COPPE/UFRJ

APLICANDO TILDECRF NA DETECÇÃO DE HOMOLOGIAS DISTANTES

Joel Bruno Santos da Costa

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia de Sistemas e Computação.

Orientadores: Gerson Zaverucha
Vítor Manuel de Morais Santos Costa

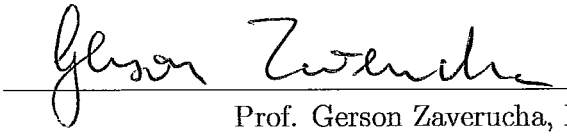
Rio de Janeiro
Setembro de 2008

APLICANDO TILDECRF NA DETECÇÃO DE HOMOLOGIAS DISTANTES

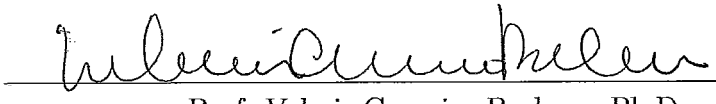
Joel Bruno Santos da Costa

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

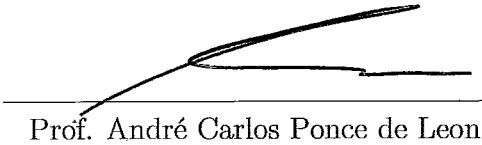
Aprovada por:



Prof. Gerson Zaverucha, Ph.D.



Prof. Valmir Carneiro Barbosa, Ph.D.



Prof. André Carlos Ponce de Leon Ferreira de Carvalho, Ph.D

RIO DE JANEIRO, RJ - BRASIL

SETEMBRO DE 2008

Costa, Joel Bruno Santos da

Aplicando TildeCRF na Detecção de Homologias Distantes/ Joel Bruno Santos da Costa. - Rio de Janeiro: UFRJ/COPPE, 2008

XIV, 76 p.: il.; 29,7 cm.

Orientadores: Gerson Zaverucha

Vítor Manuel de Morais Santos Costa

Dissertação (mestrado) - UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2008

Referências Bibliográficas: p. 64-76

1. Aprendizado de Máquina. 2. ILP. 3. Inferência Estatística. 4. Biologia Computacional. 5. Detecção de Homologias. I. Zaverucha, Gerson *et al.* II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

Dedicatória

Ao meu pai, meu exemplo de força de vontade

Agradecimentos

À minha querida mãe, Mercedes, por todo amor e carinho.

Ao meu querido pai, Aridio, por todo apoio e incentivo.

À minha querida esposa, Eliane, por não me deixar desistir e me ajudar a enfrentar todas as dificuldades.

A todos os familiares, por acreditarem.

Ao meu orientador, Professor Doutor Gerson Zaverucha, por me dedicar seu tempo e por confiar em mim para a elaboração deste trabalho.

Ao meu co-orientador, Professor Doutor Vítor Santos Costa, por responder pacientemente todas as minhas dúvidas e por estar sempre pronto a me ajudar.

A todos os amigos que fiz durante o mestrado pelo apoio, carinho e torcida, especialmente

a Juliana Bernardes, que deu início a este trabalho e me ajudou constantemente a dar continuidade a ele,

a Aline Paes e a Kate Revoredo, que estiveram sempre disponíveis e me ajudaram nas mais diversas tarefas,

e a Fábio Jimenez e a Luis Rigo, que me ajudaram com o cluster.

Obrigado a todos aqueles que torceram por mim.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

APLIACANDO TILDECRF NA DETECÇÃO DE HOMOLOGIAS DISTANTES

Joel Bruno Santos da Costa

Setembro/2008

Orientadores: Gerson Zaverucha
Vítor Manuel de Morais Santos Costa

Programa: Engenharia de Sistemas e Computação

Existe hoje em dia uma quantidade significativa de pesquisa envolvendo a detecção de homologias distantes entre seqüências de proteínas, um importante problema em Biologia Molecular Computacional. Os melhores resultados são obtidos com o uso de um método probabilístico denominado *profile hidden Markov models* (pHMM). Bernardes mostrou que a sensibilidade desses modelos aumenta significativamente ao adicionarmos informações estruturais no momento do treinamento.

Por outro lado, muitos trabalhos têm adotado modelos discriminativos, como os *Conditional Random Fields* (CRF), no lugar de generativos, como são os *hidden Markov models* (HMM), na solução de problemas de aprendizado de dados seqüencias. Em trabalho recente Gutmann e Kersting propuseram uma extensão de CRF, TildeCRF, onde as seqüências são formadas por átomos lógicos, incorporando toda a expressividade da lógica de primeira ordem às vantagens dos modelos discriminativos. Os resultados iniciais na predição de estruturas secundárias de proteínas foram promissores.

A principal contribuição desse trabalho foi desenvolver uma metodologia para usar o TildeCRF no problema de detecção de homologias distantes. Três tipos de experimentos foram feitos, cada um com um nível de informação diferente no treinamento. Os resultados foram comparados entre si, confirmando o aumento de acertividade também para o modelo discriminativo quando este dispõe de mais informações. Na comparação com programas específicos para o problema, a saber HMMER e HMMER-STRUCT, o TildeCRF não foi competitivo, mostrando que ainda muitos ajustes são necessários.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

REMOTE HOMOLOGY DETECTION APPLYING TILDECRF

Joel Bruno Santos da Costa

September/2008

Advisors: Gerson Zaverucha
Vítor Manuel de Morais Santos Costa

Program: Systems Engineering and Computer Science

Recently There has been a significant amount of research involving the detection of remote homologies between sequences of proteins, a major problem in Computational Molecular Biology. The best results are obtained using a method known as probabilistic *profile hidden Markov models* (pHMM). Bernardes showed that greater sensitivity can be achieved by using structural information when available.

In context, other fields of research have adopted discriminative models, such as *Conditional Random Field* (CRF), instead of generative ones, such as *hidden Markov model* (HMM), in the solution of problems of sequential data learning. In recent work Gutmann and Kersting proposed an extension of CRF, TildeCRF, where the sequences are formed by logical atoms, combining the expressiveness of the first order logic with the advantages of discriminative models. Initial results in predicting the secondary structures of proteins have been promising.

The main contribution of this work was to develop a methodology to use TildeCRF in the problem of detecting remote homologies. Three types of experiments were made, each with a different level of information in training. The results were compared with each other, confirming the increase of accuracy also for the discriminative model when it has more information. In comparison with specific programmes to the problem, namely HMMER and HMMER-STRUCT, the TildeCRF was not competitive, showing that many adjustments are still necessary.

Sumário

Lista de Abreviaturas

BLP *Bayesian Logic Programs*

CLP(BN) *Constraint Logic Programming*

CRF *Conditional Random Field*

FOLDT *First Order Logical Decision Tree*

glb *greatest lower bound*

HMM *Hidden Markov Model*

ICL *Independent Choice Logic*

ILP Programação em Lógica Indutiva

LOHMM *Logical Hidden Markov Model*

MGU *Most General Unifier*

pHMM *profile Hidden Markov Model*

PRM *Probabilistic Relational Models*

SLP *Stochastic Logic Program*

SLR *Statistical Relational Learning*

SCOP *Structural Classification of Proteins*

TDIDT *Top-down Induction of Decision Trees*

TILDE *Top-down Induction of Logical Decision Trees*

SVM *Support Vector Machine*

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

Capítulo 1

Introdução

1.1 Motivação

Um importante problema em Biologia Molecular Computacional é a detecção de homólogos distantes, que são proteínas que têm um ancestral comum, mas divergem significativamente na sua história evolutiva. Uma série de ferramentas tem sido desenvolvida para este fim, tais como SAM (HUGHEY, KROGH, 1995) e HMMER (EDDY, 1998). Sem dúvida, uma abordagem eficaz e popular é a que traça o perfil de uma família de proteínas por meio de um modelo probabilístico chamado *profile hidden Markov model* (pHMM) (EDDY, 1998). Então, uma nova proteína é alinhada contra esses pHMM com o intuito de definir de qual família ela faz parte. Esses modelos são treinados, na maioria das vezes, apenas com informações da *estrutura primária* (seqüência de aminoácidos) das proteínas homólogas, apesar de trabalhos recentes demonstrarem que uma maior sensibilidade pode ser alcançada por meio da utilização de informações das estruturas secundária e terciária quando disponíveis (BERNARDES, 2005).

Hidden Markov model (RABINER, 1989) fornece um modelo generativo para dados seqüenciais (dados que podem ser dispostos em seqüências). Recentemente, tem havido um grande interesse em modelos discriminativos para seqüências de dados, como *Conditional Random Field* (CRF) (LAFFERTY, et al., 2001) e *Support Vector Machine* (SVM) (VAPNIK, 1995). Guntmann e Kersting apresentam o TildeCRF (GUTMANN, KERSTING, 2006), uma extensão para CRFs na qual

uma seqüência é formada por átomos lógicos, fornecendo assim uma plataforma de trabalho natural para expressarmos dados estruturados. Os resultados iniciais em testes de previsão de estrutura secundária de proteínas foram muito promissores.

1.1.1 O problema: detecção de homologia distante

Duas proteínas são homólogas caso tenham evoluído a partir de um ancestral comum e, provavelmente, compartilham a mesma função. Uma família de proteínas agrupa as proteínas homólogas. Então, dada uma nova proteína gostaríamos de avaliar se ela pode pertencer a uma determinada família, isto é, se ela é homóloga as proteínas de uma determinada família.

O método que vem sendo mais utilizado para detecção de homologias é a comparação por similaridade. O objetivo é encontrar sinais que as identifiquem como um possível membro do conjunto padronizado. Quando a similaridade entre as seqüências homólogas é baixa, elas ainda podem ser homóloga, neste caso homólogas distantes.

Uma forma de fazer a detecção é avaliar diretamente os alinhamentos de seqüências de aminoácidos que formam as proteínas, ou os alinhamentos estruturais das proteínas, de uma mesma família. Outra forma é criar modelos matemáticos a partir desses alinhamentos e obter assim um perfil da família.

1.1.2 Trabalhos relacionados

Abordagens tradicionais para detecção de homologias são baseadas em métodos que buscam por similaridade seqüencial (ALTSCHUL, et al., 1990) (PEARSON, 1990), ou seja, elas comparam a nova proteína com base de dados públicas, tais como GENBANK (BENSON, et al., 2005), TREMBL (BOECKMANN, et al., 2003) e SWISS-PROT (BAIROCH, BOECKMANN, 1993), buscando por similaridades seqüenciais entre a nova proteína e proteínas de função conhecida. Ou usam modelos probabilísticos, construídos a partir de alinhamentos múltiplos de seqüências homólogas, que descrevem o grau de conservação dos aminoácidos das proteínas (DURBIN, et al., 1998). Esses métodos funcionam bem para o reconhecimento de proteínas que possuem fortes similaridade em suas seqüências, mas falham na

detecção de *homologias distantes*. Duas proteínas são ditas homólogas distantes quando a similaridade entre suas seqüências é um fraco indicativo de homologia, mas a *estrutura tridimensional* (disposição espacial de seus átomos) é muitas vezes bem conservada, Bernardes (BERNARDES, 2005) mostra que alinhamentos obtidos por meio de informações estruturais podem levar a modelos mais expressivos do que os alinhamentos obtidos apenas das informações das seqüência.

Isso levanta a questão de como pode-se melhorar a detecção de homólogos distantes por meio do uso de informações estruturais. Uma abordagem baseia-se nas, amplamente disponíveis, implementações de pHMM, tais como SAM (HUGHEY, KROGH, 1995) e HMMER (EDDY, 1998). Alguns sistemas estendem o HMMER e consideram informações estruturais na construção dos modelos probabilísticos. Os bons resultados experimentais obtidos pelo HMMER-STRUCT (BERNARDES, 2005) mostraram que o uso de informações estruturais secundárias e terciária melhorou a qualidade do modelo.

HMMER e SAM são implementações muito eficientes de modelos generativos proposicionais para dados seqüenciais. Recentemente, muito trabalhos se dedicaram a investigar o uso de modelos discriminativos, tal como CRF (LAFFERTY, et al., 2001) e SVM (VAPNIK, 1995). Outra linha importante de trabalho investigou o uso de modelos lógicos para seqüências estruturadas. PRISM (SATO, KAMEYA, 1997), SLPs (MUGGLETON, 2002), CLP(BN) (COSTA, et al., 2008b), LoHMMs (KERSTING, et al., 2006), *Relational Markov networks* (RMN) (TASKAR, et al., 2002) , *Relational Markov models* (ANDERSON, et al., 2002) e TildeCRF (GUTMANN, KERSTING, 2006) são diferentes abordagens, mas todas podem ser usadas para modelar dados de seqüências estruturadas. PRISM e CLP(BN) fornecem uma plataforma de trabalho geral, que permite mais compactação na descrição dos modelos, e foram usados para implementar HMMs. Em contraste, LOHMM foi desenhado especificamente para lidar com seqüências de átomos lógicos. Uma alternativa bastante diferente foi incrementar Makov Fields para suportar lógica de primeira ordem, como fizeram os modelos de *Relational Markov networks* (RMN) e *Markov logic network* (MLN) (RICHARDSON, DOMINGOS, 2006). Seguindo o mesmo conceito dos LOHMMs, o TildeCRF pode ser visto como um passo na direção de restringir es-

sas ferramentas com grande expressividade para o propósito específico de lidar com seqüências lógicas.

CRFs foram aplicados com sucesso em várias aplicações de bioinformática tais como predição de estrutura secundária (GUTMANN, KERSTING, 2006), classificação de proteínas em *folds* (GUTMANN, KERSTING, 2006) (LIU, et al., 2005), alinhamento de seqüências de proteínas (DO, et al., 2006) e alinhamentos de estrutural de RNA (SATO, SAKAKIBARA, 2005).

1.2 Objetivos

O trabalho (BERNARDES, 2005) desenvolveu um novo método para treinar pHMMs considerando alinhamentos tridimensionais e diferentes propriedades estruturais sobre um conjunto de proteínas homólogas. Nesse método diferentes pesos são atribuídos a cada aminoácido do conjunto de proteínas alinhadas de acordo com a sua importância estrutural. Na abordagem seguida, as distribuições de probabilidades de emissão em pHMMs continuam sobre o alfabeto de aminoácidos, enquanto em abordagens anteriores, tais como (CHAKRABARTI, SOWDHAMINI, 2004), (GOYON, TUFFÉRY, 2004) e (BYSTROFF, BAKER, 2000), outros alfabetos especiais eram necessários.

Seguindo a mesma direção, nossa abordagem procurou manter o foco sobre o alfabeto de aminoácidos, mas, em vez de pesos, investimos na expressividade da lógica de predicados para adicionarmos as informações estruturais. Escolhemos como ferramenta o TildeCRF (GUTMANN, KERSTING, 2006) para os experimentos. Ele gera modelos discriminativos, usando CRFs, por meio da aprendizagem incremental de árvores de regressão. Esses modelos têm algumas vantagens sobre os modelos generativos, como pHMM.

Então, o objetivo principal deste trabalho é investigar se um método discriminativo de classificação de seqüências representadas por átomos lógicos pode obter bons resultados e ser competitivo na detecção de homólogos distantes.

1.3 Organização do trabalho

Os próximos capítulos dessa dissertação estão organizados da seguinte forma:

- o capítulo 2 define os conceitos biológicos que serão utilizados no decorrer do trabalho. Inicialmente, são descritas as proteínas e a formação de suas estruturas. Depois, é definido o conceito de homologia em proteínas e de forma isso é usado para agrupá-las;

- o capítulo 3 apresenta os modelos computacionais básicos relacionados a este trabalho. Os HMMs e seus algoritmos são abordados inicialmente e é apresentado uma aplicação específica para a classificação de proteínas, pHMM. Em seguida é feita uma comparação entre modelos generativos e discriminativos. Por fim, os CRF e os seus algoritmos são descritos;

- o capítulo 4 apresenta as ferramentas computacionais aplicadas em problemas de aprendizado relacional usadas nesse trabalho, em especial, modelos que usam lógica de primeira ordem. O Tilde (GUTMANN, KERSTING, 2006) e os conceitos lógicos de primeira ordem são discutidos no início do capítulo. Em seguida, é apresentada uma proposta de ferramenta que une HMM e lógica de primeira ordem LOHMM (KERSTING, et al., 2006). Seguindo a mesma linha de LOHMM, o TildeCRF une CRF e lógica de primeira ordem. Este é descrito no final do capítulo;

- o capítulo 5 apresenta a metodologia proposta para os experimentos, sendo definido a forma de conversão das seqüências de aminoácidos para a linguagem lógica de primeira ordem, assim como a incorporação de informações adicionais. Os resultados dos procedimentos de teste são descritos e os resultados são apresentados. Ao final, é realizada uma comparação do TildeCRF com o HMMER e HMMER-STRUCT;

- o capítulo 6 conclui o trabalho, apresentando uma discussão sobre os resultados alcançados e destacando possíveis caminhos para trabalhos futuros.

Capítulo 2

Conceitos Biológicos

Este capítulo apresenta resumidamente os conceitos básicos de biologia necessários para a compreensão desta dissertação. Similaridades estruturais e funcionais podem ser usadas para determinar a homologia entre duas proteínas. A seção 2.1 define proteína e suas estruturas. A seção 2.2 aborda o conceito de homologia e similaridade.

2.1 Proteínas

As proteínas são vitais para a vida celular (ALBERTS, et al., 2002), sendo responsáveis por muitas das funcionalidades dentro do organismos vivos. Por isso, é importante estudá-las.

Proteínas são compostos orgânicos complexos constituídos a partir de aminoácidos que são sintetizadas a partir de genes presentes no DNA. Para formar uma proteína os aminoácidos são dispostos em cadeias lineares que são estabelecidas por ligações denominadas ligações peptídicas. Uma proteína é um conjunto de cem ou mais aminoácidos, sendo os conjuntos menores, de até dez aminoácidos, denominados polipeptídeos.

Existem trezentos tipos de aminoácidos, porém somente vinte são encontrados no organismo humano , eles são chamados de aminoácidos primários ou padrões, apenas esses aminoácidos podem ser sintetizados pelo DNA humano. Veja a tabela 2.1.

As proteínas foram descritas pelo químico Jöns Jakob Berzelius, inicialmente, em 1838. Porém, somente em 1926, o importante papel das proteínas para os orga-

nismos vivos foi descoberto por James B. Sumner, quando ele mostrou que a enzima urease era uma proteína (SUMNER, 1926). Já a primeira proteína a ser seqüenciada foi a Insulina, por Frederick Sanger, em 1955.

Proteína	Símbolo	Abreviação
Glicina ou Glicocola	Gly, Gli	G
Alanina	Ala	A
Leucina	leu	L
Valina	Val	V
Isoleucina	Ile	I
Prolina	Pro	P
Fenilalanina	Phe, Fen	F
Serina	Ser	S
Treonina	Thr, The	T
Cisteina	Cis	C
Tirosina	Tyr, Tir	Y
Asparagina	Asn	N
Glutamina	Gln	Q
Aspartato ou Ácido aspártico	Asp	D
Glutamato ou Ácido glutâmico	Glu	E
Arginina	Arg	R
Lisina	Lys, Lis	K
Histidina	His	H
Triptofano	Trp, Tri	W
Metionina	Met	M

Tabela 2.1: Símbolo para os 20 aminoácidos primários.

Uma proteína pode ter quatro tipos de estrutura: primária, secundária, terciária e quaternária. Cada estrutura representa um estágio, veja a figura 2.1, da formação da estrutura final da proteína, que se inicia a partir da síntese da proteína.

A *estrutura primária* é a seqüência de aminoácidos ligados pelas pontes peptídicas como uma cadeia. Essa ligações possuem natureza planar e bastante rígida.

De acordo com interações intramoleculares, os aminoácidos mais próximos sofrem um arranjo espacial, formando a *estrutura secundária.* O arranjo secundário de uma proteína pode ocorrer de forma regular e os exemplos mais comuns são alfa-hélices (PAULING, et al., 1951) e folhas-beta (BRANDEN, TOOZE, 1991a). O arranjo também pode ocorrer de forma irregular sendo formado por regiões denominadas *coils* ou *loops* (BRANDEN, TOOZE, 1991a).

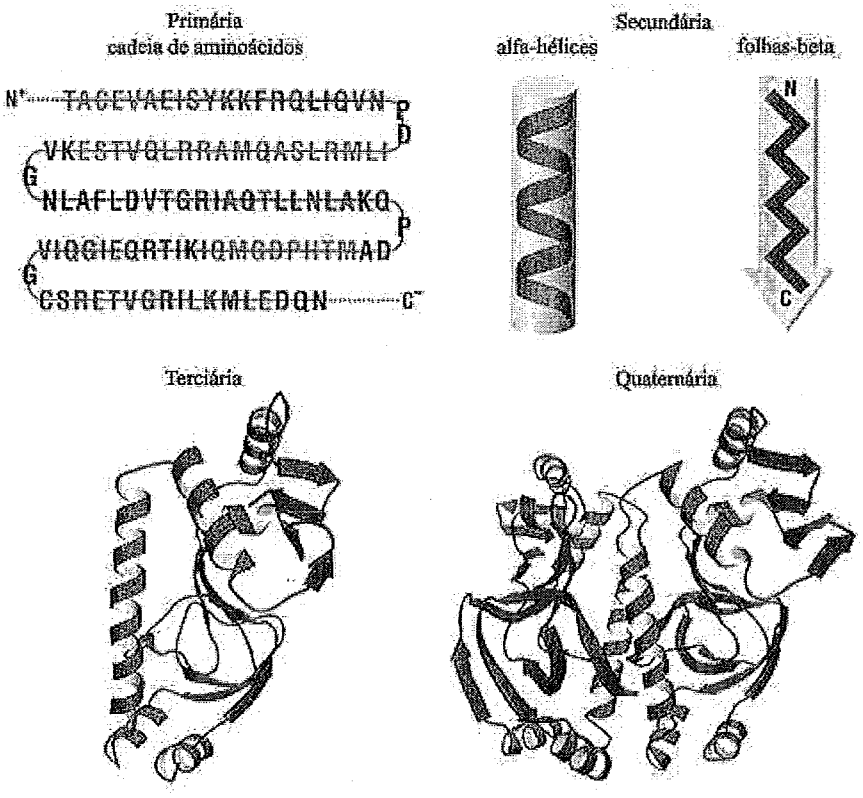


Figura 2.1: Fases da estrutura das proteínas (PETSKO, RINGE, 2004).

A *estrutura terciária* refere-se ao arranjo tridimensional de todos os átomos da proteína, completando a conformação da cadeia polipeptídica (BOURNE, WEISSIG, 2003). Ela descreve o dobramento final da cadeia sendo mantido por pontes de hidrogênio e dissulfídicas. Por um lado, a estrutura secundária apresenta interações moleculares de curta distância, por outro, a estrutura terciária apresenta as interações de longa distância e estas são entre estruturas secundárias.

Muitas dessas combinações de estruturas secundárias são freqüentemente encontradas em estruturas tridimensionais de diversas proteínas e algumas delas podem assumir papel funcional. Essas combinações são chamadas de *motivos* (BRANDEN, TOOZE, 1991b).

A maioria das proteínas é formada pela associação de várias estruturas terciárias, ou seja, várias cadeias polipeptídicas. Dessa união surge a *estrutura quaternária* que pode produzir diferentes funções para os compostos. Um exemplo de estrutura quaternária é a hemoglobina.

2.2 Homologia e Similaridade

Os organismos vivos compartilham propriedades bioquímicas, apesar das inúmeras diferenças entre eles (SCOTT, et al., 2000). Os processos evolutivos modificam as espécies dando-lhes características particulares e mantendo o que é importante dos seus ancestrais. Duas proteínas são homólogas caso tenham evoluído a partir de um ancestral comum.

Na bioinformática essa homologia é geralmente inferida com base em similaridade entre seqüências. Porém, similaridade não indica necessariamente que há um ancestral comum, logo nem sempre reflete homologia. Por outro lado, baixa similaridade não é suficiente para descartar a possibilidade de homologia. Nesta situação, os homólogos são chamados de homólogos distantes.

A seção 2.2.1 descreve os métodos para a detecção de seqüências homólogas, em especial, os programas de alinhamento, e a seção 2.2.2 define o SCOP, a base de dados que foi usada nos experimentos desta dissertação.

2.2.1 Alinhamento

As proteínas homólogas são geralmente divididas em famílias. Tais famílias servem de referência para a classificação de novas proteínas seqüenciadas em projetos genômicos, tanto pela comparação das seqüências, quanto pelo compartilhamento de funções. Os genes e proteínas seqüenciados passam por um processo conhecido como anotação, para que suas funções sejam definidas, após este processo os genes e proteínas são depositados em bancos de dados, tais como GENBANK (BENSON, et al., 2005), TREMBL (BOECKMANN, et al., 2003) e SWISS-PROT (BAIROCH, BOECKMANN, 1993). Posteriormente, essas bases de dados auxiliam o processo de anotação de novas seqüências.

A detecção semi-automática de seqüências homólogas reduz o tempo e o custo da classificação, economizando etapas nos testes laboratoriais, que têm um alto custo. Essa detecção é possível por meio de programas de alinhamento. BLAST (ALTSCHUL, et al., 1990) e FASTA (PEARSON, 1990) são exemplos de programas que alinham apenas pares de seqüências, isto é, são ferramentas de alinhamento par a par, ou seja, alinham uma proteína recém seqüenciada com cada proteína

de uma base de dados de proteínas anotadas. Já ferramentas como CLUSTALW (THOMPSON, et al., 1994), TCOFFEE (NOTREDAME, et al., 2000) e ALIGN-M (WALLE, et al., 2004) podem alinhar mais de duas seqüências e são conhecidas como ferramentas de alinhamento múltiplo.

A figura 2.2 mostra um exemplo de alinhamento de três seqüências de aminoácidos. Para alinhar as seqüências da melhor maneira possível, ou seja, tendo o máximo de resíduos idênticos alinhados, é necessário introduzir buracos no alinhamento das seqüências, esses são representados por um traço na figura 2.2. O termo resíduo, comumente, é utilizado para fazer referência a aminoácidos ou nucleotídeos. Os buracos são interpretados como inserções ou exclusões que ocorreram durante a evolução. Se duas seqüências de um alinhamento partilharem um ancestral comum, divergências ou lacunas poderão ser interpretadas como mutações pontuais provenientes do processo evolutivo. No alinhamento de proteínas, o grau de similaridade entre os aminoácidos que ocupam uma determinada posição pode ser interpretado como uma medida de conservação evolutiva.

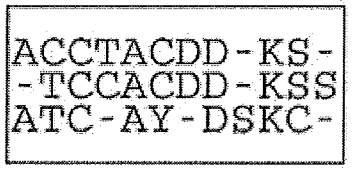


Figura 2.2: Alinhamento entre três seqüências de aminoácidos.

Uma grande variedade de algoritmos computacionais foram aplicados no problema de alinhamento de seqüências, incluindo programação dinâmica (BELLMAN, 1957) (SMITH, WATERMAN, 1981), métodos progressivos (THOMPSON, et al., 1994) (NOTREDAME, et al., 2000) e métodos probabilísticos (BATEMAN, et al., 2004) (HAFT, et al., 2003) (DO, et al., 2006).

A fim de melhorar a eficácia dos métodos a serem utilizados para a detecção de homologias distantes surge o critério de similaridade estrutural entre proteínas (CHAKRABARTI, SOWDHAMINI, 2004) (ESPADALER, et al., 2005) (ALEXANDROV, GERSTEIN, 2004) (HOU, et al., 2004). As estruturas tridimensionais das proteínas apresentam uma estrutura mínima, mesmo tendo sofrido alguma alteração quanto à organização de seus aminoácidos, que conserva as informações de sua

origem. Isso possibilita a comparação de proteínas quanto à origem. A figura 2.3 apresenta parte da estrutura tridimensional de duas proteínas homólogas distantes e a figura 2.4 mostra o alinhamento estrutural espelhado nas seqüências dessas mesmas duas proteínas. Podemos observar que as estruturas similares não correspondem necessariamente às regiões em que os aminoácidos são bem conservados.

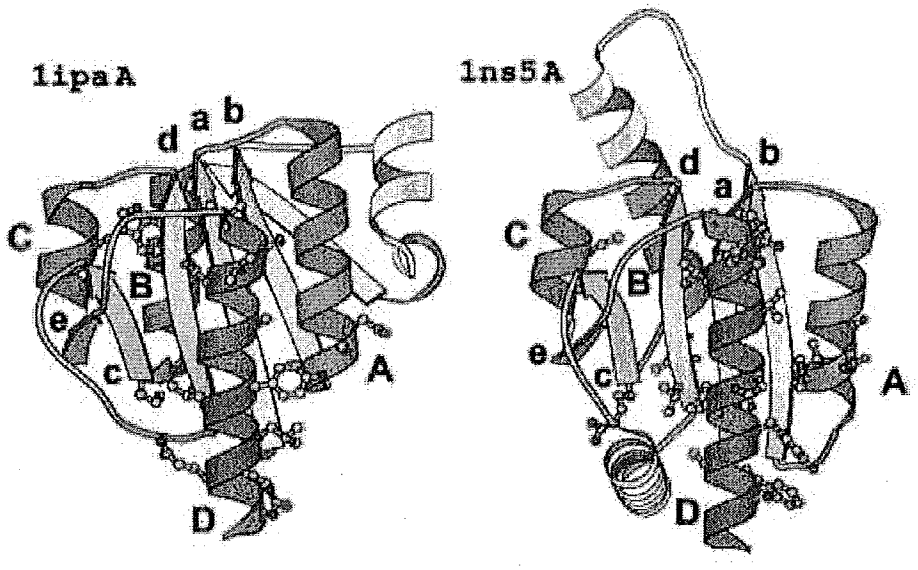


Figura 2.3: Estruturas de duas proteínas homólogas distantes (SADREYEV, GRISHIN, 2004). As estruturas secundárias similares são sinalizadas com letras maiúsculas (alfa-hélices) e minúsculas (folhas-beta).

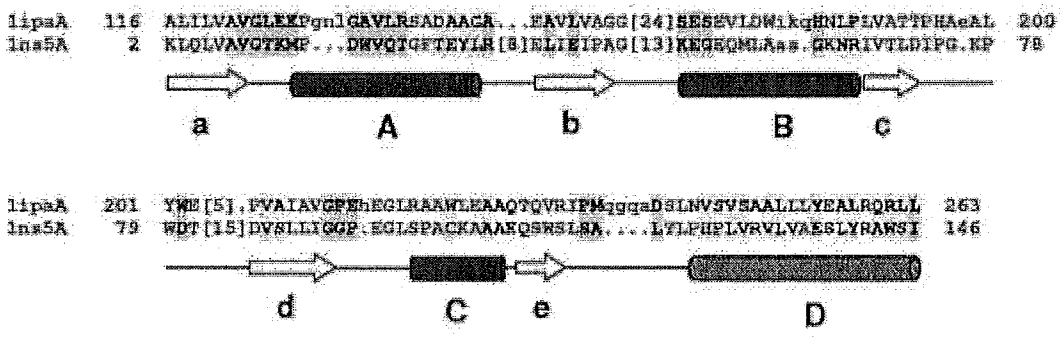


Figura 2.4: Alinhamento das seqüências das duas proteínas da figura 2.3 (SADREYEV, GRISHIN, 2004).

2.2.2 SCOP - Classificação Estrutural de Proteínas

Todas as proteínas de estrutura conhecida têm suas coordenadas espaciais inseridas na base PDB (BERMAN, et al., 2000) e, a partir dela, a base de dados

SCOP (ANDREEVA, et al., 2004) classifica todos os domínios dessas proteínas em uma ordem hierárquica com quatro níveis: família, super-família, dobramentos e classe.

No SCOP, uma *família* agrupa proteínas com identidade maior que 30% ou proteínas com funções e estruturas similares. Uma *super-família* é um conjunto de famílias em que as características estruturais sugerem um ancestral comum. As super-famílias são organizadas em *dobramentos* que possuem estruturas secundárias que compartilham ligações químicas e conexões topológicas. Por sua vez, os dobramentos são organizados em *classes* de acordo com a predominância dos elementos de estrutura secundária de suas proteínas. As classes são: *alfa*, em que as proteínas são essencialmente formadas por alfa-hélices, *beta*, onde as folhas beta são predominantes, *alfa e beta*, em que ocorrem muitas alfa-hélices e folhas beta intercaladas, *alfa mais beta*, com muitas ocorrências separadas de alfa-hélices e folhas beta e *múltiplos domínios*, no qual ocorrem diferentes dobramentos.

Muitos estudos usaram a base de dados SCOP para avaliar o desempenho de métodos voltados para a detecção de homologias, entre eles (ESPADALER, et al., 2005), (WISTRAND, SONNHAMMER, 2005), (SÖDING, 2005), (ALEXANDROV, GERSTEIN, 2004), (HOU, et al., 2004) e (BERNARDES, et al., 2007).

Capítulo 3

Modelos Computacionais

Este capítulo apresenta os métodos computacionais aplicados em problemas de classificação de proteínas usados neste trabalho. A seção 3.1 aborda os *hidden Markov models* (HMM), definindo os algoritmos mais utilizados de inferência e estimativa de parâmetros e, no final da seção, é apresentado uma aplicação específica para a classificação de proteínas. A seção 3.2 compara os modelos generativos com os discriminativos. A seção 3.3 define, a partir dos HMMs, os *conditional random fields* (CRF) e seus algoritmos de inferência e estimativa de parâmetros.

3.1 Hidden Markov Models

Inicialmente usados em aplicações de reconhecimento de voz (MENDEL, 1992), os *hidden Markov models* (RABINER, 1989) (HUGHEY, KROGH, 1996) são hoje em dia amplamente utilizados na análise de seqüências biológicas (MAMITSUKA, 2005) (EDGAR, SJÖLANDER, 2004) (KNUDSEN, MIYAMOTO, 2003) (BAE, et al., 2005) (CAMPROUX, TUFFÉRY, 2005) (LIN, et al., 2005) (BREJOVA, et al., 2005) (MAJOROS, et al., 2005).

O objetivo desta seção é descrever os conceitos básicos de HMM. Primeiramente, definiremos cadeia de Markov e HMM. E depois, as sessões 3.1.1 e 3.1.2 abordarão os algoritmos para inferência e estimativa de parâmetros, respectivamente. Profile HMMs (KROGH, et al., 1994) são usados para criar modelos de famílias de proteínas e serão apresentados na seção 3.1.3.

Geralmente *cadeias de Markov* são descritas como grafos direcionados, onde os nós são os estados e as arestas têm probabilidades associadas, que seguem a

propriedade de Markov, isto é, a probabilidade de estar em um estado só depende do estado imediatamente anterior, e não de toda a seqüência de estados trilhados até ali. Neste caso, dizemos que se trata de uma cadeia de primeira ordem. E, em uma cadeia de ordem k, o estado atual pode depender dos k estados anteriores.

As probabilidades associadas às arestas são portanto as probabilidades de transição entre os estados. Sejam dois estados s e t e uma aresta a que liga os dois, então a probabilidade de transição entre os estados s e t é $P(y_i = t|y_{i-1} = s)$. A soma das probabilidades de transição que saem de um estado deve ser igual a 1, $\sum_{\forall t} P(y_i = t|y_{i-1} = s) = 1$

A probabilidade de uma seqüência de estados é definida através do princípio da probabilidade condicional (DEGROOT, 1987). Em um modelo M, uma seqüência de estados y de tamanho n:

$$P(y, M) = P(y_n|y_{n-1})P(y_{n-1}|y_{n-2})...P(y_2|y_1)P(y_1) = P(y_1)\prod_{i=2}^{n} P(y_i|y_{i-1}) \quad (3.1)$$

A cadeia de Markov da figura 3.1 é um modelo observável, uma vez que, dada uma seqüência de observações qualquer, é possível afirmar a seqüência de estado visitados para emitir essa seqüência. Isso porque existe uma correspondência um para um entre estados e símbolos emitidos, isto é, cada estado emite apenas um símbolo.

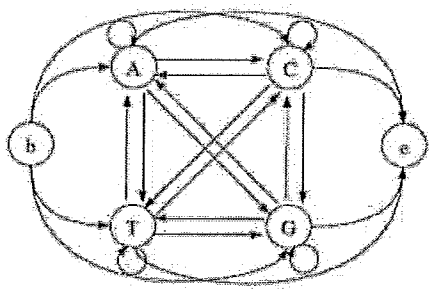


Figura 3.1: Cadeia de Markov para reconhecimento de seqüência de DNA, onde os estados A, C, T e G representam os nucleotídeos e b e e são estados de início e fim da cadeia.

Entretanto, se um estado emitir mais de um símbolo, é necessário uma função probabilística para as emissões neste estado. Suponhamos que todos os estados da figura 3.1 possam emitir os quatro símbolos usados na representação do DNA. Dada uma seqüência $x = TTGAC$, existem várias seqüências possíveis de estados

visitados. Dessa forma, a seqüência de estados que realmente emite os símbolos fica *escondida*.

Hidden Markov model (RABINER, 1989) é uma cadeia de Markov em que cada estado possui uma tabela de probabilidades de emissão para os símbolos possíveis do modelo. A soma das probabilidades desta tabela é igual a 1. Se existirem $b_1, ..., b_R$ símbolos que podem ser emitidos em um estado, então $\sum_{k=1}^{R} P(b_k) = 1$. A probabilidade conjunta de uma seqüência emitida x e uma seqüência de estados y de tamanho n é:

$$P(Y, X) = \prod_{i=1}^{n} P(y_i|y_{i-1})P(x_i|y_i) \tag{3.2}$$

A equação 3.2 especifica um HMM por meio de três distribuições de probabilidades. A primeira é a distribuição sobre os estados iniciais $P(y_1)$, igual a $P(y_1|y_0)$ para simplificar a notação. A segunda é a distribuição sobre as transições de estados, $P(y_i|y_{i-1})$. A terceira é a distribuição sobre as emissões, $P(x_i|y_i)$.

3.1.1 Inferência

Rabiner (RABINER, 1989) introduz a idéia que um HMM deve ser caracterizado por um dos três problemas a seguir:

1. dados um HMM $M(A, E)$, onde A é a matriz de probabilidades de transição e E é a matriz de probabilidades de emissão, e uma seqüência $X = X_1, X_2, ..., X_T$, computar eficientemente $P(X|M)$, isto é, a probabilidade da seqüência observada dado o modelo;

2. dados HMM $M(A, E)$ e X, determinar a melhor seqüência de estados escondidos $Q = q_1, q_2, ..., q_T$ para emitir X;

3. dados o HMM $M(A, E)$ e X, estimar A e E que maximizem $P(X|M)$.

Os dois primeiros são tratados nesta seção e o terceiro será visto na próxima.

Seja y um caminho em M composto por uma seqüência estados que inicia no estado inicial e termina no estado final. A verossimilhança (KARLIN, ALTSCHUL, 1990) da seqüência X descreve a probabilidade da seqüência ter sido gerada pelo modelo M através do caminho y. O primeiro problema descrito consiste em calcular

a verossimilhança de uma seqüência X dado o modelo M para todos os possíveis caminhos que gerem X, $\sum_y P(X|M)$. Por outro lado, a máxima verossimi-lhança indica o melhor caminho $y^* = \arg\max_y P(y|X, M)$, que é objetivo do segundo problema.

Como o número de caminhos pode crescer exponencialmente, os dois problemas podem ser computacionalmente muito difíceis. Porém, os métodos iterativos de programação dinâmica (BERTSEKAS, 1995) evitam a busca exaustiva de todos os caminhos possíveis.

Um HMM pode ser visto como um *grafo fator*, veja a figura 3.2, $p(y, x) = \prod_t \Psi_t(y_t, y_{t-1}, x_t)$ (SUTTON, MCCALLUM, 2006), onde $Z = 1$ e os fatores são definidos como:

$$\Psi_t(j, i, x) = p(y_t = j|y_{t-1} = i)p(x_t = x|y_t = j). \tag{3.3}$$

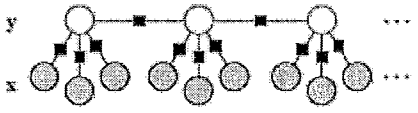


Figura 3.2: Exemplo de um HMM representado por um grafo fator.

O algoritmo *forward* (BALDI, BRUNAK, 2001) é um método que calcula a probabilidade da seqüência observada usando uma tabela para armazenar as probabilidades das sub-seqüências de X. A idéia por trás do algoritmo *forward* pode ser desvendada reescrevendo a soma $p(x) = \sum_y p(y, x)$ por meio da propriedade distributiva:

$$p(x) = \sum_y \prod_{t=1}^{T} \Psi_t(y_t, y_{t-1}, x_t) \tag{3.4}$$

$$= (\sum_y \Psi_T(y_T, y_{T-1}, x_T))(\sum_y \Psi_{T-1}(y_{T-1}, y_{T-2}, x_{T-1}))...(\sum_y \Psi_1(y_1, y_0, x_1)) \tag{3.5}$$

Podemos observar que cada uma das somas intermediárias é usada repetidas vezes para computar a soma completa. Portanto, economiza-se trabalho de forma exponencial armazenando-se essas somas internas. Podemos definir, então, um conjunto de variáveis *forward* α_t, cada uma sendo um vetor de tamanho M, que é o número de estados, para armazenar essas somas intermediárias.

$$\alpha_t(j) = p(x_1, x_2, ..., x_t|y_t = j) = \sum_{y_{1...t-1}} \Psi_t(j, y_{t-1}, x_t) \prod_{t'=1}^{t-1} \Psi_{t'}(y_{t'}, y_{t'-1}, x_{t'}), \tag{3.6}$$

onde o primeiro somatório varia sobre todas as possíveis atribuições as variáveis aleatórias $y_1, y_2, ..., y_{t-1}$. Em outras palavras, $\alpha_t(j)$ é a probabilidade de a seqüência observada até x_t, inclusive, dado que o estado atual é j ($y_t = j$). Os valores de α podem ser computados pela equação recursiva

$$\alpha_t(j) = \sum_i \Psi_t(j, i, x_t)\alpha_{t-1}(i), \tag{3.7}$$

com inicialização $\alpha_1 = \Psi_1(j, y_0, x_1)$. É fácil verificar que $p(x) = \sum_j \alpha_T(j)$ através de repetidas substituições da recursão 3.7 para obter 3.6. O pseudo-código do algoritmo *foward* é apresentado no algoritmo 3.1.

Algoritmo 3.1 Forward

entrada: o Modelo M e uma observação X de tamanho T

$\alpha_0(0) = 1$
para todo estado j de M, $j > 0$ **faça**
  inicializa $\alpha_0(j) = 0$

para todo estado j de M, $j > 0$ **faça**
  para $t = 1, ..., T$ **faça**
    $\alpha_t(j) = \sum_i \Psi_t(j, i, x_t)\alpha_{t-1}(i)$

retorna matriz α

término $P(X|M) = \sum_j \alpha_T(j)$

O algoritmo *backward* (BALDI, BRUNAK, 2001) também é um método que calcula a probabilidade da seqüências observada usando uma tabela para armazenar as probabilidades das sub-seqüências de X, mas a recursão é feita do final para o início das seqüências. Veja o algoritmo 3.2. Portanto, temos a seguinte definição:

$$\beta_t(i) =_{def} p(x_{t+1}, x_{t+2}, ..., x_T|y_t = i) = \sum_{y_{t+1}...y_T} \prod_{t'=t+1}^{T} \Psi_{t'}(y_{t'}, y_{t'-1}, x_{t'}), \tag{3.8}$$

e a equação recursiva

$$\beta_t(i) = \sum_j \Psi_{t+1}(j, i, x_{t+1})\beta_{t+1}(j), \tag{3.9}$$

a qual é inicializada com $\beta_T = 1$. Assim, como no algoritmo *forward*, pode-se computar $p(x)$ por meio das variáveis *backward*, também usadas para armazenar somas intermediárias, $p(x) = \beta_0(y_0) = \sum_j \Psi_1(y_1, y_0, x_1)\beta_1(j)$.

Algoritmo 3.2 Backward

entrada: o Modelo M e uma observação X de tamanho T

para todo estado j de M, $j > 0$ **faça**
  inicializa $\beta_T(j) = 1$

para todo estado j de M, $j > 0$ **faça**
  para $t = T - 1, ..., 1$ **faça**
    $\beta_t(j) = \sum_i \Psi_{t+1}(j, i, x_{t+1})\beta_{t+1}(i)$

retorna matriz β

término $P(X|M) = \sum_j \Psi_1(y_1, y_0, x_1)\beta_1(j)$

Combinando os resultados das recursões do *forward* e do *backward* pode-se computar a distribuição marginal necessária na estimativa de parâmetros. Novamente, aplicando a regra distributiva

$$p(y_t = i, y_{t+1} = j|x) = \alpha_t(i)\Psi_{t+1}(j, i, x_{t+1})\beta_{t+1}(j). \tag{3.10}$$

O algoritmo *Viterbi* (BALDI, BRUNAK, 2001), veja o algoritmo 3.3, é um método para encontrar o caminho mais provável y^* que gerou uma determinada sub-seqüência, $y^* = \arg\max_y P(y|x)$. Pode-se observar que o artifício usado em 3.5 também funciona se substituirmos os somatórios por funções que retornam os máximos. Então, obtemos a recursão

$$\delta_t(j) = \max_i \Psi_t(j, i, x_t)\delta_{t-1}(i). \tag{3.11}$$

Vale observar que a variável *ptr* do algoritmo 3.3 armazena os ponteiros para os estados anteriores, então, a atual seqüência de estados pode ser encontrada por meio de *backtracking*.

3.1.2 Estimativa de parâmetros

O terceiro problema definido por (RABINER, 1989) é o mais difícil: determinar um método para ajustar os parâmetros do modelo $M(A, E)$ de forma a maximizar $P(X|M)$. Os parâmetros de um modelo HMM são as probabilidades de transição A e emissão E. A estimativa desses parâmetros é geralmente feita a partir de um conjunto de seqüências de exemplo, denominado conjunto de treinamento, por meio

Algoritmo 3.3 Viterbi

entrada: o Modelo M e uma observação X de tamanho T

$\delta_0(0) = 1$
para todo estado j de M, $j > 0$ **faça**
  inicializa $\delta_0(k) = 0$

para todo estado j de M, $j > 0$ **faça**
  para $t = 1, ..., T$ **faça**
    $\delta_t(j) = \max_i \Psi_t(j, i, x_t)\delta_{t-1}(i)$
    $ptr_t(j) = \arg\max_j(\Psi_t(j, i, x_t)\delta_{t-1}(i))$

retorna matriz δ e lista ptr

término $P(X, y^*|M) = \max_j(\delta_T(j))$
  $y_T^* = \arg\max_j(\delta_T(j))$

traceback
para $t = T, ..., 1$ **faça**
  $y_{t-1}^* = ptr_t(y_t^*)$

de algum procedimento iterativo, como, por exemplo, métodos EM (*Expectation-Maximization*) (DEMPSTER, et al., 1977), técnicas com gradiente (LEVINSON, et al., 1983) (BAGOS, et al., 2004) ou métodos de otimização numérica (COLLINS, 2002)

O principal método usado para estimativa de parâmetros é conhecido por *Baum-Welch* (BAUM, 1972), um caso especial de EM. Informalmente, os novos valores das probabilidades são calculados considerando os caminhos mais prováveis sobre as seqüências de treinamento e os atuais valores das probabilidades. Esse processo é repetido até que algum critério de parada seja atingido. É possível mostrar que a log-verossimilhança do modelo cresce a cada iteração, levando o processo a convergir para um máximo local.

Formalmente, o algoritmo *Baum-Welch* calcula, nos dados de treinamento, o número esperado de ocorrências da transição de um estado i para um estado j e o número esperado de ocorrências da emissão de um símbolo b a partir de um estado j. Sejam esses valores A_{ij} e $E_j(b)$. Para isso, são usados os mesmos valores obtidos pelos algoritmos *forward* e *backward* como mostrado na equação 3.10.

Então, é possível derivar o número esperado de vezes que a transição $i \rightarrow j$ é

usada por meio da soma em todas as posições e em todos as seqüências de treinamento,

$$A_{ij} = \sum_k \frac{1}{P(x^k)} \sum_t \alpha_t^k(i) \Psi_{t+1}(j, i, x_{t+1}^k) \beta_{t+1}^k(j), \tag{3.12}$$

onde $\alpha_t^k(i)$ é a variável *forward* $\alpha_t(i)$ definida em 3.7 e $\beta_{t+1}^k(j)$ corresponde a variável *backward* $\beta_{t+1}(j)$ definida em 3.9, ambas calculadas para a seqüência k. Da mesma forma, pode-se encontrar o número esperado de vezes que um símbolo b é emitido no estado j,

$$E_j(b) = \sum_k \frac{1}{P(x^k)} \sum_{t|x_t^k=b} \alpha_t^k(j) \beta_t^k(j), \tag{3.13}$$

onde a soma interna é somente sobre as posições t onde o símbolo emitido for b.

Uma vez calculados esses valores esperados, sendo a_{ij} a probabilidade de transição de i para j e $e_j(b)$ a probabilidade de emissão do símbolo b no estado j, então os novos parâmetros do modelo são calculados por meio de

$$a_{ij} = \frac{A_{ij}}{\sum_{j'} A_{ij'}} \qquad e \qquad e_j(b) = \frac{E_j(b)}{\sum_{b'} E_j(b')}. \tag{3.14}$$

Então, inicia-se uma nova iteração usando os novos valores dos parâmetros com o objetivo de estimar novos valores para as variáveis A_{ij} e $E_j(b)$ e, a partir deles, os novos parâmetros, seguindo assim, sucessivamente, até atingir algum critério de parada estabelecido previamente, por exemplo, se a mudança na log-verossimilhança for suficientemente pequena. É conveniente também determinar um número máximo de rodadas de forma a garantir a parada.

O pseudo-código do *Baum-Welch* é apresentado no algoritmo 3.4, no qual são usados pseudo-contadores para inicializar os contadores A_{ij} e $E_j(b)$. Esse procedimento é necessário, porque as equações 3.14 de atualização dos parâmetros são muito vulneráveis a ocorrência de *overfitting*, principalmente nos casos que existem dados insuficientes. De fato, se um estado k nunca é usado pelo conjunto de seqüências de exemplos, então as equações são indefinidas para este estado, ocorrendo zero tanto no numerador quanto no denominador. Existem alguns métodos para a atribuição de pseudo-contadores, entre eles o *zero-offset* (TATUSOV, et al., 1994), que adiciona uma pequena constante z a cada contador, *pseudo contadores* (KARPLUS, 1995), que generaliza o anterior usando um conjunto de constantes z_i diferentes, e *misturas*

Algoritmo 3.4 Baum-Welch

entrada: o Modelo M, um conjunto de n seqüências e um alfabeto de símbolos

inicializa M com valores arbitrários nos parâmetros

recursão
atribui pseudo-contadores às variáveis A_{ij} e $E_j(b)$, $\forall i, j, b$
para cada seqüência $k = 1, ..., n$ **faça**
  calcula $\alpha_t(i)$ para k através do algoritmo 3.1
  calcula $\beta_t(i)$ para k através do algoritmo 3.2
  adiciona a contribuição de k nas variáveis A_{ij} e $E_j(b)$ pelas equações 3.12 e 3.13

calcula os novos parâmetros do modelo usando as equações 3.14
calcula a nova log-verossimilhança do modelo

retorna o modelos atualizado e a log-verossimilhança

término pára ao atingir o critério de parada ou o número máximo de rodadas

de Dirichlet (SJÖLANDER, et al., 1996), que combina várias distribuições *Dirichlet*, que são distribuições multinomial (DEGROOT, 1987), para estimar a probabilidade na ausência de dados reais.

3.1.3 Profile HMM

Profile hidden Markov models (KROGH, et al., 1994)é um HMM bastante apropriado para modelar alinhamentos múltiplos de seqüências de gens ou proteínas. Mais especificamente, um *profile* HMM tem três tipos de estados: "*match*", "*delete*" e "*insert*". Um estado "*match*" emite um aminoácido com uma certa probabilidade de acordo com a sua posição. Um estado "*insert*" emite aminoácidos com uma certa probabilidade de acordo com uma distribuição já existente. E um estado "*delete*" é um estado que não emite símbolos correspondendo aos buracos nos alinhamentos múltiplos de seqüências (figura 2.2).

Assumamos que as seqüências da família que se deseja modelar estejam alinhadas. Para construir um modelo capaz de representar as propriedades de uma família é necessário identificar padrões na formação das seqüências envolvidas, como, por exemplo, consenso ou padrão de formação em determinadas colunas através da prevalência de algum aminoácidos ou ausência deles.

A seqüência consenso representa o ancestral comum, a partir do qual cada

proteína derivou. Sendo que algumas perderam aminoácidos durante a evolução, outras apenas substituíram o aminoácido original por outro e finalmente algumas proteínas ganharam novos aminoácidos. Porém, a seqüência consenso permanece caracterizando os membros da família. Para maiores detalhes sobre alinhamentos de seqüências consulte a seção 2.2.1.

Para representar os padrões das seqüências por meio de pHMM, é necessário representar as colunas do alinhamento que retêm os consensos da família. Se as colunas com buracos são desconsideradas, observa-se um conjunto de blocos bem formados, que unidos contêm a seqüência consenso capaz de representar os padrões da família.

A primeira arquitetura proposta para pHMM, apresentada na figura 3.3 (HENIKOFF, HENIKOFF, 1991), considera apenas blocos de seqüências alinhadas sem buracos, fazendo uso apenas de estados "match". Os resíduos anteriores e posteriores ao bloco são representados pelos estados N e S. Os estados b e e são silenciosos e marcam o início e fim do padrão representado. Essa arquitetura foi baseada nas *positions-specific scoring matrices* (GRIBSKOV, et al., 1987).

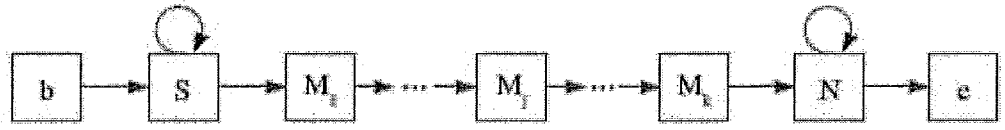


Figura 3.3: Primeira arquitetura de um pHMM.

Essa arquitetura evoluiu e passou a considerar os buracos por meio da inclusão de estados "inserts", I. Se tivermos dois blocos bem formados, os resíduos entre esses blocos serão as inserções. Essa arquitetura é adotada pelo programa META-MEME (GRUNDY, BAKER, 1997), que faz uso de pHMM para a detecção motivos. Veja sua representação gráfica na figura 3.4.

Porém, é necessário incluir o tratamento de exclusões com o uso dos estados "delete", D. Esses estados representam a ausência de resíduos em determinados estado de "match", ocasionando novas transições. Os estados D são silenciosos e seu propósito é adicionar saltos à arquitetura proporcionando uma melhor adaptação do pHMM aos padrões extraídos dos alinhamentos múltiplos de seqüências. A ar-

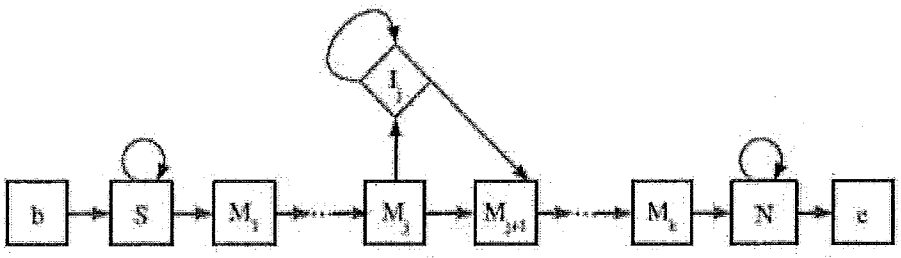


Figura 3.4: Arquitetura de um pHMM incluindo os estados de inserção.

quitetura completa, apresentada na figura 3.5, foi introduzida por (KROGH, et al., 1994). Esta arquitetura, com algumas alterações, é adotada pela maioria dos programas que implementam pHMMs na detecção de homologias, tais como HMMER (EDDY, 1998) e SAM (HUGHEY, KROGH, 1995).

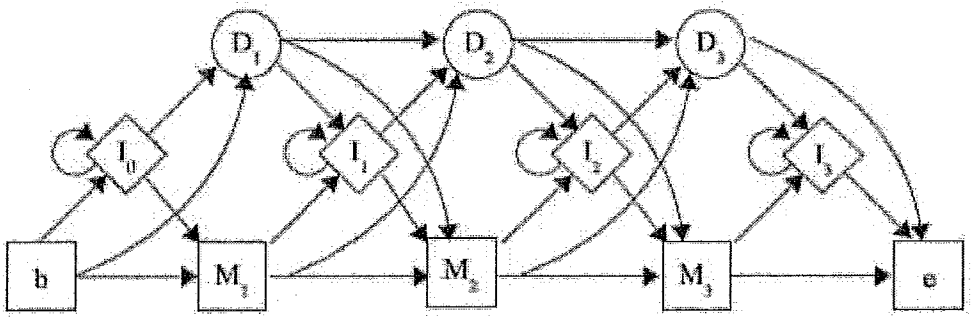


Figura 3.5: Arquitetura completa de um pHMM.

3.2 Modelos generativos e discriminativos

Os dois principais tipos de modelos nos problemas de aprendizado supervisionado são os *generativos* e os *discriminativos* (NG, JORDAN, 2002). Os modelos generativos modelam a distribuição conjunta $P(x, y)$, enquanto que os modelos discriminativos modelam a distribuição condicional $P(x|y)$.

A principal diferença entre eles é que a distribuição condicional não inclui um modelo de $p(x)$, o que não é mesmo necessário para a classificação. A dificuldade de modelar $p(x)$ é que ela geralmente contém muitas características altamente dependentes uma das outras. Há duas formas de incluir características interdependentes no modelo generativo: ou se amplia o modelo para representar estas dependências, ou se estabelece restrições de independência, assim como é feita no *naive* Bayes.

O sucesso dos modelos generativos se deve em grande parte as estas restrições assumidas nos modelos. Contudo, essas restrições nem sempre são verdadeiras. Em contraste, modelos discriminativos, como, por exemplo regressão logística (EFRON, 1975) e *suport vector machines* (VAPNIK, 1995), tipicamente fazem menos restrições sobre os dados e permitem que "os dados falem por si próprios". Foi demonstrado que em muitos domínios o modelo discriminativo é mais efetivo, como classificação de texto e mineração de dados, e existem alguns resultados empíricos mostrando que eles tendem a ter um erro assintótico mais baixo à medida que o tamanho do conjunto de treinamento aumenta (NG, JORDAN, 2002).

Lafferty *et al.* (LAFFERTY, et al., 2001) observaram que o fato de não precisarmos levar em consideração $p(x)$ pode explicar porque os *conditional random fields* tendem a ser mais robustos do que os modelos generativos ao violar as restrições de independência.

3.3 Conditional Random Fields

Os *conditional random fields* (LAFFERTY, et al., 2001) foram introduzidos como uma alternativa de modelo condicional no qual fosse possível relaxar as fortes suposições de independência feitas nos HMMs. Muito usados em processamento de linguagem natural (TASKAR, et al., 2002) (PENG, MCCALLUM, 2004) (SETTLES, 2005) (SHA, PEREIRA, 2003), os CRFs têm sido bastante aplicados em problemas de bioinformática, como por exemplo em alinhamento de seqüências de proteínas (DO, et al., 2006), alinhamento estrutural de proteínas (SATO, SAKAKIBARA, 2005) e classificação de proteínas em *folds* (LIU, et al., 2005).

Definiremos um caso particularmente importante de CRF, denominado *linear-chain* CRF (SUTTON, MCCALLUM, 2006) bastante apropriado para o modelo de seqüências. Para motivar nossa introdução, começaremos considerando a distribuição condicional $P(y|x)$ que segue a distribuição conjunta $p(y, x)$ de um HMM. O ponto chave que essa distribuição condicional é de fato a distribuição de um CRF com uma determinada escolha das *features*.

Primeiro, reescreveremos a equação 3.2 do HMM de uma forma que facilite a

generalização que precisaremos:

$$P(y,x) = \frac{1}{Z}\exp\{\sum_t \sum_{i,j\in S} \lambda_{i,j} 1_{\{y_t=i\}} 1_{\{y_{t-1}=j\}} + \sum_t \sum_{i\in S} \sum_{o\in O} \mu_{o,i} 1_{\{y_t=i\}} 1_{\{x_t=o\}}, \quad (3.15)$$

onde $\theta = \{\lambda_{i,j}, \mu_{o,i}\}$ são os parâmetros da distribuição, podendo ser qualquer número real, e $1_{\{c\}}$ é uma função que retorna o valor 1 caso a condição $\{c\}$ seja satisfeita e 0 caso contrário.

Todo HMM pode ser escrito desta forma, basta para isso definir, por exemplo, $\lambda_{ij} = \log p(y' = i|y = j)$. Uma vez que, não é mais obrigatório que os parâmetros sejam probabilidades, não é garantido que a distribuição some 1, a não ser que usemos explicitamente uma constante de normalização Z. Apesar de esta flexibilização pode-se mostrar que a equação acima descreve exatamente a classe de HMMs representados pela equação 3.2, a parametrização foi flexibilizada, mas não houve qualquer modificação na distribuição.

Depois, podemos reescrever a equação 3.15 de forma mais compacta introduzindo o conceito de *features*. Cada *feature* tem a seguinte forma: $f_k(y_t, y_{t-1}, x_t)$. Precisamos de uma *features* $f_{ij}(y, y', x) = 1_{\{y=i\}} 1_{\{y'=j\}}$ para cada transição (i, j) e uma função $f_{io}(y, y', x) = 1_{\{y=i\}} 1_{\{x=o\}}$ para cada par estado-observação (i, o). Então, podemos escrever o HMM assim:

$$P(y,x) = \frac{1}{Z}\exp\{\sum_{k=1}^{K} \lambda_k f_k(y_t, y_{t-1}, x_t)\}. \quad (3.16)$$

Novamente, a equação 3.16 define exatamente a mesma família de distribuições que a equação 3.15 e portanto, também, a mesma que a equação 3.2. Por fim, podemos escrever a probabilidade condicional $p(y|x)$ que resulta do HMM da equação 3.2.

$$P(y|x) = \frac{P(y|x)}{\sum_{y'} P(y'|x)} = \frac{\exp\{\sum_{k=1}^{K} \lambda_k f_k(y_t, y_{t-1}, x_t)\}}{\sum_{y'} \exp\{\sum_{k=1}^{K} \lambda_k f_k(y'_t, y'_{t-1}, x_t)\}} \quad (3.17)$$

Essa distribuição condicional 3.17 é um *linear-chain* CRF que, em particular, considera apenas o valor da observação corrente. Mas, outros *linear-chain* CRFs usam funções mais ricas, como as que consideram, por exemplo, o valor de observações anteriores e posteriores ou outras propriedades das observações. Essa extensão não muda a nossa notação, mas apenas permite funções $f_k(y_t, y_{t-1}, x_t)$ mais gerais que a função indicador.

Segue-se, então, a definição formal do *linear-chain* CRF. Sejam Y, X vetores de variáveis aleatórias, $\Lambda = \{\lambda_k\} \in \Re^K$ um vetor de parâmetros e $\{f_k(y, y', x_t)\}_{k=1}^K$ um conjunto de *features* que retornam um número real. Um *linear-chain* CRF é uma distribuição $p(y|x)$ que tem a seguinte forma

$$P(y|x) = \frac{1}{Z(x)} \exp\{\sum_{k=1}^{K} \lambda_k f_k(y_t, y_{t-1}, \mathbf{x}_t)\}, \tag{3.18}$$

onde $Z(x)$ é uma constante de normalização

$$Z(x) = \sum_{y} \exp\{\sum_{k=1}^{K} \lambda_k f_k(y_t, y_{t-1}, \mathbf{x}_t)\}. \tag{3.19}$$

O fato de usarmos um vetor de observações $\mathbf{x}_t$ permite que cada *feature* f_k possa depender das observações em qualquer tempo, ou seja, temos acesso a todos os componentes das observações x que precisarmos para computar a função no tempo t. A figura 3.6 mostra a representação gráfica de um *linear-chain* CRF. Finalmente, observe-se que a constante de normalização é uma soma sobre todas as possíveis seqüências, um número bem grande de termos (exponencial). Porém, ela pode ser computada eficientemente através dos algoritmos *forward* e *backward*.

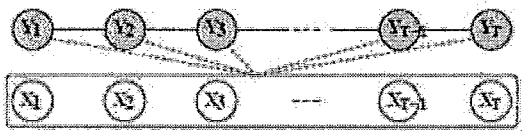


Figura 3.6: Representação gráfica de um *linear-chain* CRF (GUTMANN, KERSTING, 2006).

As duas seções seguintes, 3.3.1 e 3.3.2 abordarão os algoritmos para estimativa de parâmetros e para inferência.

3.3.1 Estimativa de parâmetros

Nesta seção discutiremos a estimativa de parâmetros $\Theta = \{\lambda_1, \lambda_2, ...\}$ de um linear-chain CRF. Estimativa de parâmetros é tipicamente feita através da maximização da verossimilhança. Como estamos modelando a distribuição condicional, é mais apropriada a log-verossimilhança dos dados de treinamento, também chamada

de log-verossimilhança condicional:

$$J(\Theta) = \log \prod_i P(Y_i|X_i). \tag{3.20}$$

Após substituirmos o modelo CRF 3.18 em 3.20, a função objetivo a ser maximizada é

$$J(\Theta) = \sum_i \log \frac{1}{Z(X_i)} \exp[\sum_t \sum_k \lambda_k f_k(y_{i,t}, y_{i,t-1}, X_i)] \tag{3.21}$$

$$J(\Theta) = \sum_i \sum_t \sum_k \lambda_k f_k(y_{i,t}, y_{i,t-1}, X_i) - \log Z(X_i). \tag{3.22}$$

Lafferty *et al.* introduziram um algoritmo iterativo para maximizar $J(\Theta)$ (LAFFERTY, et al., 2001), mas, que se mostrou excessivamente lento. Muitos grupos de pesquisa implementaram métodos de gradiente ascendente, porém implementações muito simples também são muito lentas, por isso, os vários parâmetros λ interagem uns com os outros: aumentando um parâmetro tornará necessário compensar nos outros. O sistema Mallet (MCCALLUM, 2003) emprega o algoritmo BFGS (BERTSEKAS, 1999), que é um método de aproximação de segunda ordem que lida com essas iterações entre os parâmetros.

Como uma alternativa para o BFGS, o gradiente conjugado é uma outra técnica de otimização que também faz um uso aproximado das informações de segunda ordem e tem sido usado com sucesso em CRFs. Quando técnicas como essas são usadas, a otimização baseada em gradientes é muito mais rápida que as abordagens originais apresentadas por Lafferty *et al.*, como é mostrado experimentalmente por muitos autores (SHA, PEREIRA, 2003) (WALLACH, 2002) (MALOUF, 2002) (MINKA, 2003)

As *features* f_k podem ser funções booleanas. Por exemplo, em reconhecimento de estrutura secundária de proteínas, $f_{37}(y_t, X)$ deve ser 1 quando x_t for o aminoácido "G" e y_t for "hélice", caso contrário será 0. É comum definir *features* que dependem apenas de uma janela $w_t(X)$ de X. Essa parametrização linear pode ser vista como uma extensão da regressão logística para o caso seqüencial (DIETTERICH, et al., 2004).

Um inconveniente desta parametrização linear é que ele supõe que cada *feature* contribui independentemente para as funções potenciais. É claro que é possível

definir mais *features* que capturem combinações das *features* básicas, mas isso nos leva a uma explosão combinatória no número de *features*.

O sistema Mallet inicia com uma única *feature* constante e introduz novas conjunções de *features* por meio de conjunções de *features* básicas com *features* já existentes no modelo. Conjunções candidatas são avaliadas de acordo com o seu impacto incremental na função objetivo. Ele demonstrou melhora significante na velocidade e na acurácia da classificação, quando comparado a CRFs que apenas incluem as *features* básicas.

3.3.2 Inferência

Existem dois problemas básicos de inferência para CRFs. O primeiro, computar o gradiente durante o treinamento requer as distribuições marginais de cada $p(y_t, y_{t-1}|x)$ e computar a verossimilhança requer $Z(X)$. Segundo, para classificar uma instância não vista, computamos a classificação mais adequada $y^* = \arg\max_y P(y|x)$. Ambos podem ser calculados eficientemente por variações dos algoritmos para HMM, ou seja, programação dinâmica.

Os algoritmos *forward* e *backward* para os *linear-chain* CRFs são iguais aos usados em HMMs, algoritmos 3.1 e 3.2, exceto pelo fato de os pesos de transição $\Psi_t = (j, i, x_t)$ serem definidos de forma diferente. Podemos, então, reescrever o modelo CRF (3.18):

$$p(y|x) = \frac{1}{Z(x)} \prod_{t=1}^{T} \Psi_t(y_t, y_{t-1}, \mathbf{x}_t), \tag{3.23}$$

onde

$$\Psi_t(y_t, y_{t-1}, x_t) = \exp\{\sum_k \lambda_k f_k(y_t, y_{t-1}, \mathbf{x}_t)\}. \tag{3.24}$$

Com essa definição, a equação recursiva 3.7 do *forward*, 3.9 do *backward* e 3.11 do Viterbi podem ser usadas sem modificações para os *linear-chain* CRFs e, em vez de computar $p(x)$, como no HMM, no CRF computa-se $Z(x)$.

Capítulo 4

Aprendizado relacional

Este capítulo apresenta as ferramentas computacionais aplicadas em problemas de aprendizado relacional usadas nesse trabalho. Na seção 4.1 serão apresentados a lógica de primeira ordem, programação lógica indutiva e dois modelos que se apóiam nos seus conceitos: Tilde e LOHMM. A seção 4.2 apresentará o programa TildeCRF, ferramenta que integra os CRFs com lógica de primeira ordem. Essa é a principal ferramenta usada nesse trabalho para avaliar suas propostas.

4.1 Modelos Lógicos

A lógica de primeira-ordem é um sistema robusto de representação de conhecimento, pois consegue representar de forma elegante situações complexas envolvendo vários objetos e relações entre eles, mas possui a limitação de não conseguir representar incerteza (HALPERN, 1989). Recentemente as abordagens que integram lógica de primeira-ordem com sistemas de raciocínio probabilístico têm recebido grande atenção da comunidade de pesquisa em inteligência artificial, unindo o que as duas abordagens têm de melhor. Exemplos desses sistemas híbridos, aqui chamados de teorias probabilísticas de primeira-ordem, são: *Independent Choice Logic* (ICL) (POOLE, 1993), *Probabilistic Relational Models* (PRM) (KOLLER, 1999) (FRIEDMAN, et al., 1999), *Bayesian Logic Programs* (BLP) (KERSTING, De RAEDT, 2001), *Stochastic Logic Program* (SLP) (MUGGLETON, 2002) e *Constraint Logic Programming* (CLP(BN)) (COSTA, et al., 2003).

As seções 4.1.1 apresentará os conceitos básicos de lógica de primeira ordem e programas lógicos indutivos serão definidos na seção 4.1.2. A seção 4.1.3 apresentará

Top-down Induction of Logical Decision Trees (TILDE) (BLOCKEEL, De RAEDT, 1998). A última seção apresentará *Logical Hidden Markov Model* (LOHMM) (KERSTING, et al., 2006).

4.1.1 Lógica de Primeira Ordem

Nesta seção é apresentado uma breve introdução de conceitos e terminologias referentes à lógica de primeira-ordem, também conhecida como lógica de predicados (CASANOVA, et al., 1987).

Um alfabeto de primeira-ordem Σ é formado por símbolos lógicos e não lógicos. Os símbolos lógicos são os de pontuação "(" e ")", os conectivos $\neg$, $\wedge$, $\vee$, $\leftarrow$ e $\leftrightarrow$, os quantificadores universal $\forall$ e existencial $\exists$ e variáveis. Os símbolos não lógicos com aridade associada são os predicados p/m e símbolos funcionais f/n, $m, n \geq 0$. Uma função com aridade 0, $f/0$, é uma constante e uma variável com aridade 0, $p/0$, é uma variável proposicional. Termos são variáveis, constantes e símbolos funcionais aplicados a quaisquer termos. Um símbolo de predicado seguido por um ou mais termos entre parênteses é chamado de fórmula atômica ou átomo. Um átomo é um literal positivo e a negação de um átomo é um literal negativo. Normalmente variáveis são escritas com letra maiúscula e constantes e símbolos funcionais com letras minúsculas.

Uma cláusula é uma disjunção de literais precedida por um prefixo de quantificadores universais, um para cada variável que aparece na disjunção, na seguinte forma

$$\forall X_1 \forall X_2 ... \forall X_s (L_1 \vee L_2 \vee ... \vee L_m)$$

onde cada L_i é um literal e $X_1, X_2, ..., X_s$ são todas as variáveis ocorrendo em $(L_1 \vee L_2 \vee ... \vee L_m)$. Normalmente as cláusulas são exibidas com o prefixo implícito. Uma cláusula também pode ser representada como um conjunto finito (possivelmente vazio) de literais. O conjunto $\{A_1, A_2, ..., A_h, \neg B_1, \neg B_2, ..., \neg B_b\}$, onde A_i e B_i são átomos, indica a cláusula $A_1 \vee ... A_h \vee \neg B_1 \vee ... \vee \neg B_b$, que é equivalentemente representado por $\{A_1 \vee ... \vee A_h \leftarrow B_1 \wedge ... \wedge B_b\}$. Geralmente essa mesma cláusula é escrita como $A_1, ..., A_h \leftarrow B_1, ..., B_b$, onde $A_1, ..., A_h$ é a cabeça e $B_1, ..., B_b$ é o corpo da cláusula. Uma teoria é um conjunto de cláusulas, representando a con-

junção dessas cláusulas.

Uma *cláusula de Horn* é uma cláusula com no máximo um literal positivo (um literal na cabeça). Se uma cláusula contém exatamente um literal na cabeça, então ela será chamada de cláusula definida. Um conjunto de cláusulas definidas é um programa lógico definido. Um fato é uma cláusula na forma $A \leftarrow$, ou seja, é uma cláusula definida com o corpo vazio, e geralmente denotada por A.

Uma *substituição* $\theta = \{V_1/t1, ..., V_n/t_n\}$ é uma associação de termos t_i para variáveis V_i. Aplicar uma substituição θ para um termo, átomo ou cláusula F produz o termo, átomo ou cláusula instanciado $F\theta$, onde todas as ocorrências de variáveis V_i são simultaneamente substituídas pelo termo t_i, por exemplo, aplicando a substituição $\{X/txt\}$ na cláusula $ls(X) \leftarrow vi(F, X)$ resulta em $ls(txt) \leftarrow vi(F, txt)$. Uma substituição θ é chamada de unificadora para um conjunto S finito de átomos, se e somente se $S_1\theta = ... = S_n\theta$. Um unificador θ para S é chamado de o mais geral (*Most General Unifier* (MGU)) para S se para cada unificador σ de S, existe uma substituição γ tal que $\sigma = \theta\gamma$.

Um termo, átomo ou cláusula é *ground* quando não existe nenhuma variável ocorrendo nele. Um *átomo básico* é um átomo *ground*. Uma cláusula é livre de funções (*function-free*) se ela contém somente variáveis como termo, ou seja, não contém símbolos funcionais. A base de *Herbrand* de um alfabeto Σ, denotada por hb_Σ, é o conjunto de todos os os átomos *ground* construídos com os predicados e símbolos funcionais de Σ. O conjunto $G_\Sigma(A)$ de um átomo A consiste de todos os átomos *ground* $A\theta$ existentes em hb_Σ.

Uma *interpretação* (ou atribuição de valor-verdade) é um processo que mapeia uma sentença em alguma declaração em relação a um determinado domínio. Se a interpretação atribui o valor verdadeiro para uma sentença então ela satisfaz esta sentença e tal interpretação é chamada modelo da sentença. A noção de modelo também é aplicada para uma teoria clausal: uma interpretação é um modelo para um conjunto se ela é um modelo para cada um dos membros do conjunto. Uma sentença é satisfatível, se ela tem pelo menos um modelo, caso contrário ela é insatisfatível. Uma sentença F implica logicamente uma sentença G ($F \models G$), se e somente se todo modelo de F também for modelo de G e, assim, G é uma conseqüência lógica de F.

Um sistema de inferência consiste de um conjunto inicial S de sentenças (axiomas) e um conjunto R de regras de inferência. Usando as regras de inferência, novas sentenças podem ser derivadas a partir de S e/ou a partir de outras sentenças derivadas. Assim, $S \vdash s$ denota que uma sentença s pode ser derivada a partir de S. O conjunto R é dito correto (*sound*) se $\forall$ S e s, se $S \vdash s$, então $S \models s$. De forma semelhante, ele é completo se $\forall$ S e s, se $S \models s$, então $S \vdash s$. Uma prova é uma seqüência $s_1, s_2, ..., s_n$, tal que cada s_i ou está em S, ou é derivável a partir de S e $s_1, ..., s_{i-1}$ usando R. Esta prova também é chamada uma derivação ou dedução.

4.1.2 Programação Lógica Indutiva

Nessa seção definiremos brevemente o que é Programação Lógica Indutiva (ILP - Inductive Logic Programming). Para obter informações mais detalhadas consulte (MUGGLETON, De RAEDT, 1994), (LAVRAC, DZEROSKY, 1994) e (NIENHUYS-CHENG, WOLF, 1997).

A programação em lógica indutiva foi definida como a área de pesquisa formada pela interseção de aprendizado indutivo e programação em lógica (MUGGLETON, De RAEDT, 1994). O aprendizado indutivo (MITCHELL, 1997) defende que a descoberta de uma hipótese que aproxima bem uma determinada função alvo por meio do uso de um conjunto suficientemente grande e representativa de exemplos de treinamento, também aproxima essa função alvo sobre os exemplos não observados. Já ILP tem como objetivo aprender programas lógicos a partir de exemplos e, opcionalmente, de um conhecimento preliminar. Basicamente podemos definir a tarefa tratada em ILP da seguinte maneira (LAVRAC, DZEROSKY, 1994):

- *dados:*
  - *um conjunto E^+ de exemplos positivos (tuplas que pertencem a relação alvo) e um conjunto E^- de exemplos negativos (tuplas que não pertencem a relação alvo);*
  - *uma teoria preliminar B.*
- *encontrar uma teoria de primeira-ordem H' que seja consistente e completa (que explique todos os exemplos positivos e nenhum exemplo negativo).*

Formalmente, dado um conjunto $E = E^+ \bigcup E^-$ e um conhecimento preliminar B, a tarefa é encontrar uma hipótese H' tal que $\forall e \in E^+ : H \wedge B \models e$ (H' é completa) e $\forall e \in E^- : H \wedge B \not\models e$ (H' é consistente). Essa configuração também é chamada de aprendizado a partir de implicação lógica (*learning from entailment*) (MUGGLETON, De RAEDT, 1994). Uma configuração alternativa é proposta no trabalho de De Raedt e Dzerosky (De RAEDT, DZEROSKY, 1994), onde o requerimento de que $B \wedge H \models e$ é substituído pelo requerimento de que a hipótese H seja verdadeira no modelo mínimo *Herbrand* de $B \wedge e$. Tal configuração é chamada de aprendizado a partir de interpretações (*learning from interpretation*).

Existem alguns sistemas de programação em lógica indutiva, como o PROGOL (MUGGLETON, 1995), o Aleph (SRINIVASAN, 2001) e o FOIL (QUINLAN, 1990) que é um dos primeiros sistemas ILP.

4.1.3 Tilde

Iniciaremos esta seção definindo árvores de decisão lógicas (*First Order Logical Decision Tree* (FOLDT)), que nada mais são que uma atualização das já bastante conhecidas árvores de decisão proposicional para lidar com o aprendizado lógico de primeira ordem. Uma árvore de decisão lógica é uma árvore de decisão binária que:

- os nós da árvore contém uma conjunção de literais;
- diferentes nós podem compartilhar variáveis, mas sobre a restrição de que uma variável introduzida em um nó não pode ocorrer na sub-árvore da direita desse nó. Esta restrição segue diretamente da semântica da árvore, uma variável X, introduzida em um nó, é quantificada existencialmente na conjunção daquele nó e a sub-árvore da direita só é relevante quando a conjunção falha, portanto, não teria sentido referenciá-la. Veja o exemplo da figura 4.1.

Da mesma forma que as regras proposicionais podem ser derivadas a partir de árvores de decisão (cada regra corresponde a um caminho que inicia na raiz e termina em alguma folha e os testes feitos nos nós deste caminho são condições desta regra), cláusulas podem ser derivadas a partir das árvores de decisão lógica (cada teste no caminho a partir da raiz até folha são agora um literal ou uma conjunção

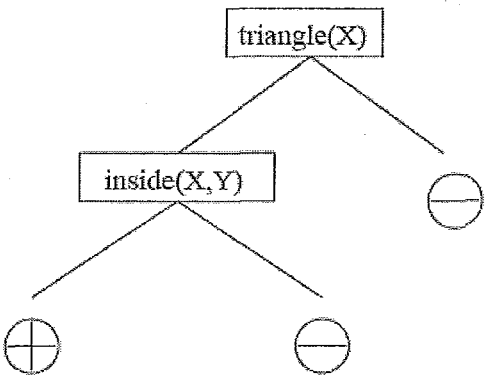


Figura 4.1: Exemplo de árvore de decisão lógica.

de literais que fazem parte da cláusula). As árvores resultantes podem ser utilizadas diretamente para a classificação do exemplos novos, mas eles também podem ser facilmente transformadas em uma programas lógicos. O algoritmo 4.1 mostra como usar as árvores de decisão lógica para classificação.

Algoritmo 4.1 Procedimento de classificação usando FOLDT proposto em (BLOCKEEL, De RAEDT, 1998).

entrada: Exemplos e

$Q = true$
$N = root$
enquanto $N \neq$ FOLHA(c) **faça**
  $N =$ NO_INTERNO($conj$, esq, dir)
  se $Q \wedge conj$ for verdadeira em $e \wedge B$ (B é o conhecimento preliminar) **então**
    $Q = Q \wedge conj$
    $N = esq$
  senão
    $N = dir$
retorna c (classe da folha)

Árvores de decisão lógica podem ser induzidas de uma forma muito similar as árvores de decisão proposicionais. O algoritmo genérico que se utiliza usualmente é referenciado como *Top-down Induction of Decision Trees* (TDIDT). Alguns exemplos de sistemas que usam esse algoritmo são o C4.5 (QUINLAN, 1993) e CART (BREIMAN, et al., 1984). O Tilde (BLOCKEEL, De RAEDT, 1998) é uma implementação de um algoritmo baseado no C4.5 usado para induzir árvores de decisão lógica. Veja o algoritmo 4.2.

Algoritmo 4.2 Procedimento de indução das árvores lógicas de primeira ordem usando pelo Tilde proposto em (BLOCKEEL, De RAEDT, 1998).

nome: CONSTROI_ARVORE
entrada: Árvore de decisão lógica T, exemplos E e consulta Q

$\leftarrow Q_b$:= elemento de $\rho(\leftarrow Q)$ com maior valor de ganho
se $\leftarrow Q_b$ não for bom (por exemplo, tiver ganho nulo) **então**
  T =FOLHA(CLASSE_MAJORITARIA(E)
senão
  $conj = Q_b - Q$
  $E_1 = \{e \in E | \leftarrow Q_b$ for verdadeira em $e \wedge B\}$
  $E_2 = \{e \in E | \leftarrow Q_b$ for falsa em $e \wedge B\}$
  esq = CONSTROI_ARVORE(E_1, Q_b)
  dir = CONSTROI_ARVORE(E_2, Q)
  T = NO_INTERNO($conj$, esq, dir)
retorna T

A principal diferença do Tilde em relação ao C4.5 está na computação dos testes a serem colocados em um nó, enquanto no C4.5 os testes são comparações entre um atributo e um valor, no Tilde é empregado o clássico operador de refinamento *θ-subsumption* (MUGGLETON, De RAEDT, 1994). Este operador especifica que literais ou conjunções de literais podem ser adicionados à consulta associada do nó. Quando um nó está para ser dividido, o conjunto de todos os refinamentos são computados e avaliados. Inicia-se com uma árvore vazia e busca-se repetidas vezes pelo melhor teste para um nó de acordo com algum critério de divisão. Então, os D exemplos do nó são divididos em $D1$ (sucesso) e $D2$ (falha) de acordo com o teste. Para cada divisão, o procedimento é recursivamente aplicado, obtendo as sub-árvores para as respectivas divisões. Podem ser usados vários critério de parada, por exemplo o número de exemplos mínimos para o nó.

Existe uma versão denominada Tilde-RT que substitui as árvores de decisão lógicas por árvores de regressão relacionais. A diferença entre as duas árvores é que enquanto a primeira tem na folhas uma classe a segunda tem um valor numérico de regressão. Neste caso, geralmente, o critério de divisão é a variância de $D1$ e $D2$ (a variância de uma variável aleatória é uma medida da sua dispersão estatística, indicando quão longe em geral os seus valores se encontram do valor esperado, se $\mu = E(X)$ é o valor esperado, então a variância é dada por $var(X) = E[(X - \mu)^2])$ e

o procedimento pára se a variância em um nó for pequena o suficiente ou um limite de profundidade for atingido.

4.1.4 HMM e sentenças lógicas: LOHMM

Apesar do seu grande sucesso, os HMMs não lidam com dados estruturados. E muitas aplicações tradicionais em aprendizado de máquina requerem exemplos estruturados, tais como os estudados no campos de programação em lógica indutiva (MUGGLETON, De RAEDT, 1994) e aprendizado relacional (DZEROSKI, 2003).

Kersting *et al.* (KERSTING, et al., 2006) propuseram um sistema de ILP chamado de *logical hidden Markov models*, ou LOHMM, que estende HMM para trabalhar com dados estruturados na forma de átomos lógicos. A grande motivação para essa abordagem deve-se aos fatos que variáveis permitem a abstração de símbolos específicos e que a unificação compartilhamento de informações entre os estados.

O HMM é essencialmente proposicional e usa símbolos fixos para representar os estados e as saídas. A idéia chave do LOHMM é trocar os símbolos fixos por símbolos abstratos, que por definição são átomos lógicos. A abstração de um símbolo A consiste no fato de que ele representa um conjunto de todos os átomos *ground*, isto é, todos os átomos sem variáveis de A sobre o alfabeto de símbolos.

Antes de definirmos formalmente LOHMM, é preciso apresentar outro conceito definido por Kersting *et al.* (KERSTING, et al., 2006): transições abstratas. Essas transições são expressões que têm a forma $p : H \leftarrow^{\square} B$, onde $p \in [0, 1]$ e H, B e $\square$ são átomos. Todas as variáveis são implicitamente assumidas como universalmente quantificadas, ou seja, elas têm escopo restrito a uma transição abstrata.

Os átomos H e B representam estados abstratos e $\square$ representa um símbolo abstrato de saída. Semanticamente, a transição abstrata significa que, se o estado atual é um átomo *ground* de B, $B\theta_B$, pode-se seguir com probabilidade p para um dos estados representados pelos átomos *ground* resultantes da unificação entre H e θ_B, $H\theta_B\theta_H$, enquanto são emitidos um átomo *ground* resultante da unificação entre $\square$ e $\theta_B\theta_H$, $\square\theta_B\theta_H\theta_{\square}$.

Estas transições não são determinísticas porque podem existir mais de uma possibilidade estado e emissão resultantes. A determinação dos estados reais e dos

símbolos emitidos é feita através de uma distribuição de probabilidade μ. A probabilidade em uma transição entre estados é o produto da probabilidade de seguir uma transição abstrata (p), a probabilidade de selecionarmos um estado real (μ_1) e a probabilidade de emitirmos determinado símbolo (μ_2).

Estados iniciais devem ser usados com probabilidades definidas e estados finais podem ser usados para os casos de seqüências de tamanhos variados.

Então, formalmente, um LOHMM é formado por um alfabeto lógico Σ, uma distribuição de probabilidade μ de seleção neste alfabeto, um conjunto de transições abstratas Δ e um conjunto de transições abstratas para os estados iniciais definidas a priori Υ. Seja $\mathbf{B}$ o conjunto de todos os átomos presentes no corpo das transições em Δ. É assumido que $\mathbf{B}$ é fechado sobre glb (*greatest lower bound*) e requerido que

$$\forall B \in \mathbf{B} : \sum_{p:H \leftarrow^{\square} B \in \Delta} p = 1 \tag{4.1}$$

e que a soma das probabilidades p das cláusulas em Υ também seja 1.

A figura 4.2 apresenta um exemplo de LOHMM para estrutura secundária de proteína, representando hélices e *strands*. Os vértices são os estados abstratos e podemos encontrar três tipos de transições: as linhas sólidas especificam as transições abstratas, as linhas pontilhadas mostram que dois estados apresentam o mesmo comportamento sendo vistas como uma forma de obter recursão e as linhas tracejadas representam um tipo de comportamento *default* para o tratamento de exceções.

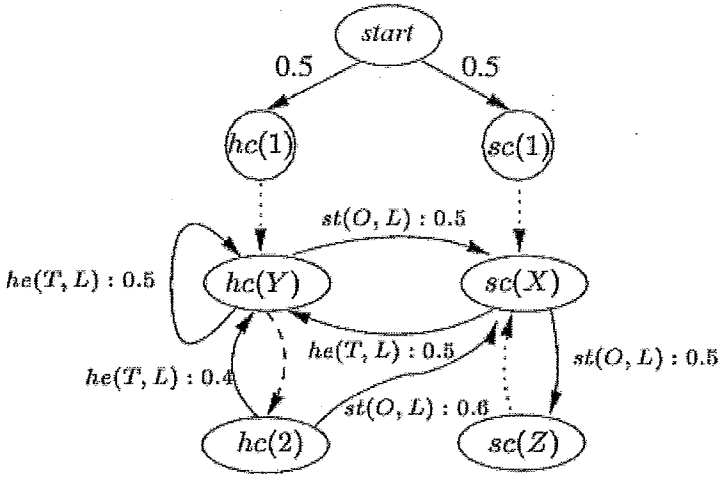


Figura 4.2: Representação gráfica de um LOHMM (KERSTING, et al., 2003).

Assim como para HMMs, três problemas de inferência são de interesse: a probabilidade $P(X|M)$, a seqüência de estados mais prováveis e a estimativa dos parâmetros do modelo. Adaptações do algoritmos de programação dinâmica *forward* e *backward*.

4.2 TildeCRF

TildeCRF (GUTMANN, KERSTING, 2006) é o primeiro método para treinar conditional random fields com seqüências sobre um alfabeto de átomos lógicos. A idéia chave é usar o algoritmo *gradient tree boosting* proposto por Dietterich *et al.* (DIETTERICH, et al., 2004) para treinar os CRFs, substituindo a representação das funções potenciais, em vez de árvores de regressão, árvores de regressão relacionais.

Na seção 4.2.1 apresentaremos o método de treinamento de CRFs por meio do método de *gradient tree boosting* e na seção 4.2.2 descreveremos as modificações necessárias para empregá-lo no aprendizado relacional.

4.2.1 Treinando CRF com Gradient Tree Boosting: TreeCRF

Nesta seção discutimos uma forma de treinamento das funções potenciais, baseada no algoritmo *gradient tree boosting* (FRIEDMAN, 2001). Especificamente, as funções potenciais definidas no CRF são representadas por somas ponderadas de árvores de regressão, que são aprendidas de forma similar ao Adaboost (FREUND, SCHAPIRE, 2003), mas com a função objetivo sendo a maximização da verossimilhança condicional $P(Y|X)$. O algoritmo tem sucesso na redução do imenso espaço de possíveis *features* por meio do treinamento de árvores de regressão de forma que apenas as combinações de *features* definidas pelas árvores são consideradas. Como resultado, o *gradient tree boosting* é linear na ordem do modelo de Markov e nas interações das *features*, enquanto algoritmos anteriores, iterativos e *gradient ascent*, são exponenciais. Uma vantagem da abordagem com o gradiente boosting é que o algoritmo tem a tendência de evitar *overfitting* devido ao efeito "*ensemble*" proveniente da combinação de múltiplas árvores de regressão.

Suponha que queremos resolver um típico problema de aprendizagem supervisionado, onde os exemplos de treinamento têm a forma $(x_i, y_i) = 1, ..., N$ e $y_i \in$

$\{1, ..., K\}$. Nós desejamos treinar o seguinte modelo:

$$P(y|x) = \frac{\exp \Psi(y, x)}{\sum_{y'} \Psi(y', x)}.$$

Descreveremos *gradient tree boosting* da mesma forma que Dietterich *et al.* (DIETTERICH, et al., 2004). Ele é baseado na idéia de gradiente funcional ascendente que parametriza Ψ de alguma forma, por exemplo, como uma função linear,

$$\Psi(y, x) = \sum_a \beta_a g_a(y, x).$$

Em aprendizagem de parâmetros por gradiente ascendente, o vetor de parâmetros após a iteração m, Θ_m, é uma soma do vetor inicial Θ_0 e uma série de passos de gradiente ascendente δ_m:

$$\Theta_m = \Theta_0 + \delta_1 + ... + \delta_m$$

onde cada δ_m é computado como um passo na direção do gradiente da função de log-verossimilhança:

$$\delta_m = \eta_m \frac{\partial}{\partial \Theta_{m-1}} \sum_i \log P(y_i|x_i; \Theta_{m-1})$$

e η_m é uma constante que controla o tamanho do passo do gradiente.

Gradiente ascendente funcional é uma técnica mais genérica (DIETTERICH, et al., 2004). Em vez de assumirmos uma parametrização linear para Ψ, apenas definimos que Ψ será representada por uma soma ponderada de funções:

$$\Psi_m = \Psi_0 + \Delta_1 + ... + \Delta_m.$$

Cada Δ_m é computado como um gradiente funcional:

$$\Delta_m = \eta_m E_{x,y}[\frac{\partial}{\partial \Psi_{m-1}} \log P(y|x; \Psi_{m-1})].$$

O gradiente funcional indica como gostaríamos que a função Ψ_{m-1} fosse modificada a fim de aumentarmos a verdadeira log-verossimilhança, isto é, em todos os possíveis pontos (x, y). Infelizmente não conhecemos a distribuição conjunta P(x, y), então não podemos calcular expectativa $E_{x,y}$. Mas, temos um conjunto exemplos de treinamento amostrado a partir desta distribuição conjunta, então podemos computar o valor do gradiente funcional em cada um dos pontos deste conjunto:

$$\Delta_m(y_i, x_i) = \frac{\partial}{\partial \Psi_{m-1}} \sum_i \log P(y_i|x_i; \Psi_{m-1}).$$

Podemos utilizar os gradientes nestes pontos para definir um outro conjunto de exemplos $((x_i, y_i), \Delta_m(y_i, x_i))$ e usar esses exemplos para treinar uma função $h_m(y, x)$ que aproxima $\Delta_m(y_i, x_i)$. Especificamente, tentaremos encontrar uma árvore de regressão h_m que minimize

$$\sum_i [h_m(y_i, x_i) - \Delta_m(y_i, x_i)]^2.$$

Podemos então dar um passo em direção a esta função:

$$\Psi_m = \Psi_{m-1} + \eta h_m.$$

Embora a função h_m apenas aproxime a função desejada Δ_m, ela apontará na mesma direção, assumindo a existência de exemplos de treinamento suficientes. Sendo ascendente na direção de h_m será uma aproximação do verdadeiro gradiente ascendente funcional.

É importante perceber que essa abordagem troca o difícil problema de maximizar o log-verossimilhança dos dados pelo problema de minimizar o erro quadrático do conjunto de exemplos de treinamento, que é, relativamente, mais simples. Friedman sugere evoluir h_m como no algoritmo *best-first* do CART (BREIMAN, et al., 1984) e parar quando a árvore de regressão atingir um número pré-definido de folhas L, através do qual pode-se controlar o *overfitting*.

É muito direto aplicar gradiente ascendente funcional ao problema de aprendizagem supervisionado. Basta representar $\Psi(y_t, X)$ e $\Psi(y_{t-1}, y_t, X)$ como somas ponderadas de árvores de regressão. Seja

$$F^{y_t}(y_{t-1}, X) = \Psi(y_t, X) + \Psi(y_{t-1}, y_t, X)$$

a função que computa o quanto é desejável o *label* y_t dados os valores do *label* y_{t-1} e das *features* de entrada X. Para cada classe de saída existem uma função F^k. Então, o CRF assume a seguinte forma:

$$P(Y|X) = \frac{1}{Z(X)} \exp \sum_t F^{y_t}(y_{t-1}, X).$$

Agora pode-se computar o gradiente funcional de $\log P(Y|X)$ com respeito a $F^{y_t}(y_{t-1}, X)$. Para simplificar a computação, nós substituímos X por $w_t(X)$, que é

uma janela dentro da seqüência X, com centro em x_t. Por hora, assumimos, sem perda de generalidade, que cada janela é única, então houve apenas uma ocorrência de $w_t(X)$ em cada seqüência X.

Dietterich *et al.* (DIETTERICH, et al., 2004) propõem a seguinte proposição: o gradiente funcional de $\log P(Y|X)$ com respeito a $F^v(u, w_d(X))$ é

$$\frac{\partial \log P(Y|X)}{\partial F^v(u, w_d(X))} = I(y_{d-1} = u, y_d = v) - P(y_{d-1} = u, y_d = v|w_d(X)) \tag{4.2}$$

onde $I(y_{d-1} = u, y_d = v)$ é 1 se a transição $u \rightarrow v$ é observada a partir da posição $d-1$ até a posição d na seqüência Y e 0 caso contrário, e onde $P(y_{d-1} = u, y_d = v|w_d(X))$ é a probabilidade desta transição de acordo com as funções potenciais correntes.

Sigamos os passos da prova desta proposição:

$$\frac{\partial \log P(Y|X)}{\partial F^v(u, w_d(X))} = \frac{\partial}{\partial F^v(u, w_d(X))} \sum_t F^{y_t}(y_{t-1}, w_t(X)) - \log Z(X)$$

Temos que exatamente um dos termos de $F^{y_t}(y_{t-1}, w_t(X))$ irá satisfazer $F^v(u, w_d(X))$, porque $w_d(X)$ é única. Esse termo terá derivada igual a 1, então podemos representá-lo por uma *função indicadora* $I(y_{d-1} = u, y_d = v)$ (uma *função indicadora* sobre um conjunto A determina se um elemento é parte de um subconjunto B de A). Então teremos no lado direito da equação

$$I(y_{d-1} = u, y_d = v) - \frac{\partial \log Z(X)}{\partial F^v(u, w_d(X))}$$

derivando-se $\log Z(X)$ temos

$$I(y_{d-1} = u, y_d = v) - \frac{1}{Z(X)} \frac{\partial Z(X)}{\partial F^v(u, w_d(X))} \tag{4.3}$$

agora aplicaremos o algoritmo forward-backward para computar Z(X). Assumiremos que y_t toma o valor $\perp$ para $t < 1$. A recursão do forward será

$$\alpha(k, 1) = \exp F^k(\perp, w_1(X))$$
$$\alpha(k, t) = \sum_{k'} [\exp F^k(k', w_t(X))] \cdot \alpha(k', t-1)$$

e a recursão do backward será

$$\beta(k, T) = 1$$

$\beta(k,t) = \sum_{k'}[\exp F^{k'}(k, w_{t+1}(X))] \cdot \beta(k', t+1)$

A iteração das variáveis k e k' consideram todas as possíveis classes. Z(X) será computada em cada posição t por

$$Z(X) = \sum_k \alpha(k,t)\beta(k,t)$$

Se voltarmos um passo na recursão de α, podemos reescrever assim

$$Z(X) = \sum_k [\sum_{k'} [\exp F^k(k', w_t(X)] \cdot \alpha(k', t-1)]\beta(k,t)$$

substituindo Z(X) na equação 4.3 e derivando, novamente porque $w_d(X)$ é única, somente o produto onde $k' = u$ and $k = v$ tem derivada diferente de zero. Obteremos então:

$$I(y_{d-1} = u, y_d = v) - \frac{1}{Z(X)}(\exp F^v(u, w_d(X)))\alpha(u, d-1)\beta(v, d).$$

A expressão mais à direita da equação acima representa precisamente a probabilidade conjunta de $y_d = u$ e $y_d = v$ dado X

$$I(y_{d-1} = u, y_d = v) - P(y_{d-1} = u, y_d = v|w_d(X))$$

C.Q.D.

Se a janela $w_d(X)$ ocorrer mais de uma vez em X, cada ocorrência contribuirá separadamente para o gradiente funcional. Esse gradiente funcional tem uma interpretação satisfatória: é o nosso erro na escala de probabilidade. Se a transição $u \longrightarrow v$ é observada nos exemplos de treinamento, então a probabilidade predita $P(u, v|X)$ deveria ser 1 para maximizar a likelihood. Se a transição não é observada, então a probabilidade deveria ser zero. O gradiente ascendente funcional simplesmente adequa as árvores de regressão a estes residuais.

O algoritmo 4.3 apresenta o pseudo-código do algoritmo *tree-boosting*. A função potencial para cada classe k é inicializada com zero. Então, M iterações de boosting são executadas. Em cada uma, para cada classe k, um conjunto S(k) de exemplos de treinamento do gradiente ascendente são gerados. A árvore de regressão com no máximo L folhas é treinada com estes exemplos de treinamento para gerar a função $h_m(k)$. Esta função é adicionada à função potencial anterior para produzir a função

Algoritmo 4.3 Gradiente tree boosting proposto em (DIETTERICH, et al., 2004).

entrada: Dados $= \{(X_i, Y_i) : i = 1, ...N\}$ e o número máximo de folhas L

para cada classe k **faça**
  inicializa $F_0^k = 0$
para $m = 1, ..., M$ **faça**
  para $k = 1, ..., K$ **faça**
    $S(k) := GERA\_EXEMPLOS(k, Dados, Pot_{m-1})$ ▷ *Algoritmo 4.4*
      ▷ *onde* $Pot_{m-1} = \{F_{m-1}^u : u = 1, ..., K\}$
    $h_m(k) := TREINA\_ARVORE(S(k), L)$
    $F_m^k := F_{m-1}^k + h_m(k)$
retorna F_m^k para todo k

potencial da próxima iteração. Em outras palavras, estamos assumindo o tamanho do passo $\eta_m = 1$. Dietterich *et al.* (DIETTERICH, et al., 2004) experimentaram uma busca linear para otimizar η_m, porém o custo constatado foi muito alto. A proposta foi permitir que a propriedade de auto-correção do *boosting* das árvores corrigisse qualquer erro para cima ou para baixo na próxima iteração.

Algoritmo 4.4 Geração de exemplos proposto em (DIETTERICH, et al., 2004).

entrada: uma classe k, Dados $= \{(X_i, Y_i) : i = 1, ...N\}$ e $Pot_m = \{F_m^u : u = 1, ..., K\}$

S := {}
para $i = 1, ..., N$ **faça**
  FORWARD_BACKWARD(X_i, Y_i, Pot_m)
    ▷ *para obter* $Z(X_i)$ *e* $\alpha(k,t)$ *e* $\beta(k,t)$ *para todo* k *e* t
  para $t = 1, ..., T_i$ **faça**
    para $k' = 1, ..., K$ **faça**
      $P(y_{i,t-1} = k', y_{i,t} = k|X_i) := \frac{\alpha(k',t-1)\exp[F_m^k(k-1, w_t(X_i))]\beta(k,t)}{Z(X_i)}$
      $\Delta(k, k', i, t) := I(y_{i,t-1} = k', y_{i,t} = k) - P(y_{i,t-1} = k', y_{i,t} = k|X_i)$
      $S := S \bigcup ((w_t(X_i), k'), \Delta(k, k', i, t))$
retorna S

O conjunto de treinamento da árvore é gerado como mostrado no algoritmo 4.4. Para cada uma das seqüências dos dados de entrada é montada uma matriz *forward-backward* para se obter as probabilidades α e β e também $Z(X)$. Então, para cada posição da seqüência e para cada classe do problema em questão é gerado um exemplo de regressão. Cada exemplo consiste de uma janela $w_t(X_i)$ da seqüência de entrada, uma possível classe k' para o exemplo e um valor target Δ, calculado

através da equação 4.2.

Uma vez que o modelo CRF tenha sido treinado, existirão pelo menos duas possibilidades para definir um classificador $Y = H(X)$ para inferência. Primeiro, podemos predizer a seqüência inteira Y que tem a maior probabilidade:

$$H(X) = \arg\max_{Y} P(Y|X)$$

Isso faz sentido em aplicações em que o objetivo é predizer seqüencialmente com coerência, por exemplo, em *part-of-speech tagging*. Neste caso podemos aplicar o algoritmo Viterbi (RABINER, 1989), que tem a vantagem de não ser necessário computar Z(X).

A segunda opção é predizer individualmente cada y_t de acordo com

$$H_t(X) = \arg\max_{v} P(y_t = v|X)$$

e concatená-las para a obter H(X). Isso faz sentido em aplicações em que o objetivo é maximizar o número de indivíduos corretamente preditos, mesmo que o resultado para a seqüência inteira seja incoerente. Por exemplo, partes de um texto podem não estar gramaticalmente corretos e mesmo assim ter o número de palavras classificadas maximizado. $P(y_t|X)$ pode ser computada pelo algoritmo *forward-backward* assim

$$P(y_t|X) = \frac{\alpha(y_t, t)\beta(y_t, t)}{Z(X)}.$$

4.2.2 CRF e sentenças lógicas

O TildeCRF se baseia na idéia dos MLNs (RICHARDSON, DOMINGOS, 2006) de representar os potenciais como conjuntos de fórmulas lógicas com um número real, ou peso, associado a cada fórmula. A abstração relacional nos potenciais oferece grande compactação e possibilita predições úteis em grandes espaços de estados, mesmo se muitos desses estados sequer sejam observados nos dados de treinamento. Mas, a compactação tem um custo alto, tornando muito mais complexo o problema da estimativa de parâmetros, pois trata-se de representações funcionais não paramétricas. Portanto, técnicas de otimização baseadas em gradientes que assume uma representação paramétrica não podem ser aplicadas. Então, o TildeCRF segue a técnica de *gradient tree boosting* apresentada na seção 4.2.1.

As árvores de regressão relacionais incrementam a representação de valor de atributo usada nas clássicas árvores de regressão: cada teste é uma conjunção de átomos relacionais. Uma variável introduzida em algum nó não pode aparecer na sub-árvore do lado direito, isto é, variáveis são delimitadas ao longo do caminho à esquerda.

Para induzir uma árvore de regressão relacional, o TildeCRF essencialmente emprega o algoritmo implementado pelo Tilde (BLOCKEEL, De RAEDT, 1998), o que também explica o nome dessa abordagem. O Tilde, como explicado em 4.1.3, aprende árvores relacionais através do aprendizado por interpretação, onde os exemplos são conjuntos de átomos *ground*. Ele basicamente segue o algoritmo bem estabelecido c4.5 (QUINLAN, 1993), diferenciando-se apenas em um ponto: na computação dos testes a serem colocados em um nó. Ele emprega o clássico operador de refinamento θ-*subsumption* nesta computação. Como critério de divisão é usado a variância dos exemplos que obtém sucesso e os que falham. E o critério de parada é a variância em um nó ou o limite de profundidade. Nas folhas, o valor de regressão médio é predito.

Algoritmo 4.5 Geração de exemplos proposto em (GUTMANN, KERSTING, 2006).

entrada: uma classe k, Dados $= \{(X_i, Y_i) : i = 1, ...N\}$ e $Pot_m = \{F_m^u : u = 1, ..., K\}$

S := {}
para $i = 1, ..., N$ **faça**
  FORWARD_BACKWARD(X_i, Y_i, Pot_m)
    ▷ *para obter* $Z(X_i)$ *e* $(\alpha(k,t), \beta(k,t))$ *para todo* k *e* t
  para $t = 1, ..., T_i$ **faça**
    para $k' = 1, ..., K$ **faça**
      ▷ ***a modificação em relação ao alg. 4.4:*** $\subseteq_\theta$
      $P(y_{i,t-1} \subseteq_\theta k', y_{i,t} \subseteq_\theta k | X_i) := \frac{\alpha(k',t-1) \exp[F_m^k(k-1, w_t(X_i))] \beta(k,t)}{Z(X_i)}$
      $\Delta(k, k', i, t) := I(y_{t-1} \subseteq_\theta k', y_t \subseteq_\theta k) - P(y_{t-1} \subseteq_\theta k', y_d \subseteq_\theta k | X_i)$
      $S := S \bigcup ((w_t(X_i), k'), \Delta(k, k', i, t))$
retorna S

Gutmann e Kersting (GUTMANN, KERSTING, 2006) adaptaram o treeCRF por meio do uso de gradiente funcional relacional. Primeiro, substitui a função de treinamento de árvores de regressão por uma que emprega o Tilde. Depois, faz uma

ligeira modificação na proposição 4.2 de forma a considerar sentenças lógicas na geração de exemplos relacionais usados para o treinamento das árvores de regressão relacionais:

$$\frac{\partial \log P(Y|X)}{\partial F^v(u, w_d(X))} = I(y_{d-1} \subseteq_\theta u, y_d \subseteq_\theta v) - P(y_{d-1} \subseteq_\theta u, y_d \subseteq_\theta v | w_d(X)) \quad (4.4)$$

onde I é a função identidade, $\subseteq_\theta$ denota que u *θ-subsumes* y e $P(y_{d-1} \subseteq_\theta u, y_d = v|w_d(X))$ é a probabilidade de que as classes u e v sejam as classes nas posições d e $d-1$. Então, o algoritmo 4.4 é modificado, como mostrado no algoritmo 4.5.

Um dos experimentos realizados por Gutmann e Kersting foi aplicar o Tilde-CRF no problema de classificação de proteínas em *folds*. Os resultados obtidos foram superiores aos obtidos com LOHMMs, veja seção 4.1.4, e *Fisher kernels* para seqüências lógicas (KERSTING, GÄRTNER, 2004).

Capítulo 5

Aplicando TildeCRF

O trabalho de Bernardes (BERNARDES, 2005) desenvolveu um novo método para treinar pHMMs considerando alinhamentos tridimensionais e diferentes propriedades estruturais sobre um conjunto de proteínas homólogas. Os experimentos constataram que a sensibilidade dos modelos gerados aumenta significativamente ao adicionarmos informações estruturais no momento do treinamento. A partir daí, definimos nossos experimentos.

Como no trabalho de Bernardes, o presente trabalho foi feito no nível de super-famílias da base de dados *Structural Classification of Proteins* (SCOP) (ANDREEVA, et al., 2004) que agrupam famílias que compartilham uma mesma origem evolutiva, mas esta não é tão evidente a partir da simples comparação das seqüências e, provavelmente, foi obtida a partir de análises de estrutura e funções (veja a seção 2.2.2). Esse nível parece ser o que melhor representa homologia distante.

Gutmann e Kersting propuseram uma extensão de CRF, TildeCRF (GUTMANN, KERSTING, 2006), um *framework* de *Statistical Relational Learning* (SLR), na qual as seqüências são formadas por átomos lógicos, incorporando toda a expressividade da lógica de primeira ordem às vantagens dos modelos discriminativos. O objetivo deste trabalho foi investigar a aplicação do TildeCRF no problema de detecção de homologias distantes. Duas questões foram levantadas nesse sentido:

i) o TildeCRF é competitivo com as tradicionais ferramentas desenvolvidas para para esta tarefa, tal como *profile Hidden Markov Model* (pHMM)?

ii) o TildeCRF pode obter resultados significativamente melhores ao fazer uso de informações estruturais das proteínas?

Na seção 5.1 descreveremos todos os passos da metodologia experimental adotada para tentar responder as duas questões propostas. Na seção seguinte, 5.2, serão apresentados e discutidos os resultados obtidos nos experimentos propostos na metodologia adotada.

5.1 Metodologia experimental

O objetivo desta seção é descrever a metodologia adotada para a aplicação do TildeCRF no problema de detecção de homologias distantes. Iniciaremos pela descrição dos arquivos usados pelo TildeCRF. Depois definiremos os experimentos propostos. Então, detalharemos os dados e como eles foram representados em lógica de primeira ordem.

O TildeCRF pode ser usado para:

- classificação, atribuir uma classe a uma seqüência inteira;
- *tagging*, atribuir uma classe a um átomo de uma seqüência.

Em qualquer caso, o TildeCRF requer pelo menos dois arquivos de entrada, um com as seqüências usadas como exemplos de treinamento e teste, e outro com o esquema que especifica a linguagem usada para formar estes exemplos. A implementação do TildeCRF que utilizamos usa uma linguagem baseada em XML para especificar os arquivos de entrada, onde *tags* marcam e agrupam as informações. As seqüências de treinamento e de teste também podem ficar em arquivos separados.

Nosso problema é de classificação e, neste caso, o arquivo de seqüências tem três *tags* XML: <**sequences**>, <**sequence**> e <**atom**>. A primeira agrupa todas as seqüências, a segunda agrupa todos os itens de uma única seqüência e a última contém cada item desta seqüência. Um item contém uma fórmula atômica *ground* (veja a seção 4.1.1). A figura 5.1 mostra um exemplo de uma seqüência de resíduos seguindo esta formatação. O arquivo de esquema define os possíveis predicados, *tag* <**schematerm**>, e os possíveis valores assumidos pelas variáveis existentes, tags <**substitutions**> e <**entry**>. Um termo para um aminoácido pode ser assim: <**schematerm**>a(X)</**schematerm**> <**substitutions variable="X"**> <**entry**>G</**entry**> <**entry**>A</**entry**>...</**substitutions**>.

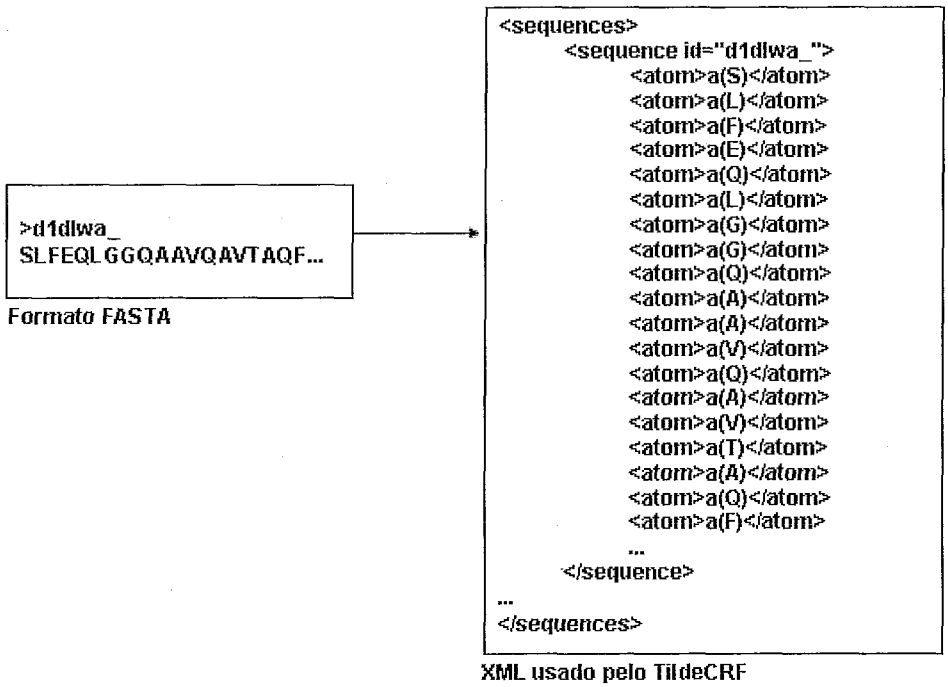


Figura 5.1: Uma seqüência de resíduos que foi usada nos experimentos deste trabalho. O predicado é "a" e a letra entre parênteses é uma constante que indica o aminoácido naquela posição.

Com o objetivo de avaliarmos a questão **ii**, definida no início deste capítulo, desenvolvemos três experimentos diferentes sobre um mesmo conjunto de super-famílias, porém, com níveis diferentes de informação usada no treinamento do modelo em cada um.

1. Foram considerados apenas os aminoácidos que compõe as proteínas, similar ao que acontece no caso proposicional.

2. As seqüências de treinamento foram primeiro alinhadas com as seqüências da mesma super-família de forma que as informações das áreas conservadas fossem melhor consideradas pelo modelo. Neste caso, o modelo considerou os tipos de colunas existentes nos alinhamentos: *matches*, *inserts* e *gaps*.

3. Adicionamos informações extraídas da estrutura secundária da proteína nas seqüências de treinamento. Especificamente, foi introduzido o tipo de formação estrutural no qual cada aminoácido faz parte (alfa-hélice, folhas-beta, *coil*, ângulo Φ).

Ao final, os resultados obtidos no segundo e terceiro experimentos são comparados aos resultados do HMMER (EDDY, 1998) e HMMER-STRUCT (BERNARDES, 2005), respectivamente, a fim de avaliarmos a questão **i**.

O primeiro experimento é implementado convertendo as seqüências de aminoácidos do formato FASTA (PEARSON, 1990) para o formato XML estabelecido pelo TildeCRF. Desenvolvemos *scripts* de conversão na linguagem nativa para *scripts* do sistema operacional Windows: *VBscript*.

O segundo experimento requer que as seqüências de treinamento estejam alinhadas. O alinhamento múltiplo de seqüências expressa o grau de conservação e os relacionamentos evolutivos entre as seqüências envolvidas. A qualidade dos alinhamentos está diretamente relacionada ao desempenho de pHMMs (EDDY, 1998) e, portanto, o mesmo vale para modelos construídos pelo TildeCRF. Para a realização dos experimentos deste trabalho foi adotada uma ferramenta de alinhamento primário muito estável e bastante utilizada: CLUSTAL W (THOMPSON, et al., 1994).

No terceiro experimento, além dos alinhamentos, informações estruturais das proteínas foram introduzidas. O programa SSTRUC, que faz parte do pacote JOY (MIZUGUCHI, et al., 1998), foi usado para extrair os elementos de estruturas secundárias do banco de dados PDB (BERMAN, et al., 2000). A figura 5.2 mostra o esquema de todos os experimentos.

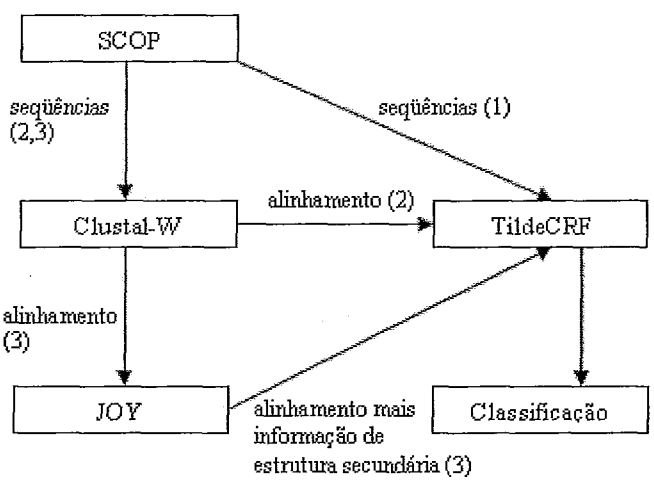


Figura 5.2: Esquema completo dos experimentos (os números 1, 2 e 3 identificam os experimentos).

Os procedimentos dos testes foram inspirados nos adotados pelo trabalho (BERNARDES, 2005). A base de dados SCOP foi particionada por nível de super-família. Foram selecionadas seis super-famílias: três da classe alfa e três da classe beta; seguindo os requisitos de terem mais de vinte seqüências e pelo menos duas famílias. Veja a tabela 5.1 as super-famílias selecionadas.

Super-famílias	Número de famílias	Número de seqüências
a.1.1	4	31
a.3.1	8	35
a.4.1	12	43
b.6.1	5	39
b.47.1	4	34
b.121.4	9	30

Tabela 5.1: Número de famílias e seqüências das super-famílias consideradas nos experimentos.

Para uma dada super-família x com n famílias, foram construídos n modelos, tomando $n-1$ famílias para o conjunto de treinamento. Para testar cada modelo o conjunto de teste foi formado pelas seqüências da família que não participou do treinamento. A família que não participa do treinamento mede a capacidade de detecção de homólogos distantes. A validação cruzada (HAYKIN, 2001) foi utilizada permitindo que cada família fosse excluída do treinamento uma vez (*leave one family out*).

Consideramos portanto como um problema de múltiplas classes, uma classe para cada super-família. Seguimos uma abordagem *round robin* (FÜRNKRANZ, 2002) em que cada par de classes é tratado como problema de classificação separado e a classificação final de uma instância é determinada por uma votação majoritária sobre todos os pares de problemas. Esta técnica foi usada por Gutmann e Kersting para o problema de classificação de proteínas em *folds* (GUTMANN, KERSTING, 2006).

Definimos, então, uma forma simples de representar os aminoácidos das seqüências de proteínas em sentenças lógicas de primeira ordem: um único predicado "a" podendo ter aridade 1, 2 ou 3, dependendo do experimento realizado.

O primeiro termo deste predicado representa o aminoácido propriamente dito

e pode receber uma das vinte letras usadas para representar os aminoácidos na biologia (veja a tabela 2.1).

O segundo codifica a informação referente a estrutura secundária da proteína cujo aquele aminoácido é parte e pode receber "C" para as regiões irregulares conhecidas como *coil*, "H" para as alfa-hélices, "E" para a folhas-beta e "P" para o ângulo Φ, seguindo as mesmas letras usadas pelo JOY.

O último termo codifica a informação referente ao tipo de coluna a que este termo se refere no alinhamento das seqüências da sua super-família, podendo receber "m" para as colunas de *match*, "i" para as colunas de inserção e "g" para o caso de ausência de um aminoácido dessa seqüência naquela coluna, isto é, no caso de um buraco (*gap*).

O termo $a(N, H, m)$, por exemplo, define que nesta posição da seqüência temos uma asparagina que faz parte de uma alfa-hélice na estrutura secundária da proteína e que está em uma coluna de *matches* no alinhamento primário para a sua super-família. A figura 5.3 é um exemplo de formação de seqüência para o terceiro experimento. No primeiro experimento, o predicado tem apenas um termo, referente ao aminoácido, por exemplo $a(N)$, como na figura 5.1. No segundo experimento, não existe o termo referente a informação estrutural, por exemplo $a(N, m)$. Na ocorrência de um buraco não há aminoácido e nem informação estrutural, sendo estes representados nas seqüências por duas aspas simples seguidas e o termo é assim definido: $a('', '', g)$.

5.2 Resultados experimentais

A avaliação dos resultados foi feita por meio da acurácia (*accuracy*) que é uma medida da correlação entre o valor estimado e os valores das fontes de informação, ou seja, mede o quanto a estimativa que obtivemos é relacionada com o valor real do parâmetro.

Em cada *fold* da validação cruzada, a versão java do TildeCRF fornece um arquivo com os resultados e o conjunto de árvores geradas em cada passo do *tree boosting*. No arquivo de resultados, são encontradas as matrizes de confusão tanto para os dados de treinamento quanto para os dados de teste. Uma matriz de con-

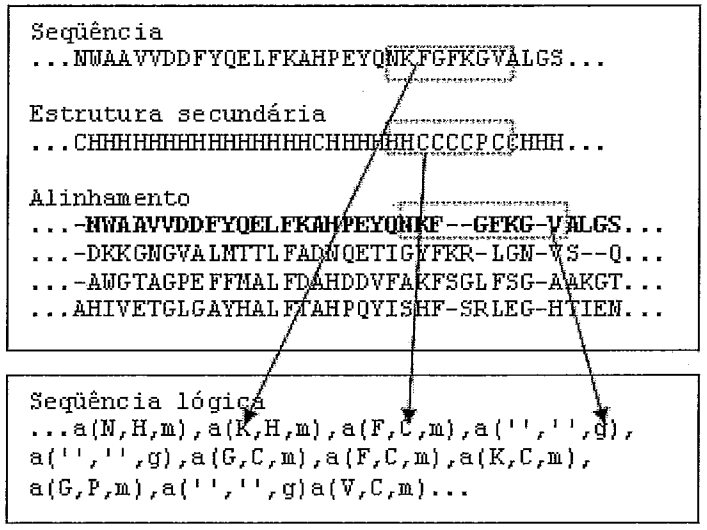


Figura 5.3: Esquema de formação das seqüências de entrada no terceiro experimento.

fusão (*confusion matrix*) (KOHAVI, PROVOST, 1998) é uma forma de visualização tipicamente usada em tarefas de aprendizado supervisionado. Cada coluna da matriz representa a instância da classe predita enquanto cada linha representa as instâncias da classe real. Esta matriz recebe este nome porque nela é simples notar que um sistema está se confundido entre duas classes, isto é, trocando uma pela outra. A tabela 5.2 apresenta um exemplo no qual pode-se perceber que o modelo teve dificuldade em classificar os exemplos da classe C, confundindo-os com os da classe B.

	classe A	classe B	classe C
classe A	26	1	0
classe B	0	34	0
classe C	0	27	15

Tabela 5.2: Matriz de confusão envolvendo três classes.

A partir da matriz de confusão calcula-se a acurácia, bastando para isso computar os acertos, diagonal principal da matriz, e dividi-los pelo total de exemplos. No exemplo da tabela 5.2, somam-se 75 $(26 + 34 + 15)$ acertos em um total de 103 exemplos. Portanto, a acurácia é de aproximadamente 73% (75/103).

Os seguintes parâmetros do TildeCRF foram modificados nos procedimentos dos testes: tamanho da janela igual a 11 (valor *default* 5) e profundidade máxima

permitida para as árvores de regressão relacionais igual a 5 (valor *default* 7). Tentamos aumentar a expressividade dos exemplos gerados pelo TildeCRF para o treinamento das árvores de regressão por meio do aumento do tamanho da janela. E, por outro lado, precisamos diminuir os tamanhos das árvores para viabilizar os experimentos. Os outros parâmetros foram deixados com os valores *defaults*.

As tabelas 5.3 e 5.4 apresentam os resultados obtidos pelo TildeRF nos três experimentos propostos, descritos na seção anterior, a primeira do conjunto de treinamento e a segunda do conjunto de teste. Na primeira tabela, pode-se verificar claramente que os modelos obtêm melhores resultados já que mais informações são usadas para o treinamento dos mesmos. O mesmo também é verificado nos resultados para o conjunto de teste em três das seis super-famílias. A figura 5.4 apresenta dois gráficos com os resultados para cada família da super-família $a.1.1$ que mostram que esse comportamento também é percebido no nível de famílias.

Super-família	Aminoácidos	Alinhamento	Estrutura secundária
a.1.1	0,50	0,97	0,98
a.3.1	0,68	0,99	0,99
a.4.1	0,52	0,98	0,98
b.6.1	0,39	0,86	0,89
b.47.1	0,39	0,85	0,86
b.121.4	0,38	0,94	0,95

Tabela 5.3: Acurácia do TildeCRF no conjunto de treinamento.

Super-família	Aminoácidos	Alinhamento	Estrutura secundária
a.1.1	0,40	0,29	0,54
a.3.1	0,41	0,75	0,87
a.4.1	0,58	0,71	0,79
b.6.1	0,37	0,32	0,38
b.47.1	0,10	0,21	0,35
b.121.4	0,25	0,86	0,40

Tabela 5.4: Acurácia do TildeCRF no conjunto de teste.

Então, os resultados nos levam a crer que a questão **ii**, definida no início deste capítulo, pode ter uma resposta afirmativa. Mas ainda é preciso atestar a significância estatísticas desses resultados.

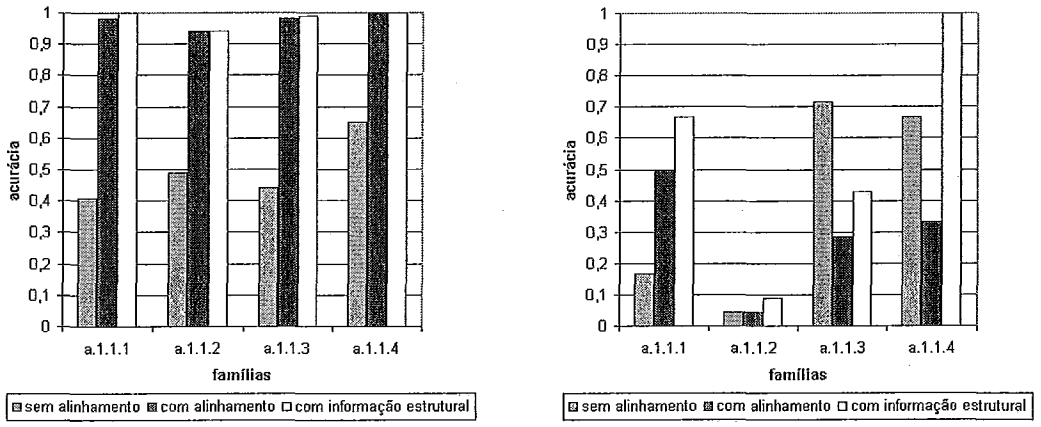


Figura 5.4: Acurácia para as famílias da super-família $a.1.1$ nos três experimentos, primeiro no conjunto de treinamento (esq.) e depois no conjunto de teste (dir.).

Os testes onde as hipóteses são avaliadas com amostras idênticas são chamados de *paired tests* (MITCHELL, 1997). *Paired tests* normalmente produzem intervalos de confiança mais justos porque as diferenças nos erros observados em um *paired test* são certamente devido as diferenças entre as hipóteses. Em contrapartida, quando as hipóteses são testadas com amostras diferentes, as diferenças nos erros de duas amostra podem ser parcialmente devido as diferenças na composição das duas amostras.

Então, a fim de avaliar a significância estatística dos resultados, foi utilizado o *paired t-test* e os resultados foram considerados como significativos quando obtiveram p menor que $0,05$.

As tabelas 5.5 e 5.6 apresentam os resultados do *paired t-test*. A primeira apresenta os resultados para todas as super-famílias consideradas nos experimentos. Pode-se verificar que não foi obtida significância estatística apenas no modelo que usou informações estruturais em relação ao modelo que usou informações dos alinhamentos. A segunda destaca os resultados quando consideramos apenas as super-famílias da classe alfa do SCOP e estes foram significativos na comparação entre os três modelos.

Então, é possível afirmar que a questão **ii**, definida no início deste capítulo, tem resposta afirmativa, isto é, o TildeCRF obtém resultados significativamente melhores ao fazer uso de informações estruturais das proteínas.

	Aminoácidos	Estrutura secundária
Aminoácidos		0,00298 (sim)
Alinhamento	0,00114 (sim)	0,96747 (não)

Tabela 5.5: Resultados do *paired t-test* e significância estatística sobre os resultados da tabela 5.4 para todas as super-famílias consideradas.

	Aminoácidos	Estrutura secundária
Aminoácidos		0,00229 (sim)
Alinhamento	0,003622 (sim)	0,04692 (sim)

Tabela 5.6: Resultados do *paired t-test* e significância estatística sobre os resultados da tabela 5.4 para as super-famílias da classe alfa do SCOP.

Porém, identificamos que a acurácia dos resultados oscila muito de um passo para outro do *tree boosting* e isso pode explicar alguns resultados ruins, mas também coloca em dúvida alguns bons resultados. Veja nos gráficos da figura 5.5 que provavelmente ocorreu *overfitting* no treinamento, por isso, decidimos considerar sempre os resultados dos passos do *tree boosting* com a melhor acurácia no treinamento e a acurácia do conjunto de teste nesse mesmo passo, conseqüentemente, desconsiderando os passos restantes. Os resultados considerando todos os passos do *tree boosting* foram publicados recentemente (COSTA, et al., 2008a).

As figuras 5.6 e 5.7 apresentam duas árvores de regressão geradas pelo TildeCRF, uma de um *fold* da validação cruzada em que a família *a*.1.1.1 ficou excluída do treinamento e a outra de um *fold* em que a família excluída foi a *b*.6.1.1. Pode-se notar a ausência de aminoácidos e informações estruturais na segunda árvore em relação a primeira, os dois primeiros termos do átomo lógico, prevalecendo os casos de variáveis *unbounded* (_0). Este comportamento ocorreu com todas as famílias das super-famílias da classe beta, o que indica uma baixa qualidade nos exemplos gerados pelo TildeCRF para o treinamento dos modelos.

Por fim, reproduzimos os mesmos experimentos feitos com o TildeCRF utilizando o HMMER, indiscutivelmente, uma das mais populares ferramentas de detecção de homólogos distantes, e com o HMMER-STRUCT, que é uma versão modificada do HMMER, que considera informações estruturais na construção do modelo, criando cinco pHMMs, cada um para uma propriedade estrutural específica.

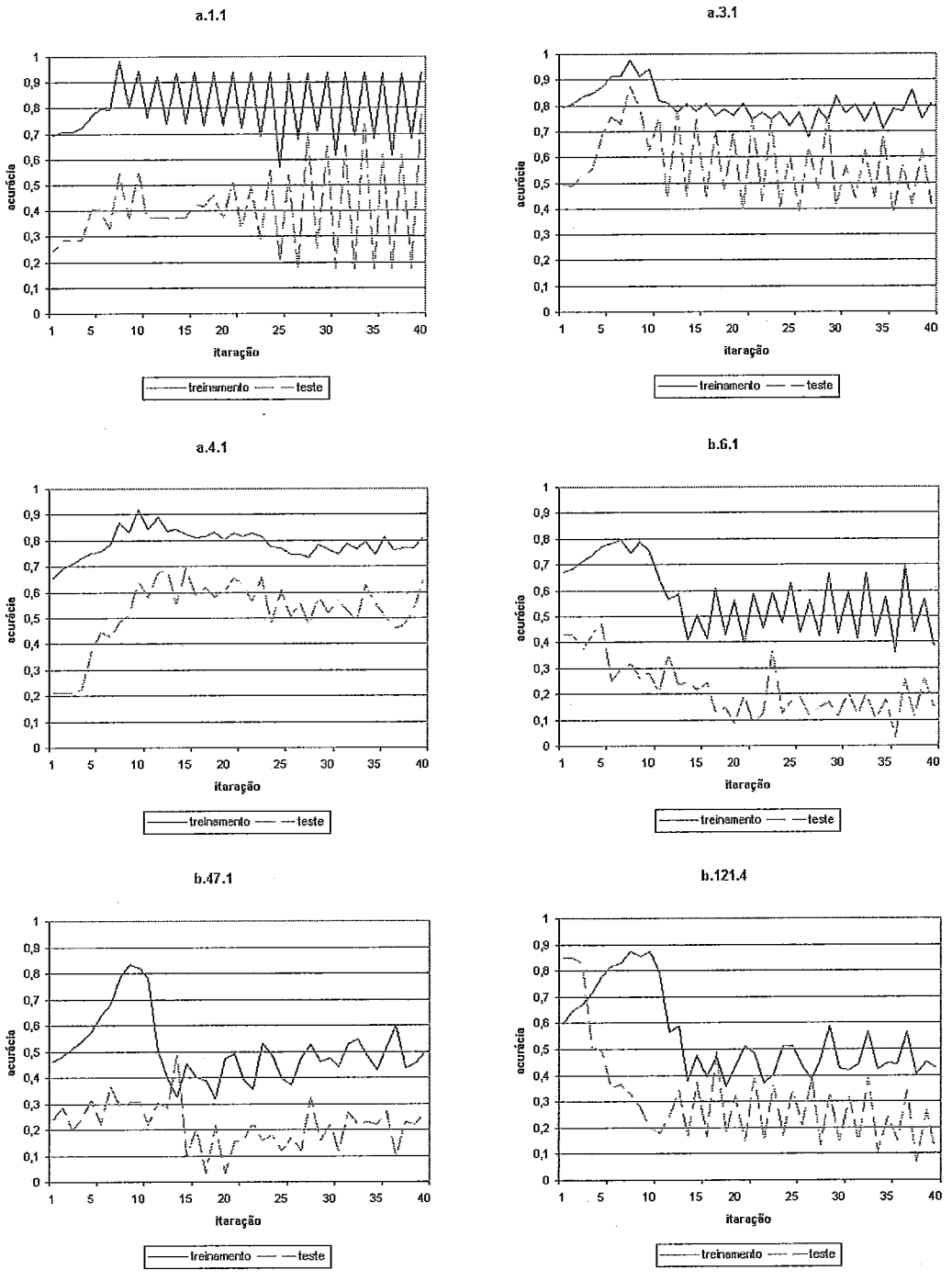


Figura 5.5: Acurácia versus o números de passos do *tree boosting* no experimento com informações estruturais.

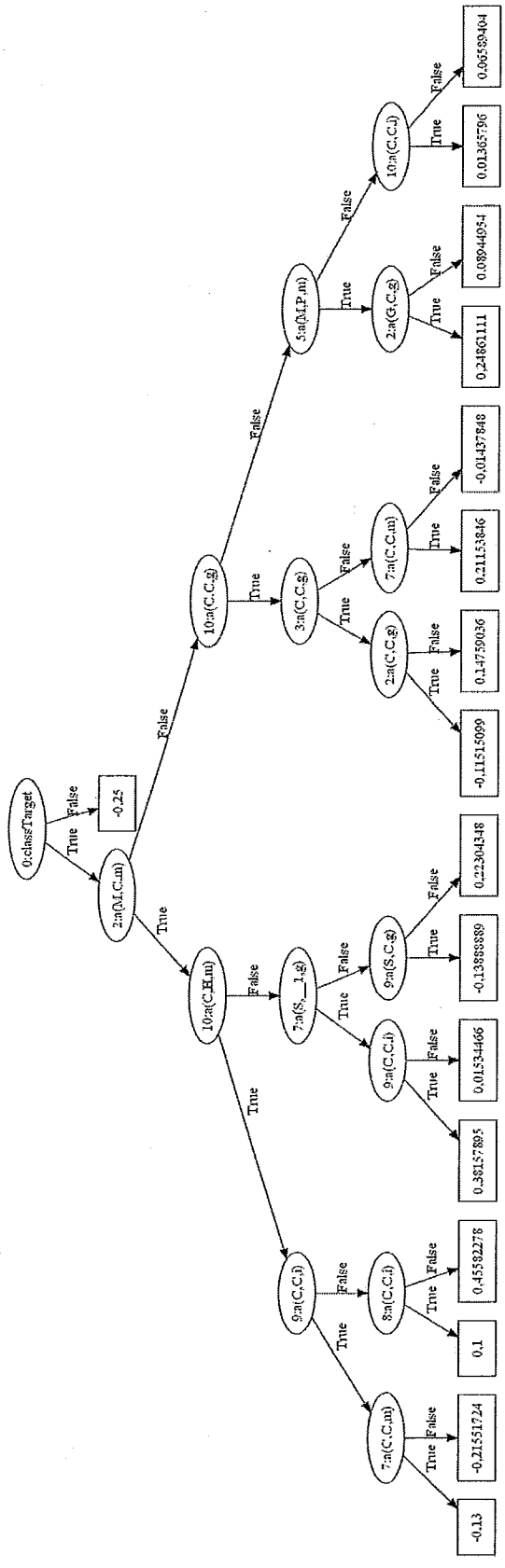


Figura 5.6: Árvore para experimento com informações estruturais em que a família $a.1.1.1$ fica excluída do treinamento.

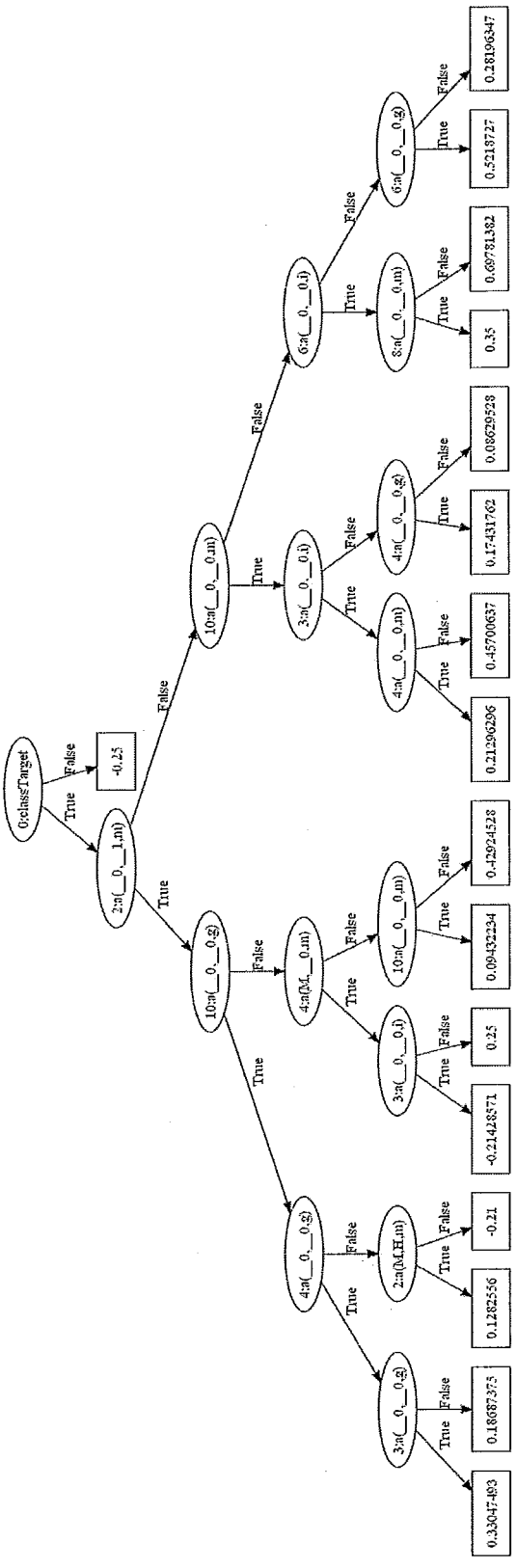


Figura 5.7: Árvore para experimento com informações estruturais em que a família *b*.6.1.1 fica excluída do treinamento.

As propriedades utilizadas pelo HMMER-STRUCT são: estruturas primária, secundária e terciária, acessibilidade e empacotamento (*packing*) de aminoácidos. Neste trabalho, foram usados apenas os modelos do HMMER-STRUCT referentes as estruturas primária e secundária, uma vez que não incorporamos as outras informações nas seqüências usadas no TildeCRF.

Os modelos foram construídos utilizando o programa *hmmbuild*. O programa *hmmpfam* foi utilizado para realizar busca na base de dados de modelos. Bibliotecas de modelos têm sido utilizadas por diversos trabalhos (BATEMAN, et al., 2004) (HAFT, et al., 2003) (GOUGH, et al., 2001) e apresentam melhores resultados quando comparadas a modelos individuais. Em geral, se uma proteína é classificada por vários modelos relacionados, a chance de sua classificação estar correta é maior quando esta é comparada a outras classificações que utilizam um único modelo.

Quando um pHMM é comparado com um banco de dados de seqüências genômicas, a medida ideal de significância relacionada aos *scores* é o *e-value* (EDDY, 1998). O *e-value* relacionado a uma seqüência de *score* s e a um banco de dados de seqüências é calculado através do número esperado de seqüências no banco de dados que possuem um *score* maior do que s. O *hmmpfam* apresenta os *e-values* das seqüências avaliadas em cada um dos modelos e a classificação final da seqüência é do modelo em que for apresentado o menor *e-value* entre 0 e 1. A acurácia é calculada da mesma forma que mostramos para a matriz de confusão dividindo-se os acertos pelo total de exemplos. A tabela 5.7 mostra a comparação dos resultados por super-família.

Super-família	HMMER	HMMER-STRUCT	TildeCRF
a.1.1	0,67	0,74	0,54
a.3.1	0,97	0,97	0,87
a.4.1	0,54	0,56	0,79
b.6.1	0,72	0,99	0,38
b.47.1	0,81	0,84	0,35
b.121.4	0,69	0,69	0,40

Tabela 5.7: Comparação da acurácia no conjunto de teste: TildeCRF com informação estrutural secundária, HMMER e HMMER-STRUCT apenas com os modelos de informação estrutural secundária.

O HMMER alcançou uma acurácia média de 73% e o HMMER-struct 80%. Em contraste, o TildeCRF atingiu apenas 55%. Este resultado demonstra que o TildeCRF não teve *performance* comparável com a que os outros atingiram, muito provavelmente em função do *overfitting* observado nos gráficos da figura 5.5. Mas, o fato dele ter sido melhor em uma das seis super-famílias consideradas, na super-família a.4.1, sugere que a questão i ainda pode ser respondida de forma satisfatória.

Como última observação dos experimentos, vale ressaltar que foi necessário muito tempo de processamento para executar estes experimentos. E à medida que incorporamos mais informações nos exemplos de treinamento, mais tempo é necessário. Na observação dos *logs* do sistema verificamos que a etapa de treinamento das árvores é o ponto do algoritmo mais demorado. Os experimentos foram executados em um *cluster* com quinze máquinas com dois processadores cada. Veja os tempos de um fold da validação cruzada para cada super-família na tabela 5.8.

Super-família	Aminoácidos	Alinhamento	Estrutura secundária
a.1.1 (4)	5	17	78
a.3.1 (8)	4	19	67
a.4.1 (12)	3	23	68
b.6.1 (5)	4	16	106
b.47.1 (4)	4	17	114
b.121.4 (8)	4	14	110

Tabela 5.8: Tempo de processamento médio aproximado em horas de um *fold* da validação cruzada para cada super-família, entre parênteses o número de famílias de cada super-família.

Capítulo 6

Conclusão

6.1 Contribuições

A precisão dos métodos de detecção de homologias é de vital importância para os processo de anotação de novas seqüências. O principal desafio dessa área é direcionado à detecção de homologias distantes. O objetivo é diminuir o número de novas seqüências sem correspondência em bancos de dados públicos empregando métodos mais eficientes.

O crescimento das bases de dados que armazenam informações estruturais de proteínas, pode ser visto como uma motivação ao aprimoramento e surgimento de métodos baseados nessas informações. Os alinhamentos estruturais captam as similaridades espaciais entre as coordenadas dos átomos de um conjunto de proteínas. Esses alinhamentos fornecem informações significativas sobre as relações das amostras do conjunto, principalmente quando as as seqüências analisadas são muito divergentes.

A proposta deste trabalho foi investigar o problema de detecção de homólogos distantes em seqüências de proteínas por meio do uso de modelos estatísticos discriminativos relacionais. Então, foi adotado o TildeCRF, que une CRF e lógica de primeira ordem, como ferramenta base para os experimentos. A seção 4.2 descreve os algoritmos de forma detalhada.

A principal contribuição desse trabalho foi desenvolver uma metodologia para usar o TildeCRF no problema de detecção de homologias distantes. O capítulo 5

mostrou que informações estruturais utilizadas na construção dos CRFs produzem melhores resultados em relação a alinhamentos primários. O mesmo já havia sido constatado para pHMMs (BERNARDES, 2005).

Porém, a comparação dos nosso resultados com os obtidos usando ferramentas específicas para o problema, há muito tempo estabelecidas, mostrou que há ainda muito o que se experimentar na direção de adaptar os modelos discriminativos, como os CRFs, ao problema de detecção de homologias distantes.

6.2 Trabalhos Futuros

Este trabalho ateve-se ao uso de informações de estruturas secundária, porém, é interessante explorar outras características que não foram consideradas, tais como acessibilidade e empacotamento de aminoácidos e pontes de hidrogênio, uma vez que, essas características, quando devidamente exploradas, aumentam a sensibilidade dos modelos.

Seria interessante realizar uma análise adicional para avaliar se a representação lógica adotada para as proteínas foi a mais adequada. Então, poderíamos definir novas regras lógicas que melhor combinem os resultados dos CRFs. Isto aliado ao uso de conhecimento preliminar (*background knowledge*) pode levar a construção de árvores mais significativas no treinamento e, conseqüentemente, a redução do *overfitting*.

Um dos grandes entraves dos experimentos foi o alto custo computacional do TildeCRF, principalmente o tempo de processamento, como foi visto no capítulo 5. Uma representação mais compacta seria importante para diminuir o processamento necessário para o aprendizado dos modelos, assim como uma definição mais consciente dos parâmetros utilizados. Outra possibilidade de melhoria nesse sentido seria modificar o algoritmo, por exemplo, introduzindo um comportamento randômico na estimativa das árvores de regressão (GUTMANN, KERSTING, 2007).

Referências Bibliográficas

ALBERTS, B., JOHNSON, A., LEWIS, J., RAFF, M., ROBERTS, K., WALTER, P., 2002, *Molecular Biology of the Cell*, Garland Science.

ALEXANDROV, V., GERSTEIN, M., 2004, "Using 3D Hidden Markov Models that explicitly represent spatial coordinates to model and compare protein structures", *BMC Bioinformatics*, v. 5, n. 2, pp. 1–10.

ALTSCHUL, F., GISH, W., MILLER, W., MYERS, E., LIPMAN, D., 1990, "A basic local alignment search tool", *Journal of Molecular Biology*, v. 215, n. 3.

ALTSCHUL, F., MADDEN, L., SCHAFFER, A., ZHANG, J., ZHANG, Z., MILLER, W., , LIPMAN, D., 1997, "Gapped BLAST and PSI-blast: a new generation of protein database search programs", *Nucleic Acids Research*, v. 25, n. 17.

ANDERSON, C. R., DOMINGOS, P., WELD, D., 2002, "Relational Markov Models and their Application to Adaptive Web Navigation", In: *Proceedings of the 8th Int. Conf. on Knowledge Discovery and Data Mining (KDD-02)*, pp. 143–152.

ANDREEVA, A., HOWORTH, D., BRENNER, S., CHOTHIA, T. H. C., MURZIN, A., 2004, "SCOP database in 2004: refinements integrate structure and sequence family data", *Nucleic Acids Research*, v. 32, pp. 226–229.

BAE, K., MALLICK, B. K., ELSIK, C. G., 2005, "Prediction of protein interdomain

linker regions by a hidden Markov model", *Bioinformatics*, v. 21, n. 10, pp. 2264–2270.

BAGOS, P. G., LIAKOPOULOS, T., HAMODRAKAS, S. J., 2004, "Faster gradient descent training of hidden Markov models, using individual learning rate adaptation", In:Paliouras, G., Sakakibara, Y., editors, *International Colloquium on Grammatical Inference, Lecture Notes in Computer Science, Springer-Verlag*, volume 3264, pp. 40–52.

BAIROCH, A., BOECKMANN, B., 1993, "The SWISS-PROT protein sequence data bank, recent developments", *Nucleic Acids Research*, v. 21, pp. 3093–3096.

BALDI, P., BRUNAK, S., 2001, *Bioinformatics: The Machine Learning Approach*, 2nd ed., The MIT Press, Massachusetts, USA.

BATEMAN, A., COIN, L., DURBIN, R., FINN, R., HOLLICH, V., GRIFFITHS-JONES, S., KHANNA, A., MARSHALL, M., S. MOXON, E. S., STUDHOLME, D., YEATS, C., , EDDY, S. R., 2004, "The Pfam Protein Families Database", *Nucleic Acids Research*, v. 32, pp. 138–141.

BAUM, L., 1972, "An equality and associated maximization technique in statistical estimation for probabilistic functions of markov process", *Inequalities*, v. 12, n. 3, pp. 1–8.

BELLMAN, R., 1957, *Dynamic Programming*, Princeton University Press.

BENSON, D. A., KARSCH-MIZRACHI, I., LIPMAN, D. J., OSTELL, J., WHEELER, D. L., 2005, "GenBank", *Nucleic Acids Research*, v. 33, pp. 34–38.

BERMAN, H., WESTBROOK, J., FENG, Z., GILLILAND, G., BHAT, T. N., SHINDYALOV, H. W. I., , BOURNE, P., 2000, "The Protein Data Bank", *Nucleic Acids Research*, v. 28, pp. 235–242.

BERNARDES, J. S., 2005, *Detecção de Homologias distantes utilizando HMMs e informações estruturais*, Master's thesis, Universidade Federal do Rio de Janeiro, COPPE/UFRJ.

BERNARDES, J. S., DAVILA, A. M. R., COSTA, V. S., ZAVERUCHA, G., 2007, "Improving Model Construction of Profile HMMs for Remote Homology Detection Through Structural Alignment", *BMC Bioinformatics*, v. 8, n. 435, pp. 1–12.

BERTSEKAS, D. P., 1995, *Dynamic Programming and Optimal Control*, Athena Scientific.

BERTSEKAS, D. P., 1999, *Nonlinear Programming*, 2nd ed., Athena Scientific.

BLOCKEEL, H., De RAEDT, L., 1998, "Top-down Induction of First-order Logical Decision Trees", *Artificial Intelligence*, v. 101, n. 1-2, pp. 285–297.

BOECKMANN, B., BAIROCH, A., APWEILER, R., BLATTER, M., ESTREICHER, A., GASTEIGER, E., MARTIN, M. J., MICHOUD, K., O'DONOVAN, C., PHAN, I., PILBOUT, S., , , SCHNEIDER, M., 2003, "The Swiss-Prot protein knowledgebase and its supplement TrEMBL in 2003", *Nucleic Acids Research*, v. 31, pp. 365–370.

BOURNE, P., WEISSIG, H., 2003, *Structural Bioinformatics*, chapter Fundamentals of protein structure, pp. 15–36 Sinauer Associates.

BRANDEN, C., TOOZE, J., 1991a, *Introduction to Protein Structure*, chapter The building blocks, pp. 3–9 Garland Publishing.

BRANDEN, C., TOOZE, J., 1991b, *Introduction to protein Structure*, chapter Motifs of protein structure, pp. 11–29 Garland Publishing.

BREIMAN, L., FRIEDMAN, J. H., OLSHEN, R. A., STONE, C. J., 1984, *Classification and regression trees*, Wadsworth and Brooks, Belmont, CA.

BREJOVA, B., BROWN, D., LI, M., VINAR, T., 2005, "ExonHunter: a comprehensive approach to gene finding", *Bioinformatics*, v. 21, n. 1, pp. 57–65.

BYSTROFF, C., BAKER, D., 2000, "HMMSTR: A hidden Markov model for local sequence-structure correlation in proteins", *Jornal of Molecular Biology*, v. 301, n. 18, pp. 173–190.

CAMPROUX, A. C., TUFFÉRY, P., 2005, "Hidden Markov model-derived structural alphabet for proteins: the learning of protein local shapes captures sequence specificity", *Biochimica et Biophysica Acta (BBA)*, v. 1724, n. 3, pp. 394–403.

CASANOVA, M. A., GIORNO, F. A. C., FURTADO, A. L., 1987, *Programação em Lógica e a Linguagem Prolog*, Edgard Blücher.

CHAKRABARTI, S., SOWDHAMINI, R., 2004, "Regions of minimal structural variation among members of protein domain superfamilies: application to remote homology detection and modelling using distant relationships", *FEBS Letters*, v. 569, n. 1, pp. 31–36.

COLLINS, M., 2002, "Discriminative training methods for hidden Markov models: theory and experiments with perceptron algorithms", In: *Proceedings of the Association for Computational Linguistics conference on Empirical methods in natural language processing*, volume 10, pp. 1–8.

COSTA, J. B. S., BERNARDES, J. S., COSTA, V. S., ZAVERUCHA, G., 2008a, "Remote Homology Detection Through Discriminative Statistical Relational Learning", In: *Statistical and Relational Learning in Bioinformatics (StReBIO), Proceedings of the 18th European Conference on Machine Learning.*

COSTA, V. S., PAGE, D., CUSSENS, J., 2008b, "CLP(BN): Constraint Logic Programming for Probabilistic Knowledge", *Probabilistic Inductive Logic Programming*, pp. 156–188.

COSTA, V. S., PAGE, D., QAZI, M., , CUSSENS, J., 2003, "CLP(BN): Constraint Logic Programming for Probabilistic Knowledge", In: *Proceedings of the Nineteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI-03)*, pp. 517–524.

De RAEDT, L., DZEROSKY, S., 1994, "First order jk-clausal theories are PA-Clearnable", *Artificial Intelligence*, v. 70, pp. 375–392.

DEGROOT, M., 1987, *Probability and Statistics*, 3nd ed., Addison-Wesley.

DEMPSTER, A. P., LAIRD, N. M., RUBIN, D. B., 1977, "Maximun likelihood from incomplete data via the EM algorithm", *Journal of the Royal Statistical Society*, v. 39, n. 1, pp. 1–38.

DIETTERICH, T., ASHENFELTER, A., BULATOV, Y., 2004, "Training conditional random Fields via gradient tree boosting", In: *Proceedings of the 21th International Conference on Machine Learning*, ACM, pp. 217–224.

DO, C. B., GROSS, S. S., BATZOGLOU, S., 2006, "CONTRAlign: Discriminative Training for Protein Sequence Alignment", In: *Proceedings of the Tenth Annual International Conference on Computational Molecular Biology (RECOMB06).*

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

DZEROSKI, S., 2003, "Multi-relational data mining: an introduction", *SIGKDD Explorations*, v. 5, n. 1, pp. 1–16.

EDDY, S., 1998, "Profile hidden Markov models", *Bioinformatics*, v. 14, n. 9, pp. 755–763.

EDGAR, R., SJÖLANDER, K., 2004, "COACH: profile-profile alignment of protein families using hidden Markov models", *Bioinformatics*, v. 20, n. 8, pp. 1309–1318.

EFRON, B., 1975, "The efficiency of logistic regression compared to normal discriminant analysis", *Journal of the American Statistical Association*, v. 70, pp. 892–898.

ESPADALER, J., ARAGÜÉS, R., ESWAR, N., MARTI-RENOM, M., QUEROL, E., AVILÉS, F., SALI, A., , OLIVA, B., 2005, "Detecting remote related proteins by their interactions and sequence similarity", In: *Proceedings of the National Academy of Sciences of the United States of America*, volume 102, pp. 7151–7156.

FREUND, Y., SCHAPIRE, R. E., 2003, "Experiments with a new boosting algorithm", In: *Proceedings of the 13th International Conference on Machine Learning*, pp. 148–156 Morgan Kaufmann.

FRIEDMAN, J. H., 2001, "Greedy function approximation: A gradient boosting machine", In: *Annals of Statistics*, volume 29.

FRIEDMAN, N., GETOOR, L., KOLLER, D., PFEFFER, A., 1999, "Learning Probabilistic Relational Models", In: *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI-99)*, pp. 1300–1309.

FÜRNKRANZ, J., 2002, "Round Robin Classification", *Journal of Machine Learning Research (JMLR)*, v. 2, pp. 721–747.

GOUGH, J., KARPLUS, K., HUGHEY, R., CHOTHIA, C., 2001, "Assignment of homology to genome sequences using a library of hidden Markov models that represent all proteins ok known structure", *Journal of Molecular Biology*, v. 313, n. 4, pp. 903–919.

GOYON, F., TUFFÉRY, P., 2004, "SA-Search: A web tool for protein structure mining based on structural alphabet", *Nucleic Acids Research*, v. 32, pp. 545–548.

GRIBSKOV, M., MCLACHLAN, A., EISENBERG, D., 1987, "Profile analysis: detection of distantly related proteins", In: *Proceedings of the National Academy of Sciences of the United States of America*, volume 84, pp. 4355–4358.

GRUNDY, W., BAKER, M., 1997, "Meta-MEME: Motif-based Hidden Markov Models of Protein Families", *Computer Applications in the Biosciences*, v. 13, n. 4, pp. 397–406.

GUTMANN, B., KERSTING, K., 2006, "TildeCRF: Conditional Random Fields for Logical Sequences", In:Boulicaut, J.-F., Esposito, F., Giannotti, F., Pedreschi, D., editors, *Proceedings of the 17th European Conference on Machine Learning*, volume 4212, pp. 174–185, Berlim, Germany.

GUTMANN, B., KERSTING, K., 2007, "Stratified Gradient Boosting for Fast Training of CRFs", In:Frasconi, P., Kersting, K., Tsuda, K., editors, *Proceedings of the 5th International Workshop on Mining and Learning with Graphs (MLG 2007)*, pp. 131–134, Florence, Italy.

HAFT, D. H., SELENGUT, J. D., WHITE, O., 2003, "The TIGRFAMs database of protein families", *Nucleic Acids Research*, v. 31, pp. 371–373.

HALPERN, J. Y., 1989, "An analysis of first-order logics of probability", *Artificial Intelligence*, v. 46, pp. 311–350.

HAYKIN, S., 2001, *Redes Neurais: Princípios e prática*, Bookman.

HENIKOFF, S., HENIKOFF, J., 1991, "Automated assembly of protein blocks for database searching", *Nucleic Acids Research*, v. 19, n. 23, pp. 6565–6572.

HOU, Y., HSU, W., LEE, M. L., BYSTROFF, C., 2004, "Remote homology detection using local sequence-structure correlations", *Proteins: Structure, Function and Bioinformatics*, v. 57, n. 3, pp. 518–530.

HUGHEY, R., KROGH, A., 1995, "SAM: Sequence Alignment and Modeling Software System", Technical Report UCSC-CRL-95-7, University of California at Santa Cruz, Santa Cruz, CA, USA.

HUGHEY, R., KROGH, A., 1996, "Hidden Markov models for sequence analysis: extension and analysis of basic method", *Computer Applications in the Biosciences*, v. 12, n. 2, pp. 95–108.

KARLIN, S., ALTSCHUL, S. F., 1990, "Methods for assessing the statistical significance of molecular sequence features by using general scoring schemes", In: *Proceedings of the National Academy of Sciences of the United States of America*, volume 87, pp. 2264–2268.

KARPLUS, K., 1995, "Regularizers for estimation distributions of aminoacids from small samples", Technical report, University of California, Santa Cruz, CA.

KERSTING, K., De RAEDT, L., 2001, "Towards Combining Inductive Logic Programming with Bayesian Networks", In: *Proceedings of the Eleventh Conference on Inductive Logic Programming (ILP-01), LNAI 2157, Springer Verlag*, pp. 118–131.

KERSTING, K., De RAEDT, L., RAIKO, T., 2006, "Logical Hidden Markov Models", *Journal of Artificial Intelligence Research*, v. 25, pp. 425–456.

KERSTING, K., GÄRTNER, T., 2004, "Fisher Kernels for Logical Sequences", In:J. Fürnkranz, T. S., Spiliopoulou, M., editors, *Proceedings of the 15th European Conference on Machine Learning*, pp. 205–216, Pisa, Italy.

KERSTING, K., RAIKO, T., KRAMER, S., De RAEDT, L., 2003, "Towards discovering structural signatures of protein folds based on logical hidden Markov models", In: *Proceedings of the eighth Pacific Symposium on Biocomputing (PSB)*, volume 8, pp. 192–203.

KNUDSEN, B., MIYAMOTO, M., 2003, "Sequence alignments and pair hidden Markov models using evolutionary history", *Journal of Molecular Biology*, v. 333, n. 2, pp. 453–460.

KOHAVI, R., PROVOST, F., 1998, "Glossary of terms", *Machine Learning Journal*, v. 30, pp. 271–274.

KOLLER, D., 1999, "Probabilistic Relational Models", In: *Proceedings of the Ninth International Conference on Inductive Logic Programming (ILP-99), LNAI 1634, Springer Verlag*, pp. 3–13.

KROGH, A., BROWN, M., MIAN, I. S., SJOLANDER, K., HAUSSLER, D., 1994, "Hidden Markov Models in Computational Biology: Applications to Protein Modeling", *Journal of Molecular Biology*, v. 235, pp. 1501–1531.

LAFFERTY, J., MCCALLUM, A., PEREIRA, F., 2001, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data", In: *Proceedings of the 18h International Conference of Machine Learning*, pp. 282–289, San Francisco, CA Morgan Kaufmann.

LAVRAC, N., DZEROSKY, S., 1994, *Inductive Logic Programming: Techniques and Applications*, Ellis Horwood, New York.

LEVINSON, S. E., RABINER, L. R., SONDHI, M. M., 1983, "An introduction to the application of the theory of probabilistic functions of a Markov process to automatic speech recognition", *Bell System Technical Journal*, v. 39, n. 1, pp. 1035–1074.

LIN, K., SIMOSSIS, V. A., TAYLOR, W. R., HERINGA, J., 2005, "A simple and fast secondary structure prediction method using hidden neural networks", *Bioinformatics*, v. 21, n. 2, pp. 152–159.

LIU, Y., CARBONELL, J., WEIGELE, P., GOPALAKRISHNAN, V., 2005, "Segmentation conditional random fields (SCRFs): A new approach for protein fold recognition", In: *ACM International conference on Research in Computational Molecular Biology (RECOMB05)*.

MAJOROS, W. H., PERTEA, M., SALZBERG, S. L., 2005, "Eficient implementation of a generalized pair hidden Markov model for comparative gene finding", *Bioinformatics*, v. 21, n. 9, pp. 1782–1788.

MALOUF, R., 2002, "A comparison of algorithms for maximum entropy parameter estimation", In:Roth, D., van den Bosch, A., editors, *Proceedings of the Sixth Conference on Natural Language Learning (CoNLL-2002)*, pp. 49–55.

MAMITSUKA, H., 2005, "Finding the biologically optimal alignment of multiple sequences", *Artificial Intelligence in Medicine*, v. 35, n. 1-2, pp. 9–18.

MCCALLUM, A., 2003, "Efficiently inducing features of conditional random fields", In: *Proceedings of the 19h Conference on Uncertainty in Artificial Intelligence*, pp. 403–410, San Francisco, CA Morgan Kaufmann.

MENDEL, M., 1992, "A commercial large-vocabulary discrete speech recognition system: Dragon Dictate", *Language and Speech*, v. 35, n. 1-2, pp. 237–246.

MINKA, T. P., 2003, "A comparsion of numerical optimizers for logistic regression", Technical report, Microsoft.

MITCHELL, T. M., 1997, *Machine Learning*, McGraw-Hill, New York.

MIZUGUCHI, K., DEANE, C., BLUNDELL, T., JOHNSON, M., OVERINGTON, J., 1998, "JOY: protein sequence-structure representation and analysis, Bioinformatics", *Bioinformatics*, v. 14, n. 7, pp. 617–623.

MUGGLETON, S., 1995, "Inverse Entailment and Progol", *New Generation Computing Journal*, v. 13, pp. 245–286.

MUGGLETON, S., 2002, "Learning structure and parameters of stochastic logic programs", In: *Proceedings of the Twelth International Conference on Inductive Logic Prgramming (ILP-02),LNAI 2583, Springer Verlag*, pp. 198–206.

MUGGLETON, S., De RAEDT, L., 1994, "Inductive logic programming: Theory and methods", *Journal of Logic Programming*, v. 19/20, pp. 629–679.

NG, A. Y., JORDAN, M. I., 2002, "On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes", In:Dietterich, T. G., Becker, S., Ghahramani, Z., editors, *In Advances in Neural Information Processing Systems 14*, volume 2, pp. 841–848, Cambidge, MA MIT Press.

NIENHUYS-CHENG, S.-H., WOLF, R. D., 1997, *Foundations of Inductive Logic Programming*, Springer, Berlin.

NISHIKAWA, K., OOI, T., 1986, "Radial locations of amino acid residues in a globular protein: correlation with the sequence", *Journal of Biochemistry*, v. 100, n. 4, pp. 1043–1047.

NOTREDAME, C., HIGGINS, D. G., HERINGA, J., 2000, "T-coffee: a novel method for fast and accurate multiple sequence alignment", *Journal of Molecular Biology*, v. 302, n. 1, pp. 205–217.

PAULING, L., COREY, R., BRANSON, H., 1951, "The structure of protein: two hydrogen-bonded helical configurations of the polypeptide chain", In: *Proceedings of the National Academy of Sciences of the United States of America*, volume 31, pp. 205–211.

PEARSON, W. R., 1990, "Rapid and sensitive sequence comparisons with FASTP and FASTA", *Methods Enzymology*, v. 183, n. 1, pp. 63–98.

PENG, F., MCCALLUM, A., 2004, "Accurate information extraction from research papers using conditional random fields", In: *Proceedings of Human Language Technology Conference and North American Chapter of the Association for Computational Linguistics (HLT-NAACL04)*.

PETSKO, G. A., RINGE, D., 2004, *Protein Structure and Function*, New Science Press.

POOLE, D., 1993, "Probabilistic Horn abduction and Bayesian networks", *Artificial Intelligence*, v. 64, n. 1, pp. 81–129.

QUINLAN, J. R., 1990, "Learning Logical Definitions from Relations", *Machine Learning*, v. 5, n. 3, pp. 239–266.

QUINLAN, J. R., 1993, "C4.5: Programs for Machine Learning", *Machine Learning*, v. 16, n. 3, pp. 235–240.

RABINER, L. R., 1989, "A tutorial on hidden Markov models and selected applications in speech recognition", In: *Proceedings of the IEEE*, volume 77, pp. 257–286, San Francisco Morgan Kaufmann.

RICHARDSON, M., DOMINGOS, P., 2006, "Markov Logic Networks", *Machine Learning*, v. 62, n. 1-2, pp. 107–136.

SADREYEV, R. I., GRISHIN, N. V., 2004, "Estimates of statistical significance for comparison of individual positions in multiple sequence alignments", *BMC Bioinformatics*, v. 5, n. 106.

SATO, K., SAKAKIBARA, Y., 2005, "RNA secondary structural alignment with conditional random fields", *Bioinformatics*, pp. 237–242.

SATO, T., KAMEYA, Y., 1997, "PRISM: A symbolic-statistical modeling language", In: *Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 1330–1335.

SCOTT, M., MATSUDAIRA, P., LODISH, H., DARNELL, J., ZIPURSKY, L., KAISER, C., BERK, A., , KRIEGER, M., 2000, *Molecular Cell Biology*, Von Hoffman Press.

SÖDING, J., 2005, "Protein Homology detection by HMM-HMM comparison", *Bioinformatics*, v. 21, n. 7, pp. 951–960.

SETTLES, B., 2005, "Abner: an open source tool for automatically tagging genes, proteins, and other entity names in text", *Bioinformatics*, pp. 3191–3192.

SHA, F., PEREIRA, F., 2003, "Shallow parsing with conditional random fields", In: *Proceedings of Human Language Technology Conference and North American Chapter of the Association for Computational Linguistics (HLT-NAACL03)*.

SJÖLANDER, K., KARPLUS, K., BROWN, M., HUGHEY, R., KROGH, A., MIAN, I. S., , HAUSSLER, D., 1996, "Dirichlet mixtures: a method for improving detection of weak but significant protein sequence homology", *Computer Applications in the Biosciences*, v. 12, n. 4, pp. 327–345.

SMITH, T., WATERMAN, M., 1981, "Identification of common molecular subsequences", *Journal of Molecular Biology*, v. 147, n. 1, pp. 195–197.

SRINIVASAN, A., 2001, "The Aleph manual", Technical report, Oxford University.

SUMNER, J. B., 1926, "The Isolation and Crystallization of the Enzyme Urease. Preliminary Paper", *Journal of Biological Chemistry*, v. 69, pp. 435–441.

SUTTON, C., MCCALLUM, A., 2006, *Introduction to Statistical Relational Learning*, chapter An Introduction to Conditional Random Fields for Relational Learning MIT Press.

TASKAR, B., ABBEEL, P., KOLLER, D., 2002, "Discriminative probabilistic models for relational data", In: *Proceedings of 8th Conference on Uncertainty in Artificial Intelligence (UAI02)*, pp. 485–492.

TATUSOV, R. L., ALTSCHUL, S. F., KOONIN, E. V., 1994, "Detection of conserved segments in proteins: iterative scanning of sequence databases with align-

ment blocks", In: *Proceedings of the National Academy of Sciences of the United States of America*, volume 91, pp. 12091–12095.

THOMPSON, J. D., HIGGINS, D. G., GIBSON, T. J., 1994, "CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, positionspecific gap penalties and weight matrix choice", *Nucleic Acids Research*, v. 22, pp. 4673–4680.

VAPNIK, V., 1995, *The Nature of Statistical Learning Theory*, Springer-Verlag New York.

WALLACH, H., 2002, *Efficient training of conditional random fields*, Master's thesis, University of Edinburgh.

WALLE, I., LASTERS, I., WYNS, L., 2004, "Align-m: a new algorithm for multiple alignment of highly divergent sequences", *Bioinformatics*, v. 20, n. 9, pp. 1428–1435.

WISTRAND, M., SONNHAMMER, E. L., 2005, "Improved profile HMM performance by assessment of critical algorithmic in SAM and HMMER", *BMC Bioinformatics*, v. 6, pp. 99.