0.0882353;
 "C" | "S6": 0.00980392;
 "G" | "S6": 0.784314;
 "U" | "S6": 0.117647;
 "A" | "S7": 0.0147059;
 "C" | "S7": 0.00980392;
 "G" | "S7": 0.156863;
 "U" | "S7": 0.818627;
 "A" | "S8": 0.0784314;
 "C" | "S8": 0.0588235;
 "G" | "S8": 0.818627;
 "U" | "S8": 0.0441176;
 "A" | "S9": 0.107843;
 "C" | "S9": 0.362745;
 "G" | "S9": 0.0588235;
 "U" | "S9": 0.470588;
 "A" | "C2, A": 0.0324064;
 "C" | "C2, A": 0.0145349;
 "G" | "C2, A": 0.0245057;
 "U" | "C2, A": 0.928553;
 "A" | "C2, C": 0.0416693;
 "C" | "C2, C": 0.0378619;
 "G" | "C2, C": 0.774208;
 "U" | "C2, C": 0.146261;
 "A" | "C2, G": 0.371922;
 "C" | "C2, G": 0.105982;
 "G" | "C2, G": 0.209392;
 "U" | "C2, G": 0.312704;
 "A" | "C2, U": 0.721718;
 "C" | "C2, U": 0.00656485;
 "G" | "C2, U": 0.252126;
 "U" | "C2, U": 0.0195913;
 "A" | "S10": 0.0832486;
 "C" | "S10": 0.746988;
 "G" | "S10": 0.126363;
 "U" | "S10": 0.0434007;
 "A" | "C1, A": 0.0862229;
 "C" | "C1, A": 0.243979;
 "G" | "C1, A": 0.0861277;
 "U" | "C1, A": 0.583671;
 "A" | "C1, C": 0.217106;
 "C" | "C1, C": 0.122135;
 "G" | "C1, C": 0.635614;
 "U" | "C1, C": 0.0251448;
 "A" | "C1, G": 0.107527;
 "C" | "C1, G": 0.76612;
 "G" | "C1, G": 0.0597733;
 "U" | "C1, G": 0.0665805;
 "A" | "C1, U": 0.52884;
 "C" | "C1, U": 0.134959;
 "G" | "C1, U": 0.292367;
 "U" | "C1, U": 0.043835)
initial_probabilities = ("INI":  1)
```

Apêndice B

Modelo de Markov de Estados Ocultos Sensível ao Contexto Para Perfil de Sequências

Apresentamos neste apêndice dados utilizados nos testes do *Profile*-csHMM.

B.1 Profile-csHMM com parâmetros reestimados

Abaixo o *Profile*-csHMM com os parâmetros reestimados com base no alinhamento da Figura 2.24.

Profile-csHMM

```
model_name = "ProfileCSHMM"
state_names = ("M0","I0","M1","D1","I1","M2","D2","I2","M3","D3","I3","M4",
"D4","I4","M5","D5","I5","M6")
pairwise_states = ("M2","M1")
context_sensitive_states = ("M4","M5")
search_region = 5
observation_symbols = ("A","C","G","U")
transitions = ("I0" | "M0": 0.125;
 "M1" | "M0": 0.75;
 "D1" | "M0": 0.125;
 "I0" | "I0": 0.333333;
 "M1" | "I0": 0.333333;
 "D1" | "I0": 0.333333;
 "I1" | "M1": 0.125;
 "M2" | "M1": 0.625;
 "D2" | "M1": 0.25;
 "I1" | "D1": 0.333333;
 "M2" | "D1": 0.333333;
 "D2" | "D1": 0.333333;
 "I1" | "I1": 0.333333;
 "M2" | "I1": 0.333333;
 "D2" | "I1": 0.333333;
 "I2" | "M2": 0.142857;
 "M3" | "M2": 0.571429;
 "D3" | "M2": 0.285714;
 "I2" | "D2": 0.25;
 "M3" | "D2": 0.5;
 "D3" | "D2": 0.25;
 "I2" | "I2": 0.333333;
 "M3" | "I2": 0.333333;
 "D3" | "I2": 0.333333;
 "I3" | "M3": 0.142857;
```

```
 "M4" | "M3": 0.571429;
 "D4" | "M3": 0.285714;
 "I3" | "D3": 0.25;
 "M4" | "D3": 0.5;
 "D4" | "D3": 0.25;
 "I3" | "I3": 0.333333;
 "M4" | "I3": 0.333333;
 "D4" | "I3": 0.333333;
 "I4" | "M4, empty": 0.142857;
 "M5" | "M4, empty": 0.714286;
 "D5" | "M4, empty": 0.142857;
 "I4" | "D4": 0.25;
 "M5" | "D4": 0.5;
 "D5" | "D4": 0.25;
 "I4" | "I4": 0.333333;
 "M5" | "I4": 0.333333;
 "D5" | "I4": 0.333333;
 "I5" | "M5, empty": 0.142857;
 "M6" | "M5, empty": 0.857143;
 "I5" | "D5": 0.5;
 "M6" | "D5": 0.5;
 "I5" | "I5": 0.5;
 "M6" | "I5": 0.5;
 "M6" | "M6": 1)
emission_probabilities = ("A" | "I0": 0.25;
 "C" | "I0": 0.25;
 "G" | "I0": 0.25;
 "U" | "I0": 0.25;
 "A" | "M1": 0.2;
 "C" | "M1": 0.1;
 "G" | "M1": 0.3;
 "U" | "M1": 0.4;
 "A" | "I1": 0.25;
 "C" | "I1": 0.25;
 "G" | "I1": 0.25;
 "U" | "I1": 0.25;
 "A" | "M2": 0.333333;
 "C" | "M2": 0.111111;
 "G" | "M2": 0.333333;
 "U" | "M2": 0.222222;
 "A" | "I2": 0.25;
 "C" | "I2": 0.25;
 "G" | "I2": 0.25;
 "U" | "I2": 0.25;
 "A" | "M3": 0.222222;
 "C" | "M3": 0.222222;
 "G" | "M3": 0.222222;
 "U" | "M3": 0.333333;
 "A" | "I3": 0.25;
 "C" | "I3": 0.25;
 "G" | "I3": 0.25;
 "U" | "I3": 0.25;
 "A" | "M4, A": 0.2;
 "C" | "M4, A": 0.2;
 "G" | "M4, A": 0.2;
 "U" | "M4, A": 0.4;
 "A" | "M4, C": 0.25;
 "C" | "M4, C": 0.25;
 "G" | "M4, C": 0.25;
```

```
 "U" | "M4, C": 0.25;
 "A" | "M4, G": 0.166667;
 "C" | "M4, G": 0.5;
 "G" | "M4, G": 0.166667;
 "U" | "M4, G": 0.166667;
 "A" | "M4, U": 0.4;
 "C" | "M4, U": 0.2;
 "G" | "M4, U": 0.2;
 "U" | "M4, U": 0.2;
 "A" | "I4": 0.25;
 "C" | "I4": 0.25;
 "G" | "I4": 0.25;
 "U" | "I4": 0.25;
 "A" | "M5, A": 0.25;
 "C" | "M5, A": 0.25;
 "G" | "M5, A": 0.25;
 "U" | "M5, A": 0.25;
 "A" | "M5, C": 0.25;
 "C" | "M5, C": 0.25;
 "G" | "M5, C": 0.25;
 "U" | "M5, C": 0.25;
 "A" | "M5, G": 0.166667;
 "C" | "M5, G": 0.5;
 "G" | "M5, G": 0.166667;
 "U" | "M5, G": 0.166667;
 "A" | "M5, U": 0.571429;
 "C" | "M5, U": 0.142857;
 "G" | "M5, U": 0.142857;
 "U" | "M5, U": 0.142857;
 "A" | "I5": 0.25;
 "C" | "I5": 0.25;
 "G" | "I5": 0.25;
 "U" | "I5": 0.25)
initial_probabilities = ("M0":  1)
```

B.2 Números de acesso dos alinhamentos

Os números de acesso do banco de dados Rfam para famílias utilizadas nos testes de validação cruzada para o *Profile*-csHMM são:

- Conjunto **C1.1**: RF01315, RF01316, RF01318, RF01319, RF01320, RF01321, RF01322, RF01323, RF01324, RF01325, RF01326, RF01327, RF01328, RF01329, RF01330, RF01331, RF01332, RF01333, RF01334, RF01335, RF01336, RF01337, RF01338, RF01339, RF01340, RF01341, RF01342, RF01343, RF01344, RF01345, RF01346, RF01347, RF01348, RF01349, RF01350, RF01351, RF01352, RF01353, RF01354, RF01355, RF01356, RF01357, RF01358, RF01359, RF01360, RF01361, RF01362, RF01363, RF01364, RF01365, RF01366, RF01367, RF01368, RF01369, RF01370, RF01371, RF01373, RF01374, RF01375, RF01376, RF01377, RF01378, RF01379;
- Conjunto **C1.2**: RF01317;
- Conjunto **C2.1**: RF01852;
- Conjunto **C2.2**: RF00005;

Referências Bibliográficas

[Amb04] Victor Ambros. The functions of animal microrna. *Nature*, 431(7006):350–355, 2004. 44

[BA08] Djamel Bouchaffra e Abbes Amira. Structural hidden Markov models for biometrics: Fusion of face and fingerprint. *Pattern Recognition*, 41(3):852 – 867, 2008. Part Special issue: Feature Generation and Machine Learning for Robust Multimodal Biometrics. 11

[Bon13] Ígor Bonadio. *Desenvolvimento de um arcabouço probabilístico para implementação de campos aleatórios condicionais*. Tese de Doutorado, Universidade de São Paulo, 2013. 52

[Cho59] Noam Chomsky. On certain formal properties of grammars. *Information and control*, 2(2):137–167, 1959. 11

[DEKM98] Richard Durbin, Sean R. Eddy, Anders Krogh e Graeme Mitchison. *Biological sequence analysis: probabilistic models of proteins and nucleic acids*. Cambridge university press, 1998. 1, 3, 9, 10, 11

[DMY+03] Josée Dostie, Zissimos Mourelatos, Michael Yang, Anup Sharma e Gideon Dreyfuss. Numerous microRNPs in neuronal cells containing novel microRNAs. *RNA (New York, N.Y.)*, 9(2):180–186, Fevereiro 2003. 1

[ED94] Sean R. Eddy e Richard Durbin. Rna sequence analysis using covariance models. *Nucleic Acids Research*, 22(11):2079–2088, 1994. 4, 30

[Edd98] Sean R. Eddy. Profile hidden Markov models. *Bioinformatics*, 14(9):755–763, 1998. 11

[Edd01] Sean R. Eddy. Non-coding RNA genes and the modern RNA world. *Nature Reviews Genetics*, 2(12):919–929, 2001. 1

[Edd02] Sean R Eddy. Computational genomics of noncoding rna genes. *Cell*, 109(2):137–140, 2002. 12

[EKS06] Aurora Esquela-Kerscher e Frank J Slack. Oncomirs—micrornas with a role in cancer. *Nature Reviews Cancer*, 6(4):259–269, 2006. 44

[Est11] Manel Esteller. Non-coding RNAs in human disease. *Nature Reviews Genetics*, 12(12):861–874, 2011. 1

[GHJV94] E. Gamma, R. Helm, R. Johnson e J. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Pearson Education, 1994. 33

[HGP+97] Steven Henikoff, Elizabeth A. Greene, Shmuel Pietrokovski, Peer Bork, Teresa K. Attwood e Leroy Hood. Gene families: The taxonomy of protein paralogs and chimeras. *Science*, 278(5338):609–614, 1997. 29

[HK96] Matthias W Hentze e Lukas C Kühn. Molecular control of vertebrate iron metabolism: mrna-based regulatory circuits operated by iron, nitric oxide, and oxidative stress. *Proceedings of the National Academy of Sciences*, 93(16):8175–8182, 1996. 36

[Hof03] Ivo L. Hofacker. Vienna rna secondary structure server. *Nucleic Acids Research*, 31(13):3429–3431, 2003. 3

[IFL+05] Marilena V Iorio, Manuela Ferracin, Chang-Gong Liu, Angelo Veronese, Riccardo Spizzo, Silvia Sabbioni, Eros Magri, Massimo Pedriali, Muller Fabbri, Manuela Campiglio et al. Microrna gene expression deregulation in human breast cancer. *Cancer research*, 65(16):7065–7070, 2005. 44

[Kas12] A. Y. Kashiwabara. MYOP/ToPS/SGEval: Um ambiente computacional para estudo sistemático de predição de genes. 2012. 33, 52

[KBM+94] Anders Krogh, Michael Brown, I.Saira Mian, Kimmen Sjolander e David Haussler. Hidden Markov Models in Computational Biology: Applications to Protein Modeling. *Journal of Molecular Biology*, 235(5):1501 – 1531, 1994. 30

[KKD+03] Anna M. Krichevsky, Kevin S. King, Christine P. Donahue, Konstantin Khrapko e Kenneth S. Kosik. A microRNA array reveals extensive regulation of microRNAs during brain development. *RNA*, 9(10):1274–1281, 2003. 1

[Kro97] Anders Krogh. Two methods for improving performance of an HMM and their application for gene finding. *Center for Biological Sequence Analysis*, 45:4525, 1997. 11

[LQRLT01] Mariana Lagos-Quintana, Reinhard Rauhut, Winfried Lendeckel e Thomas Tuschl. Identification of Novel Genes Coding for Small Expressed RNAs. *Science*, 294(5543):853–858, 2001. 1

[LY90] Karim Lari e Steve J Young. The estimation of stochastic context-free grammars using the inside-outside algorithm. *Computer speech & language*, 4(1):35–56, 1990. 17

[LZD+08] Ming Lu, Qipeng Zhang, Min Deng, Jing Miao, Yanhong Guo, Wei Gao e Qinghua Cui. An analysis of human microrna and disease associations. *PloS one*, 3(10):e3420, 2008. 44

[MDC+04] David H. Mathews, Matthew D. Disney, Jessica L. Childs, Susan J. Schroeder, Michael Zuker e Douglas H. Turner. Incorporating chemical modification constraints into a dynamic programming algorithm for prediction of RNA secondary structure. *Proceedings of the National Academy of Sciences of the United States of America*, 101(19):7287–7292, 2004. 3

[MLDPD08] Ariane Machado-Lima, Hernando A Del Portillo e Alan Mitchell Durham. Computational methods in noncoding rna research. *Journal of mathematical biology*, 56(1-2):15–49, 2008. 3

[MLKD10] Ariane Machado-Lima, André Y Kashiwabara e Alan M Durham. Decreasing the number of false positives in sequence classification. *BMC genomics*, 11(Suppl 5):S10, 2010. 49

[MSS04] H. Matsui, K. Sato e Y. Sakakibara. Pair stochastic tree adjoining grammars for aligning and predicting pseudoknot RNA structures. Em *Computational Systems Bioinformatics Conference, 2004. CSB 2004. Proceedings. 2004 IEEE*, páginas 290–299, 2004. 4

[NBB+14] Eric P Nawrocki, Sarah W Burge, Alex Bateman, Jennifer Daub, Ruth Y Eberhardt, Sean R Eddy, Evan W Floden, Paul P Gardner, Thomas A Jones, John Tate et al. Rfam 12.0: updates to the rna families database. *Nucleic acids research*, página gku1063, 2014. 38, 48

[NE13] Eric P Nawrocki e Sean R Eddy. Infernal 1.1: 100-fold faster rna homology searches. *Bioinformatics*, 29(22):2933–2935, 2013. 49

[NM07] Kwang Loong Stanley Ng e Santosh K Mishra. De novo svm classification of precursor microrna from genomic pseudo hairpins using global and intrinsic folding measures. *Bioinformatics*, 23(11):1321–1330, 2007. 43, 45

[NPGK78] R. Nussinov, G. Pieczenik, J. Griggs e D. Kleitman. Algorithms for Loop Matchings. *SIAM Journal on Applied Mathematics*, 35(1):68–82, 1978. 3

[Onu12] Vitor Onuchic. Inovações em técnicas de alinhamentos múltiplos e predições de genes. 2012. 52

[Pla02] Ronald H. A. Plasterk. RNA silencing: the genome's immune system. *Science Signaling*, 296(5571):1263, 2002. 1

[Rab89] L. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989. 6, 9, 11

[Sak03] Yasubumi Sakakibara. Pair hidden Markov models on tree structures. *Bioinformatics*, 19(suppl 1):i232–i240, 2003. 1, 4

[SZB02] Frank Sleutels, Ronald Zwart e Denise P Barlow. The non-coding Air RNA is required for silencing autosomal imprinted genes. *Nature*, 415(6873):810–813, 2002. 1

[TK98] Howard M Taylor e Samuel Karlin. *An introduction to stochastic modeling.* Boston, MA: Academic Press, 1998. 5

[WHS04] Christina Witwer, Ivo L Hofacker e Peter F Stadler. Prediction of consensus rna secondary structures including pseudoknots. *Computational Biology and Bioinformatics, IEEE/ACM Transactions on*, 1(2):66–77, 2004. 43, 45

[WZS99] Karen Montzka Wassarman, Aixia Zhang e Gisela Storz. Small RNAs in Escherichia coli. *Trends in Microbiology*, 7(1):37 – 45, 1999. 1

[YV04a] Byung-Jun Yoon e P. R. Vaidyanathan. RNA secondary structure prediction using context-sensitive hidden Markov models. Em *Biomedical Circuits and Systems, 2004 IEEE International Workshop on*, páginas S2/7/INV–S2/7/1–4, 2004. 1

[YV04b] Byung-Jun Yoon e P. P. Vaidynathan. HMM with auxiliary memory: a new tool for modeling RNA structures. 2:1651–1655 Vol.2, 2004. 1, 4, 12

[YV06] Byung-Jun Yoon e P. P. Vaidyanathan. Context-Sensitive Hidden Markov Models for Modeling Long-Range Dependencies in Symbol Sequences. *Signal Processing, IEEE Transactions on*, 54(11):4169–4184, 2006. 13, 14, 16, 27

[YV07] Byung-Jun Yoon e P.P. Vaidyanathan. Computational identification and analysis of noncoding RNAs - Unearthing the buried treasures in the genorne. *Signal Processing Magazine, IEEE*, 24(1):64–74, 2007. 3

[YV08a] Byung-Jun Yoon e P.P. Vaidyanathan. Fast Structural Alignment of RNAs by Optimizing the Adjoining Order of Profile-csHMMs. *Selected Topics in Signal Processing, IEEE Journal of*, 2(3):400–411, 2008. 1

[YV08b] Byung-Jun Yoon e P.P. Vaidyanathan. Structural Alignment of RNAs Using Profile-csHMMs and Its Application to RNA Homology Search: Overview and New Results. *Automatic Control, IEEE Transactions on*, 53(Special Issue):10–25, 2008. 1, 4, 29, 30, 38

[YZLZ01] Zhiyuan Yang, Qingwei Zhu, Kunxin Luo e Qiang Zhou. The 7SK small nuclear RNA inhibits the CDK9/cyclin T1 kinase to control transcription. *Nature*, 414(6861):317–322, 2001. 1

[ZS81] Michael Zuker e Patrick Stiegler. Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information. *Nucleic Acids Research*, 9(1):133–148, 1981. 3