**UNIVERSIDADE DE SÃO PAULO
INSTITUTO DE FÍSICA**

**Aprendizado em Modelos de Markov com Variáveis de
Estado Escondidas**

Roberto Castro Alamino

*Tese apresentada ao Instituto
de Física da Universidade de
São Paulo para a obtenção
do título de Doutor em Ciên-
cias.*

Orientador : Prof. Dr. Nestor Felipe Caticha Alfonso

Banca Examinadora:

Prof. Dr. Nestor Felipe Caticha Alfonso (IF-USP)

Profa. Dra. Carmem Pimentel Cintra do Prado (IF-USP)

Prof. Dr. Rogério Rosenfeld (IFT-UNESP)

Prof. Dr. Marco Aurélio Pires Idiart (UFRGS)

Prof. Dr. Renato Vicente (EACH-USP)

SBI-IFUSP

305M810T4409

Prof. Celso Luiz Lima
Presidente da Comissão de Pós-Graduação

SÃO PAULO

2005

FICHA CATALOGRÁFICA
Preparada pelo Serviço de Biblioteca e Informação
do Instituto de Física da Universidade de São Paulo

Alamino, Roberto Castro

Aprendizado em Modelos de Markov com Variáveis de Estado Escondidas. São Paulo, 2005.

Tese (Doutoramento) - Universidade de São Paulo
Instituto de Física – Depto de Física Geral

Orientador: Prof. Dr. Nestor Caticha
Área de Concentração: Física

Unitermos:

1. Mecânica Estatística Clássica-aprendizado em máquinas;
2. Mecânica Estatística Clássica-Modelos Markovianos;
3. Inferência Bayesiana-Inferência estatística.

USP/IF/SBI-082/2005

Sumário

1	Introdução	1
1.1	Visão Geral	1
1.2	Organização da Tese	2
1.3	Convenções	4
2	Fundamentos	5
2.1	Complexidade Computacional	5
2.1.1	Introdução	5
2.1.2	Eficiência Computacional	6
2.1.3	Classes de Complexidade	7
2.2	Processos Markovianos	9
2.2.1	Processos Markovianos Discretos	9
2.2.2	Processos Markovianos Contínuos	12
2.3	Inferência Estatística	14
2.3.1	Introdução	14
2.3.2	Máxima Verossimilhança	15
2.3.3	O Algoritmo EM	16
2.3.4	Inferência Bayesiana	17
2.4	Modelos Gráficos	20
2.4.1	Introdução	20
2.4.2	Modelos Gráficos Direcionados	21
2.4.3	Modelos Gráficos Não-Direcionados	22
2.4.4	Inferência	24
2.5	Aprendizado em Máquinas	24
2.5.1	O Cenário de Aprendizado	26
2.5.2	Algoritmos <i>Off-line</i> e <i>On-line</i>	27
2.5.3	Performance e Generalização	28

2.5.4	O Algoritmo Bayesiano <i>On-line</i>	29
3	Modelos de Markov com Variáveis de Estado Escondidas	32
3.1	Teoria	32
3.2	Algoritmos	36
3.3	Aplicações	41
3.3.1	Bioinformática	41
3.3.2	Econometria	43
3.3.3	Meteorologia	43
3.3.4	Música	44
3.3.5	Reconhecimento de Fala	44
4	Aprendizado Não-Bayesiano	45
4.1	Distância entre HMMs	46
4.2	Simulações	50
4.3	Fórmulas de Reestimação de Baum-Welch	51
4.3.1	Algoritmo	51
4.3.2	Simulações	54
4.4	Método de Baum-Welch <i>On-line</i>	58
4.4.1	Algoritmo	58
4.4.2	Simulações	61
4.5	Algoritmo de Baldi-Chauvin	67
4.5.1	Algoritmo	67
4.5.2	Relação entre BC e BWO	72
4.5.3	Simulações	73
5	Aprendizado Bayesiano	80
5.1	Algoritmo Bayesiano <i>On-line</i>	80
5.1.1	Algoritmo	80
5.1.2	Simulações	84
5.2	Posterior Média	85
5.2.1	Algoritmo	85
5.2.2	Simulações	86
5.3	Estudo Teórico: A Moeda Simples	92
5.3.1	BKL	97
5.3.2	PM	98
5.4	Conceitos Variáveis	98

5.5	Aplicação : Previsão de Índices	100
5.6	Aprendizado e Quebra de Simetria	106
6	Conclusões e Perspectivas	111
6.1	Conclusões	111
6.1.1	Distâncias entre HMMs	111
6.1.2	Algoritmos	112
6.2	Perspectivas	112
6.2.1	Estudos Teóricos	113
6.2.2	Aplicações	114
A	O Método de Descida pelo Gradiente	115
B	Distribuições de Dirichlet	116
B.1	Definições	116
B.2	Normalização	116
B.3	Momentos	119
B.4	Obtenção pelo Princípio da Máxima Entropia	121
B.5	Mistura de Dirichlets	122
C	O Sistema Digama	123
D	Projeção de uma Distribuição em uma Família Paramétrica	124
D.1	Obtenção da Família Paramétrica via Máxima Entropia	124
D.2	Projeção de uma Distribuição na Família Paramétrica	125
E	Notação	127
E.1	Abreviações	127
E.2	Símbolos	127

Lista de Figuras

2.1	O problema $P \times NP$: A figura <i>a</i> mostra o caso em que existem 3 classes e a figura <i>b</i> o caso onde apenas duas existem.	9
2.2	Grafo de Transição de uma CMH: Os círculos denotam os estados com os respectivos nomes e as setas as possíveis transições com os valores das probabilidades.	11
2.3	Distribuição a Priori: Exemplo de distribuição a priori da variável <i>A</i> no intervalo real $[a, b]$ tal que os valores mais prováveis de <i>A</i> se encontram perto do ponto <i>a</i>	17
2.4	Modelos Gráficos: Modelo gráfico representando relações de interdependência entre 4 variáveis aleatórias diferentes.	20
2.5	Modelos Gráficos Direcionados: Modelo gráfico direcionado representando os 4 primeiros passos de um processo markoviano de ordem 2. . . .	22
2.6	Modelos Gráficos Não-Direcionados: Modelo gráfico representando o modelo de Ising em uma rede 3×3	23
2.7	Percéptron: Representação gráfica de um perceptron.	25
2.8	Percéptron: Representação gráfica de um perceptron com uma camada intermediária (chamada de camada escondida).	26
2.9	Algoritmo Bayesiano On-line: a) 1o. passo: quando um novo dado é incorporado, a distribuição antiga é modificada de tal modo que sai do espaço formado por sua família paramétrica. b) 2o. passo: a distribuição é então projetada de volta no espaço das distribuições paramétricas originais através da minimização da distância de Kullback-Leibler. . . .	30
3.1	HMM: Representação gráfica de um HMM.	33
3.2	Topologias de um HMM: a) modelo <i>left-right</i> ; b) modelo de <i>loop</i>	35

- 4.1 **Algoritmo de Baum-Welch:** a) Gráfico mostrando a distância euclidiana entre o professor e o aluno após a aplicação do algoritmo para uma sequência de T símbolos fazendo a média em 110 professores com 2 estados escondidos e 3 estados observados. A linha tracejada mostra o limite do aprendizado; b) os mesmos dados do gráfico *a* subtraindo-se o valor limite 0.3, porém em log-log. A linha tracejada indica o ajuste de uma lei de potência $(d - 0.3) = \alpha T^{-\xi}$ para os últimos 100 pontos com resultados $\alpha = 0.25$ e $\xi = 0.69$ 56
- 4.2 **Algoritmo de Baum-Welch:** a) O gráfico ilustra o efeito de aumentar o número de estados escondidos do HMM; b) mesmo gráfico em escala log-log. 57
- 4.3 **Algoritmo de Baum-Welch:** a) O gráfico mostra a comparação entre duas simulações com alunos iniciais diferentes do algoritmo de Baum-Welch: os círculos mostram o resultado obtido com um aluno inicial simétrico e os pontos com as respectivas barras de erro mostram a média em 50 alunos iniciais aleatórios; b) o mesmo gráfico em escala log-log. . . 59
- 4.4 **Método de Baum-Welch On-line:** a) Gráfico mostrando a distância de Kullback-Leibler entre o professor e o aluno após a aplicação do algoritmo para 4000 sequências de 2 símbolos fazendo a média em 500 professores com 2 estados escondidos e 3 estados observados. Os números próximos a cada linha indicam o valor de η_{BW} utilizado. b) O mesmo que o gráfico *a*, porém em log-log. 62
- 4.5 **Método de Baum-Welch On-line:** a) Gráfico mostrando a distância Euclidiana entre o professor e o aluno após a aplicação do algoritmo para 4000 sequências de 2 símbolos fazendo a média em 500 professores com 2 estados escondidos e 3 estados observados. Os números próximos a cada linha indicam o valor de η_{BW} utilizado. b) O mesmo que o gráfico *a*, porém em log-log. 63
- 4.6 **Método de Baum-Welch On-line:** a) Gráfico comparando a curva de aprendizado de um aluno inicial simétrico (linha tracejada) com a média de 60 alunos aleatórios (linha contínua); b) o mesmo gráfico em escala log-log. 64
- 4.7 **Método de Baum-Welch On-line:** Comparação entre o algoritmo BW e o algoritmo BWO adaptado para aprender uma sequência de tamanho T quebrando-a em subsequências menores com tamanho indicado junto às curvas correspondentes. 65

- 4.8 **Método de Baum-Welch *On-line*:** Valor médio da distância entre aluno e professor nos últimos 100 pontos das curvas de aprendizado de BWO para seqüências únicas com tamanho de 2 até 100 símbolos em função do tamanho das subsequências. 66
- 4.9 **Algoritmo de Baldi-Chauvin:** a) Gráfico mostrando a distância de Kullback-Leibler entre o professor e o aluno após a aplicação do algoritmo para 4000 seqüências de 2 símbolos fazendo a média em 200 professores com 2 estados escondidos e 3 estados observados. Os números próximos a cada linha indicam o valor de η_{BC} utilizado. b) O mesmo que o gráfico *a*, porém em log-log e para 10000 seqüências. 74
- 4.10 **Algoritmo de Baldi-Chauvin:** a) Gráfico mostrando a distância Euclideana entre o professor e o aluno após a aplicação do algoritmo para 4000 seqüências de 2 símbolos fazendo a média em 200 professores com 2 estados escondidos e 3 estados observados. Os números próximos a cada linha indicam o valor de η_{BC} utilizado. b) O mesmo que o gráfico *a*, porém em log-log e para 10000 seqüências. 75
- 4.11 **Algoritmo de Baldi-Chauvin:** Gráfico mostrando a comparação entre as curvas de aprendizado para o algoritmo BC (linha contínua) e para o algoritmo BWO com a taxa de aprendizado estimada (linha tracejada) e com a melhor taxa obtida por tentativa e erro (linha pontilhada). As curvas são traçadas para $\eta_{BC} = 1$ fixo e para 3 valores distintos de λ , de baixo para cima, 0.01, 0.001 e 0.0001. 76
- 4.12 **Algoritmo de Baldi-Chauvin:** Gráfico mostrando a distância euclideana entre aluno e professor para o algoritmo BC aplicado ao aprendizado de uma única seqüência de tamanho T utilizando cinco tamanhos diferentes de subsequências indicadas ao lado de cada curva comparando o resultado com o obtido utilizando o algoritmo BW. 77
- 4.13 **Algoritmo de Baldi-Chauvin:** Gráfico mostrando a distância euclideana entre aluno e professor para o algoritmo BC aplicado ao aprendizado de uma única seqüência de tamanho T utilizando subsequências de tamanho 6 para cinco diferentes valores da taxa de aprendizado η_{BC} indicados ao lado de cada curva. 78

- 5.1 **Posterior Média:** a) Gráfico mostrando a distância de Kullback-Leibler entre o professor e o aluno após a aplicação do algoritmo para 10000 seqüências de 2 símbolos fazendo a média em 500 professores com 2 estados escondidos e 3 estados observados. b) O mesmo que o gráfico *a*, porém em log-log. 87
- 5.2 **Posterior Média:** a) Gráfico mostrando a distância Euclideana entre o professor e o aluno após a aplicação do algoritmo para 10000 seqüências de 2 símbolos fazendo a média em 500 professores com 2 estados escondidos e 3 estados observados. b) O mesmo que o gráfico *a*, porém em log-log. 88
- 5.3 **Posterior Média:** a) A curva superior representa o valor da variância de π ao longo do aprendizado, as duas curvas inferiores, as variâncias de A . b) Variâncias de B ao longo do aprendizado. 90
- 5.4 **Posterior Média:** O gráfico log-log acima mostra a distância d_{KL} em função do número de seqüências aprendidas para cinco valores iniciais de u : 10^{-4} , 0.5, 1.0, 2.0 e 20.0. 91
- 5.5 **Posterior Média:** a) Comparação entre PM e BC. BC foi rodado com $\eta_{BC} = 0.5$ e para 5 diferentes valores de λ indicados junto às respectivas curvas. b) Comparação entre PM e BWO. BWO foi rodado para 4 diferentes valores de η_{BW} indicados junto às respectivas curvas. Para os dois gráficos: $n = 2$, $m = 3$, $T = 2$ e $N_\tau = 500$ e a linha inferior corresponde à curva de aprendizado de PM. 93
- 5.6 **Posterior Média:** a) Comparação entre PM e BC. BC foi rodado com $\eta_{BC} = 0.5$ e para 5 diferentes valores de λ indicados junto às respectivas curvas. Escalas em log-log. b) Comparação entre PM e BWO. BWO foi rodado para 4 diferentes valores de η_{BW} indicados junto às respectivas curvas. Para os dois gráficos: $n = 2$, $m = 3$, $T = 2$ e $N_\tau = 500$ e a linha inferior corresponde à curva de aprendizado de PM. Escalas em log-log. 94
- 5.7 **Posterior Média:** a) Gráfico comparativo da performance de PM versus BKL. Ambos foram rodados para os seguintes valores: $n = 2$, $m = 3$, $T = 2$ e $N_\tau = 150$, para um total de 300 seqüências. Os pequenos círculos mostram o resultado de BKL, enquanto a linha tracejada mostra o resultado de PM; b) mesmo gráfico em escala log-log. 95

5.8	Posterior Média: a) Gráfico mostrando as curvas de aprendizado para uma situação onde há apenas uma única sequência de tamanho T a ser aprendida dividida em diferentes tamanhos de subsequências indicados na legenda; b) mesmo gráfico em escala log-log.	96
5.9	Teste de Wisconsin para HMMs: Gráfico comparativo da performance de PM versus BC. A linha contínua representa PM e a linha tracejada representa BC. O professor é mudado aleatoriamente a cada 500 sequências. 100	
5.10	Mudança Aleatória Contínua: Gráfico comparativo da performance de PM versus BC. A linha contínua representa PM e a linha tracejada representa BC. O professor sofre uma mudança aleatória pequena a cada sequência.	101
5.11	Previsão do Ibovespa: a) Gráfico comparativo da performance de 4 diferentes estratégias: apostando aleatoriamente (Aleat.), apostando de acordo com o resultado do dia anterior (Dia Ant.), e usando os HMMs treinados pelos algoritmos PM e BC com subsequências de tamanho 2. b) Gráfico mostrando a evolução no tempo do índice Ibovespa deflacionado. 105	
5.12	Quebra de Simetria: Gráfico mostrando a distância entre aluno e professor e os valores das entradas dos parâmetros do HMM aluno em função do número de sequências aprendidas para o algoritmo BC.	107
5.13	Quebra de Simetria: Gráfico mostrando a distância entre aluno e professor e os valores das entradas dos parâmetros do HMM aluno em função do número de sequências aprendidas para o algoritmo PM.	108
5.14	Quebra de Simetria: Gráfico mostrando a distância entre aluno e professor e os valores das entradas dos parâmetros do HMM aluno em função do número de sequências aprendidas para o algoritmo BWO.	109
B.1	Normalização da Distribuição de Dirichlet: Caminho de integração no plano complexo utilizado para o caso $k > 0$	118
B.2	Normalização da Distribuição de Dirichlet: Caminho de integração no plano complexo utilizado para o caso $k < 0$	119

Resumo

Neste trabalho estudamos o aprendizado em uma classe específica de modelos probabilísticos conhecidos como **modelos de Markov com variáveis de estado escondidas** (em inglês, *Hidden Markov Models* ou *HMMs*). Primeiramente discutimos sua teoria básica e em seguida fazemos um estudo detalhado do comportamento de cinco diferentes algoritmos de aprendizado, dois deles já conhecidos na literatura e os outros três propostos por nós neste trabalho.

Os cinco algoritmos estão descritos abaixo e são estudados na sequência apresentada:

- **Algoritmo de Baum-Welch (BW):** consiste em um célebre algoritmo *off-line* obtido através da aplicação do algoritmo EM ao caso particular dos HMMs. Na literatura, é comum referir-se a ele pelo nome de *Fórmulas de Reestimação de Baum-Welch*.
- **Algoritmo de Baum-Welch On-line (BWO):** versão *on-line* de BW proposta por nós.
- **Algoritmo de Baldi-Chauvin (BC):** algoritmo *on-line* proposto por Baldi e Chauvin em [5] onde uma representação do tipo *softmax* é utilizada para as probabilidades dos HMMs e cujo objetivo é, a cada passo de iteração, maximizar a verossimilhança do modelo.
- **Algoritmo Bayesiano On-line (BKL):** algoritmo desenvolvido por nós baseado numa proposta de Oppér [74], onde, após a atualização da distribuição de probabilidades do modelo a cada novo dado, projeta-se a densidade obtida em uma família paramétrica de distribuições tratáveis minimizando-se a distância de Kullback-Leibler entre as duas.
- **Algoritmo Posterior Média (PM):** uma simplificação de BKL onde a projeção após a atualização é feita na distribuição posterior média.

Para cada um dos algoritmos acima, obtemos curvas de aprendizado através de simulações onde utilizamos duas medidas distintas de erro de generalização: a distância de Kullback-Leibler (d_{KL}) e a distância euclidiana (d_E).

Com exceção do algoritmo BW, que só pode ser utilizado em situações de aprendizado *off-line*, estudamos para todos os outros algoritmos as curvas de aprendizado tanto para a situação *on-line* quanto para a *off-line*. Comparamos as performances dos algoritmos entre si e discutimos os resultados obtidos mostrando que, apesar de um tempo de computação maior, o algoritmo bayesiano PM, proposto por nós, é superior aos outros algoritmos não-bayesianos quanto à generalização em situações de aprendizado estáticas e possui uma performance muito próxima do algoritmo bayesiano BKL.

Fazemos, também, uma comparação entre os algoritmos PM e BC em situações de aprendizado variáveis com o tempo, com dados gerados artificialmente e em uma situação com dados reais, porém com um cenário simplificado, onde os utilizamos para prever o comportamento do índice da bolsa de valores de São Paulo (IBOVESPA), mostrando que, embora necessitem de um período longo de aprendizado, após essa fase inicial as previsões obtidas por esses algoritmos são surpreendentemente boas.

Por fim, apresentamos uma discussão sobre aprendizado e quebra de simetria baseada nos estudos feitos.

Abstract

In this work we study learning in a specific class of probabilistic models known as **Hidden Markov Models (HMMs)**. First we discuss its basic theory and after we make a detailed study of the behavior of five different learning algorithms, two of them already known in the literature and the other three proposed by us in this work.

The five algorithms are described below in the sequence they are presented in the thesis:

- **Baum-Welch Algorithm(BW):** consists of a renowned offline algorithm obtained by applying the EM-algorithm to the particular case of HMMs. Through the literature it is common to refer to it by the name *Baum-Welch Reestimation Formulas*.
- **Baum-Welch Online Algorithm (BWO):** online version of BW proposed by us.
- **Baldi-Chauvin Algorithm (BC):** online algorithm proposed by Baldi and Chauvin in [5] where a softmax representation for the probabilities of the HMMs is used and where the aim is to maximize the model likelihood at each iteration step.
- **Online Bayesian Algorithm (BKL):** an algorithm developed by us based on the work of Oppé [74] where, after updating the probability distribution of the model with each new data, the obtained density is projected into a parametric family of tractable distributions minimizing the Kullback-Leibler distance between both.
- **Mean Posterior Algorithm (PM):** a simplification of BKL where the projection after the update is made on the mean posterior distribution.

For each one of the above algorithms, we obtain learning curves by means of simulations where we use two distinct measures of generalization error: the Kullback-Leibler distance (d_{KL}) and the euclidean distance (d_E).

With exception of the BW algorithm, which can be used only in offline learning situations, we study for all the other algorithms the learning curves for both learning situations:

online and offline. We compare the performance of the algorithms with one another and discuss the results showing that, besides its larger computation time, the bayesian algorithm PM, proposed by us, is superior to the other non-bayesian algorithms with respect to the generalization in static learning situations and that it has a performance that is very close to the bayesian algorithm BKL.

We also make a comparison between algorithms PM and BC in learning situations that change with time using artificially generated data and in one situation with real data, with a simplified scenario, where we use them to predict the behavior of the São Paulo Stock Market Index (IBOVESPA) showing that, although they need a large learning period, after that initial phase the predictions obtained by both algorithms are surprisingly good.

Finally, we present a discussion about learning and symmetry breaking based on the presented studies.

Prefácio

Este trabalho teve início com a idéia do Prof. Caticha de que HMMs talvez pudessem ter alguma relação com procedimentos de renormalização. A hipótese de que configurações renormalizadas de redes de *spin* poderiam ser “emitidas” pelas configurações originais da mesma maneira que estados observados são emitidos pelos estados escondidos foi levantada e nos propusemos a examiná-la. Além disso, surgiu-nos a idéia de tentar tratar os algoritmos existentes para aprendizado em HMMs através de métodos padrões de Mecânica Estatística, principalmente pelo Método de Réplicas, a exemplo do que foi feito para percéptrons com grande sucesso.

Depois de estudarmos essas possibilidades durante um longo período de tempo, ficou claro para nós que a complexidade do modelo era tal que precisaríamos estudar muito mais a fundo tanto o modelo quanto os algoritmos existentes para tentar extrair deles as informações que procurávamos. Durante os estudos dos algoritmos, surgiu a idéia de que poderíamos adaptar o algoritmo bayesiano *on-line* para percéptrons desenvolvido pelo Prof. Oppen para HMMs e resolvemos dedicar tempo de nossa pesquisa para esse fim. Como é comum em ciência, a tarefa se mostrou muito mais complicada do que esperávamos. As particularidades dos HMMs fizeram com que percorrêssemos um longo caminho de tentativas infrutíferas, que nos levaram ao estudo de técnicas computacionais avançadas de otimização convexa, como o método da barreira [12], e de solução de equações matriciais acopladas, entre elas, equações algébricas de Riccati e Sylvester. No final, todo o estudo dessas técnicas e sua aplicação ao algoritmo procurado nos mostrou o caminho para a adaptação correta do algoritmo ao caso dos HMMs. Dessa forma conseguimos obter o algoritmo BKL que, no entanto, mostrou-se excessivamente custoso computacionalmente, a ponto de ter pouca utilidade em situações práticas. Novamente, após uma série de tentativas sem sucesso, conseguimos finalmente encontrar uma simplificação de BKL na forma do algoritmo PM. Esse processo, relatado de forma bastante resumida, foi fruto de muito tempo de estudo, trabalho, computação e muitas discussões.

Um outro tópico estudado durante este trabalho não foi incluído nesta tese. A razão fundamental para isso foi que ainda não conseguimos extrair nenhuma conclusão sólida

dos estudos feitos. Esse estudo foi originário do estágio de três meses que eu, juntamente com Evaldo Oliveira, fiz junto ao Grupo de Pesquisa sobre Computação Neural (NCRG) da Universidade de Aston, na Inglaterra, sob a supervisão do Prof. Manfred Oppel. Durante este estágio, tivemos a idéia de estudar o comportamento assintótico do algoritmo bayesiano *on-line* para um perceptron linear através do método de réplicas utilizando uma distribuição posterior efetiva que representasse as equações de atualização do algoritmo. Com a supervisão entusiasmada do Prof. Oppel, começamos a fazer os cálculos na Inglaterra e, de volta ao Brasil, demos continuidade a esse trabalho até obtermos o comportamento assintótico, mas não conseguimos encontrar o valor correspondente ao obtido através de simulações. Mesmo com nossa volta ao Brasil, a contribuição do Prof. Oppel continuou presente através de *e-mails* e continuamos revisando os cálculos feitos, uma vez que eles envolvem várias aproximações sutis onde podem haver falhas. Além disso, estamos estudando a possibilidade de aproximações alternativas para as variáveis envolvidas de tal forma que possamos reproduzir as curvas experimentais.

Roberto Castro Alamino,
São Paulo, 10 de novembro de 2005.

Agradecimentos

Agradeço à minha família com a qual eu sempre posso contar. À minha noiva que soube me compreender nos momentos em que mais precisei. Ao Prof. Nestor Caticha pela sua orientação e ajuda. A Evaldo Oliveira pela colaboração que desenvolvemos em nossos trabalhos. Ao Prof. Manfred Oppen por ter me recebido de forma tão hospitaleira durante o estágio na Inglaterra juntamente com outros professores, alunos e funcionários do NCRG na Universidade de Aston. Aos membros da banca examinadora, cujas críticas e sugestões contribuíram enormemente para a melhora do formato final deste trabalho.

Agradeço também aos inúmeros amigos, colegas, professores e funcionários aqui no Brasil que compartilharam este tempo e me ajudaram ao longo deste trabalho cujos nomes são tão numerosos que um capítulo inteiro dessa tese não daria conta de todos eles e que, mesmo que me propusesse a escrevê-los, com certeza acabaria por esquecer alguns injustamente. Dessa forma, agradeço a todos de uma só vez sem correr o risco de deixar alguém sem os devidos méritos.

Agradeço, por fim, à FAPESP pelo financiamento dos 4 primeiros anos deste projeto.

Capítulo 1

Introdução

1.1 Visão Geral

Pode-se considerar que o estudo de sistemas artificiais capazes de aprendizado, uma área da ciência hoje chamada de Aprendizado em Máquinas (*Machine Learning*), tem como marco inicial o primeiro modelo simplificado de um neurônio introduzido por McCulloch e Pitts em 1943 [62]. A partir desse trabalho, iniciou-se o estudo de redes neurais artificiais (RNAs), compostas pela união de vários neurônios simplificados conectados entre si de forma a imitar a maneira como os neurônios reais se conectam no cérebro dos seres vivos. Em 1958, Rosenblatt introduziu o célebre modelo do perceptron (ver seção 2.5), uma RNA organizada em camadas que se tornou o principal modelo estudado nessa área [87].

O estudo das RNAs chamou a atenção da comunidade de Mecânica Estatística devido à possibilidade de mapeamento dessas redes em redes de spin (como o modelo de Ising, por exemplo), o que tornou viável o uso de técnicas desenvolvidas nessa área para a análise e compreensão desses modelos [25, 40].

Embora tenham uma gama de aplicações muito grande, sendo usadas em várias áreas de ciência e tecnologia, avanços teóricos indicam que redes neurais artificiais (RNAs) inspiradas em redes biológicas podem ter um desempenho inferior a outros sistemas adaptativos de processamento de informação em determinadas situações.

Devido a isso, surgiu um grande interesse no estudo de estratégias de processamento de informação alternativas, em particular daquelas que se utilizam de métodos bayesianos para realizar inferências devido à superioridade destes quanto à eficiência na generalização. Uma área cujo interesse tem aumentado cada vez mais, por exemplo, é a de modelagem através de técnicas conhecidas como **Processos Gaussianos** (*Gaussian Processes*

ou GPs) [111] e **Máquinas de Vetores de Suporte** (*Support Vector Machines* ou SVMs) [100]. Estas ferramentas pertencem a um grupo de técnicas baseadas em **métodos de núcleo** (*kernel methods*) [2, 69] que, em geral, apresentam grande flexibilidade por lidar com representações internas de alta dimensionalidade, incluindo casos com dimensão infinita. Mesmo que essas representações em dimensões altas possam trazer complicações na sua implementação prática, tais complicações não são, em princípio, intransponíveis, o que justifica o alto interesse nesses modelos.

Além de GPs e SVMs, também há um interesse crescente nos modelos Markovianos para o tratamento de dados seqüenciais, em particular, nos chamados **Modelos de Markov com Variáveis de Estado Escondidas** (*Hidden Markov Models* ou HMMs) [8, 23], que são o principal objeto de estudo desta tese e que podem ser considerados como casos particulares de modelos mais gerais denominados **Modelos Gráficos** [44, 70].

Os HMMs são modelos de aprendizado bastante conhecidos na comunidade de inteligência artificial, engenharia e matemática, principalmente pelas suas aplicações em reconhecimento de fala [31, 82, 84] e modelagem de cadeias de DNA e estrutura de proteínas [6, 24, 39, 53, 54], esta última com um crescente interesse tecnológico e econômico, mas são relativamente desconhecidos pelos físicos, o que pode ser observado em [14] onde, embora uma extensa bibliografia seja apresentada, apenas um trabalho em revistas de física aparece. Algumas aplicações desses modelos são feitas em conjunto com RNAs, estas últimas podem ser pré-processadores ou ainda trabalhar em conjunto com o HMM para extrair dependências não-lineares dos dados, mas em grande parte das aplicações esta interdependência não se faz necessária.

Obviamente o conjunto de técnicas citadas acima não esgota o repertório de técnicas alternativas às RNAs inspiradas em modelos biológicos. Existe uma quantidade enorme de técnicas de reconhecimento e classificação de padrões [90] que, em certos casos, se confundem com os modelos de RNAs, sendo formas de representação diferentes mas equivalentes. No entanto, em muitos outros casos, elas são realmente diferentes das RNAs e não podem ser mapeadas nestas de forma alguma.

1.2 Organização da Tese

Nosso intuito, nesta tese, é estudar e comparar o comportamento dos HMMs quando treinados por diferentes algoritmos em situações que envolvem aprendizado. Para tal fim, organizamos a tese na seqüência de capítulos detalhada a seguir.

No capítulo 2, apresentamos uma introdução teórica aos principais conceitos utilizados nesta tese. O primeiro conceito a ser introduzido é o de **Complexidade Computa-**

cional, na seção 2.1. Embora algumas informações nessa seção estejam fora do escopo principal desta tese (como o problema $P \neq NP$), os pontos mais importantes são a grande diferença entre algoritmos polinomiais e não-polinomiais e a invariância da complexidade computacional de um algoritmo quando calculada com base em diferentes operações. A seção seguinte (seção 2.2) fala sobre **Processos Markovianos**, onde encontram-se as definições e notações básicas. A sua importância no estudo dos HMMs é evidente. A seção sobre **Inferência Estatística** (seção 2.3) é uma seção extremamente importante na qual é explicado o conceito de inferência em modelos estatísticos de dados, tanto tradicional como bayesiana, na qual são explicadas as técnicas mais utilizadas para seu uso. A seção 2.4 apresenta os chamados **Modelos Gráficos** que representam uma maneira unificada de se entender vários modelos diferentes utilizados para inferência estatística sob a perspectiva da inferência bayesiana. Os HMMs podem ser vistos como casos particulares desses modelos. Por fim, a seção 2.5 contém, de forma compacta, os conceitos básicos da teoria geral do **Aprendizado em Máquinas**. Uma subseção bastante importante para o nosso trabalho é a referente ao Algoritmo Bayesiano *On-line*, que é o ponto de partida para dois dos algoritmos que desenvolvemos e estudamos nesta tese.

No capítulo 3, finalmente chegamos à teoria geral dos HMMs. A seção 3.1 explica a teoria básica desses modelos e fixa a notação que será utilizada; a seção 3.2 apresenta os três problemas considerados como os “problemas clássicos” dos HMMs juntamente com os algoritmos necessários para sua solução e a seção 3.3 descreve as aplicações deste tipo de modelo de aprendizado em algumas áreas da ciência e da tecnologia.

O capítulo 4 aborda a teoria de aprendizado em HMMs e apresenta os nossos estudos referentes aos algoritmos de aprendizado não-bayesianos. Na seção 4.1 descrevemos as duas distâncias entre HMMs que farão o papel do **erro de generalização** em nossos estudos, indicando quão distante um HMM aluno está do HMM professor. A seção 4.2 contém uma explicação sobre a forma geral como serão realizadas as simulações ao longo do trabalho. Na seção 4.3 descrevemos as **Fórmulas de Reestimação de Baum-Welch**, que formam o algoritmo *off-line* padrão utilizado no aprendizado em HMMs. Nela também são apresentados estudos referentes ao aprendizado nesse algoritmo realizados por nós. Na seção 4.4 sugerimos e estudamos em diferentes situações de aprendizado uma forma *on-line* das fórmulas de reestimação de Baum-Welch. Na seção 4.5 deduzimos de forma detalhada o algoritmo de Baldi-Chauvin (BC) e estuda o comportamento de suas curvas de aprendizado.

No capítulo 5 estudamos os algoritmos bayesianos que desenvolvemos em nosso trabalho. Na seção 5.1 derivamos, baseados em um algoritmo criado por Oppor [74], um algoritmo bayesiano *on-line* (BKL) para HMMs. Na seção 5.2 apresentamos uma sim-

plificação do algoritmo BKL que consome menos tempo de computação e possui um desempenho muito semelhante. Estudamos, então, o comportamento desse algoritmo, que chamaremos de **Posterior Média (PM)**, comparando-o com os outros algoritmos já apresentados. A título de ilustração, a seção 5.3 apresenta um exemplo teórico da aplicação dos algoritmos BKL e PM à análise de lançamentos de uma moeda simples. A seção 5.4 compara os algoritmos PM e BC em situações teóricas onde o professor varia com o tempo e em seguida, na seção 5.5, eles são utilizados em uma aplicação real, porém simplificada, de previsão do índice da bolsa de valores de São Paulo (IBOVESPA). A seção 5.6, a última deste capítulo, apresenta uma discussão sobre a relação entre aprendizado e quebra de simetria.

Finalmente, o capítulo 6 apresenta as conclusões obtidas a partir do nosso trabalho e aponta questões não resolvidas e possíveis desenvolvimentos futuros.

1.3 Convenções

Procuraremos manter uma notação coerente durante toda a tese. Em geral, cada símbolo utilizado terá apenas um significado, embora algumas exceções possam ocorrer, as quais esperamos não causarem confusão por serem facilmente diferenciadas de acordo com o contexto no qual serão utilizadas.

Para facilitar a leitura, descreveremos a seguir as convenções mais importantes. A probabilidade ou densidade de probabilidade de um determinado evento A será indicada através de $\mathcal{P}(A)$. Note que $\mathcal{P}$ funciona como um operador cujo significado é “a probabilidade de” ou “a densidade de probabilidade de” e não uma forma funcional específica das variáveis entre parênteses. Usamos $\mathcal{P}(A|B)$ para indicar a probabilidade condicional do evento A dado o evento B .

Matrizes, em geral, serão indicadas por letras latinas maiúsculas e a notação A^T representa a matriz transposta da matriz A .

Uma sequência de variáveis $\{y_1, y_2, \dots, y_T\}$ será escrita de forma abreviada através da notação y_1^T , onde o índice inferior representa o índice da variável inicial da sequência e o superior o índice da variável final.

Ao longo da tese, à medida que se tornarem necessárias, outras serão introduzidas. No apêndice E encontram-se tabelas com as abreviações e notações usadas durante o trabalho.

Capítulo 2

Fundamentos

2.1 Complexidade Computacional

2.1.1 Introdução

Antes do advento dos computadores, resolver problemas de forma numérica era um procedimento extremamente complicado e viável apenas em casos simples. Mesmo assim, muitos algoritmos numéricos foram desenvolvidos, como o algoritmo para encontrar a raiz quadrada de um inteiro, o método de Newton para encontrar raízes de uma função e o método de Gauss para inversão de matrizes. Isso se deve à importância de se obter valores numéricos para problemas em inúmeras situações práticas.

Hoje, no entanto, até mesmo os computadores pessoais já conseguem realizar cálculos a velocidades da ordem de GHz, isto é, 10^9 operações por segundo e o recorde atual para supercomputadores é da ordem de 10THz, ou seja, 10^{13} operações por segundo. Dada tal velocidade, resoluções numéricas de problemas complicados, como cálculos de órbitas de corpos celestes, adquiriram velocidades que as tornaram não apenas viáveis mas indispensáveis em todas as áreas do desenvolvimento científico e tecnológico. Assim, o estudo dos algoritmos numéricos adquiriu uma importância tão grande quanto o dos algoritmos analíticos e sua análise deu origem a um ramo da matemática denominado **Complexidade Computacional** [42, 64].

É conveniente, para a discussão que faremos a seguir, diferenciar um determinado problema daquilo que chamaremos de suas **instâncias**. Uma instância de um problema é uma determinada configuração de seus parâmetros. Por exemplo, determinar a distância percorrida por um veículo dada sua velocidade e o tempo durante o qual ele se deslocou é um problema que depende dos parâmetros (v, t) , isto é, velocidade e tempo de deslo-

camento. Uma instância desse problema seria uma determinada configuração de valores como, por exemplo, $(v, t) = (5m/s, 7s)$. Um outro exemplo um pouco mais sofisticado seria o cálculo do estado fundamental de um vidro de *spin* dada uma hamiltoniana dependente de certos parâmetros. Cada conjunto de valores assumidos por esses parâmetros corresponderia a uma instância do problema.

De forma não rigorosa, podemos definir um **algoritmo** como um procedimento que, dada qualquer instância de um certo problema, produz, após um número finito de operações sobre os dados da instância, ou a solução ou a constatação de que ela não existe.

Quando usamos algoritmos computacionais para resolver problemas numericamente, uma das questões de interesse é saber qual a eficiência desse algoritmo, isto é, quanto ele irá demorar para resolver o problema. Para todo problema existe um algoritmo chamado de **trivial**: simplesmente testamos as candidatas a soluções uma a uma e verificamos qual a correta, caso nenhuma solucione o problema, concluímos que não existe solução. Para o exemplo que demos anteriormente sobre o cálculo da distância percorrida por um veículo, esse algoritmo equivaleria a testar, a partir do zero, todos os números para encontrar qual satisfaz a relação velocidade = distância/tempo. Mesmo nesse exemplo simples, o número de soluções a serem testadas é infinito e o algoritmo é infinitamente mais demorado do que a solução analítica, sendo esta extremamente simples.

Devido a situações como essa, é extremamente importante a análise da eficiência dos algoritmos computacionais e esse é o objetivo da área da complexidade computacional.

2.1.2 Eficiência Computacional

A noção de eficiência para um algoritmo depende de algumas outras definições preliminares. Definiremos, primeiramente, o **tamanho de uma instância** S_p de um determinado problema p , o que simbolizaremos por $|S_p|$. Define-se $|S_p|$ como o número de símbolos de um alfabeto dado necessários para representar formalmente a instância S_p . Simbolizaremos por $\theta_A(S_p)$ o número de operações necessárias para resolver a instância S_p com o algoritmo A .

Note que tanto o conceito de alfabeto utilizado quanto o de operação não estão precisamente definidos. Porém, os resultados da teoria são invariantes, dentro de certos limites, por mudanças de alfabeto, tipo de operação ou computador utilizado [32, 35], o que nos dá essa liberdade na definição de tais conceitos.

Sendo assim, definimos a **eficiência** g_A do algoritmo A com relação a um problema p como

$$g_A(n) = \max_{|S_p| \leq n} \theta_A(S_p), \quad (2.1)$$

isto é, o maior número de operações necessárias para solucionar qualquer instância S_p de tamanho igual ou menor a um dado número n com o algoritmo A . Portanto, a eficiência de um algoritmo considera a sua atuação no pior caso. Nem sempre o fato de um algoritmo ser superior ao outro no pior caso implica que ele será sempre superior, porém, é muito difícil estimar a eficiência de um algoritmo pelo seu comportamento médio já que é impraticável, e muitas vezes impossível, determinar a sua atuação para todas as instâncias possíveis do problema. Para o pior caso, por outro lado, podemos geralmente encontrar métodos de estimação. Recentemente, porém, métodos de mecânica estatística têm sido utilizados na análise da complexidade não somente no pior caso, mas também em casos típicos e na maneira como se dá a transição entre situações nas quais instâncias podem tipicamente ser resolvidas facilmente para situações nas quais tanto o caso típico quanto o pior caso são difíceis.[29, 51, 63, 64, 65, 66, 67, 109].

2.1.3 Classes de Complexidade

Definiremos a **classe de complexidade P** de problemas polinomiais como a classe que contém os problemas para os quais existe um algoritmo polinomial A para resolvê-los, ou seja, tal que $g_A(n)$ esteja limitado por um polinômio em n . Considera-se que os problemas contidos nessa classe são problemas eficientemente solúveis do ponto de vista computacional e muitas vezes, na literatura, costumam ser chamados simplesmente de **problemas fáceis**.

É importante notar que essa definição leva em conta os algoritmos já descobertos para a resolução de problemas, isto é, um problema não está em **P** enquanto não for descoberto um algoritmo polinomial para solucioná-lo, mas logo que um algoritmo qualquer polinomial seja descoberto, o problema passa automaticamente a fazer parte de **P**.

Definida dessa forma, a classe **P** possui certas propriedades que a tornam passível de tratamento matemático. Ela possui, por exemplo, invariância com respeito aos alfabetos, tipos de operações e computadores utilizados¹. Outra propriedade leva em conta o fato de que o conjunto dos polinômios é fechado por composição. Isto significa que se existe uma função polinomial que leva um problema p' em outro problema p que sabemos ser polinomial, então o problema p' também é polinomial.

Existe uma outra classe importante de problemas que podemos definir. Entre os pro-

¹Aqui cabe uma observação breve. Os computadores referidos nesse caso são computadores clássicos, isto é, que seguem as regras clássicas de lógica booleana. Nos últimos anos, surgiram especulações teóricas sobre a possibilidade de construção de computadores quânticos e talvez a classificação dos problemas em classes nesse tipo de computador difira da classificação para computadores clássicos [73].

blemas que não sabemos se pertencem ou não a P existe uma classe especial, que denominamos NP , composta pelos problemas para os quais dada uma solução possível, existe um algoritmo polinomial que testa sua validade. Os problemas pertencentes a essa classe são também chamados de **problemas difíceis**. Note que $P \subseteq NP$, pois uma vez que achamos um algoritmo polinomial para a solução de um problema ele mesmo confere se uma possível solução é válida.

O problema clássico da área da complexidade computacional é o de descobrir se $P = NP$. A princípio, isso parece um problema impraticável, já que precisaríamos conhecer todos os problemas NP e demonstrar que todos possuem um algoritmo de resolução polinomial. No entanto, existe um teorema, chamado de **teorema de Cook** [18], que diz que *todo o problema da classe NP pode ser reduzido polinomialmente ao problema de Satisfatibilidade*.

Descreveremos o problema da Satisfatibilidade (SAT). Suponha que tenhamos variáveis $x_1, \dots, x_n$ que representam proposições. Cada uma dessas variáveis pode assumir os valores V (verdadeiro) ou F (falso). Simbolizaremos a negação de x_i por $\bar{x}_i$. As operações booleanas usuais *AND* e *OR* serão simbolizadas, respectivamente, por $\wedge$ e $\vee$. Uma **cláusula** é definida como uma expressão da forma

$$C = x_{i_1} \vee x_{i_2} \vee \dots \vee x_{i_k}, \quad (2.2)$$

onde os índices $i_1, \dots, i_k$ são quaisquer dos índices que as variáveis x_i podem possuir. Uma **fórmula** é definida como uma expressão do tipo

$$\Phi = C_1 \wedge C_2 \wedge \dots \wedge C_l, \quad (2.3)$$

onde cada C_i representa uma cláusula. O problema SAT consiste em saber se existe ou não uma determinada configuração de valores das variáveis x_i de tal forma que uma dada fórmula Φ assumo o valor V .

Logo, se encontrarmos um algoritmo polinomial para resolver SAT, todos os problemas NP são resolvidos polinomialmente e, portanto, $P = NP$. Suponha, agora, que SAT possa ser reduzido polinomialmente a outro problema $p \in NP$. Nesse caso, se um algoritmo polinomial for encontrado para resolver p , todos os problemas NP se reduzem a P . Podemos, então, definir uma nova classe dentro de NP chamada de classe dos problemas NP -completos. Dizemos que um problema p será NP -completo se:

- (a) $p \in NP$ e
- (b) $\forall p' \in NP$ se reduz polinomialmente a p .

Figura 2.1: **O problema $P \times NP$:** A figura *a* mostra o caso em que existem 3 classes e a figura *b* o caso onde apenas duas existem.

Sendo assim, demonstrando-se que qualquer problema NP-completo está em P , prova-se que $P = NP$ ou, por outro lado, demonstrando-se que qualquer um deles não está em P , mostra-se que $P \neq NP$. Atualmente, a quantidade de problemas NP-completos conhecida é da ordem de centenas [64], porém, ainda não se conseguiu encontrar nenhum que seja redutível polinomialmente a um problema em P e também não se conseguiu provar para nenhum deles a não-existência de um algoritmo polinomial. Um ponto interessante é que, no momento, quase todos os problemas NP conhecidos ou foram reduzidos a P ou provou-se serem NP-completos, o que sugere que qualquer problema pertença somente a uma dessas duas classes, porém essa hipótese também não foi provada ou refutada até o momento. A figura 2.1 ilustra as duas situações possíveis.

2.2 Processos Markovianos

A maior parte desta seção é baseada no livro citado em [13].

2.2.1 Processos Markovianos Discretos

Uma sequência $\{q_t\}_{t \geq 1}$, $t \in \mathbb{N}$, de variáveis aleatórias com valores em um conjunto S é chamada de um **processo estocástico discreto com espaço de estados S** .

Considere o processo estocástico discreto finito $q_1^T \equiv \{q_1, q_2, \dots, q_T\}$. Usando

$$\mathcal{P}(AB) = \mathcal{P}(A|B) \mathcal{P}(B), \quad (2.4)$$

obtemos que a distribuição de probabilidades desse processo é dada por

$$\begin{aligned}\mathcal{P}(q_1^T) &= \mathcal{P}(q_T|q_1^{T-1})\mathcal{P}(q_1^{T-1}) \\ &= \mathcal{P}(q_1) \prod_{t=1}^{T-1} \mathcal{P}(q_{t+1}|q_1^t).\end{aligned}\quad (2.5)$$

Definição 2.1 Seja $\{q_t\}_{t \geq 1}$ um processo estocástico discreto com um estado de espaços contável S . Se para todo $t \geq 1$ e para todos os estados q_t

$$\mathcal{P}(q_{t+1}|q_1^t) = \mathcal{P}(q_{t+1}|q_{t-k+1}^t), \quad (2.6)$$

o processo é denominado **processo markoviano** ou **modelo de Markov de ordem k** . Isto significa que a probabilidade de q_t só depende dos k últimos estados da sequência.

Definição 2.2 Um processo markoviano de ordem 1, isto é, para o qual

$$\mathcal{P}(q_{t+1}|q_1^t) = \mathcal{P}(q_{t+1}|q_t), \quad (2.7)$$

para todo $t \geq 1$ e todos os $q_1, \dots, q_{t-1}$, é chamado de **cadeia de Markov**. Em particular, ele será chamado de **cadeia de Markov homogênea (CMH)** se, além disso, o lado direito da equação for independente de t .

Para uma cadeia de Markov finita, a distribuição de probabilidades é dada por

$$\mathcal{P}(q_1^T) = \mathcal{P}(q_1) \prod_{t=1}^{T-1} \mathcal{P}(q_{t+1}|q_t). \quad (2.8)$$

Seja o espaço de estados dado pelo conjunto $S = \{s_1, \dots, s_n\}$. A matriz $A(t)$, de tamanho $n \times n$, cujas entradas são dadas por

$$A_{ij}(t) = \mathcal{P}(q_{t+1} = s_j | q_t = s_i), \quad (2.9)$$

é denominada **matriz de transição** da cadeia de Markov. No caso de uma CMH, a matriz de transição não depende de t . A matriz de transição pode ser representada pelo seu **grafo de transição**, que é um grafo orientado cujos vértices são os estados de S , isto é, o vértice i está associado ao estado s_i . Esse grafo possui um elo orientado de i para j se, e somente se, $A_{ij} \neq 0$. Ao lado de cada elo escreve-se o valor de A_{ij} .

A figura 2.2 mostra o grafo de transição de uma CMH onde $S = \{\alpha, \beta, \gamma\}$ e cuja matriz de transição é

$$A = \begin{pmatrix} 1/2 & 1/2 & 0 \\ 3/4 & 0 & 1/4 \\ 1/2 & 0 & 1/2 \end{pmatrix}. \quad (2.10)$$

Figura 2.2: **Grafo de Transição de uma CMH:** Os círculos denotam os estados com os respectivos nomes e as setas as possíveis transições com os valores das probabilidades.

A distribuição de probabilidades

$$\pi_i(t) = \mathcal{P}(q_t = s_i), \quad (2.11)$$

forma um vetor coluna $\pi(t)$ que dá a distribuição de probabilidades da variável q_t do modelo. A distribuição $\pi(1)$ é denominada **distribuição inicial** da cadeia de Markov. Para uma CMH, temos

$$\begin{aligned}
 \pi_i(t+1) &= \mathcal{P}(q_{t+1} = s_i) \\
 &= \sum_{j=1}^n \mathcal{P}(q_{t+1} = s_i, q_t = s_j) \\
 &= \sum_{j=1}^n \mathcal{P}(q_{t+1} = s_i | q_t = s_j) \mathcal{P}(q_t = s_j) \\
 &= \sum_{j=1}^n A_{ji} \pi_j(t),
 \end{aligned} \quad (2.12)$$

ou seja

$$\pi(t+1) = A^T \pi(t), \quad (2.13)$$

onde A^T é a matriz transposta de A . Iterando esse resultado, obtemos

$$\pi(t) = (A^T)^{t-1} \pi(1), \quad (2.14)$$

o que mostra que a distribuição de probabilidades em qualquer instante depende apenas da distribuição inicial e da matriz de transição.

2.2.2 Processos Markovianos Contínuos

Na discussão que faremos a seguir, nos restringiremos somente ao caso de cadeias homogêneas. Seja S um espaço de estados contável e seja $\{q(t)\}_{t \geq 0}$ um processo estocástico com valores em S e indexado pelo conjunto dos reais não negativos $\mathbb{R}_+$. Então, temos a seguinte definição:

Definição 2.3 O processo estocástico $\{q(t)\}_{t \geq 0}$ com valores em um dado conjunto S é chamado **cadeia de Markov com tempo contínuo** ou **cadeia de Markov contínua** se para todo $s_i, s_j, s_{i_1}, \dots, s_{i_k} \in S$, todo $t, r \geq 0$ e todo $r_1, \dots, r_k \geq 0$ com $r_l \leq r$ para todo $l \in [1, k]$

$$\mathcal{P}(q(t+r) = s_j | q(r) = s_i, q(r_1) = s_{i_1}, \dots, q(r_k) = s_{i_k}) = \mathcal{P}(q(t+r) = s_j | q(r) = s_i), \quad (2.15)$$

desde que os dois lados estejam bem definidos.

As matrizes de transição $A(t)$ com $t \geq 0$ e com entradas

$$A_{ij}(t) = \mathcal{P}(q(t+r) = s_j | q(r) = s_i), \quad (2.16)$$

são matrizes estocásticas que formam um semigrupo $\mathcal{T}$ chamado de **semigrupo de transição** da cadeia de Markov contínua, isto é, dados $s, t \in \mathbb{R}_+$, temos que

- 1) $A(t) \in \mathcal{T}$,
- 2) $I \in \mathcal{T}$ (onde $I = A(0)$),
- 3) $A(t+s) = A(t)A(s)$ (**equação de Chapman-Kolmogorov**).

Se, além dessas três propriedades, tivermos que

$$\lim_{h \rightarrow 0^+} A(h) = A(0) = I, \quad (2.17)$$

onde a convergência é ponto a ponto e em cada entrada, então o semigrupo é dito contínuo na origem. A continuidade na origem implica a continuidade para qualquer tempo $t \geq 0$

$$\lim_{h \rightarrow 0} A(t+h) = A(t), \quad (2.18)$$

e assim dizemos simplesmente que o semigrupo é contínuo.

A distribuição de probabilidades no instante t é dada pela matriz coluna

$$\pi_i(t) = \mathcal{P}(q(t) = s_i), \quad (2.19)$$

e é obtida através de

$$\pi(t) = [A(t)]^T \pi(0). \quad (2.20)$$

A matriz

$$W = \lim_{h \rightarrow 0^+} \frac{A(h) - A(0)}{h}, \quad (2.21)$$

é chamada de **gerador infinitesimal** do semigrupo e suas entradas W_{ij} são denominadas de **características locais** do semigrupo. Um semigrupo é dito **estável** se os elementos diagonais de W forem finitos e **conservativo** se

$$W_{ii} = - \sum_{j(j \neq i)} W_{ij}. \quad (2.22)$$

O semigrupo de transição é estável para a maior parte dos problemas práticos e tem que ser conservativo, pois as matrizes estocásticas $A(h)$ devem satisfazer

$$\sum_j A_{ij}(h) = 1 \Rightarrow \frac{1 - A_{ii}(h)}{h} = \sum_{j(j \neq i)} \frac{A_{ij}(h)}{h}, \quad (2.23)$$

que no limite $h \rightarrow 0^+$ é equivalente à equação (2.22).

Note que se considerarmos h como um intervalo infinitesimal de tempo, as características locais W_{ij} de um semigrupo $\mathcal{T}$ estável e conservativo fornecem as probabilidades de transição por unidade de tempo do estado s_i para o estado s_j da cadeia de Markov e são conhecidas pelo nome de **taxas de transição** de s_i para s_j . Dessa forma, as cadeias de Markov contínuas podem ser consideradas como o limite contínuo das cadeias de Markov discretas quando o intervalo de tempo entre dois estados vai a zero.

Para todo $t, h \in \mathbb{R}_+$, podemos escrever

$$\frac{A(t+h) - A(t)}{h} = A(t) \frac{A(h) - I}{h} = \frac{A(h) - I}{h} A(t), \quad (2.24)$$

se o limite existe dos dois lados, o que sempre acontece quando o espaço de estados é finito, então temos o chamado **sistema diferencial de Kolmogorov**

$$\frac{d}{dt} A(t) = A(t) W = W A(t). \quad (2.25)$$

Se $\mathcal{T}$ é estável e conservativo, e se valer a condição

$$- \sum_i W_{ii} \pi_i(t) < \infty, \quad (2.26)$$

para todo $t \in \mathbb{R}_+$, então as matrizes $\pi(t)$ e W satisfazem a equação

$$\frac{d}{dt} \pi(t) = W^T \pi(t), \quad (2.27)$$

que é conhecida como **equação mestra**.

2.3 Inferência Estatística

2.3.1 Introdução

Inferência estatística, também chamada de **inferência probabilística**, consiste em estimar as propriedades de uma distribuição de probabilidades desconhecida através de um conjunto D_P composto de P dados por ela gerados. É comum aproximar a distribuição procurada restringindo-a a uma família de distribuições específica. Geralmente assume-se que as propriedades da distribuição são descritas por determinados parâmetros simbolizados coletivamente por θ , ou seja, que estamos tratando de uma **distribuição paramétrica**. Denominaremos o conjunto de todas as variáveis aleatórias dessa distribuição por $\mathcal{X} = \mathcal{Y} \cup \mathcal{Q}$, onde $\mathcal{Q} = \{q_1, \dots, q_N\}$ representa o conjunto das variáveis a cujos valores o observador não tem acesso, as quais denominaremos **variáveis escondidas**, enquanto $\mathcal{Y} = \{y_1, \dots, y_M\}$ representa as variáveis a cujos valores têm-se acesso, que chamaremos de **variáveis observadas**. Para os elementos de $\mathcal{X}$, sejam eles variáveis escondidas ou observadas, usaremos a notação geral $x_i, i = 1, \dots, N + M$.

Para a distribuição de probabilidades de $\mathcal{X}$, usaremos a notação $\mathcal{P}(\mathcal{X}|\theta)$, onde θ representa o conjunto de todos os parâmetros da distribuição de probabilidades em questão. Por exemplo, para um conjunto $\mathcal{X} = \{x_1, x_2, x_3\}$ com uma distribuição gaussiana nessas variáveis, temos

$$\mathcal{P}(\mathcal{X}|\theta) = \frac{1}{\sqrt{(2\pi)^3 \det C}} \exp \left[-\frac{1}{2} (x - \bar{x})^T C^{-1} (x - \bar{x}) \right], \quad (2.28)$$

onde x é o vetor

$$x \equiv \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}, \quad (2.29)$$

e os parâmetros da distribuição são

$$\theta = \{\bar{x}, C\}, \quad (2.30)$$

tais que

$$\bar{x} \equiv \begin{pmatrix} \bar{x}_1 \\ \bar{x}_2 \\ \bar{x}_3 \end{pmatrix}, \quad C = \begin{pmatrix} C_{11} & C_{12} & C_{13} \\ C_{21} & C_{22} & C_{23} \\ C_{31} & C_{32} & C_{33} \end{pmatrix}, \quad (2.31)$$

são, respectivamente, a média e a covariância da distribuição de probabilidades.

O conjunto de dados D_P é então uma sequência de P observações do conjunto de variáveis $\mathcal{Y}$, que são as variáveis passíveis de serem observadas, indexadas por números

naturais, o que simbolizaremos por

$$D_P = \{\mathcal{Y}^1, \mathcal{Y}^2, \dots, \mathcal{Y}^P\}. \quad (2.32)$$

Chama-se **verossimilhança** dos parâmetros θ dado o conjunto de dados D_P à função

$$\mathcal{L}(\theta|D_P) \equiv \mathcal{P}(D_P|\theta). \quad (2.33)$$

Considerando as observações de $\mathcal{Y}$ como i.i.d. (independentes e identicamente distribuídas) segundo a distribuição $\mathcal{P}(\mathcal{Y}|\theta)$, a verossimilhança dos parâmetros é dada por

$$\mathcal{L}(\theta|D_P) = \mathcal{P}(D_P|\theta) = \prod_{i=1}^P \mathcal{P}(\mathcal{Y}^i|\theta). \quad (2.34)$$

2.3.2 Máxima Verossimilhança

O método da **máxima verossimilhança**, que costuma-se abreviar pela sigla ML (do inglês, *maximum likelihood*), consiste em estimar os parâmetros de uma distribuição com base em um conjunto de dados observados, quando não existem variáveis escondidas no problema, de tal forma que a sua verossimilhança seja maximizada. Simbolizando os parâmetros estimados da distribuição por $\hat{\theta}$, temos então que a estimativa de máxima verossimilhança é dada por

$$\hat{\theta} = \arg \max_{\theta} \mathcal{L}(\theta|D_P) = \arg \max_{\theta} \mathcal{P}(D_P|\theta). \quad (2.35)$$

Como o logaritmo é uma função monotonicamente crescente, muitas vezes é mais conveniente (principalmente quando as probabilidades envolvidas são números muito pequenos) maximizar a chamada **verossimilhança logarítmica** (*log-likelihood*) dada por

$$l(\theta|D_P) = \ln \mathcal{L}(\theta|D_P) = \ln \mathcal{P}(D_P|\theta), \quad (2.36)$$

cujo resultado final é o mesmo.

Suponha que o conjunto $\mathcal{Y}$ de variáveis observadas consista apenas de uma variável y que pode assumir um entre m diferentes valores distintos, ou seja, $y \in O$, com $O = \{o_1, \dots, o_m\}$. Se ignorarmos a sequência temporal na qual os dados foram obtidos, o conjunto de dados D_P pode ser expresso na forma de números de ocorrência n_i do i -ésimo valor possível para y . Pode-se mostrar [24] que a estimativa de máxima verossimilhança para as frequências relativas da distribuição original nada mais são que as frequências relativas calculadas pelo conjunto de dados. Simbolizando a frequência relativa do i -ésimo valor possível para y por $\Theta_i \equiv \mathcal{P}(y = o_i)$, temos

$$\hat{\Theta}_i = \frac{n_i}{\sum_i n_i}. \quad (2.37)$$

2.3.3 O Algoritmo EM

O **algoritmo EM** (*Expectation-Maximization*) [11, 22, 24, 36, 68] é um método criado para encontrar-se uma estimativa de máxima verossimilhança dos parâmetros de uma distribuição no caso em que existem variáveis escondidas. Ele consiste em dois passos principais: o primeiro, chamado de *expectation step*, que traduziremos por **passo da média** ou ainda **Passo E**, é uma média sobre as variáveis escondidas condicionada às observações, o segundo, denominado *maximization step*, que traduziremos por **passo de maximização** ou **Passo M**, dá a estimativa dos novos parâmetros através da maximização da função obtida no primeiro passo.

Suponha que nossa intenção seja, a partir de um conjunto de P dados observados D_P , maximizar a verossimilhança de nossos parâmetros $\mathcal{L}(\theta|D_P)$ sabendo que a cada observação $\mathcal{Y}^i$ corresponde uma configuração das variáveis escondidas que denominaremos por $\mathcal{Q}^i$.

O objetivo passa a ser maximizar a verossimilhança dada por

$$\mathcal{L}(\theta|D_P) = \prod_i \mathcal{P}(\mathcal{Y}^i|\theta) = \prod_i \sum_{\mathcal{Q}^i} \mathcal{P}(\mathcal{Y}^i, \mathcal{Q}^i|\theta). \quad (2.38)$$

Chamaremos o conjunto dos dados não-observados de H_P , ou seja, $H_P = \{\mathcal{Q}^1, \dots, \mathcal{Q}^P\}$. A verossimilhança pode ser escrita, então, como

$$\mathcal{L}(\theta|D_P) = \sum_{H_P} \mathcal{P}(D_P, H_P|\theta). \quad (2.39)$$

O cálculo dessa verossimilhança requer uma soma sobre as variáveis escondidas do modelo, o que torna um cálculo direto algo extremamente custoso do ponto de vista computacional. Assim, para maximizar $\mathcal{L}(\theta|D_P)$, o algoritmo EM obtém, recursivamente, a n -ésima estimativa dos parâmetros θ , que simbolizaremos por θ^n , maximizando, a cada iteração, a média nas variáveis escondidas de $\ln \mathcal{P}(D_P, H_P|\theta)$ dado o conjunto de variáveis observadas D_P e uma estimativa anterior θ^{n-1} para os parâmetros do modelo. Como já explicamos, isso é feito através dos dois seguintes passos:

1. Passo E (*expectation step*): calculamos a média

$$Q(\theta|\theta^{n-1}) = E_{H_P}[\ln \mathcal{P}(D_P, H_P|\theta)|D_P, \theta^{n-1}], \quad (2.40)$$

onde a média é sobre a distribuição $\mathcal{P}(H_P|D_P, \theta^{n-1})$.

2. Passo M (*maximization step*): adotamos como nova estimativa para θ os parâmetros que maximizam $Q(\theta|\theta^{n-1})$:

Figura 2.3: **Distribuição a Priori:** Exemplo de distribuição a priori da variável A no intervalo real $[a, b]$ tal que os valores mais prováveis de A se encontram perto do ponto a .

$$\theta^n = \arg \max_{\theta} Q(\theta | \theta^{n-1}). \quad (2.41)$$

Enquanto as referências [24, 68] são introduções básicas ao método, a referência [22] é mais detalhada abordando detalhes técnicos como, entre outros, a convergência do algoritmo.

2.3.4 Inferência Bayesiana

Existe uma estratégia em inferência estatística que é comumente chamada de **inferência bayesiana**. Esta estratégia está fundamentada na interpretação bayesiana da teoria de probabilidades que tem seu nome em homenagem a Thomas Bayes, um reverendo e matemático inglês do século 18. Em um de seus trabalhos, publicado postumamente, Bayes estabeleceu as bases para uma maneira diferente de interpretar probabilidades como graus de plausibilidade e não apenas como frequências, à maneira ortodoxa.

Uma discussão detalhada sobre inferência bayesiana pode ser encontrada em [43] e [95]. Em termos práticos, o que diferencia a estatística bayesiana da ortodoxa é a possibilidade de incluirmos algum conhecimento *a priori* que tenhamos na forma de uma **distribuição a priori** de probabilidades. Suponha, por exemplo, que desejamos estimar o valor de um determinado parâmetro θ e que possuímos o conhecimento de que seus valores possíveis se encontram dentro de um certo intervalo real $[a, b]$, ou seja, $\theta \in [a, b]$. Além disso, suponha que também saibamos que é mais provável que θ esteja localizado no começo que no final do intervalo estando, portanto, mais próximo de a que de b . A nossa distribuição *a priori* $P^0(\theta)$ teria, então, aproximadamente a forma do gráfico representado na figura 2.3.

Quando uma nova informação é obtida através, por exemplo, de um experimento que gera um conjunto de dados D_P , a probabilidade $\mathcal{P}^0(\theta)$ é alterada e a nova distribuição de probabilidades para θ , que passa a ser chamada de **distribuição posterior**, é simbolizada por $\mathcal{P}(\theta|D_P)$, ou seja, a distribuição da variável θ dado que agora temos a informação extra D_P . A forma como essa atualização ocorre é através da fórmula

$$\mathcal{P}(AB) = \mathcal{P}(A|B)\mathcal{P}(B). \quad (2.42)$$

Aplicando-a ao nosso caso, temos

$$\mathcal{P}(\theta, D_P) = \mathcal{P}(\theta|D_P)\mathcal{P}(D_P) = \mathcal{P}(D_P|\theta)\mathcal{P}(\theta). \quad (2.43)$$

Assumindo que $\mathcal{P}(\theta)$ é a distribuição a priori para o parâmetro θ , a equação nos dá

$$\mathcal{P}(\theta|D_P) = \frac{\mathcal{P}(D_P|\theta) \mathcal{P}^0(\theta)}{\mathcal{P}(D_P)}, \quad (2.44)$$

onde $\mathcal{P}(D_P)$ é apenas um fator de normalização que pode ser escrito como

$$\mathcal{P}(D_P) = \int d\theta \mathcal{P}(D_P|\theta) \mathcal{P}^0(\theta) = \int d\theta \mathcal{P}(D_P, \theta). \quad (2.45)$$

A cada nova informação, o processo se repete e temos uma melhor estimativa da distribuição de probabilidades do parâmetro θ é obtida. A equação (2.44) é comumente chamada de **Teorema de Bayes** e forma a base para todos os métodos bayesianos de inferência.

Máximo a Posteriori

Uma vez que utilizamos todos os dados disponíveis e fizemos a atualização da nossa distribuição para o parâmetro a ser estimado através da fórmula (2.44), precisamos encontrar uma estimativa final para ele. Uma opção é utilizar como estimador o valor que maximiza a distribuição posterior, ou seja,

$$\hat{\theta}_{MAP} \equiv \arg \max_{\theta} \mathcal{P}(D_P|\theta) \mathcal{P}^0(\theta). \quad (2.46)$$

Esse tipo de estimativa é chamada de **máximo a posteriori** ou MAP.

Posterior Média

Uma outra alternativa para estimar parâmetros é a chamada estimativa **posterior média** ou PME (do inglês, *posterior mean estimator*).

A idéia é estimar o parâmetro como sendo a média na distribuição posterior, ou seja,

$$\hat{\theta}_{PME} \equiv \int d\theta \theta \mathcal{P}(\theta|D_P). \quad (2.47)$$

Podemos justificar a tomada da média através do fato de que ela é o estimador que minimiza o erro quadrático médio. Sendo assim, esse tipo de estimativa minimiza o erro quadrático médio dos parâmetros na distribuição posterior.

Escolha da Distribuição *a Priori*

Um dos pontos importantes quando utilizamos a inferência bayesiana diz respeito à distribuição *a priori*. Queremos que essa distribuição codifique toda a informação que possuímos a respeito de um determinado parâmetro, mas que não inclua nenhum tipo de pressuposição de natureza subjetiva. Qual a maneira de encontrar uma distribuição que obedeça tais condições?

A maneira mais utilizada para isso é baseada no chamado **Método da Máxima Entropia** ou, abreviadamente, **MaxEnt** [15, 86, 95]. Ele consiste em assumir que a melhor distribuição *a priori* é aquela que maximiza a entropia de Shannon [91] referente à distribuição procurada

$$S = - \int d\theta \mathcal{P}(\theta) \ln \mathcal{P}(\theta), \quad (2.48)$$

observando os vínculos aos quais ela obedece. Existe pelo menos um vínculo correspondente ao fato da distribuição estar normalizada, ou seja,

$$\int d\theta \mathcal{P}(\theta) = 1. \quad (2.49)$$

Os vínculos são introduzidos na maximização através de multiplicadores de Lagrange. Por exemplo, no caso onde não há mais nenhum vínculo além da normalização, a função a ser maximizada é

$$\mathcal{L} = S + \lambda \left(\int d\theta \mathcal{P}(\theta) - 1 \right), \quad (2.50)$$

onde λ é o multiplicador de Lagrange.

Uma justificativa para esse método pode ser encontrada em [43]. Em casos onde não possuímos nenhuma informação *a priori*, o método da máxima entropia equivale ao princípio da razão insuficiente de Laplace.

Figura 2.4: **Modelos Gráficos:** Modelo gráfico representando relações de interdependência entre 4 variáveis aleatórias diferentes.

2.4 Modelos Gráficos

2.4.1 Introdução

Modelos gráficos [44, 45, 57, 70, 77] são grafos onde cada vértice representa uma variável aleatória e a ausência de um elo entre dois vértices significa independência condicional entre as variáveis que eles representam. Tais independências condicionais permitem que a probabilidade conjunta de todos os vértices seja representada de forma fatorizada, o que diminui o número de parâmetros a serem calculados e aumenta a velocidade de inferência dos modelos.

Considere, por exemplo, o modelo gráfico dado pela figura 2.4, que veremos mais à frente tratar-se de um modelo gráfico não-direcionado, onde temos quatro variáveis aleatórias denominadas A , B , C e D . Se cada variável puder assumir n diferentes estados, para caracterizar a probabilidade conjunta $\mathcal{P}(A, B, C, D)$ precisaríamos, no caso geral, de $n^4 - 1$ parâmetros (a normalização representa um vínculo que diminui o número de parâmetros livres em uma unidade). Para o caso $n = 3$ teríamos que encontrar 80 parâmetros.

Aplicando em seqüência a fórmula usual para probabilidade condicional, podemos escrever a probabilidade conjunta dos vértices como

$$\mathcal{P}(A, B, C, D) = \mathcal{P}(D|A, B, C) \mathcal{P}(C|A, B) \mathcal{P}(B|A) \mathcal{P}(A). \quad (2.51)$$

Utilizando as relações de interdependência simbolizadas na figura 2.4, a probabilidade conjunta torna-se

$$\mathcal{P}(A, B, C, D) = \mathcal{P}(D|B, C) \mathcal{P}(C|A) \mathcal{P}(B|A) \mathcal{P}(A). \quad (2.52)$$

O que diminui o número de parâmetros segundo a tabela abaixo

Termo	Número de Parâmetros Gerados
$\mathcal{P}(D B, C)$	n^3
$\mathcal{P}(C A)$	n^2
$\mathcal{P}(B A)$	n^2
$\mathcal{P}(A)$	n

Reduzindo, no caso $n = 3$, o número total de parâmetros para 47, contando com o vínculo referente à normalização. A redução é muito mais drástica para redes com um número muito grande de vértices e com mais estados em cada variável, pois como é fácil observar, o número de parâmetros a ser calculado sem as relações de interdependência cresce exponencialmente com o número de variáveis aleatórias.

Os modelos gráficos podem ser classificados em dois tipos básicos: Modelos Direcionados e Não-Direcionados.

Os modelos gráficos podem ser vistos como uma forma unificada de estudar vários modelos existentes tanto em física e engenharia quanto em probabilidade. Alguns de seus casos particulares mais conhecidos são os filtros de Kalman, as cadeias de Markov, os HMMs e o próprio modelo de Ising. Dada a sua versatilidade, têm sido muito estudados como modelos de aprendizado através de diversas técnicas que incluem aproximações de campo médio [88] e métodos de núcleo [2, 36, 69, 93].

2.4.2 Modelos Gráficos Direcionados

Os **modelos gráficos direcionados** também são conhecidos como **redes bayesianas** ou **redes de crenças** [77] e são muito populares na comunidade de inteligência artificial e muito tem sido estudado sobre o aprendizado neles [38]. As cadeias de Markov, os HMMs e os filtros de Kalman [48] são casos particulares deste tipo de modelo.

Num modelo gráfico direcionado, uma seta saindo de um vértice A e chegando a um vértice B indica uma relação de dependência entre eles. Caso não haja uma seta saindo de A e chegando em B , isso implica que

$$\mathcal{P}(B|A) = \mathcal{P}(B). \quad (2.53)$$

Dado um determinado vértice, todos os vértices que levam a ele através de uma seta são denominados seus **pais**. Para um conjunto de n vértices V_i , a probabilidade conjunta pode, dadas tais relações de interdependência, ser escrita como

$$\mathcal{P}(V_1, \dots, V_n) = \prod_{i=1}^n \mathcal{P}(V_i | \Pi(V_i)), \quad (2.54)$$

Figura 2.5: **Modelos Gráficos Direcionados:** Modelo gráfico direcionado representando os 4 primeiros passos de um processo markoviano de ordem 2.

onde $\Pi(V_i)$ simboliza os vértices pais do vértice V_i .

A figura 2.5 mostra o modelo gráfico que representa os quatro primeiros passos de um processo markoviano de ordem 2 (ver seção 2.2). As variáveis aleatórias q_i representam o estado do processo no instante $t = i$. A probabilidade conjunta dos quatro estados é dada, usando-se a equação (2.54), por

$$\mathcal{P}(q_1, q_2, q_3, q_4) = \mathcal{P}(q_1|\Pi(q_1))\mathcal{P}(q_2|\Pi(q_2))\mathcal{P}(q_3|\Pi(q_3))\mathcal{P}(q_4|\Pi(q_4)). \quad (2.55)$$

Pelo modelo gráfico, temos que

$$\Pi(q_1) = \emptyset, \quad (2.56)$$

$$\Pi(q_2) = \{q_1\},$$

$$\Pi(q_3) = \{q_1, q_2\},$$

$$\Pi(q_4) = \{q_2, q_3\}$$

e a probabilidade conjunta dos vértices é dada, então, por

$$\mathcal{P}(q_1, q_2, q_3, q_4) = \mathcal{P}(q_1)\mathcal{P}(q_2|q_1)\mathcal{P}(q_3|q_1, q_2)\mathcal{P}(q_4|q_2, q_3). \quad (2.57)$$

Note que nesse tipo de modelo gráfico, os elos orientados podem ser interpretados em determinadas situações como a indicação de uma relação de dependência causal, isto é, os pais de um determinado vértice são a causa de tal vértice.

2.4.3 Modelos Gráficos Não-Direcionados

Modelos gráficos não-direcionados também são chamados, às vezes, de **campos aleatórios de Markov** (*Markov Random Fields*, ou *MRFs*) [57] e são aqueles onde os elos não são orientados, representando relações de dependência condicional, porém sem uma interpretação causal. Para um grafo com um conjunto V de vértices, a probabilidade conjunta desses vértices pode ser escrita como

$$\mathcal{P}(V) = \frac{1}{Z} \prod_{i \in V} \Psi_i(V^i), \quad (2.58)$$

Figura 2.6: **Modelos Gráficos Não-Direcionados:** Modelo gráfico representando o modelo de Ising em uma rede 3×3 .

onde C é o conjunto de cliques maximais do grafo, Ψ_i são funções potenciais associadas ao clique V^i e Z é um fator de normalização.

Um **clique** de um grafo não direcionado G é um subconjunto K de vértices de G tal que quaisquer dois vértices de K estão ligados por um elo, ou seja, é um subgrafo completo de G . O **tamanho** de um clique é o número de vértices que ele possui. Um clique é denominado **maximal** se não existir nenhum clique do qual ele seja um subconjunto próprio.

Vejamos como o modelo de Ising se enquadra neste grupo. Consideremos a rede 3×3 dada na figura 2.6, onde a cada sítio i associamos uma variável aleatória σ_i representando o *spin* nesse ponto. Interpretando esse gráfico como um modelo gráfico não-direcionado, temos que a probabilidade conjunta das variáveis é

$$\mathcal{P}(\sigma) = \frac{1}{Z} \prod_{i \in C} \Psi_i(\sigma_i). \quad (2.59)$$

A rede da figura forma um grafo cujos cliques maximais correspondem aos pares de primeiros vizinhos. Usando como função potencial para cada clique cujos componentes são os vértices i e j a função

$$\Psi(\sigma_i, \sigma_j) = e^{\beta J \sigma_i \sigma_j}, \quad (2.60)$$

a probabilidade conjunta resultante tem a forma

$$\mathcal{P}(\sigma) = \frac{e^{-\beta H(\sigma)}}{Z}, \quad (2.61)$$

onde

$$H(\sigma) = -J \sum_{\langle i,j \rangle} \sigma_i \sigma_j, \quad (2.62)$$

com a soma se estendendo sobre os primeiros vizinhos. Essas equações são as mesmas que descrevem o modelo de Ising com interação de primeiros vizinhos sem campo externo.

2.4.4 Inferência

Os métodos mais populares de inferência em modelos gráficos são:

1) **Monte Carlo:** onde uma variação do método comum chamada de **Markov Chain Monte Carlo** [36] é utilizada em altas dimensões, pois possui uma eficiência maior e tem, como casos particulares, amostragem de Gibbs e o algoritmo de Metropolis.

2) **Métodos Variacionais:** o mais utilizado é a aproximação de campo médio na qual todos os vértices são desacoplados e um parâmetro variacional é introduzido de forma a minimizar a divergência de Kullback-Leibler entre a distribuição de probabilidades real e a estimada. Existem aproximações melhores que a de campo médio com estimativas mais refinadas, porém são mais complicadas.

3) **Propagação de Crenças:** consiste em aplicar o chamado **algoritmo de Pearl** [77] no grafo original. Embora o algoritmo de Pearl tenha convergência garantida apenas para grafos sem caminhos fechados, tal procedimento é aplicado a qualquer grafo a princípio e parece funcionar muito bem em situações práticas como na área de visão de nível primário. Os **turbo codes** e as equações de Thouless-Anderson-Palmer (TAP) [88, 103] são instâncias desse algoritmo.

2.5 Aprendizado em Máquinas

A área da ciência que estuda como dispositivos projetados artificialmente podem aprender e reconhecer padrões é conhecida como **Aprendizado em Máquinas**. Essa é uma área interdisciplinar extremamente vasta, incluindo elementos de biologia, ciência da computação, física, linguística, lógica e até mesmo psicologia. Sendo assim, nosso intuito nesta seção será somente introduzir os conceitos que utilizaremos ao longo do nosso trabalho,

Figura 2.7: **Perceptron**: Representação gráfica de um perceptron.

conceitos esses com ênfase na teoria estatística de aprendizado, de forma a tornar compreensível o estudo que faremos a seguir.

A teoria estatística do aprendizado pode ser formulada como uma forma de inferência estatística (ver seção 2.3) onde um determinado dispositivo deve ajustar uma série de parâmetros θ baseado em um conjunto de P dados $D_P = \{y^1, \dots, y^P\}$ com a finalidade de realizar alguma tarefa. A natureza da tarefa e o dispositivo em questão irão determinar a relação entre os dados e os parâmetros de aprendizado, o que nos fornecerá um determinado modelo matemático que denominaremos de **modelo de aprendizado**.

Um dos modelos de aprendizado mais estudado é o **perceptron booleano** [11, 25, 40, 87]. Nesse modelo, o dispositivo é alimentado com um vetor n -dimensional $\xi \in [-1, +1]^n$ e devolve um número $\sigma \in [-1, +1]$ através da fórmula

$$\sigma = \text{sgn} \left(\sum_{i=1}^n (w_i \xi_i + \beta_i) \right). \quad (2.63)$$

O perceptron é inspirado em um modelo simplificado de um neurônio e, devido a isso, é usual chamar os pesos w_i na fórmula acima de **pesos sinápticos**. Os parâmetros β_i são denominados **limiares de ativação** e, na maioria das vezes, são colocados como zero. A figura 2.7 ilustra graficamente um perceptron.

Como um modelo de aprendizado, a tarefa do perceptron é classificar cada vetor ξ em dois conjuntos definidos pelos números $+1$ e -1 . Em uma situação de aprendizado, então, o nosso conjunto de dados seria dado por pares $y = (\xi, \sigma)$, que representam qual deve ser a resposta do modelo para cada vetor, e os parâmetros a ajustar seriam os pesos sinápticos w_i .

O perceptron da figura 2.7 possui apenas duas camadas: a de entrada, onde estão as unidades que recebem os dados, e a de saída, onde a resposta é dada. Pode-se, no entanto, colocar-se quantas camadas se queira entre essas duas. Tais camadas recebem o nome de

Figura 2.8: **Percéptron**: Representação gráfica de um perceptron com uma camada intermediária (chamada de **camada escondida**).

camadas escondidas e a importância do perceptron está no fato de que utilizando-se de modelos com pelo menos uma camada escondida, como na figura 2.8, eles são capazes de computação universal e podem aproximar qualquer função.

Dado um modelo, um conjunto de dados e os parâmetros a serem ajustados, o que falta é um procedimento para realizar essa tarefa. Tais procedimentos, que são denominados de **algoritmos de aprendizado**, são o principal objeto de estudo no aprendizado em máquinas. Como já dissemos, eles são, em geral, variações dos métodos de inferência estatística que já apresentamos na seção 2.3 associados a variadas técnicas matemáticas que dependem dos detalhes do modelo e do conjunto de dados. Mais que uma ciência, pode-se considerar que o desenvolvimento de algoritmos de aprendizado cada vez mais eficientes é uma arte.

A seguir, apresentaremos os pontos mais importantes do estudo dos algoritmos de aprendizado e introduziremos os conceitos que utilizaremos durante o nosso trabalho.

2.5.1 O Cenário de Aprendizado

A forma mais usual de se descrever o aprendizado em máquinas assume que o conjunto de dados é gerado através de alguma **regra** determinada por um **professor**. O conceito de professor é extremamente amplo e ele pode ser uma pessoa que determina a regra, um outro modelo ao qual o modelo em questão tentará imitar ou simplesmente a própria Natureza de onde o modelo tentará extrair um determinado padrão (uma série diária de temperaturas máximas e mínimas, por exemplo). Em contrapartida, o modelo de aprendizado passa a ser chamado de **aluno**.

Nos casos mais simples, a regra definida pelo professor é fixa, mas o interesse em

situações onde a regra muda é cada vez maior [101]. Atualmente, aplicações em diversas áreas onde séries temporais seguem padrões que podem mudar com o tempo têm aparecido. Exemplos podem ser encontrados na previsão do comportamento de mercados financeiros, no monitoramento de eletrocardiogramas ou eletroencefalogramas em medicina e sistemas de controle e navegação em engenharia.

O aprendizado, descrito da forma acima, pode ainda ser classificado em duas categorias distintas: aprendizado **supervisionado** e **não-supervisionado**.

No aprendizado supervisionado, o aluno tem sempre a tarefa de, recebido um determinado dado, emitir uma resposta que coincida com a resposta gerada pelo professor, sendo que o aluno tem sempre acesso a essa resposta durante a sua fase de aprendizado.

Já no aprendizado não-supervisionado, a tarefa do aluno pode variar: tanto pode ser gerar respostas a dados recebidos pelo professor como pode ser apenas reconhecer padrões em alguma seqüência de dados para gerar uma seqüência semelhante posteriormente. O ponto principal é que o aluno não tem acesso a respostas do professor com a qual ele poderia se guiar durante o aprendizado.

2.5.2 Algoritmos *Off-line* e *On-line*

Os algoritmos de aprendizado podem ser separados em duas grandes classes segundo a sua dinâmica: algoritmos *on-line* e algoritmos *off-line*, cujas diferenças explicitaremos a seguir.

Algoritmos *Off-line*

Um algoritmo é chamado de *off-line* quando a situação de aprendizado fornece, de uma única vez, um conjunto de dados D_P com o qual os parâmetros do modelo devem ser ajustados. Isso implica que o algoritmo deve utilizar uma certa quantidade de memória para guardar o conjunto de dados. Esse conjunto de dados pode, então, ser utilizado em qualquer ordem e quantas vezes forem necessárias para extrair dele a maior quantidade possível de informação para que o ajuste dos parâmetros se dê da melhor forma possível.

Algoritmos *On-line*

Um algoritmo é chamado de *on-line* quando a situação de aprendizado não fornece de uma única vez todo o conjunto de dados D_P com o qual os parâmetros do modelo devem ser ajustados. Ao invés disso, cada dado y^i do conjunto é fornecido seqüencialmente e, a cada vez que isso acontece, o algoritmo deve fazer uma nova estimativa dos parâmetros.

Obviamente esse cenário é mais restritivo que o cenário *off-line*, no entanto, possui a grande vantagem de que, se o algoritmo utilizado for devidamente projetado, pode ser capaz de captar mudanças na regra do professor à medida que o aprendizado ocorre.

2.5.3 Performance e Generalização

Durante o aprendizado, faz-se necessário uma maneira de decidir se o aluno está aprendendo corretamente através do conjunto de dados. Além disso, para podermos comparar diferentes algoritmos, é necessário saber com que velocidade o aluno está aprendendo [11].

Existem duas funções usualmente empregadas que medem a performance do aprendizado. A primeira delas é denominada de **erro de treinamento** e consiste em uma medida de quão bem o aluno está aprendendo os dados do conjunto D_P fornecido. Note bem que neste caso apenas o aprendizado do conjunto está sendo avaliado. Não existe uma forma fixa para essa função, pois ela depende do formato dos dados e de qual é a grandeza que o aprendizado pretende minimizar. Ela pode ser, por exemplo, o erro quadrático médio de uma série de medidas ou simplesmente uma função que atribui 1 a cada exemplo classificado erradamente e 0 a cada exemplo classificado corretamente em uma situação de classificação e soma esses números.

A segunda função chama-se **erro de generalização** e mede um conceito mais complexo, o conceito de **generalização** de uma regra a partir de exemplos. Suponha que D é o conjunto de todos os dados possíveis que possam ser gerados por um professor com base em uma determinada regra. Chamamos de generalização a capacidade de um aluno inferir a regra completa através de um subconjunto $D_P \subset D$ menor que o conjunto total de exemplos. O erro de generalização mede, então, quão bem o aluno aprendeu a regra geral com um determinado número limitado de exemplos. Assim como o erro de treinamento, a forma funcional do erro de generalização depende do modelo de aprendizado e do conjunto de dados. Para o *perceptron*, por exemplo, ele é uma função do produto escalar entre os pesos sinápticos do professor e do aluno e mede a probabilidade do aluno classificar de forma errada um determinado vetor gerado de forma aleatória. Para modelos mais complexos, onde o produto escalar dos parâmetros não possui uma interpretação tão clara, ou para situações onde o aprendizado não requer uma classificação, esse tipo de função torna-se pouco prático ou não pode ser utilizado.

A performance de um algoritmo é geralmente medida através das chamadas **curvas de aprendizado**. Essas curvas são, normalmente, gráficos do erro de generalização em função da quantidade de dados utilizada durante o processo de aprendizado. Dependendo

da situação (quando, por exemplo, não temos acesso aos parâmetros do professor), no entanto, não é possível obter o erro de generalização e às vezes as curvas de aprendizado são traçadas apenas para o erro de treinamento.

Pode-se, assim, avaliar os algoritmos de aprendizado com relação, principalmente, a duas características: a velocidade de aprendizado, que é maior quanto menor for a quantidade de exemplos necessária para atingir um dado erro de generalização, e a generalização em si, que é representada pelo valor final do erro de generalização ao fim do aprendizado.

O ponto principal da nossa tese será estudar algoritmos de aprendizado para HMMs de acordo com as duas características acima apresentadas.

2.5.4 O Algoritmo Bayesiano *On-line*

Para algoritmos de aprendizado baseados em exemplos de um professor, os métodos bayesianos saturam o limite de Bayes correspondente ao menor erro de generalização possível para qualquer algoritmo. Tal vantagem seria suficiente para tornar os métodos bayesianos o padrão para o aprendizado de máquinas não fosse pelo grande inconveniente desses métodos serem, em geral, exponencialmente lentos no que diz respeito à complexidade computacional dos programas neles baseados. Isso se deve ao fato de exigirem uma soma sobre as variáveis do problema a respeito das quais não temos nenhuma informação.

Essa desvantagem é extremamente restritiva, já que um modelo de aprendizado pode precisar de um número muito grande de exemplos para generalizar uma dada regra. Isso é especialmente verdade para aplicações práticas, principalmente na área de linguística e reconhecimento de imagens. No entanto, a eficiência dos algoritmos bayesianos quanto à capacidade de generalização é uma forte motivação para a pesquisa de algoritmos que possam ser implementados de forma polinomial ou pelo menos que sejam cada vez mais rápidos. Nos últimos tempos, a utilização desses algoritmos vem crescendo, uma vez que a velocidade dos computadores está aumentando, o que faz com que já possam ser aplicados em algumas situações práticas, e vários avanços nessa área têm aparecido. A seguir, descrevemos um algoritmo desenvolvido por Oppen [74] para o aprendizado de parâmetros $\theta \in \mathbb{R}$ que tem se tornado bastante popular.

Considere uma sequência de dados gerada por um professor τ na forma

$$D_P = \{y^1, \dots, y^P\}. \quad (2.64)$$

O algoritmo encontra uma distribuição de probabilidades $\mathcal{P}(\theta)$, onde θ são os parâmetros de possíveis modelos que gerariam a sequência dada. Se considerarmos cada dado na

Figura 2.9: **Algoritmo Bayesiano On-line:** a) 1o. passo: quando um novo dado é incorporado, a distribuição antiga é modificada de tal modo que sai do espaço formado por sua família paramétrica. b) 2o. passo: a distribuição é então projetada de volta no espaço das distribuições paramétricas originais através da minimização da distância de Kullback-Leibler.

seqüência como independente, a sua distribuição de probabilidades pode ser escrita como

$$\mathcal{P}(D_P|\theta) = \prod_{i=1}^P \mathcal{P}(y^i|\theta). \quad (2.65)$$

Usando o teorema de Bayes, temos

$$\mathcal{P}(\theta|D_P) = \frac{\mathcal{P}(\theta)\mathcal{P}(D_P|\theta)}{\int d\theta' \mathcal{P}(\theta')\mathcal{P}(D_P|\theta')}, \quad (2.66)$$

e, portanto

$$\begin{aligned} \mathcal{P}(\theta|D_{P+1}) &= \frac{\mathcal{P}(\theta)\mathcal{P}(D_P|\theta)\mathcal{P}(y^{P+1}|\theta)}{\int d\theta' \mathcal{P}(\theta')\mathcal{P}(D_P|\theta')\mathcal{P}(y^{P+1}|\theta')} = \\ &= \frac{\mathcal{P}(\theta|D_P)\mathcal{P}(y^{P+1}|\theta)}{\int d\theta' \mathcal{P}(\theta'|D_P)\mathcal{P}(y^{P+1}|\theta')}. \end{aligned} \quad (2.67)$$

A fórmula acima representa uma atualização *off-line* da distribuição para θ , já que pressupõe o conhecimento de todo o conjunto anterior de dados para uma atualização. Para transformarmos essa equação em uma forma *on-line*, vamos aproximar a distribuição $\mathcal{P}(\theta|D_P)$ por uma forma paramétrica $\mathcal{P}(\theta|\lambda^P)$, onde λ^P é um vetor de parâmetros que descrevem a distribuição após o exemplo P ter sido apresentado. Utilizaremos, então, o seguinte algoritmo, representado graficamente na figura 2.9, para atualização dessa distribuição à medida que novos exemplos são apresentados:

1) Atualiza-se a distribuição parametrizada utilizando o teorema de Bayes

$$\mathcal{P}(\theta|\lambda^P, y^{P+1}) = \frac{\mathcal{P}(\theta|\lambda^P)\mathcal{P}(y^{P+1}|\theta)}{\int d\theta' \mathcal{P}(\theta'|\lambda^P)\mathcal{P}(y^{P+1}|\theta')}. \quad (2.68)$$

2) Projeta-se a distribuição obtida num espaço de distribuições gaussianas

$$\mathcal{P}(\theta|\lambda^P, y^{P+1}) \rightarrow \mathcal{P}(\theta|\lambda^{P+1}). \quad (2.69)$$

A parametrização gaussiana que faremos será do tipo

$$\mathcal{P}(\theta|\lambda^P) = \frac{1}{\sqrt{(2\pi)^N \det C^P}} \exp \left[-\frac{1}{2}(\theta - \bar{\theta}^P)^T (C^P)^{-1} (\theta - \bar{\theta}^P) \right], \quad (2.70)$$

onde $\bar{\theta}^P$ é o vetor da média da distribuição após o exemplo P , C^P é a matriz de correlação e N é a dimensão do vetor θ . Assim, temos que

$$\lambda^P = (\bar{\theta}^P, C^P). \quad (2.71)$$

Devido à sua tratabilidade [74, 97], a projeção da nova distribuição no espaço das gaussianas é feita minimizando-se a distância de Kullback-Leibler entre as duas, isto é, minimizando

$$D_{KL}(\mathcal{P}(\theta|\lambda^P, y^{P+1}) || \mathcal{P}(\theta|\lambda^{P+1})) = \int \mathcal{P}(\theta|\lambda^P, y^{P+1}) \ln \frac{\mathcal{P}(\theta|\lambda^P, y^{P+1})}{\mathcal{P}(\theta|\lambda^{P+1})} d\theta, \quad (2.72)$$

ou ainda, levando em conta que $\mathcal{P}(\theta|\lambda^P, y^{P+1})$ é fixa, extremizando a expressão

$$\mathcal{L}_0 = - \int \mathcal{P}(\theta|\lambda^P, y^{P+1}) \ln \mathcal{P}(\theta|\lambda^{P+1}) d\theta, \quad (2.73)$$

ou, equivalentemente, maximizando

$$\mathcal{L}'_0 = \int \mathcal{P}(\theta|\lambda^P, y^{P+1}) \ln \mathcal{P}(\theta|\lambda^{P+1}) d\theta, \quad (2.74)$$

com relação a λ^{P+1} .

Essa extremização levará às equações para atualização dos parâmetros λ . Usamos as equações resultantes para cada dado novo que for apresentado e, ao final do conjunto de dados, teremos uma gaussiana cuja média indicará o valor mais provável dos parâmetros do modelo que representa os dados.

Capítulo 3

Modelos de Markov com Variáveis de Estado Escondidas

3.1 Teoria

Daremos a seguir uma introdução aos conceitos básicos da teoria dos HMMs, que podem ser encontrados também nas referências [8, 23, 81]. Uma exposição matemática mais rigorosa dos conceitos envolvidos pode ser encontrada em [26]. Tentaremos sempre seguir as convenções de notação descritas nesta seção ao longo de toda esta tese. Possíveis exceções ou desvios desta notação padrão serão explicitamente citados quando necessário.

Considere um processo markoviano finito de ordem k dado pela sequência $y_1^T = \{y_1, \dots, y_T\}$. Às vezes, k pode ser um número muito grande, o que gera problemas para lidar com tais modelos, pois a complexidade computacional para o cálculo da distribuição de probabilidades dessa sequência cresce exponencialmente com k . Caso existam m valores possíveis para cada y_t , o número de parâmetros crescerá com m^k . Por outro lado, as relações de interdependência entre as variáveis da sequência podem ser muito mais complicadas que processos markovianos de qualquer ordem e, assim, muito difíceis de serem modeladas.

Para tentar contornar dificuldades como estas, introduz-se a idéia de variável escondida. Assumimos que as relações entre os dados passados podem ser englobadas em uma variável de estado não observada que satisfaria um processo markoviano de ordem baixa, geralmente 1.

Consideremos a sequência y_1^T como a sequência das variáveis observadas do modelo de Markov. As variáveis de estado escondidas, que simbolizaremos por q_t e que poderão assumir n diferentes valores, formarão uma sequência q_1^T de tal forma que tenhamos, por

Figura 3.1: **HMM**: Representação gráfica de um HMM.

hipótese,

$$\begin{aligned}\mathcal{P}(y_t|q_1^t, y_1^{t-1}) &= \mathcal{P}(y_t|q_t), \\ \mathcal{P}(q_{t+1}|q_1^t, y_1^t) &= \mathcal{P}(q_{t+1}|q_t).\end{aligned}\tag{3.1}$$

Note que aqui ocorre, devido à estrutura de dependências assumida das variáveis, uma redução de parâmetros que na primeira linha é de $m^t \times n^t$ para $m \times n$ parâmetros e na segunda é de $m^t \times n^{t+1}$ para m^2 .

Isto significa que as variáveis q_t trazem informação completa (dentro do modelo) dos valores passados das variáveis observadas e escondidas no que diz respeito à previsão de y_t ou q_{t+1} . O fato da dependência dos estados observados a cada instante t ser apenas com relação ao estado escondido também em t justifica o fato de, às vezes, esses modelos serem chamados de funções probabilísticas de cadeias de Markov. A figura 3.1 mostra a representação de um HMM como um modelo gráfico.

De acordo com as equações (3.1), a distribuição conjunta dos estados escondidos e observados é dada por

$$\begin{aligned}\mathcal{P}(y_1^T, q_1^T) &= \mathcal{P}(y_T, y_1^{T-1}, q_1^T) = \\ &= \mathcal{P}(y_T|y_1^{T-1}, q_1^T) \mathcal{P}(y_1^{T-1}, q_1^T) = \\ &= \mathcal{P}(y_T|q_T) \mathcal{P}(q_T, q_1^{T-1}, y_1^{T-1}) = \\ &= \mathcal{P}(y_T|q_T) \mathcal{P}(q_T|q_1^{T-1}, y_1^{T-1}) \mathcal{P}(q_1^{T-1}, y_1^{T-1}) = \\ &= \mathcal{P}(y_T|q_T) \mathcal{P}(q_T|q_{T-1}) \mathcal{P}(q_1^{T-1}, y_1^{T-1}) = \\ &= \left[\prod_{t=2}^T \mathcal{P}(y_t|q_t) \right] \left[\prod_{t=2}^T \mathcal{P}(q_t|q_{t-1}) \right] \mathcal{P}(y_1, q_1) = \\ &= \mathcal{P}(q_1) \prod_{t=1}^{T-1} \mathcal{P}(q_{t+1}|q_t) \prod_{t=1}^T \mathcal{P}(y_t|q_t).\end{aligned}\tag{3.2}$$

Utilizaremos a nomenclatura

$$\begin{aligned}
\mathcal{P}(q_1) &\equiv \text{Probabilidade do estado inicial.} \\
\mathcal{P}(q_{t+1}|q_t) &\equiv \text{Probabilidade de transição.} \\
\mathcal{P}(y_t|q_t) &\equiv \text{Probabilidade de emissão.}
\end{aligned}$$

Como não podemos observar os valores da variável escondida, ao invés de trabalharmos com a distribuição conjunta, nós a marginalizamos para obtermos a distribuição de probabilidades da sequência de variáveis observadas

$$\mathcal{P}(y_1^T) = \sum_{q_1^T} \mathcal{P}(y_1^T, q_1^T). \quad (3.3)$$

Veremos, mais adiante, que existem algoritmos polinomiais em T e n (número de possíveis estados da variável escondida), para a obtenção dessas probabilidades.

As variáveis de estado escondidas são sempre consideradas discretas para um HMM e denominaremos o conjunto dos seus n possíveis valores por $S = \{s_1, \dots, s_n\}$. Existe uma generalização para o caso de variáveis escondidas contínuas, chamada de Modelos de Estado-Espaço [33], mas que não será abordada nesta tese. As variáveis observadas podem ser contínuas ou discretas. No caso discreto, simbolizaremos o conjunto dos seus m possíveis valores por $O = \{o_1, \dots, o_m\}$.

A **matriz de transição** do modelo será a matriz $A(t)$ de transição da variável q_t de tamanho $n \times n$, ou seja,

$$A_{ij}(t) = \mathcal{P}(q_{t+1} = s_j | q_t = s_i), \quad (3.4)$$

e a matriz $B(t)$, de tamanho $n \times m$, dada por

$$B_{ij}(t) = \mathcal{P}(y_t = o_j | q_t = s_i), \quad (3.5)$$

será chamada de **matriz de emissão**. O vetor $\pi \equiv \pi(1)$ das probabilidades iniciais do modelo é o vetor coluna das probabilidades iniciais da variável q_t

$$\pi_i = \mathcal{P}(q_1 = s_i). \quad (3.6)$$

Eventualmente, quando os estados não estiverem explicitados, usaremos as notações

$$\begin{aligned}
A_{q_t, q_{t+1}} &\equiv \mathcal{P}(q_{t+1} | q_t), \\
B_{q_t, y_t} &\equiv \mathcal{P}(y_t | q_t), \\
\pi_{q_1} &\equiv \mathcal{P}(q_1).
\end{aligned} \quad (3.7)$$

Figura 3.2: **Topologias de um HMM:** a) modelo *left-right*; b) modelo de *loop*.

Dado um determinado estado escondido s_i , ele certamente evoluirá para um dos outros estados ou para ele mesmo e, portanto,

$$\sum_{j=1}^n A_{ij}(t) = 1, \quad i = 1, \dots, n. \quad (3.8)$$

Da mesma forma, o estado s_i sempre emitirá um dos estados observados o_j e, portanto,

$$\sum_{j=1}^m B_{ij}(t) = 1, \quad i = 1, \dots, n. \quad (3.9)$$

Por fim, o vetor π obviamente satisfaz

$$\sum_{i=1}^n \pi_i = 1. \quad (3.10)$$

Quando as matrizes de transição e de emissão não dependerem de t , diremos que o HMM é **homogêneo**. Os HMMs que trataremos ao longo da tese são quase todos homogêneos, com a exceção sendo o estudo da seção 5.4 e da aplicação na seção 5.5, que tratam de professores com parâmetros que podem variar no tempo, ou seja, ao longo da aprendizagem.

Chamaremos de modelo de Markov de variável escondida (HMM) ao conjunto simbolizado por $\omega = \{S, O, \pi, A, B\}$.

Algumas vezes, quando o contexto deixar claro, omitiremos os índices em y_1^T e q_1^T escrevendo apenas y e q para as seqüências de estados observados e escondidos, respectivamente.

Costuma-se chamar de **topologia** do HMM à forma do grafo de transição (ver seção 2.2) do modelo de Markov subjacente, isto é, da matriz de transição A . Algumas topologias particulares, usadas com certa frequência em aplicações, recebem nomes especiais como *left-right models* (figura 3.2a), onde os estados escondidos possuem uma ordem

definida e apenas transições na direção definida por essa ordem são permitidas, e *loop models* (figura 3.2b), onde os estados são organizados de forma que o grafo de transição tenha o formato de um círculo onde as transições são permitidas apenas entre estados vizinhos e em uma única direção.

Existem diversas variações do modelo básico descrito nesta seção para HMMs que podem ser tão complicadas quanto se queira para tentar adaptar seu desempenho ao problema sendo tratado, como Input/Output-HMMs [9], HMMs Fatoriais [34] e outras [7], mas elas não serão estudadas neste trabalho.

3.2 Algoritmos

Tornou-se comum na literatura considerar três problemas clássicos referentes a HMMs: verossimilhança, reconhecimento e aprendizado. A solução dos dois primeiros problemas resulta em algoritmos que formam a base para a solução do terceiro. O terceiro problema tem um grande interesse por si mesmo, pois está relacionado à questão do aprendizado em HMMs e é a base de todas as aplicações mais avançadas desses modelos. A seguir, apresentaremos esses três problemas clássicos. Para os dois primeiros, faremos uma descrição detalhada tanto do problema quanto da solução. O terceiro problema, estando intimamente relacionado ao tema central desta tese, será apenas apresentado formalmente nesta seção para ser desenvolvido posteriormente ao longo deste trabalho.

Problema 1: Verossimilhança

Dada uma sequência de estados observados y_1^T , qual a probabilidade de se obtê-la a partir de um determinado HMM ω ?

O problema equivale a encontrar $\mathcal{P}(y_1^T|\omega)$, ou seja, a verossimilhança. Se calculássemos essa probabilidade diretamente pelas fórmulas (3.2) e (3.3), veríamos que o número de multiplicações necessárias para o cálculo dessa soma seria $(2T - 1)n^T$, isto é, um número exponencial em T . No entanto, existem pelo menos dois algoritmos com complexidade polinomial em T para a solução desse problema que são chamados de *Forward Procedure* e *Backward Procedure*. A diminuição da complexidade para esses dois algoritmos é um caso especial de uma teoria mais geral que engloba diversos outros algoritmos e está relacionada diretamente com propriedades de fatorização das expressões a serem calculadas [1].

Forward Procedure:

Definindo a **variável forward** como

$$\alpha_t(i) \equiv \mathcal{P}(y_1^t, q_t = s_i | \omega), \quad (3.11)$$

o algoritmo é dado pelos seguintes passos:

1) Calculamos para os n possíveis valores de q_1 , isto é, para $i = 1, \dots, n$, as probabilidades

$$\begin{aligned} \alpha_1(i) &\equiv \mathcal{P}(y_1, q_1 = s_i | \omega) = \\ &= \mathcal{P}(q_1 = s_i | \omega) \mathcal{P}(y_1 | q_1 = s_i, \omega) = \\ &= \pi_i B_{i, y_1}. \end{aligned} \quad (3.12)$$

Esse passo precisa de n multiplicações.

2) Calculamos recursivamente para $t = 2, \dots, T$ e para $j = 1, \dots, n$ a variável *forward* através de

$$\begin{aligned} \alpha_t(j) &\equiv \mathcal{P}(y_1^t, q_t = s_j | \omega) = \\ &= \mathcal{P}(y_t | y_1^{t-1}, q_t = s_j, \omega) \mathcal{P}(y_1^{t-1}, q_t = s_j | \omega) = \\ &= \mathcal{P}(y_t | q_t = s_j, \omega) \sum_i \mathcal{P}(y_1^{t-1}, q_t = s_j, q_{t-1} = s_i | \omega) = \\ &= \mathcal{P}(y_t | q_t = s_j, \omega) \sum_i \mathcal{P}(q_t = s_j | q_{t-1} = s_i, y_1^{t-1}, \omega) \mathcal{P}(y_1^{t-1}, q_{t-1} = s_i | \omega) = \\ &= \mathcal{P}(y_t | q_t = s_j, \omega) \sum_i \mathcal{P}(q_t = s_j | q_{t-1} = s_i, \omega) \mathcal{P}(y_1^{t-1}, q_{t-1} = s_i | \omega) = \\ &= B_{j, y_t} \sum_i A_{ij} \alpha_{t-1}(i). \end{aligned} \quad (3.13)$$

Note que é exatamente a penúltima passagem que permite a redução da complexidade computacional do problema.

Cada passo da recursão precisará de $n + 1$ multiplicações. Como já temos $\alpha_1(i)$ para cada i , calcularemos recursivamente desde $\alpha_2(i)$ até $\alpha_T(i)$, isto é, $T - 1$ vezes. Ao todo, então, essa etapa utilizará $n(T - 1)(n + 1)$ multiplicações, pois temos n possíveis valores para i .

3) Finalmente, obtemos a verossimilhança

$$\mathcal{P}(y_1^T | \omega) = \sum_i \mathcal{P}(y_1^T, q_T = i | \omega) = \sum_i \alpha_T(i), \quad (3.14)$$

que requer apenas somas. O número total de multiplicações é, então, $n + n(T-1)(n+1)$, que é polinomial de ordem 2 em n e de ordem 1 em T . É importante notar a diferença entre as equações (3.3) e (3.14). Na primeira, a soma é sobre todas as possíveis seqüências q_1^T , cujo número é n^T , que é exponencial em T , enquanto que na segunda, a soma é sobre os possíveis estados q_T que a variável escondida pode assumir no tempo T , o que é simplesmente n e, portanto, linear.

Backward Procedure:

Definindo a **variável backward** como

$$\beta_t(i) \equiv \mathcal{P}(y_{t+1}^T | q_t = s_i, \omega), \quad (3.15)$$

o algoritmo é dado pelos seguintes passos:

1) Inicializamos a variável *backward* para $i = 1, \dots, n$ apenas formalmente como

$$\beta_T(i) = \mathcal{P}(y_{T+1}^T | q_T = s_i, \omega) = 1. \quad (3.16)$$

2) Calculamos recursivamente a variável *backward* para $t = T-1, \dots, 1$ e $i = 1, \dots, n$

$$\begin{aligned} \beta_t(i) &= \mathcal{P}(y_{t+1}^T | q_t = s_i, \omega) = \\ &= \sum_j \mathcal{P}(y_{t+1}^T, q_{t+1} = s_j | q_t = s_i, \omega) = \\ &= \sum_j \mathcal{P}(y_{t+1}^T | q_{t+1} = s_j, q_t = s_i, \omega) \mathcal{P}(q_{t+1} = s_j | q_t = s_i, \omega) = \\ &= \sum_j \mathcal{P}(q_{t+1} = s_j | q_t = s_i, \omega) \mathcal{P}(y_{t+1}^T | q_{t+1} = s_j, \omega) = \\ &= \sum_j \mathcal{P}(q_{t+1} = s_j | q_t = s_i, \omega) \mathcal{P}(y_{t+1} | q_{t+1} = s_j, \omega) \mathcal{P}(y_{t+2}^T | q_{t+1} = s_j, \omega) = \\ &= \sum_j A_{ij} B_{j, y_{t+1}} \beta_{t+1}(j). \end{aligned} \quad (3.17)$$

Esse passo necessita de $2n \cdot n(T-1)$ multiplicações.

3) Obtemos, finalmente

$$\begin{aligned}
 \mathcal{P}(y_1^T | \omega) &= \sum_i \mathcal{P}(y_1^T, q_1 = s_i | \omega) = \\
 &= \sum_i \mathcal{P}(y_1^T | q_1 = s_i, \omega) \mathcal{P}(q_1 = s_i | \omega) = \\
 &= \sum_i \mathcal{P}(q_1 = s_i | \omega) \mathcal{P}(y_1 | q_1 = s_i, \omega) \mathcal{P}(y_2^T | q_1 = s_i, \omega) = \\
 &= \sum_i \pi_i B_{i, y_1} \beta_1(i),
 \end{aligned} \tag{3.18}$$

o que necessita de mais $2n$ multiplicações. Ao todo, o algoritmo precisa de $2n + 2n^2(T - 1)$ multiplicações que, assim como a *forward procedure*, também é polinomial de ordem 2 em n e de ordem 1 em T .

Qualquer um dos dois procedimentos, acima pode ser usado para o cálculo de $\mathcal{P}(y_1^T)$ o que, a princípio, parece significar que um dos dois (preferencialmente o mais lento) é totalmente desnecessário. No entanto, quando estudarmos as fórmulas de Baum-Welch (seção 4.3), veremos que ambos serão necessários para os cálculos.

Problema 2: Reconhecimento

Dada uma determinada seqüência de observações y_1^T , qual a seqüência de estados q_1^T que maximiza a probabilidade $\mathcal{P}(y_1^T, q_1^T)$?

Esse problema equivale a reconhecer qual seqüência de estados escondidos provavelmente gerou a seqüência observada. A solução desse problema é dada por um algoritmo de programação dinâmica conhecido pelo nome de algoritmo de Viterbi:

Algoritmo de Viterbi: o objetivo é encontrar

$$q_1^{T*} = \arg \max_{q_1^T} \mathcal{P}(y_1^T, q_1^T). \tag{3.19}$$

Definimos a função peso $U(A)$ para um determinado evento A tal que $\mathcal{P}(A) \neq 0$ como

$$U(A) \equiv -\ln \mathcal{P}(A) \Rightarrow \mathcal{P}(A) = \exp[-U(A)]. \tag{3.20}$$

Assim, temos a equivalência

$$\max_{q_1^T} \mathcal{P}(y_1^T, q_1^T) \Leftrightarrow \min_{q_1^T} U(y_1^T, q_1^T). \tag{3.21}$$

Podemos, então, ao invés de procurar o máximo da probabilidade, procurar o mínimo da função peso. Pela equação (3.2) temos

$$\begin{aligned} U(y_1^T, q_1^T) &= -\ln[\mathcal{P}(q_1)\mathcal{P}(y_1|q_1)] - \sum_{t=2}^T \ln[\mathcal{P}(q_t|q_{t-1})\mathcal{P}(y_t|q_t)] = \\ &= [U(q_1) + U(y_1|q_1)] + \sum_{t=2}^T [U(q_t|q_{t-1}) + U(y_t|q_t)]. \end{aligned} \quad (3.22)$$

Cada termo dessa função é positivo e, portanto, minimizar a função é equivalente a minimizar qualquer soma parcial desses termos. Essa propriedade será utilizada pelo algoritmo.

Definimos as seguintes variáveis:

$\delta_t(s_i) \equiv$ peso acumulado quando se está no estado s_i no instante t .

$\psi_t(s_i) \equiv$ estado em $t-1$ que possui o menor custo para ir para s_i no instante t .

O procedimento é descrito a seguir:

1) Calculamos para $1 \leq i \leq n$

$$\begin{aligned} \delta_1(s_i) &= U(q_1 = s_i) + U(y_1|q_1 = s_i), \\ \psi_1(s_i) &= 0. \end{aligned} \quad (3.23)$$

O número de adições necessárias é n .

2) Calculamos para $1 \leq i \leq n$ e $2 \leq t \leq T$

$$\begin{aligned} \delta_t(s_i) &= \min_{1 \leq j \leq n} [\delta_{t-1}(s_j) + U(q_t = s_i|q_{t-1} = s_j)] + U(y_t|q_t = s_i), \\ \psi_t(s_i) &= \arg \min_{1 \leq j \leq n} [\delta_{t-1}(s_j) + U(q_t = s_i|q_{t-1} = s_j)]. \end{aligned} \quad (3.24)$$

São necessárias $2n^2(T+1)$ adições.

3) Temos, então, que

$$q_T^* = \arg \min_{1 \leq j \leq n} [\delta_T(s_j)]. \quad (3.25)$$

4) Obtemos a sequência de estados q_1^{T*} através da recorrência

$$q_t^* = \psi_{t+1}(q_{t+1}^*), \quad t = T, \dots, 1. \quad (3.26)$$

Ao todo, o número de adições necessárias é $n + 2n^2(T+1)$, que é polinomial em n e em T .

O algoritmo de Viterbi, embora seja o comumente usado na esmagadora maioria das aplicações, principalmente devido à sua simplicidade, não é a única maneira existente para resolver o segundo problema e métodos alternativos existem na literatura, inclusive métodos bayesianos (veja, por exemplo, [49]).

Problema 3: Aprendizado

Como treinar o HMM ω tal que, após observar um determinado número de seqüências, ele poderá lembrá-las posteriormente?

O problema equivale a, dado um número p de seqüências de treinamento

$$(y_1^T)^1, \dots, (y_1^T)^p, \quad (3.27)$$

obter um HMM ω^* tal que $\mathcal{P}[(y_1^T)^i, (q_1^T)^i | \omega^*]$ seja um máximo com relação aos parâmetros do HMM para todo i , onde $(q_1^T)^i$ é a seqüência de estados obtida através do algoritmo de Viterbi.

As duas soluções mais comuns encontradas na literatura são conhecidas pelo nome de *Segmental K-means* [47] e *Fórmulas de Reestimação de Baum-Welch*. Este último será apresentado e estudado com mais detalhes no capítulo 4.

3.3 Aplicações

Como é possível observar em [14], ao longo das últimas duas décadas o campo de aplicação dos HMMs apresentou um espantoso crescimento, devido principalmente à obtenção de algoritmos práticos para o aprendizado, abrangendo atualmente diversas áreas do conhecimento. No texto a seguir, procuraremos apresentar uma visão geral das principais aplicações dos HMMs e de como eles são utilizados em cada caso. Note que essas aplicações não esgotam as possibilidades e muitas outras áreas podem se utilizar desses modelos, principalmente aquelas onde a análise de séries temporais de dados é imprescindível [20, 30].

3.3.1 Bioinformática

Em Biologia, o uso dos HMMs têm se destacado em duas grandes áreas: DNA e Proteínas. Descreveremos essas aplicações a seguir.

DNA/RNA

A molécula de DNA é composta de uma fita dupla organizada em forma helicoidal. Cada fita é uma seqüência de 4 diferentes bases nitrogenadas simbolizadas por letras: A (Adenina), T (Timina), C (Citosina) e G (Guanina). Dessa forma, uma fita pode ser considerada simplesmente como uma seqüência de 4 símbolos. As duas fitas contêm basicamente a mesma informação, uma vez que estão ligadas de tal forma que uma base A de uma das

fitas sempre está conectada a uma base T da outra e uma base C sempre está conectada a uma base G.

A molécula de RNA é composta de apenas uma fita e seu formato depende de sua função na célula. Ela também é uma sequência de 4 bases nitrogenadas, a saber, T, C e G como no DNA, mas ao invés de A (Adenina), ela contém a base U (Uracila) que, quando necessário, conecta-se a uma Timina (por exemplo, no processo de duplicação de DNA para a divisão celular).

Estas moléculas são responsáveis pela codificação e decodificação de toda a informação necessária para a construção e funcionamento dos organismos vivos no planeta. Avanços tecnológicos nas últimas duas décadas permitiram que grupos em todo o mundo conseguissem sequenciar essas moléculas em diferentes organismos (isto é, obter a sequência de símbolos correspondente às bases nitrogenadas) e montar bancos de dados que podem inclusive ser acessados através da Internet. No entanto, o grande desafio agora consiste em interpretar tais sequências de símbolos e extrair delas informações úteis. É nesse ponto que os HMMs são aplicados [6, 24] e foram inicialmente introduzidos nessa área por Krogh *et al.* [52].

A informação contida nas moléculas de DNA e RNA varia tanto em quantidade quanto em complexidade e identificar e classificar toda essa informação é uma tarefa extremamente complicada. Algumas das tarefas onde os HMMs têm encontrado larga aplicação nessa área são [6]: localização de genes em procariontes e eucariontes, HMMs de introns, exons e sítios de separação e descoberta de padrões periódicos em exons e introns.

Proteínas

Proteínas são moléculas compostas de longas cadeias de aminoácidos organizados segundo variadas geometrias e que servem para inúmeras funções nos organismos vivos. Todas as proteínas em um determinado organismo são produzidas segundo informações contidas no seu DNA onde sequências de bases, denominadas **códons**, codificam um determinado aminoácido.

Ao todo, existem 20 diferentes aminoácidos que combinados em sequências compõem todas as proteínas possíveis. Assim como no caso do DNA, nas últimas décadas foram acumulando-se vários bancos de dados com informações detalhadas de uma grande quantidade de proteínas, incluindo a sequência de aminoácidos que as formam, denominada de **proteoma**, e detalhes sobre a sua configuração geométrica [78, 80].

Existem muitas aplicações de HMMs nessa área. Os HMMs, por exemplo, podem ser usados para inferir a geometria através da sequência de aminoácidos, considerando que a

geometria pode ser descrita como uma sequência de unidades básicas.

Além disso, assim como os organismos, as proteínas sofrem mutações ao longo da evolução e formam grandes famílias. Muitas vezes, o parentesco entre proteínas é obtido do que se chama **alinhamento**, onde as seqüências de aminoácidos de duas proteínas são comparadas alinhando-os de uma determinada forma convencional. Como essa forma nem sempre é única, os HMMs podem ser usados para escolher-se o melhor alinhamento e obter uma classificação aprimorada das proteínas em suas famílias.

Muitas outras aplicações são feitas e pode-se encontrar a maior parte delas nas referências [6, 24].

3.3.2 Econometria

Uma grande quantidade de esforço é gasta em Economia na tentativa de obter modelos que possam permitir a previsão, dentro de determinados limites, do comportamento de uma grande quantidade de índices referentes a bolsas de valores, cotações de moedas, bonds e outros produtos.

A análise das séries temporais formadas por tais índices é tema de uma enorme literatura em Economia e os HMMs têm muito a contribuir nessa área. Como o número de interações que influencia cada índice é extremamente grande e a forma dessas interações muito complexa, uma modelagem direta é uma tarefa praticamente impossível. Os HMMs podem extrair padrões probabilísticos dessas séries mesmo em situações onde o mercado sofre mudanças. Na seção 5.5 aplicaremos alguns dos algoritmos estudados nesta tese para a previsão do índice de bolsa de valores e discutiremos os resultados obtidos.

3.3.3 Meteorologia

A meteorologia é um caso clássico onde o número de variáveis é extremamente grande e as interações são tão complexas a ponto de incluírem efeitos caóticos (não-lineares). O objetivo, no entanto, é prever um número simples de estados partindo de séries temporais de medidas de temperatura, pressão, quantidade de chuva, entre outras.

Os estados a serem previstos podem ser desde simples previsões binárias como “vai chover”/“não vai chover” até seqüências de condições atmosféricas ou de tempo mais complexas.

3.3.4 Música

HMMs podem ser utilizados para produzir ou identificar músicas de acordo com determinados estilos se treinados com bancos de dados apropriados [16, 76].

3.3.5 Reconhecimento de Fala

A aplicação mais popular dos HMMs atualmente é no reconhecimento da linguagem falada [82]. Existem várias maneiras pelas quais esse reconhecimento pode ser feito, mas a mais comum delas, que se utiliza da característica dos HMMs de modelarem uma sequência de dados através de estados ao qual não temos acesso na aprendizagem, será o que explicaremos a seguir. Tal processo de reconhecimento possui uma série de passos. O primeiro consiste em codificar a palavra falada no formato de entrada correto da máquina que fará o reconhecimento. Seguem-se, então, os processos responsáveis pelo reconhecimento das palavras que incluem análises fonéticas, sintáticas e semânticas. Embora o processo inteiro seja complicado e cada etapa possa ser implementada com técnicas e maneiras diferentes, descreveremos uma forma de aplicação dos HMMs chamada de reconhecimento de palavras isoladas.

Considere que possuímos um vocabulário com P palavras p_i . Para cada palavra, um determinado HMM ω_i é exposto a uma série de exemplos de pronúncia por uma ou mais pessoas e os seus parâmetros são ajustados de tal forma a maximizar a probabilidade das sequências de sons de treinamento. Esse passo pode ser implementado através, por exemplo, das fórmulas de reestimação de Baum-Welch.

Após o treinamento, dada uma palavra falada, ela é quebrada em uma sequência de estados observados y_1^T . Pode-se, então, encontrar o valor exato de

$$w_i = \mathcal{P}(y_1^T | \omega_i), \quad (3.28)$$

através dos procedimentos *forward-backward* ou uma aproximação onde considera-se apenas o caminho mais provável no espaço das variáveis escondidas através algoritmo de Viterbi. O reconhecimento é feito, então, associando à sequência observada a palavra correspondente ao HMM cujo peso w_i seja maior.

Esse método possui o inconveniente de exigir o treinamento de um HMM para cada palavra, o que pode significar um número muito grande (da ordem de 10^5) para tarefas de reconhecimento gerais.

Capítulo 4

Aprendizado Não-Bayesiano

HMMs são modelos extremamente versáteis para análise de dados seqüenciais. No entanto, isso leva a complicações no estudo do aprendizado nesse tipo de modelo. Para HMMs, o aprendizado pode assumir diferentes formas dependendo de qual a aplicação que se deseja fazer. A seguir, propomos que essas formas podem ser divididas em duas grandes classes de problemas de aprendizado, a saber

1) Aprendizado de uma única seqüência

Neste tipo de aprendizado, o conjunto de dados consiste apenas em uma única seqüência de estados observados de tamanho T . Em geral, o algoritmo de aprendizado procura ajustar os parâmetros do aluno de tal forma a maximizar alguma função custo, usualmente a verossimilhança, com base na seqüência dada.

Normalmente, os algoritmos usados para este tipo de aprendizado são algoritmos *off-line*, uma vez que já se tem acesso à seqüência toda.

Este caso é o mais comum para reconhecimento de fala, já que nesse tipo de aplicação apresenta-se uma seqüência completa para a máquina, a palavra falada, e ela deve aprendê-la de tal forma a reconhecê-la posteriormente.

É possível utilizar-se da estratégia de quebrar a seqüência em várias subseqüências menores e utilizar algoritmos *on-line* que aprendem os parâmetros do HMM baseados nessas subseqüências.

2) Aprendizado de $P \neq 1$ seqüências

Este tipo de aprendizado ocorre quando o conjunto de dados é formado por P diferentes seqüências geradas pelo HMM professor e que devem ser aprendidas uma de cada

vez. O tamanho de cada sequência é arbitrário, mas há casos onde é conveniente trabalhar com sequências de um mesmo tamanho fixo, como veremos mais adiante.

Nesse caso, o interesse está em obter um HMM que gere sequências observadas que se pareçam com os exemplos, sendo que o conceito *parecer* nesse caso é traduzido nas probabilidades de determinados símbolos sucederem outros. Isso é o equivalente a estabelecer regras análogas às regras gramaticais em diversas línguas, como por exemplo, a regra em português que diz que antes das letras *p* e *b* utiliza-se, sempre que necessário, a letra *m* e nunca a letra *n*. Dessa analogia, vem o motivo de chamar-se, às vezes, tais regras de **regras gramaticais**.

Aprender, no caso dos HMMs, significa ajustar os parâmetros π , A e B do aluno de tal forma que a probabilidade das sequências de estados observados que podem ser geradas por ele se aproximem daquelas geradas pelo HMM professor. Assumiremos que o HMM aluno e o HMM professor possuem o mesmo número de estados escondidos e observados e que esses números são conhecidos. O caso onde tais números são parâmetros a serem aprendidos e extraídos do conjunto de dados é mais complicado e não será estudado nesta tese.

Faremos a seguir uma análise de cinco diferentes algoritmos de aprendizado para HMMs observando e comparando a sua velocidade de generalização e complexidade computacional. As fórmulas de reestimação de Baum-Welch (seção 4.3) e o algoritmo de Baldi-Chauvin (seção 4.5) já são conhecidos na literatura. Embora seja uma extensão simples e direta das fórmulas de reestimação, não encontramos na literatura referências à versão *on-line* de Baum-Welch (seção 4.4). Os dois algoritmos restantes, o primeiro um algoritmo bayesiano baseado no trabalho de Oppen [74] (seção 5.1) e o segundo uma simplificação deste que remete a uma técnica conhecida como posterior média (seção 5.2), são dois algoritmos bayesianos propostos por nós.

Obviamente, os algoritmos estudados nesta tese não são os únicos existentes e é possível encontrar inúmeras outras propostas (veja, por exemplo, [3, 17, 21, 27, 58, 92, 98, 108]). No entanto, os algoritmos analisados aqui compreendem os mais simples e mais utilizados atualmente para a maior parte das aplicações feitas.

4.1 Distância entre HMMs

Como discutido na seção 2.5, um objeto de grande interesse no estudo do aprendizado em máquinas é o erro de generalização de um determinado modelo. O erro de generalização é um meio quantitativo de sabermos se o modelo em questão está ou não aprendendo a

partir do conjunto de dados e com que velocidade isto está acontecendo.

No caso do perceptron, isso é medido através de uma função do produto escalar entre o vetor que representa os pesos sinápticos do professor e o do aluno e, para esse caso específico, é definido como a probabilidade do aluno classificar um exemplo de forma diferente do professor [25].

Para HMMs, no entanto, o aprendizado ocorre de maneira não-supervisionada e, para a maioria das aplicações, não consiste em um par exemplo-classe, ou seja, não há como definir o erro de generalização com base em erros de classificação.

Sendo assim, o que procuramos é alguma função que possa nos dar uma medida de quão diferentes entre si são dois HMMs, isto é, que nos dê uma indicação da distância entre eles. A seguir, analisaremos as duas medidas de distância (num sentido matemático não-rigoroso) entre HMMs que usaremos em nosso trabalho destacando suas vantagens e desvantagens. Existem outras possíveis formas de medir a distância entre dois HMMs, mas que não serão discutidas aqui (veja, por exemplo, [4, 106])

1) Divergência de Kullback-Leibler

Como os HMMs são modelos probabilísticos, a idéia mais natural para uma medida de distância entre eles baseia-se no uso de medidas de divergência entre duas distribuições de probabilidades. Existe, na literatura, uma enorme quantidade de medidas de divergência entre distribuições de probabilidades [75], mas a mais utilizada entre elas, principalmente devido à sua relação com teoria de informação, é a chamada **divergência de Kullback-Leibler** [19, 55, 56, 61], que é uma medida de dissimilaridade entre duas distribuições de probabilidades definida da seguinte forma:

Definição 4.1 *Sejam $p_1(x)$ e $p_2(x)$ duas distribuições de probabilidade sobre a variável x . A **divergência de Kullback-Leibler** entre essas duas distribuições é dada por*

$$D_{KL}(p_1||p_2) = \int_{-\infty}^{+\infty} p_1(x) \ln \left[\frac{p_1(x)}{p_2(x)} \right] dx, \quad (4.1)$$

no caso contínuo e por

$$D_{KL}(p_1||p_2) = \sum_x p_1(x) \ln \left[\frac{p_1(x)}{p_2(x)} \right], \quad (4.2)$$

no caso discreto.

Em muitos textos, principalmente em Teoria de Informação, D_{KL} também é conhecida como **entropia cruzada** [85]. É importante notar que essa medida de distância não

é simétrica e, portanto, não pode ser associada a uma métrica no sentido matemático rigoroso. Embora possamos definir uma distância simétrica através da expressão

$$D_s(p_1||p_2) \equiv D_{KL}(p_1||p_2) + D_{KL}(p_2||p_1), \quad (4.3)$$

a utilidade disto é discutível e dificilmente faremos uso desta expressão ao longo do nosso trabalho.

De forma mais rigorosa, deveríamos exigir que as distribuições de probabilidades fossem absolutamente contínuas uma em relação a outra, o que significa que elas nunca se anulam em pontos diferentes. Com esta exigência, torna-se possível demonstrar várias propriedades de D_{KL} , entre elas, a de que essa distância é sempre maior ou igual a zero, sendo que a igualdade se dá se e somente se as duas distribuições de probabilidade forem idênticas, ou seja, ela é não-degenerada [56].

Consideremos então dois HMMs discretos, os quais simbolizaremos por ω_1 e ω_2 . Intuitivamente, queremos uma medida de dissimilaridade entre ambos tal que, quanto mais parecidas forem as distribuições das seqüências de estados observados por eles geradas, mais próximos eles serão considerados. Uma maneira natural de definirmos essa medida de dissimilaridade é usarmos a distância de Kullback-Leibler entre as distribuições dos estados observados

$$d_{KL}(\omega_1, \omega_2) \equiv \sum_{y_1^T} \mathcal{P}(y_1^T|\omega_1) \ln \left[\frac{\mathcal{P}(y_1^T|\omega_1)}{\mathcal{P}(y_1^T|\omega_2)} \right], \quad (4.4)$$

onde a soma é feita sobre todas as possíveis seqüências observáveis de *tamanho fixo* T . Não encontramos na literatura resultados analíticos com relação a distâncias baseadas em D_{KL} , salvo em situações particulares (geralmente modelos simplificados), como em [94], por exemplo.

No entanto, existem várias aproximações sugeridas para o cálculo de distâncias baseadas em D_{KL} . Descreveremos a seguir a mais conhecida delas, sugerida por Juang e Rabiner [46] e apresentada de forma simplificada em [23]. Existem outras menos utilizadas que podem ser encontradas em [28, 105].

A primeira aproximação a ser feita consiste em aproximar a soma sobre as seqüências de estados escondidos no argumento do logaritmo pelo termo correspondente à seqüência de estados escondidos ótima q_1^{T*} , que pode ser calculada pelo algoritmo de Viterbi (seção 3.2) para cada HMM. Assim

$$\sum_{y_1^T} \mathcal{P}(y_1^T|\omega_1) \ln \left[\frac{\mathcal{P}(y_1^T|\omega_1)}{\mathcal{P}(y_1^T|\omega_2)} \right] \approx \sum_{y_1^T} \mathcal{P}(y_1^T|\omega_1) \ln \left[\frac{P^*(y_1^T|\omega_1)}{P^*(y_1^T|\omega_2)} \right], \quad (4.5)$$

onde

$$P^*(y_1^T | \omega_1) \equiv \mathcal{P}(y_1^T, q_1^{T*} | \omega_1). \quad (4.6)$$

Note que a expressão representa uma média do logaritmo com relação à distribuição $\mathcal{P}(y_1^T | \omega_1)$. Assim, se gerarmos aleatoriamente uma sequência de estados e assumirmos a ergodicidade da cadeia de Markov subjacente, podemos definir uma medida de dissimilaridade para valores grandes de T como

$$D_{JR}(\omega_1, \omega_2) \equiv \lim_{T \rightarrow \infty} \frac{1}{T} [\ln P^*(y_1^T | \omega_1) - \ln P^*(y_1^T | \omega_2)], \quad (4.7)$$

e a distância simetrizada como

$$D_{JR}^s(\omega_1, \omega_2) \equiv \frac{1}{2} [D_{JR}(\omega_1, \omega_2) + D_{JR}(\omega_2, \omega_1)]. \quad (4.8)$$

Note bem que as seqüências são geradas sempre pelo HMM que entra como primeiro argumento na fórmula da distância. Simulações mostram que o termo entre colchetes na fórmula (4.7) se aproxima de um limite para $T \sim 4000$ [23] para qualquer valor de n e m .

Note que a distância de Kullback-Leibler definida pela equação (4.4) pode ser interpretada como a média no conjunto de todas as possíveis seqüências observadas geradas pelo professor da função custo

$$\epsilon \equiv [\ln \mathcal{P}(y_1^T | \omega_\tau) - \ln \mathcal{P}(y_1^T | \omega_\sigma)] = \ln \frac{\mathcal{P}(y_1^T | \omega_\tau)}{\mathcal{P}(y_1^T | \omega_\sigma)}, \quad (4.9)$$

onde ω_τ representa os parâmetros do professor e ω_σ os parâmetros do aluno. Isso justifica considerarmos $d_{KL}(\omega_\tau, \omega_\sigma)$ como sendo o erro de generalização para o HMM

$$\epsilon_g = \langle \epsilon \rangle = d_{KL}(\omega_\tau, \omega_\sigma), \quad (4.10)$$

onde

$$\langle \cdot \rangle \equiv \sum_{y_1^T} (\cdot) \mathcal{P}(y_1^T | \omega_\tau). \quad (4.11)$$

Usando essa interpretação, poderíamos ser levados a definir, de forma análoga ao que é feito para o perceptron [25], o **erro de treinamento** do HMM como a média da função custo sobre o conjunto de dados usados para o treinamento

$$\epsilon_t \equiv \langle \epsilon \rangle_{D_P}, \quad (4.12)$$

onde $D_P = \{y^1, \dots, y^P\}$ é o conjunto das P seqüências apresentadas durante a fase de treinamento. No entanto, como a média não é feita sobre todas as seqüências possíveis, o

erro de treinamento definido dessa forma pode não ser positivo, o que torna a sua interpretação pouco clara e, sendo assim, não o utilizaremos em nosso trabalho.

Um outro problema com relação à distância $d_{KL}(\omega_1, \omega_2)$ ocorre quando existe a possibilidade de que $\mathcal{P}(y_1^T|\omega_2) = 0$ e $\mathcal{P}(y_1^T|\omega_1) \neq 0$ para a mesma seqüência de estados observados, ou seja, quando elas não são absolutamente contínuas uma em relação à outra. Nesse caso, $d_{KL}(\omega_1, \omega_2)$ é infinita. Embora muitos algoritmos restrinjam os parâmetros do HMM de forma que não existam probabilidades nulas, isso pode acontecer em particular no algoritmo de Baum-Welch e, dessa forma, não será possível utilizarmos essa distância para calcular curvas de aprendizado nesses casos.

2) Distância Euclideana

Uma alternativa mais rápida à distância d_{KL} , porém menos justificada do ponto de vista conceitual, é utilizar a chamada **distância euclideana** entre as linhas da matriz de emissão B [28], dada por

$$d_E(\omega_1, \omega_2) \equiv \sqrt{\frac{1}{n \cdot m} \sum_{i=1}^n \sum_{j=1}^m |B_{ij}^1 - B_{ij}^2|^2}, \quad (4.13)$$

onde B^1 e B^2 são, respectivamente, as matrizes de emissão de ω_1 e ω_2 .

Essa distância é obviamente positiva e simétrica, no entanto, dois HMMs com a mesma distribuição de probabilidades para as seqüências observadas podem ter uma distância euclideana diferente de zero.

A vantagem da utilização dessa distância é o fato de que não são necessárias somas sobre os estados escondidos e, portanto, a distância pode ser calculada rapidamente sem nenhuma aproximação. Além disso, ela não depende do particular tipo de aprendizado ou do tamanho das seqüências observadas.

Nas próximas seções, ao estudarmos diferentes algoritmos de aprendizado para HMMs, faremos comparações entre as duas distâncias apresentadas nesta seção e as suas vantagens e desvantagens para cada algoritmo e cada caso de aprendizado ficarão mais claras.

4.2 Simulações

A maior parte das simulações feitas nesta tese representam curvas comparativas do desempenho dos algoritmos. Para tanto, uma ou mais seqüências observadas são geradas

artificialmente através de um HMM pré-definido (chamado de HMM professor) e compõem o conjunto de dados que será utilizado pelo HMM aluno para aprender de acordo com cada algoritmo.

Para cada conjunto de dados D_P aprendido com determinado algoritmo, contruiremos uma **curva de aprendizado**, um gráfico cujo eixo horizontal representará o tamanho da sequência aprendida no caso *off-line* ou o número de sequências aprendidas no caso *on-line*. O eixo vertical contém os valores da distância entre aluno e professor, que será uma das duas distâncias descritas na seção anterior.

Simbolizaremos por N_r o número de professores utilizados em cada simulação. Isto significa que cada um desses professores irá gerar um conjunto de dados D_P que será utilizado para o aprendizado. A curva de aprendizado final será dada pela média das curvas para cada professor. Os professores podem ser gerados aleatoriamente segundo alguma distribuição ou definidos de acordo com o que se deseja estudar em cada situação.

Em alguns casos, traçaremos curvas com um número N_σ de alunos. Isso significará que a simulação inteira será feita para cada aluno e que o resultado final será a média sobre todas as curvas.

Os detalhes específicos de cada simulação feita serão explicitados no respectivo texto. Os estudos incluirão também outros tipos de simulações que também serão explicadas em detalhes quando necessário.

4.3 Fórmulas de Reestimação de Baum-Welch

4.3.1 Algoritmo

As **fórmulas de reestimação de Baum-Welch** (ou simplesmente **algoritmo BW**)[23, 81] formam um algoritmo *off-line* resultante da aplicação do algoritmo EM (seção 2.3.3) ao caso particular dos HMMs [10, 24]. Se os parâmetros de um HMM ω forem simbolizados de forma geral por um vetor θ com componentes θ_i , o passo de maximização (ou Passo M) do algoritmo EM para este caso particular implica na maximização da função

$$\mathcal{L} = \sum_q \mathcal{P}(q|y, \theta) \ln \mathcal{P}(q, y|\hat{\theta}), \quad (4.14)$$

onde y é a sequência observada para a qual faremos a maximização requerida e $\hat{\theta}$ são os novos parâmetros. Utilizando multiplicadores de Lagrange para implementar os vínculos correspondentes ao fato de que os parâmetros são probabilidades, o algoritmo EM fornece fórmulas que permitem reestimar os parâmetros do HMM iterativamente até que um ponto fixo seja atingido. As fórmulas resultantes da extremização são dadas por

$$\begin{aligned}
\hat{\pi}_i &= \gamma_1(i), \\
\hat{A}_{ij} &= \frac{\sum_{t=1}^{T-1} \xi_t(i, j)}{\sum_{t=1}^{T-1} \gamma_t(i)}, \\
\hat{B}_{ij} &= \frac{\sum_{t=1, y_t=o_j}^T \gamma_t(i)}{\sum_{t=1}^T \gamma_t(i)}.
\end{aligned} \tag{4.15}$$

O chapéu em cima das variáveis indica que elas são os parâmetros reestimados do modelo. As variáveis γ e ξ são definidas como

$$\begin{aligned}
\gamma_t(i) &= \mathcal{P}(q_t = s_i | y_1^T, \omega) = \\
&= \frac{\mathcal{P}(q_t = s_i, y_1^T | \omega)}{\mathcal{P}(y_1^T | \omega)},
\end{aligned} \tag{4.16}$$

e

$$\begin{aligned}
\xi_t(i, j) &= \mathcal{P}(q_t = s_i, q_{t+1} = s_j | y_1^T, \omega) = \\
&= \frac{\mathcal{P}(q_t = s_i, q_{t+1} = s_j, y_1^T | \omega)}{\mathcal{P}(y_1^T | \omega)},
\end{aligned} \tag{4.17}$$

onde $\mathcal{P}(y_1^T | \omega)$ pode ser calculada tanto pela *forward procedure* quanto pela *backward procedure* (seção 3.2).

Observe que

$$\begin{aligned}
\mathcal{P}(q_t = s_i, y_1^T | \omega) &= \mathcal{P}(y_{t+1}^T | q_t = s_i, y_1^t, \omega) \mathcal{P}(q_t = s_i, y_1^t | \omega) = \\
&= \mathcal{P}(y_{t+1}^T | q_t = s_i, \omega) \mathcal{P}(y_1^t, q_t = s_i | \omega) = \\
&= \alpha_t(i) \beta_t(i),
\end{aligned} \tag{4.18}$$

e

$$\begin{aligned}
\mathcal{P}(q_t = s_i, q_{t+1} = s_j, y_1^T | \omega) &= \mathcal{P}(q_t = s_i, q_{t+1} = s_j, y_1^t, y_{t+1}, y_{t+2}^T | \omega) \\
&= \mathcal{P}(y_{t+2}^T | q_t = s_i, q_{t+1} = s_j, y_1^t, y_{t+1}, \omega) \cdot \\
&\quad \cdot \mathcal{P}(q_t = s_i, q_{t+1} = s_j, y_1^t, y_{t+1} | \omega) \\
&= \beta_{t+1}(j) \mathcal{P}(y_{t+1} | q_t = s_i, q_{t+1} = s_j, y_1^t, \omega) \cdot \\
&\quad \cdot \mathcal{P}(q_t = s_i, q_{t+1} = s_j, y_1^t | \omega) \\
&= \beta_{t+1}(j) \mathcal{P}(y_{t+1} | q_{t+1} = s_j, \omega) \cdot \\
&\quad \cdot \mathcal{P}(q_{t+1} = s_j | q_t = s_i, y_1^t, \omega) \mathcal{P}(q_t = s_i, y_1^t | \omega) \\
&= \beta_{t+1}(j) \mathcal{P}(q_{t+1} = s_j | q_t = s_i, \omega) \mathcal{P}(y_{t+1} | q_{t+1} = s_j, \omega) \alpha_t(i),
\end{aligned} \tag{4.19}$$

lembrando que

$$\begin{aligned}\alpha_t(i) &= \mathcal{P}(y_1^t, q_t = s_i | \omega), \\ \beta_t(i) &= \mathcal{P}(y_{t+1}^T | q_t = s_i, \omega).\end{aligned}\quad (4.20)$$

Dessa forma, podemos escrever

$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{\mathcal{P}(y_1^T | \omega)}, \quad (4.21)$$

e

$$\xi_t(i, j) = \frac{\alpha_t(i)\mathcal{P}(q_{t+1} = s_j | q_t = s_i, \omega)\mathcal{P}(y_{t+1} | q_{t+1} = s_j, \omega)\beta_{t+1}(j)}{\mathcal{P}(y_1^T | \omega)}. \quad (4.22)$$

Note que

$$\sum_{t=1}^{T-1} \gamma_t(i) = \sum_{t=1}^{T-1} \mathcal{P}(q_t = s_i | y_1^T, \omega), \quad (4.23)$$

é o número de transições a partir do estado s_i ,

$$\sum_{t=1}^T \gamma_t(i) = \sum_{t=1}^T \mathcal{P}(q_t = s_i | y_1^T, \omega), \quad (4.24)$$

é o número de vezes em que ocorre o estado s_i e

$$\sum_{t=1}^{T-1} \xi_t(i, j) = \sum_{t=1}^{T-1} \mathcal{P}(q_t = s_i, q_{t+1} = s_j | y_1^T, \omega), \quad (4.25)$$

é o número de transições do estado s_i para o estado s_j .

Durante as iterações, apenas uma das duas alternativas abaixo ocorre:

1. O modelo inicial é um ponto fixo do algoritmo, caso em que $\hat{\theta} = \theta$;
2. $\mathcal{P}(y | \hat{\theta}) > \mathcal{P}(y | \theta)$;

o que significa que a cada passo de iteração a verossimilhança do modelo dada a sequência observada aumenta ou se estabiliza, nunca decrescendo.

As expressões acima nos permitem observar que as fórmulas de reestimação podem ser interpretadas simplesmente como uma contagem de frequência, ou seja, os novos parâmetros reestimados são obtidos calculando as frequências médias com as quais ocorrem transições e emissões na sequência de aprendizado. Esse procedimento tem uma desvantagem que, em inglês, costuma-se chamar de *overfitting*, que poderia ser traduzido como

sobreajuste, e que significa que os parâmetros do modelo reproduzirão apenas seqüências contidas nos dados, excluindo qualquer outra que não esteja neles representada e, portanto, não obtendo uma boa generalização para seqüências que nunca foram observadas pelo HMM aluno.

A cada passo de iteração, são necessárias:

- a) $n + n(n + 1)(T - 1)$ multiplicações para se obter ao mesmo tempo todos os $\alpha_t(i)$ e mais $\mathcal{P}(y_1^T)$;
- b) $2n^2(T - 1)$ multiplicações para a obtenção de todos os $\beta_t(i)$;
- c) $2nT$ multiplicações para calcular todos os $\gamma_t(i)$, uma vez que, dados $\alpha_t(i)$ e $\beta_t(i)$, cada $\gamma_t(i)$ necessita de mais 2 multiplicações (mais precisamente, uma multiplicação e uma divisão, a qual, para fins de cálculo de complexidade computacional, contaremos como multiplicação);
- d) $4n^2(T - 1)$ multiplicações para calcular $\xi_t(i, j)$ e
- e) $2n^2$ multiplicações que correspondem às divisões de somas em $\hat{A}$ e $\hat{B}$.

Somando todas as multiplicações necessárias, concluímos que cada passo, e conseqüentemente o algoritmo inteiro, é polinomial de ordem 2 em n e de ordem 1 em T .

Pelas fórmulas (4.15) também é possível perceber que podem ocorrer casos onde os parâmetros acabam assumindo o valor zero, o que impossibilita o uso de d_{KL} para medir a distância entre o aluno e o professor. Isso é facilmente notado se observarmos que na fórmula de reestimação para a matriz B , dada pela terceira equação em (4.15), a soma no numerador é restrita a $y_t = o_j$. Caso o estado o_j não apareça na seqüência observada, a soma torna-se zero e a probabilidade de qualquer seqüência onde tal estado seja observado também é zero.

4.3.2 Simulações

A seguir, apresentaremos simulações com o intuito de avaliar a performance do algoritmo em tarefas de aprendizado. Para os cinco algoritmos que estudamos neste trabalho, adotaremos algumas convenções.

Utilizaremos o termo **aluno inicial simétrico** para indicar um aluno inicial onde todas as transições, emissões e estados iniciais são equiprováveis, isto é, todas as entradas em

cada uma das matrizes π , A e B têm o mesmo valor.

Nas curvas de aprendizado apresentadas, o símbolo d representa a distância entre o professor e o aluno, que pode ser d_{KL} ou d_E dependendo da situação (que será explicitada no texto referente).

Para o algoritmo BW, utilizaremos como medida de distância a distância euclidiana d_E . A justificativa para a utilização de d_E ao invés de d_{KL} neste caso se deve ao fato de que, após o término do aprendizado pelas fórmulas de reestimação, como já observado, algumas seqüências que têm uma probabilidade não nula de serem geradas pelo professor podem ter probabilidade nula de serem geradas pelo aluno, o que faz com que d_{KL} simplesmente seja infinita quando isto acontecer, mesmo que apenas para uma seqüência. Isso não só é possível de acontecer, como observamos que realmente acontece na grande maioria dos casos, o que nos impossibilita de traçar uma curva de aprendizado com essa medida.

Curva de Aprendizado

As fórmulas de reestimação de Baum-Welch permitem o aprendizado de apenas uma seqüência de estados observados. Assim, a curva de aprendizado que obteremos para esse algoritmo será dada pelo gráfico da distância d entre o HMM professor (aquele que gerou a seqüência utilizada como dado) e o HMM aluno em função do tamanho da seqüência aprendida, que será simbolizado por T .

Assim, para construirmos o gráfico mostrado na figura 4.1, simulamos o aprendizado para um aluno inicial simétrico e fizemos a média em um número $N_T = 110$ professores. Tanto para o HMM professor quanto para o HMM aluno foram utilizados 2 estados escondidos ($n = 2$) e 3 estados observados ($m = 3$).

Ambos os gráficos, normal e log-log, parecem indicar um limite no aprendizado, correspondendo à linha $d = 0.3$ tracejada. No entanto, não podemos tirar a conclusão de que o modelo pára de aprender antes de atingir uma generalização completa, pois devido à forma como d_E é definida, dois HMMs podem ter as mesmas probabilidades para todas as seqüências emitidas e ainda assim terem $d_E \neq 0$ por terem matrizes de emissão diferentes (veja seção 4.1).

Note que o erro de generalização para esse caso se aproxima do valor limite 0.3 com um comportamento $\sim 1/T^\xi$ com $\xi = 0.69$.

Figura 4.1: **Algoritmo de Baum-Welch:** a) Gráfico mostrando a distância euclidiana entre o professor e o aluno após a aplicação do algoritmo para uma sequência de T símbolos fazendo a média em 110 professores com 2 estados escondidos e 3 estados observados. A linha tracejada mostra o limite do aprendizado; b) os mesmos dados do gráfico *a* subtraindo-se o valor limite 0.3, porém em log-log. A linha tracejada indica o ajuste de uma lei de potência $(d - 0.3) = \alpha T^{-\xi}$ para os últimos 100 pontos com resultados $\alpha = 0.25$ e $\xi = 0.69$.

Figura 4.2: **Algoritmo de Baum-Welch:** a) O gráfico ilustra o efeito de aumentar o número de estados escondidos do HMM; b) mesmo gráfico em escala log-log.

Número de Estados Escondidos

Para observar como o comportamento pode mudar com o número de estados escondidos (n), fizemos a mesma simulação apresentada na figura 4.1 para 3 valores diferentes de n . A figura 4.2 mostra os resultados obtidos. Como o HMM aluno continua tendo o mesmo número de estados escondidos e observados do professor, o fato da distância na qual o aprendizado pára estar crescendo com n pode ser entendido se observarmos que, à medida que n cresce, cresce também o número de linhas na matriz de emissão, ou seja, existe um maior número de matrizes de emissão diferentes capazes de gerar as mesmas probabilidades para as mesmas seqüências emitidas, o que faz com que d_E seja, em média, maior.

Ainda assim, pode-se notar que o comportamento qualitativo das curvas não muda para um número diferente de estados escondidos. Esse comportamento com relação ao número de estados escondidos se repete em todos os algoritmos estudados e, sendo assim, apresentaremos apenas os resultados das simulações para valores pequenos de n e m por serem mais rápidas.

Influência das Condições Iniciais

A figura 4.3 mostra o resultado comparativo entre uma simulação onde o aluno inicial é simétrico e uma onde fazemos a média em $N_\sigma = 50$ alunos iniciais aleatórios (distribuídos uniformemente). Os valores utilizados foram $n = 2$, $m = 3$, $N_\tau = 100$ e a distância medida foi d_E entre os HMMs aluno e professor. Neste caso, em particular, a performance do aluno inicial simétrico é superior à média.

Observações Gerais

Os gráficos acima confirmam a noção intuitiva de que, quanto maior a seqüência utilizada como exemplo, mais informação consegue-se extrair sobre os parâmetros do HMM e melhor é o aprendizado, ou seja, mais próximo fica o aluno do professor no caso do aprendizado *off-line*.

4.4 Método de Baum-Welch *On-line*

4.4.1 Algoritmo

Vimos, na seção anterior, que cada iteração para as fórmulas de reestimação de Baum-Welch (4.15) representa um passo na direção de um máximo local de $\mathcal{P}(q, y)$, onde y é a

Figura 4.3: **Algoritmo de Baum-Welch:** a) O gráfico mostra a comparação entre duas simulações com alunos iniciais diferentes do algoritmo de Baum-Welch: os círculos mostram o resultado obtido com um aluno inicial simétrico e os pontos com as respectivas barras de erro mostram a média em 50 alunos iniciais aleatórios; b) o mesmo gráfico em escala log-log.

seqüência de tamanho T gerada pelo HMM professor e utilizada como dado no aprendizado. Embora essas fórmulas formem um algoritmo *off-line* onde apenas uma seqüência pode ser aprendida, propomos nesta seção uma modificação de tal forma a torná-lo *on-line*, permitindo que o HMM aluno aprenda, através dele, com um conjunto de $P > 1$ seqüências observadas.

Essa versão *on-line*, que chamaremos de **Método de Baum-Welch On-line**, ou simplesmente **BWO**, pode ser obtida trocando-se y em cada iteração, ou seja, a cada vez que os parâmetros são reestimados através das fórmulas (4.15), por y^p , a p -ésima seqüência observada, e usar o incremento das fórmulas de reestimação multiplicado por um parâmetro η_{BW} positivo, que fará o papel da taxa de aprendizado. Esse procedimento equivale a usar o método de descida pelo gradiente para a função a ser extremizada, que no caso é $\mathcal{P}(q, y)$ (ver apêndice A).

As novas fórmulas ficam, então

$$\begin{aligned}\hat{\pi}_i^{p+1} &= \hat{\pi}_i^p + \eta_{BW} \hat{\Delta} \pi_i^p, \\ \hat{A}_{ij}^{p+1} &= \hat{A}_{ij}^p + \eta_{BW} \hat{\Delta} A_{ij}^p, \\ \hat{B}_{ij}^{p+1} &= \hat{B}_{ij}^p + \eta_{BW} \hat{\Delta} B_{ij}^p,\end{aligned}\tag{4.26}$$

onde

$$\begin{aligned}\hat{\Delta} \pi_i^p &= \gamma_1(i) - \hat{\pi}_i^p, \\ \hat{\Delta} A_{ij}^p &= \frac{\sum_{t=1}^{T-1} \xi_t(i, j)}{\sum_{t=1}^{T-1} \gamma_t(i)} - \hat{A}_{ij}^p, \\ \hat{\Delta} B_{ij}^p &= \frac{\sum_{t=1, y_t=o_j}^T \gamma_t(i)}{\sum_{t=1}^T \gamma_t(i)} - \hat{B}_{ij}^p,\end{aligned}\tag{4.27}$$

são as variações dos parâmetros dadas pelas fórmulas de reestimação de Baum-Welch (4.15) com as funções γ e ξ calculadas usando-se a seqüência observada y^p . Note que para $\eta_{BW} = 1$, as fórmulas são exatamente iguais às fórmulas de reestimação originais.

Usando a notação

$$\tilde{\Delta} \theta^p \equiv \theta^p - \theta^{p-1},\tag{4.28}$$

para a variação dos parâmetros em BWO, fica claro que essa variação é simplesmente proporcional à variação obtida pelas fórmulas de reestimação de Baum-Welch.

A complexidade computacional deste algoritmo para cada passo é exatamente a mesma das fórmulas de reestimação de Baum-Welch, ou seja, polinomial de ordem 2 em n e de ordem 1 em T .

4.4.2 Simulações

Curvas de Aprendizado On-line

Para montarmos a curva de aprendizado para este algoritmo, analisaremos primeiramente a situação onde $P > 1$ seqüências de tamanho fixo T são emitidas por um número N_τ de professores gerados aleatoriamente. As distâncias obtidas são, então, médias tomadas nos professores.

A figura 4.4 mostra a curva de aprendizado obtida nesta situação para os valores $n = 2$, $m = 3$, $T = 2$ e $N_\tau = 500$. Utilizamos seis diferentes valores para a taxa de aprendizado: $\eta_{BW} = 0.3, 0.1, 0.05, 0.01, 0.005, 0.001$. A medida de distância entre os HMMs aluno e professor utilizada neste caso foi a distância de Kullback-Leibler d_{KL} e o aluno inicial é simétrico. Os gráficos mostram dois efeitos interessantes da diminuição da taxa de aprendizado. O primeiro efeito diz respeito à estabilidade do algoritmo. Nota-se que, quanto menor a taxa de aprendizado, menores são as flutuações no valor da distância entre o professor e o aluno observadas para valores grandes do número de seqüências, o que pode ser explicado pelo fato de que o aluno fica oscilando em torno de um mínimo local, mas não consegue se fixar nele por dar passos grandes demais. O segundo efeito, que é bem conhecido, é o de que, quanto menor a taxa de aprendizado, mais lentamente o aluno se aproxima do professor, no entanto, mais próximo do professor real ele chega.

A figura 4.5 mostra os mesmos gráficos, porém utilizando como medida de distância a distância euclidiana d_E . É claramente visível que os efeitos descritos acima observados para a distância d_{KL} continuam valendo para d_E e os gráficos são qualitativamente os mesmos, embora quantitativamente sejam diferentes. Isso parece ser um argumento em favor de se utilizar d_E para casos onde o tamanho das seqüências é muito grande, o que faz com que o cálculo de d_{KL} seja extremamente custoso. Continuaremos com tais comparações nos próximos algoritmos analisados para verificarmos se isso é justificado de uma forma geral.

Influência das Condições Iniciais

É esperado que o comportamento do aprendizado mude se mudarmos o aluno inicial. Sendo assim, a figura 4.6 compara a curva de aprendizado do aluno inicial simétrico com uma curva obtida através da média de $N_\sigma = 60$ alunos iniciais gerados aleatoriamente. Os outros valores para a simulação foram $n = 2$, $m = 3$, $N_\tau = 60$ e $\eta_{BW} = 0.01$. Pode-se notar que o aluno inicial simétrico comporta-se, para fins de aprendizado, pior que a média. Note que a curva de aprendizado para a média nos alunos não mostra, como no

Figura 4.4: **Método de Baum-Welch On-line:** a) Gráfico mostrando a distância de Kullback-Leibler entre o professor e o aluno após a aplicação do algoritmo para 4000 seqüências de 2 símbolos fazendo a média em 500 professores com 2 estados escondidos e 3 estados observados. Os números próximos a cada linha indicam o valor de η_{BW} utilizado. b) O mesmo que o gráfico a, porém em log-log.

Figura 4.5: **Método de Baum-Welch On-line:** a) Gráfico mostrando a distância Euclidiana entre o professor e o aluno após a aplicação do algoritmo para 4000 seqüências de 2 símbolos fazendo a média em 500 professores com 2 estados escondidos e 3 estados observados. Os números próximos a cada linha indicam o valor de η_{BW} utilizado. b) O mesmo que o gráfico a, porém em log-log.

Figura 4.6: **Método de Baum-Welch On-line:** a) Gráfico comparando a curva de aprendizado de um aluno inicial simétrico (linha tracejada) com a média de 60 alunos aleatórios (linha contínua); b) o mesmo gráfico em escala log-log.

Figura 4.7: **Método de Baum-Welch On-line:** Comparação entre o algoritmo BW e o algoritmo BWO adaptado para aprender uma sequência de tamanho T quebrando-a em subsequências menores com tamanho indicado junto às curvas correspondentes.

caso do aluno simétrico, uma parada no aprendizado a partir de um certo ponto, mas sim continua aprendendo indefinidamente. A explicação desse efeito está relacionada com a quebra de simetria entre os parâmetros do aluno durante o aprendizado e será feita com mais detalhes na seção 5.6. O argumento básico é de que o aprendizado ocorre quando ocorrem tais quebras de simetria e, no caso de alunos que não sejam simétricos, a simetria nos parâmetros já está quebrada desde o começo, resultando num aprendizado mais rápido [89].

Curvas de Aprendizado Off-line

Analisamos, a seguir, o que acontece quando aplicamos o algoritmo BWO numa situação onde temos apenas uma sequência de aprendizado de tamanho T . Nesse caso, utiliza-

Figura 4.8: **Método de Baum-Welch *On-line***: Valor médio da distância entre aluno e professor nos últimos 100 pontos das curvas de aprendizado de BWO para seqüências únicas com tamanho de 2 até 100 símbolos em função do tamanho das subsequências.

remos a estratégia de quebrar essa seqüência em subsequências menores para podermos aplicar o algoritmo na sua forma *on-line*. A figura 4.7 mostra um gráfico comparando a distância d_E entre o professor e o aluno para o algoritmo BW e para BWO com cinco tamanhos diferentes de subsequências indicados ao lado da curva correspondente: 2, 4, 6, 8 e 10. O gráfico para BW é o mesmo da figura 4.1 e os parâmetros utilizados para as simulações de BWO foram $n = 2$, $m = 3$, $N_\tau = 50$ e $\eta_{BW} = 10^{-4}$ com alunos iniciais simétricos.

Note que nos três primeiros casos, onde o tamanho das subsequências é respectivamente 2, 4 e 6, o valor final da distância cai com o tamanho da subsequência, chegando até mesmo a superar o algoritmo *off-line* BW. No entanto, quando simulamos para subsequências de tamanho igual a 8 e 10, embora o desempenho continue superior à de BW, ele é pior que as simulações anteriores. Para analisarmos de uma forma melhor o que

acontece, fizemos um gráfico com a média de d_E nos últimos 100 pontos de cada curva de BWO em função do tamanho da subsequência. Esse gráfico é mostrado na figura 4.8, onde podemos notar que o valor de d_E oscila sem um padrão aparente, embora não tenhamos feito estudos mais detalhados desses dados.

Assim, embora existam casos em que BWO supere BW dependendo do tamanho da subsequência, as simulações feitas não parecem indicar uma regularidade que sirva como base para obter o tamanho ideal de subsequência tal que BWO atinja um aprendizado ótimo.

4.5 Algoritmo de Baldi-Chauvin

4.5.1 Algoritmo

O **algoritmo de Baldi-Chauvin** [5], ao qual nos referiremos a partir deste ponto pela sigla **BC**, é um algoritmo *on-line* que se utiliza do método de descida pelo gradiente (ver apêndice A) para obter uma estimativa de ML para os parâmetros de um HMM. Além disso, o algoritmo se utiliza de uma representação do tipo *softmax* [60] para os parâmetros do HMM introduzindo assim uma temperatura efetiva λ que regula a intensidade das flutuações ao longo da trajetória de maximização. Assim, para um HMM $\omega = \{S, O, \pi, A, B\}$, temos

$$\pi_i = \frac{e^{\lambda p_i}}{\sum_j e^{\lambda p_j}}, \quad A_{ij} = \frac{e^{\lambda a_{ij}}}{\sum_k e^{\lambda a_{ik}}}, \quad B_{i\alpha} = \frac{e^{\lambda b_{i\alpha}}}{\sum_\beta e^{\lambda b_{i\beta}}}. \quad (4.29)$$

Consideremos o conjunto de dados como contendo P seqüências observadas independentes dadas por

$$D_P = \{y^1, y^2, \dots, y^P\}. \quad (4.30)$$

O algoritmo consiste em fazer incrementos aos parâmetros $u = (p, a, b)$ na direção que maximiza a verossimilhança de todo o conjunto de dados. O gradiente da verossimilhança $\mathcal{L}$, será dado por

$$\nabla \mathcal{L} = \left(\frac{\partial \mathcal{L}}{\partial p}, \frac{\partial \mathcal{L}}{\partial a}, \frac{\partial \mathcal{L}}{\partial b} \right), \quad (4.31)$$

$$\mathcal{L} = \prod_{i=1}^P \mathcal{L}_i, \quad (4.32)$$

$$\mathcal{L}_i = \mathcal{P}(y^i | \xi), \quad \xi = (u, \lambda). \quad (4.33)$$

Calculemos, então, as componentes do gradiente que envolvem as derivadas com relação ao hiperparâmetro a

$$\frac{\partial \mathcal{L}}{\partial a_{ij}} = \sum_{k,l} \frac{\partial A_{kl}}{\partial a_{ij}} \frac{\partial \mathcal{L}}{\partial A_{kl}}, \quad (4.34)$$

o que exige o cálculo das derivadas

$$\begin{aligned} \frac{\partial}{\partial A_{ij}} \mathcal{L} &= \frac{\partial}{\partial A_{ij}} \prod_k \mathcal{L}_k = \\ &= \sum_k \left(\prod_{l \neq k} \mathcal{L}_l \right) \frac{\partial}{\partial A_{ij}} \mathcal{L}_k, \end{aligned} \quad (4.35)$$

onde

$$\mathcal{L}_k = \mathcal{P}(y^k | \xi) = \sum_{q^k} \mathcal{P}(y^k, q^k | \xi), \quad (4.36)$$

e, portanto,

$$\begin{aligned} \frac{\partial}{\partial A_{ij}} \mathcal{L}_k &= \sum_{q^k} \frac{\partial}{\partial A_{ij}} [\mathcal{P}(y^k, q^k | \xi)] = \\ &= \sum_{q^k} \frac{c_{ij}(q^k)}{A_{ij}} \mathcal{P}(y^k, q^k | \xi), \end{aligned} \quad (4.37)$$

onde c_{ij} é a frequência da transição $s_i \rightarrow s_j$ na sequência de estados escondidos q^k . Assim

$$\begin{aligned} \frac{\partial A_{kl}}{\partial a_{ij}} &= \frac{\partial}{\partial a_{ij}} \left(\frac{e^{\lambda a_{kl}}}{\sum_m e^{\lambda a_{km}}} \right) = \\ &= \frac{\lambda \delta_{ik} \delta_{jl} e^{\lambda a_{kl}}}{\sum_m e^{\lambda a_{km}}} - \frac{(\lambda \sum_m e^{\lambda a_{km}} \delta_{ik} \delta_{jm}) e^{\lambda a_{kl}}}{(\sum_m e^{\lambda a_{km}})^2} = \\ &= \lambda A_{kl} \delta_{ik} \delta_{jl} - \lambda A_{kl} A_{kj} \delta_{ik} = \\ &= \lambda A_{kl} \delta_{ik} (\delta_{jl} - A_{kj}). \end{aligned} \quad (4.38)$$

Portanto, temos que

$$\frac{\partial \mathcal{L}}{\partial a_{ij}} = \sum_{k,l} \lambda A_{kl} \delta_{ik} (\delta_{jl} - A_{kj}) \left[\sum_m \left(\prod_{n \neq m} \mathcal{L}_n \right) \frac{n_{kl}^m}{A_{kl}} \right], \quad (4.39)$$

onde

$$n_{ij}^k = \sum_{q^k} c_{ij}(q^k) \mathcal{P}(y^k, q^k | \xi). \quad (4.40)$$

A derivada fica, então,

$$\begin{aligned}
 \frac{\partial \mathcal{L}}{\partial a_{ij}} &= \lambda \sum_{k,l} \delta_{ik} (\delta_{jl} - A_{kj}) \left[\sum_m \frac{\mathcal{L}}{\mathcal{L}_m} n_{kl}^m \right] = \\
 &= \lambda \mathcal{L} \sum_m \frac{1}{\mathcal{L}_m} \left[\sum_{k,l} \delta_{ik} (\delta_{jl} - A_{kj}) n_{kl}^m \right] = \\
 &= \lambda \mathcal{L} \sum_m \frac{1}{\mathcal{L}_m} \left[\sum_l (\delta_{jl} - A_{ij}) n_{il}^m \right] = \\
 &= \lambda \mathcal{L} \sum_m \frac{1}{\mathcal{L}_m} \left(n_{ij}^m - A_{ij} \sum_l n_{il}^m \right).
 \end{aligned} \tag{4.41}$$

Se, ao invés de minimizarmos diretamente a verossimilhança, optarmos por minimizar o seu logaritmo, obtemos

$$\frac{\partial \ln \mathcal{L}}{\partial a_{ij}} = \frac{1}{\mathcal{L}} \frac{\partial \mathcal{L}}{\partial a_{ij}} = \lambda \sum_m \frac{1}{\mathcal{L}_m} \left(n_{ij}^m - A_{ij} \sum_l n_{il}^m \right). \tag{4.42}$$

Com cálculos análogos para b e p , chegamos às fórmulas

$$\begin{aligned}
 \Delta p_i &= \frac{\eta_{BC}}{\mathcal{L}} \left(l_i - \pi_i \sum_j l_j \right), \\
 \Delta a_{ij} &= \frac{\eta_{BC}}{\mathcal{L}} \left(n_{ij} - A_{ij} \sum_l n_{il} \right), \\
 \Delta b_{i\alpha} &= \frac{\eta_{BC}}{\mathcal{L}} \left(m_{i\alpha} - B_{i\alpha} \sum_\beta m_{i\beta} \right),
 \end{aligned} \tag{4.43}$$

com

$$l_i^j = \sum_{q^j} f_i(q^j) \mathcal{P}(y^j, q^j | \xi), \tag{4.44}$$

$$n_{ij}^k = \sum_{q^k} c_{ij}(q^k) \mathcal{P}(y^k, q^k | \xi), \tag{4.45}$$

$$m_{i\alpha}^j = \sum_{q^j} d_{i\alpha}(y^j, q^j) \mathcal{P}(y^j, q^j | \xi), \tag{4.46}$$

onde $f_i(q^j)$ é a frequência com a qual o estado inicial é o estado s_i na sequência q^j , $d_{i\alpha}(y^j, q^j)$ é o número de emissões de o_α por s_i se o exemplo y^j fosse gerado por q^j e η_{BC} regula o passo de aprendizado.

Esse é o algoritmo de Baldi-Chauvin na forma dada em [5], com a diferença que estamos deduzindo também as equações para o estado inicial do HMM. As equações acima ainda podem ser simplificadas e colocadas em uma forma mais conveniente para a implementação do algoritmo.

Ignorando momentaneamente os índices j apenas por simplicidade, podemos escrever

$$l_i = \sum_q f_i(q) \mathcal{P}(y, q|\xi), \quad (4.47)$$

Assim

$$\begin{aligned} l_i &= \sum_q \mathcal{P}(q_1 = s_i|q, \xi) \mathcal{P}(y, q|\xi) = \\ &= \sum_q \mathcal{P}(q_1 = s_i|q, \xi) \mathcal{P}(q|y, \xi) \mathcal{P}(y|\xi) = \\ &= \mathcal{P}(y|\xi) \sum_q \mathcal{P}(q_1 = s_i, q|y, \xi) = \\ &= \mathcal{P}(y|\xi) \mathcal{P}(q_1 = s_i|y, \xi), \end{aligned} \quad (4.48)$$

$$\begin{aligned} \sum_i l_i &= \sum_i \mathcal{P}(y|\xi) \mathcal{P}(q_1 = s_i|y, \xi) = \\ &= \sum_i \mathcal{P}(q_1 = s_i, y|\xi) = \\ &= \mathcal{P}(y|\xi). \end{aligned} \quad (4.49)$$

Portanto,

$$\Delta p_i = \eta_{BC} [\mathcal{P}(q_1 = s_i|y, \xi) - \pi_i]. \quad (4.50)$$

Da mesma forma, temos para os outros parâmetros

$$\begin{aligned} n_{ij} &= \sum_q c_{ij}(q) \mathcal{P}(y, q|\xi) = \\ &= \sum_q \left[\sum_{t=1}^{T-1} \mathcal{P}(q_t = s_i, q_{t+1} = s_j|q, y, \xi) \right] \mathcal{P}(q|y, \xi) \mathcal{P}(y|\xi) = \\ &= \mathcal{P}(y|\xi) \sum_{t=1}^{T-1} \left[\sum_q \mathcal{P}(q_t = s_i, q_{t+1} = s_j|q, y, \xi) \mathcal{P}(q|y, \xi) \right] = \\ &= \mathcal{P}(y|\xi) \sum_{t=1}^{T-1} \sum_q \mathcal{P}(q_t = s_i, q_{t+1} = s_j, q|y, \xi) = \\ &= \mathcal{P}(y|\xi) \sum_{t=1}^{T-1} \mathcal{P}(q_t = s_i, q_{t+1} = s_j|y, \xi). \end{aligned} \quad (4.51)$$

$$\begin{aligned}
\sum_j n_{ij} &= \mathcal{P}(y|\xi) \sum_{t=1}^{T-1} \sum_j \mathcal{P}(q_t = s_i, q_{t+1} = s_j | y, \xi) = \\
&= \mathcal{P}(y|\xi) \sum_{t=1}^{T-1} \mathcal{P}(q_t = s_i | y, \xi),
\end{aligned} \tag{4.52}$$

e portanto

$$\Delta a_{ij} = \eta_{BC} \left[\sum_{t=1}^{T-1} \mathcal{P}(q_t = s_i, q_{t+1} = s_j | y, \xi) - A_{ij} \sum_{t=1}^{T-1} \mathcal{P}(q_t = s_i | y, \xi) \right], \tag{4.53}$$

e também

$$m_{i\alpha} = \sum_q d_{i\alpha}(y, q) \mathcal{P}(y, q | \xi), \tag{4.54}$$

onde $d_{i\alpha}$ é o número de emissão de o_α por s_i .

$$\begin{aligned}
m_{i\alpha} &= \sum_q \sum_t \mathcal{P}(q_t = s_i, y_t = o_\alpha | q, y, \xi) \mathcal{P}(y, q | \xi) = \\
&= \mathcal{P}(y|\xi) \sum_t \delta_{y_t, o_\alpha} \sum_q \mathcal{P}(q_t = s_i | q, y, \xi) \mathcal{P}(q | y, \xi) = \\
&= \mathcal{P}(y|\xi) \sum_t \delta_{y_t, o_\alpha} \mathcal{P}(q_t = s_i | y, \xi) = \\
&= \mathcal{P}(y|\xi) \sum_{t=1, y_t=o_\alpha}^T \mathcal{P}(q_t = s_i | y, \xi),
\end{aligned} \tag{4.55}$$

$$\sum_\alpha m_{i\alpha} = \mathcal{P}(y|\xi) \sum_{t=1}^T \mathcal{P}(q_t = s_i | y, \xi), \tag{4.56}$$

resultando em

$$\Delta b_{i\alpha} = \eta_{BC} \left[\sum_{t=1, y_t=o_\alpha}^T \mathcal{P}(q_t = s_i | y, \xi) - B_{i\alpha} \sum_{t=1}^T \mathcal{P}(q_t = s_i | y, \xi) \right]. \tag{4.57}$$

Usando a notação

$$\begin{aligned}
\gamma_t(i) &= \mathcal{P}(q_t = s_i | y, \xi), \\
\xi_t(i, j) &= \mathcal{P}(q_t = s_i, q_{t+1} = s_j | y, \xi),
\end{aligned} \tag{4.58}$$

e lembrando das fórmulas de reestimação de Baum-Welch (4.15), temos que

$$\begin{aligned}
\Delta p_i &= \eta_{BC} (\hat{\pi}_i - \pi_i), \\
\Delta a_{ij} &= \eta_{BC} \left[\sum_{t=1}^{T-1} \gamma_t(i) \right] (\hat{A}_{ij} - A_{ij}), \\
\Delta b_{i\alpha} &= \eta_{BC} \left[\sum_{t=1}^T \gamma_t(i) \right] (\hat{B}_{i\alpha} - B_{i\alpha}).
\end{aligned} \tag{4.59}$$

Ou seja, a variação dos parâmetros exponenciais é proporcional à variação dos parâmetros do HMM calculada através das fórmulas de reestimação de Baum-Welch.

Essas fórmulas tornam claro, também, que a complexidade computacional de cada passo do algoritmo BC é a mesma de BWO, ou seja, polinomial de ordem 2 em n e de ordem 1 em T .

Um efeito importante a ser notado nestas equações é o que acontece quando iniciamos o aprendizado com um aluno inicial simétrico, isto é, onde as probabilidades são iguais. Analisemos o que acontece para o parâmetro π . Nesse caso, o aluno ser simétrico significa que $p_i = p$ e $\pi_i = 1/n$ para todo i , e a equação de atualização fica

$$\begin{aligned}\Delta\pi_i &= \frac{e^{\lambda(p_i + \Delta p_i)}}{\sum_j e^{\lambda(p_j + \Delta p_j)}} - \frac{e^{\lambda p_i}}{\sum_j e^{\lambda p_j}} = \\ &= \frac{e^{\lambda(p + \Delta p_i)}}{\sum_j e^{\lambda(p + \Delta p_j)}} - \frac{1}{n} = \\ &= \frac{e^{\lambda \Delta p_i}}{\sum_j e^{\lambda \Delta p_j}} - \frac{1}{n}.\end{aligned}\tag{4.60}$$

Vimos que Δp_i é proporcional a η_{BC} e, portanto, o acréscimo em π_i só irá depender de η_{BC} e λ através do produto entre eles. É importante frisar que **para qualquer outro aluno inicial isso já não é mais válido**.

4.5.2 Relação entre BC e BWO

Considere agora a variação dos parâmetros do HMM no algoritmo de Baldi-Chauvin, que simbolizaremos por Δ , e a variação utilizando as fórmulas de reestimação de Baum-Welch, que simbolizaremos por $\hat{\Delta}$. Temos que

$$\Delta\pi_i = \frac{e^{\lambda(p_i + \Delta p_i)}}{\sum_j e^{\lambda(p_j + \Delta p_j)}} - \frac{e^{\lambda p_i}}{\sum_j e^{\lambda p_j}}.\tag{4.61}$$

Para λ pequeno temos, em primeira ordem

$$\begin{aligned}\Delta\pi_i &\approx \frac{1 + \lambda(p_i + \Delta p_i)}{\sum_j [1 + \lambda(p_j + \Delta p_j)]} - \frac{1 + \lambda p_i}{\sum_j (1 + \lambda p_j)} = \\ &= \frac{1 + \lambda p_i + \lambda \Delta p_i}{n + \lambda \zeta} - \frac{1 + \lambda p_i}{n + \lambda \zeta},\end{aligned}\tag{4.62}$$

com

$$\sum_j \Delta p_j = \sum_j \eta_{BC}(\hat{\pi}_j - \pi_j) = 0,\tag{4.63}$$

$$\zeta = \sum_j p_j. \quad (4.64)$$

Assim

$$\begin{aligned} \Delta\pi_i &= \frac{\lambda\Delta p_i}{n + \lambda\zeta} = \\ &= \frac{\lambda\eta_{BC}}{n} \hat{\Delta}\pi_i \left(1 + \frac{\lambda\zeta}{n}\right)^{-1} \approx \\ &\approx \frac{\lambda\eta_{BC}}{n} \hat{\Delta}\pi_i. \end{aligned} \quad (4.65)$$

Cálculos análogos levam às seguintes três equações em primeira ordem em λ

$$\begin{aligned} \Delta\pi_i &= \frac{\lambda\eta_{BC}}{n} \hat{\Delta}\pi_i = \frac{\lambda}{n} \frac{\eta_{BC}}{\eta_{BW}} \tilde{\Delta}\pi_i, \\ \Delta A_{ij} &= \frac{\lambda\eta_{BC}}{n} \left[\sum_{t=1}^{T-1} \gamma_t(i) \right] \hat{\Delta}A_{ij} = \frac{\lambda}{n} \frac{\eta_{BC}}{\eta_{BW}} \left[\sum_{t=1}^{T-1} \gamma_t(i) \right] \tilde{\Delta}A_{ij}, \\ \Delta B_{i\alpha} &= \frac{\lambda\eta_{BC}}{n} \left[\sum_{t=1}^T \gamma_t(i) \right] \hat{\Delta}B_{i\alpha} = \frac{\lambda}{n} \frac{\eta_{BC}}{\eta_{BW}} \left[\sum_{t=1}^T \gamma_t(i) \right] \tilde{\Delta}B_{i\alpha}. \end{aligned} \quad (4.66)$$

onde $\tilde{\Delta}$ representa a variação do parâmetro utilizando o algoritmo BWO. Utilizando essas fórmulas, vemos que o valor de η_{BW} para que os dois algoritmos sejam próximos para λ pequeno pode ser estimado como

$$\eta_{BW} \approx \frac{\lambda\eta_{BC}}{n}. \quad (4.67)$$

Verificaremos a validade dessas equações e a qualidade dessa estimativa fazendo simulações comparando os dois algoritmos para λ pequeno mais adiante.

Vemos, então, que para λ suficientemente pequeno, os incrementos dados pelo algoritmo BC são proporcionais aos dados por BWO, porém com taxas de aprendizado diferentes para cada matriz. Note que a taxa de aprendizado para as matrizes de emissão e de transição em BC vão depender da palavra lida pelo HMM.

4.5.3 Simulações

Curvas de Aprendizado *On-line*

Considerando o caso onde temos $P > 1$ seqüências de aprendizado, fizemos simulações com o intuito de traçar a curva de aprendizado para o algoritmo BC. A figura 4.9 mostra curvas de d_{KL} em função do número de seqüências aprendidas com $n = 2$, $m = 3$, $N_\tau = 200$, $T = 2$ e $\lambda = 0.01$ para 4 diferentes valores de η_{BC} : 10.0, 5.0, 1.0 e 0.5. A figura 4.10 mostra a mesma simulação, porém a distância utilizada é d_E .

Figura 4.9: **Algoritmo de Baldi-Chauvin:** a) Gráfico mostrando a distância de Kullback-Leibler entre o professor e o aluno após a aplicação do algoritmo para 4000 seqüências de 2 símbolos fazendo a média em 200 professores com 2 estados escondidos e 3 estados observados. Os números próximos a cada linha indicam o valor de η_{BC} utilizado. b) O mesmo que o gráfico a, porém em log-log e para 10000 seqüências.

Figura 4.10: **Algoritmo de Baldi-Chauvin:** a) Gráfico mostrando a distância Euclidiana entre o professor e o aluno após a aplicação do algoritmo para 4000 seqüências de 2 símbolos fazendo a média em 200 professores com 2 estados escondidos e 3 estados observados. Os números próximos a cada linha indicam o valor de η_{BC} utilizado. b) O mesmo que o gráfico a, porém em log-log e para 10000 seqüências.

Figura 4.11: **Algoritmo de Baldi-Chauvin:** Gráfico mostrando a comparação entre as curvas de aprendizado para o algoritmo BC (linha contínua) e para o algoritmo BWO com a taxa de aprendizado estimada (linha tracejada) e com a melhor taxa obtida por tentativa e erro (linha pontilhada). As curvas são traçadas para $\eta_{BC} = 1$ fixo e para 3 valores distintos de λ , de baixo para cima, 0.01, 0.001 e 0.0001.

Relação entre BC e BWO

Também fizemos testes para tentar verificar as relações entre o algoritmo BC e o algoritmo BWO dadas pelas equações (4.66). Fizemos simulações com os parâmetros $n = 2$, $m = 3$, $T = 2$ e $N_\tau = 100$ para um total de 2000 sequências. O aluno inicial é tomado simétrico, o que, segundo já discutimos, faz com que BC dependa apenas do produto $\lambda\eta_{BC}$. Assim, fixamos $\eta_{BC} = 1$ e traçamos as curvas para BC com três valores distintos $\lambda = 0.01, 0.001, 0.0001$. Para cada valor de λ , traçamos duas curvas utilizando o algoritmo BWO: uma com um valor estimado através da fórmula (4.67), simbolizado por $\hat{\eta}_{BW}$, e o outro com o valor encontrado através de tentativa e erro que torna as curvas de

Figura 4.12: **Algoritmo de Baldi-Chauvin:** Gráfico mostrando a distância euclidiana entre aluno e professor para o algoritmo BC aplicado ao aprendizado de uma única sequência de tamanho T utilizando cinco tamanhos diferentes de subsequências indicadas ao lado de cada curva comparando o resultado com o obtido utilizando o algoritmo BW.

aprendizado o mais parecidas possível, simbolizado por η_{BW}^* . A tabela a seguir mostra esses valores:

λ	$\hat{\eta}_{BW}$	η_{BW}^*
0.01	0.005	0.003
0.001	0.0005	0.00035
0.0001	0.00005	0.000034

Vemos que η_{BW}^* é muito próximo e da mesma ordem de grandeza do valor estimado $\hat{\eta}_{BW}$. Além disso, quanto menor o valor de λ , maior o acordo entre as curvas dos dois algoritmos, sendo que para o menor valor de λ , a curva com η_{BW}^* está tão próxima da

Figura 4.13: **Algoritmo de Baldi-Chauvin:** Gráfico mostrando a distância euclidiana entre aluno e professor para o algoritmo BC aplicado ao aprendizado de uma única sequência de tamanho T utilizando subsequências de tamanho 6 para cinco diferentes valores da taxa de aprendizado η_{BC} indicados ao lado de cada curva.

curva do algoritmo BC que praticamente não é visível no gráfico. Isso confirma a relação (4.66). As curvas de aprendizado estão mostradas na figura 4.11.

Curvas de Aprendizado *Off-line*

A seguir, analisamos o resultado da aplicação do algoritmo BC para o caso onde temos apenas uma sequência de aprendizado e utilizamos a estratégia de quebrá-la em subsequências menores a figura 4.12 mostra o resultado de simulações onde calculamos a distância euclidiana d_E com parâmetros $n = 2$, $m = 3$, $N_\tau = 100$, $\lambda = 0.0005$ e $\eta_{BC} = 10.0$ para um aluno inicial simétrico e com cinco diferentes tamanhos de subsequências: 2, 4, 6, 8 e 10. Comparamos esses dados com os obtidos através da aplicação

do algoritmo BW. Note como os alunos se aproximam mais do professor no início, mas à medida que o tamanho da subsequência aumenta, o algoritmo se torna instável para valores cada vez menores de T e o aluno começa a se afastar do professor cada vez mais cedo. Esse problema pode ser controlado diminuindo-se a taxa de aprendizado η_{BC} , como mostrado na figura 4.13, onde apresentamos o resultado de simulações com parâmetros $n = 2$, $m = 3$, $N_\tau = 100$, $\lambda = 0.0005$ e tamanho das subsequências igual a 6. Cada curva representa o aprendizado com um valor de η_{BC} dado pelo número ao seu lado no gráfico. Note que quanto menor o valor da taxa de aprendizado, mais tempo o algoritmo demora para ficar instável. No entanto, há uma desvantagem em aumentar η_{BC} : o algoritmo demora mais para aprender. Porém, como esse retardo no aprendizado é pequeno, pode-se considerar que abaixar a taxa de aprendizado é uma boa solução para o problema da instabilidade. O que pode ser feito é abaixar a taxa de aprendizado ao longo do processo à mão, um recurso chamado de *annealing*, o que apresenta a vantagem de termos uma taxa alta no começo, aumentando a velocidade inicial do aprendizado, e uma taxa baixa no final, fazendo com que o aluno se aproxime mais do professor. No caso dos algoritmos bayesianos *on-line*, o *annealing* ocorre automaticamente [74] e isso será mostrado para o algoritmo que introduziremos nas duas próximas seções do capítulo seguinte (seções 5.1 e 5.2).

Capítulo 5

Aprendizado Bayesiano

Neste capítulo, derivamos e estudamos em detalhes dois algoritmos bayesianos para o aprendizado em HMMs. Os estudos incluem comparações entre esses algoritmos e aqueles já estudados no capítulo anterior e aplicações em situações práticas.

5.1 Algoritmo Bayesiano *On-line*

5.1.1 Algoritmo

Para o aprendizado de um HMM, os parâmetros que queremos ajustar são a matriz de transição A , a matriz de emissão B e o vetor das probabilidades iniciais π . Definiremos a seguinte notação para tais parâmetros em um determinado HMM ω

$$\xi = (\xi^1, \dots, \xi^V), \quad V = 2n + 1, \quad (5.1)$$

$$\xi^1 = \begin{pmatrix} \pi_1 \\ \dots \\ \pi_n \end{pmatrix} = \begin{pmatrix} \xi_1^1 \\ \dots \\ \xi_n^1 \end{pmatrix}, \quad (5.2)$$

$$\xi^{1+i} = \begin{pmatrix} A_{i1} \\ \dots \\ A_{in} \end{pmatrix} = \begin{pmatrix} \xi_1^{1+i} \\ \dots \\ \xi_n^{1+i} \end{pmatrix}, \quad (5.3)$$

$$\xi^{1+n+i} = \begin{pmatrix} B_{i1} \\ \dots \\ B_{im} \end{pmatrix} = \begin{pmatrix} \xi_1^{1+n+i} \\ \dots \\ \xi_m^{1+n+i} \end{pmatrix}, \quad (5.4)$$

de tal forma que cada um dos vetores ξ^i obedece ao vínculo $\sum_j \xi_j^i = 1$.

A intenção é adaptar o método descrito na seção 2.5.4 e que pode ser encontrado em [74] no qual projetaremos as distribuições de probabilidade posteriores

$$Q^{p+1}(\xi) \equiv \mathcal{P}(\xi|\theta^p, y^{p+1}), \quad (5.5)$$

isto é, as distribuições obtidas após adicionarmos o dado $p + 1$ à distribuição paramétrica $P^p(\xi)$, em uma outra distribuição paramétrica da mesma família

$$P^{p+1}(\xi) \equiv \mathcal{P}(\xi|\theta^{p+1}), \quad (5.6)$$

minimizando a distância de Kullback-Leibler entre as duas. Devido a esta minimização, daqui para diante nos referiremos a este algoritmo como BKL (*Bayesian/Kullback-Leibler*).

No caso de HMMs, optaremos por uma distribuição paramétrica da forma

$$P(\xi|u) \equiv \prod_{i=1}^V \mathcal{D}_i, \quad (5.7)$$

$$\mathcal{D}_i \equiv \mathcal{D}(\xi^i|u^i), \quad (5.8)$$

$$u^i = (u_1^i, u_2^i, \dots), \quad (5.9)$$

que é uma produtória de distribuições de Dirichlet (ver Apêndice B). Tais distribuições são as mais apropriadas quando as variáveis são em si mesmas probabilidades e sua normalização já leva em conta o simplex onde os vínculos necessários são satisfeitos. Nesse caso, os parâmetros u das distribuições de Dirichlet são os parâmetros θ das equações (5.5) e (5.6).

A distribuição que escolhemos considera como independentes os grupos de parâmetros que formam conjuntos independentes de probabilidades. Isto é obviamente uma aproximação onde ignoramos qualquer correlação entre linhas diferentes das matrizes de transição e emissão e qualquer correlação existente entre π , A e B .

A minimização da distância de Kullback-Leibler, neste caso, é equivalente à minimização da expressão

$$\begin{aligned} \mathcal{L} &= - \int d\xi Q^{p+1} \ln P^{p+1} \equiv \\ &\equiv - \langle \ln P^{p+1} \rangle_Q = \\ &= - \sum_i \langle \ln \mathcal{D}_i \rangle_Q. \end{aligned} \quad (5.10)$$

Fazendo

$$\frac{\partial \mathcal{L}}{\partial u_j^i} = 0, \quad (5.11)$$

obtemos as equações

$$\psi(u_j^i) - \psi\left(\sum_k u_k^i\right) = \langle \ln \xi_j^i \rangle_Q \equiv \mu_j^i, \quad (5.12)$$

onde $\psi(x)$ é a função digama, definida como a derivada logarítmica da função gama

$$\psi(x) = \frac{d}{dx} \ln \Gamma(x) = \frac{\Gamma'(x)}{\Gamma(x)}, \quad (5.13)$$

e onde o índice k da somatória varia sobre cada conjunto fechado de probabilidades. Nos referiremos a esse conjunto de equações como o **sistema digama**. Assim, nos restam dois cálculos a fazer: os coeficientes μ_j^i e a solução do sistema digama (5.12).

A distribuição posterior Q^{p+1} após a aprendizagem do exemplo número $p + 1$ é dada por

$$Q^{p+1}(\xi) = \frac{1}{Z_Q} P^p(\xi) \sum_{I \in \mathcal{S}} \left(\prod_{(i,j) \in I} \xi_j^i \right), \quad (5.14)$$

$$Z_Q = \int d\xi P^p(\xi) \sum_{I \in \mathcal{S}} \left(\prod_{(i,j) \in I} \xi_j^i \right). \quad (5.15)$$

O conjunto I é um conjunto de índices composto pelos índices dos estados observados e de um determinado conjunto de estados escondidos capaz de gerar tais estados observados. O conjunto $\mathcal{S}$ corresponde a todos os possíveis grupos de índices que podem gerar os estados observados dados como exemplo.

Para tornar a notação mais clara, vejamos um exemplo. Suponha que temos um HMM com $S = \{1, 2\}$ e $O = \{1, 2, 3\}$ e que a seqüência a ser aprendida seja dada por $y = \{1, 3\}$. Temos as seguintes possibilidades para as seqüências de estados escondidos que poderiam gerar y :

$$\begin{aligned} q_1 &= \{1, 1\}, \\ q_2 &= \{1, 2\}, \\ q_3 &= \{2, 1\}, \\ q_4 &= \{2, 2\}. \end{aligned} \quad (5.16)$$

Os possíveis conjuntos são

$$\begin{aligned} \{\pi_1, A_{11}, B_{11}, B_{13}\} &= \{\xi_1^1, \xi_1^2, \xi_1^4, \xi_3^4\} \Rightarrow I_1 = \{(1, 1), (2, 1), (4, 1), (4, 3)\} \\ \{\pi_1, A_{12}, B_{11}, B_{23}\} &= \{\xi_1^1, \xi_2^2, \xi_1^4, \xi_3^5\} \Rightarrow I_2 = \{(1, 1), (2, 2), (4, 1), (5, 3)\}, \\ \{\pi_2, A_{21}, B_{21}, B_{13}\} &= \{\xi_2^1, \xi_1^3, \xi_1^5, \xi_3^4\} \Rightarrow I_3 = \{(1, 2), (3, 1), (5, 1), (4, 3)\}, \\ \{\pi_2, A_{22}, B_{21}, B_{23}\} &= \{\xi_2^1, \xi_2^3, \xi_1^5, \xi_3^5\} \Rightarrow I_4 = \{(1, 2), (3, 2), (5, 1), (5, 3)\}. \end{aligned} \quad (5.17)$$

E, portanto, o conjunto que chamamos de $\mathcal{S}$ é dado por

$$\mathcal{S} = \{I_1, I_2, I_3, I_4\}. \quad (5.18)$$

Para facilitar os cálculos analíticos que se seguirão, definiremos o conjunto de variáveis $r = \{r_{ij}\}$, onde cada r_{ij} é igual ao número de vezes em que a variável ξ_j^i aparece em uma certa verossimilhança. Dessa forma, podemos escrever

$$Q^{p+1}(\xi) = \frac{1}{Z_Q} P^p(\xi) \sum_r \left[\prod_{i,j} (\xi_j^i)^{r_{ij}} \right], \quad (5.19)$$

e

$$Z_Q = \int d\xi P^p(\xi) \sum_r \left[\prod_{i,j} (\xi_j^i)^{r_{ij}} \right] = \quad (5.20)$$

$$= \sum_r \prod_i \left\langle \prod_j (\xi_j^i)^{r_{ij}} \right\rangle_i, \quad (5.21)$$

onde

$$\langle \cdot \rangle_i \equiv \int d\xi^i(\cdot) \mathcal{D}_i. \quad (5.22)$$

No exemplo acima, teríamos

$$\begin{aligned} Q^{p+1}(\xi) &= \frac{1}{Z_Q} \sum_r \prod_i \left\langle \prod_j (\xi_j^i)^{r_{ij}} \right\rangle_i = \\ &= \frac{(\prod_{i=1}^5 \mathcal{D}_i)}{Z_Q} [\xi_1^1 \xi_1^2 \xi_1^4 \xi_3^4 + \xi_1^1 \xi_2^2 \xi_1^4 \xi_3^5 + \xi_2^1 \xi_1^3 \xi_3^4 \xi_1^5 + \xi_2^1 \xi_2^3 \xi_1^5 \xi_3^5], \end{aligned} \quad (5.23)$$

e

$$\begin{aligned} Z_Q &= \langle \xi_1^1 \rangle_1 \langle \xi_1^2 \rangle_2 \langle \xi_1^4 \xi_3^4 \rangle_4 + \langle \xi_1^1 \rangle_1 \langle \xi_2^2 \rangle_2 \langle \xi_1^4 \rangle_4 \langle \xi_3^5 \rangle_5 + \\ &+ \langle \xi_2^1 \rangle_1 \langle \xi_1^3 \rangle_3 \langle \xi_3^4 \rangle_4 \langle \xi_1^5 \rangle_5 + \langle \xi_2^1 \rangle_1 \langle \xi_2^3 \rangle_2 \langle \xi_1^5 \xi_3^5 \rangle_5. \end{aligned} \quad (5.24)$$

Dessa forma, vemos que para calcularmos os coeficientes $\mu_j^i = \langle \ln \xi_j^i \rangle_Q$, temos que calcular termos da forma geral

$$\mu_j = \left\langle \left[\prod_i (\xi_i)^{r_i} \right] \ln \xi_j \right\rangle, \quad (5.25)$$

numa distribuição de Dirichlet $\mathcal{D}(\xi|u)$. Os detalhes desse cálculo se encontram no Apêndice B, fórmula B.31, e o resultado final é

$$\mu_j = \left\langle \prod_i (\xi_i)^{r_i} \right\rangle [\psi(u_j + r_j) - \psi(u_0 + r_0)], \quad (5.26)$$

com $u_0 = \sum_i u_i$, $r_0 = \sum_i r_i$ e

$$\left\langle \prod_i (\xi_i)^{r_i} \right\rangle = \frac{\Gamma(u_0)}{\prod_i \Gamma(u_i)} \frac{\prod_i \Gamma(u_i + r_i)}{\Gamma(u_0 + r_0)}. \quad (5.27)$$

Assim, temos como calcular numericamente o coeficiente μ_j^i para as equações digama. Para encontrar a solução de um sistema de equações do tipo

$$\psi(x_i) - \psi\left(\sum_j x_j\right) = \mu_i, \quad i = 1, \dots, d, \quad (5.28)$$

desenvolvemos um método muito simples que consiste em transformar esse conjunto de equações em um mapa unidimensional e encontrar o seu ponto fixo escolhendo um ponto inicial e iterando. Para isso, isolamos x_i nas equações acima e obtemos

$$x_i = \psi^{-1} \left[\mu_i + \psi\left(\sum_j x_j\right) \right]. \quad (5.29)$$

Somando em i e definindo $x_0 \equiv \sum_i x_i$, obtemos

$$x_0 = \sum_i \psi^{-1}[\mu_i + \psi(x_0)], \quad (5.30)$$

que, transformado em um mapa, fica

$$x_0^{n+1} = \sum_i \psi^{-1}[\mu_i + \psi(x_0^n)]. \quad (5.31)$$

Testamos esse método para valores de d entre 1 e aproximadamente 300 e conseguimos uma convergência extremamente rápida para o ponto fixo (linear em d).

O algoritmo BKL, assim como todo algoritmo bayesiano, sofre de um grave problema: o fato de ser necessário fazer a soma sobre as variáveis escondidas faz com que a complexidade do algoritmo escale com n^T , ou seja, ela é exponencial no tamanho das palavras. No caso de BKL a situação é agravada devido á grande quantidade de funções digama que precisam ser calculadas, acrescentando mais tempo ao algoritmo. Na seção seguinte obteremos um algoritmo mais rápido, embora o problema da complexidade ainda perdure.

5.1.2 Simulações

O algoritmo bayesiano *on-line* como apresentado aqui é extremamente lento. Como exemplo, uma simulação para 300 seqüências de tamanho $T = 2$ e 150 professores de 2 estados escondidos e 3 observados levou 340 minutos para rodar em um processador de 1.5 GHz. Sendo assim, nos limitaremos a comparar a sua performance com o algoritmo apresentado na próxima seção.

5.2 Posterior Média

O algoritmo BKL apresentado na seção anterior (5.1) é extremamente custoso do ponto de vista computacional, o que o torna pouco útil em situações práticas. Dessa forma, nesta seção, desenvolveremos uma simplificação desse algoritmo que o torna mais rápido e compararemos com os outros algoritmos já estudados neste trabalho.

5.2.1 Algoritmo

Essa versão do algoritmo bayesiano, que simbolizaremos pela sigla PM, também foi inspirada no algoritmo sugerido por Oppen em [74]. Em BKL, usamos a idéia de projetar a posterior na mesma distribuição paramétrica da priori minimizando a distância de Kullback-Leibler. No artigo citado, Oppen usa esse algoritmo para redes neurais com distribuições gaussianas e mostra que ele é equivalente a igualar a média e a covariância da posterior e da distribuição projetada¹. Observando esse fato, resolvemos, ao invés da minimização de d_{KL} , utilizar a estratégia de igualar a média e uma das covariâncias da posterior e da projetada como forma de simplificação do algoritmo. Isso resultará em manter o coeficiente u_0^i sempre fixo ao longo do aprendizado. Veremos isso em detalhe a seguir.

Utilizando a mesma notação do algoritmo anterior (BKL), consideremos a nossa distribuição *a priori* para os parâmetros do HMM como um produto de Dirichlets também neste algoritmo

$$P(\xi|u) \equiv \prod_{i=1}^V \mathcal{D}_i, \quad (5.32)$$

Os valores estimados dos parâmetros u e ξ após o processamento de cada dado serão indicados por $\hat{u}$ e $\hat{\xi}$ respectivamente. A cada nova palavra y , a distribuição posterior será

$$Q(\xi|u) = \frac{1}{Z_Q} \left(\prod_{i=1}^V \mathcal{D}_i \right) \sum_q P(y, q|\xi), \quad (5.33)$$

$$P(y, q|\xi) = \prod_{i,j} (\xi_j^i)^{r_{ij}}, \quad (5.34)$$

onde r_{ij} é o número de vezes que ξ_j^i aparece na verossimilhança e Z_Q tem exatamente a mesma forma do item anterior, ou seja,

$$Z_Q = \sum_r \prod_i \left\langle \prod_j (\xi_j^i)^{r_{ij}} \right\rangle_i. \quad (5.35)$$

¹Isso é um caso particular de um teorema simples mais geral que se encontra no apêndice D

Note que Q é uma mistura de Dirichlets. Agora, como já dissemos de forma resumida no começo desta seção, este algoritmo também se utiliza da idéia de projetar a distribuição posterior em uma nova distribuição de Dirichlet, mas ao invés de usar a distância de Kullback-Leibler para isso, simplesmente faremos com que as médias da nova distribuição coincidam com as médias da posterior, ou seja,

$$\hat{\xi}_j^i = \langle \xi_j^i \rangle_Q = \frac{\hat{u}_j^i}{\hat{u}_0^i} \Rightarrow \hat{u}_j^i = \hat{\xi}_j^i \hat{u}_0^i. \quad (5.36)$$

Isso nos deixa com um parâmetro livre para cada conjunto de probabilidades, $\hat{u}_0^i$. Para fixar esse parâmetro, escolhemos a covariância da primeira variável de cada conjunto na distribuição posterior e fazemos com que ela seja igual à covariância da primeira variável de cada conjunto na nova distribuição projetada de Dirichlet. Isso nos dá a fórmula

$$\hat{u}_0^i = \frac{\hat{\xi}_1^i - \langle (\xi_1^i)^2 \rangle_Q}{\langle (\xi_1^i)^2 \rangle_Q - (\hat{\xi}_1^i)^2}. \quad (5.37)$$

Assim, ficam determinados os valores de $\hat{u}$ para a nova distribuição de Dirichlet. Embora a complexidade computacional deste algoritmo também seja da ordem de n^T como em BKL, a simplificação proposta diminui radicalmente o tempo real de computação, tornando este algoritmo bem mais acessível na prática.

5.2.2 Simulações

Curvas de Aprendizado *On-line*

Traçamos as curvas de aprendizado para este algoritmo utilizando os valores $T = 2$, $N_\tau = 500$, $n = 2$, $m = 3$ e com um aluno inicial simétrico com todos os parâmetros iniciais u iguais a 1. A figura 5.1 mostra o gráfico de d_{KL} , enquanto a figura 5.2 mostra o gráfico de d_E . De forma diferente do que havia acontecido para BWO e BC, neste caso o comportamento qualitativo da curva de aprendizado é bem diferente quando usamos d_{KL} e d_E para descrevê-la. Isso indica que, embora sendo muito mais rápida de se calcular, é necessário cuidado ao se comparar algoritmos através de d_E . Como d_{KL} é obtida através de princípios teóricos bem definidos, é de se esperar que ela seja a descrição mais adequada.

Comportamento das Variâncias

É interessante analisarmos o comportamento das variâncias durante o aprendizado. Para o perceptron, como explicado em [74], a matriz de covariância do algoritmo bayesiano *on-*

Figura 5.1: **Posterior Média:** a) Gráfico mostrando a distância de Kullback-Leibler entre o professor e o aluno após a aplicação do algoritmo para 10000 seqüências de 2 símbolos fazendo a média em 500 professores com 2 estados escondidos e 3 estados observados. b) O mesmo que o gráfico a, porém em log-log.

Figura 5.2: **Posterior Média:** a) Gráfico mostrando a distância Euclidiana entre o professor e o aluno após a aplicação do algoritmo para 10000 seqüências de 2 símbolos fazendo a média em 500 professores com 2 estados escondidos e 3 estados observados. b) O mesmo que o gráfico a, porém em log-log.

line faz o papel da taxa de aprendizado e tem um *annealing* automático, ou seja, diminui à medida que mais dados são apresentados como consequência direta das equações de atualização. Como no algoritmo PM não temos equações explícitas de atualização na forma $\theta(t+1) = \theta(t) + f(t)$, é interessante observar através de simulações se o mesmo efeito está ocorrendo.

A figura 5.3 mostra os gráficos das variâncias dos parâmetros π , A e B em uma simulação com $T = 2$, $n = 2$, $m = 3$ e $N_\tau = 500$ para um total de 4000 seqüências. Como acontece no caso do percéptron, aqui também a variância decresce ao longo do aprendizado. Para π e A , ela decresce lentamente, mas para B o efeito é muito mais drástico. Essa diferença se deve ao fato de que, enquanto π e A refletem o comportamento de variáveis que o algoritmo não tem acesso, e portanto a informação ganha a cada passo sobre elas é muito pequena, B está diretamente relacionada ao comportamento dos estados observados, o que faz com que uma grande quantidade de informação sobre essa variável possa ser adquirida implicando numa diminuição rápida de sua variância.

Influência das Condições Iniciais

Estudaremos agora qual a influência dos parâmetros iniciais u para o aprendizado *on-line*. Consideremos k como sendo n ou m de acordo com o grupo de parâmetros u associado a um certo conjunto de probabilidades correspondente (por exemplo, o conjunto formado pelas componentes do vetor π).

Consideremos a situação onde todos os parâmetros iniciais u são iguais e analisemos o caso de um grupo de probabilidades com um número de elementos igual a k cujos elementos serão simbolizados por ξ_i , $i = 1, \dots, k$. Nesse caso, os correspondentes parâmetros iniciais são dados por $u_i = u$, $i = 1, \dots, k$ e

$$\begin{aligned} \langle \xi_i \rangle &= \frac{1}{k}, \\ \text{Var}(\xi_i) &= \frac{k-1}{k^2(ku+1)}. \end{aligned} \quad (5.38)$$

Note que, embora as médias das variáveis não dependam de u , o mesmo não acontece com as variâncias, pois quanto maior o valor de u , menor elas são. Para $u \rightarrow \infty$, as variâncias tendem a zero. Para $u \rightarrow 0$, elas tendem a $(k-1)/k^2$.

A figura 5.4 mostra o comportamento de PM para simulações onde $T = 2$, $n = 2$, $m = 3$, $N_\tau = 500$ com alunos iniciais simétricos definidos pelos valores de $u = 10^{-4}, 0.5, 1.0, 2.0, 20.0$. Note que em cada curva há um momento em que ocorre um salto na generalização. Esses saltos estão associados a quebras de simetria nos parâmetros do

Figura 5.3: **Posterior Média:** a) A curva superior representa o valor da variância de π ao longo do aprendizado, as duas curvas inferiores, as variâncias de A . b) Variâncias de B ao longo do aprendizado.

Figura 5.4: **Posterior Média:** O gráfico log-log acima mostra a distância d_{KL} em função do número de seqüências aprendidas para cinco valores iniciais de u : 10^{-4} , 0.5, 1.0, 2.0 e 20.0.

aluno e representam um avanço súbito no aprendizado (seção 5.6). O gráfico mostra que, quanto maior o valor de u , mais tarde ocorre essa quebra de simetria. Isso é um resultado esperado, uma vez que o aumento de u implica na diminuição da variância. A variância no algoritmo bayesiano representa o tamanho do passo na direção de um mínimo local e, se ela é menor, mais difícil é atingir esse mínimo.

Performance Comparativa

A curva de aprendizado deste algoritmo apresenta uma característica diferente de BWO e BC: o aluno não pára de aprender a partir de um determinado ponto. Pode-se observar que a distância entre aluno e professor continua caindo, mesmo após o aprendizado de 10000 seqüências, o que é mais claramente visualizado no gráfico log-log da figura 5.1.

As figuras 5.5 e 5.6 comparam as performances de BWO, BC e PM, sendo que a segunda figura contém as mesmas curvas da primeira, porém em escala log-log. Usamos a distância d_{KL} para fazer as comparações entre as performances. O gráfico superior mostra PM comparado com BC e o inferior, PM comparado com BWO. Vemos que, mesmo variando os parâmetros de BC e BWO, a performance de PM tanto em relação à velocidade de generalização quanto em relação a proximidade com o professor sempre é superior.

Note que a vantagem de PM é obtida sem a necessidade de nenhum parâmetro ajustável no algoritmo. Fizemos, também, alguns testes calculando a média de d_{KL} em alunos gerados aleatoriamente. Apesar das curvas de aprendizado de todos os algoritmos sofrerem modificações neste caso, a curva de aprendizado do algoritmo PM continua demonstrando a sua superioridade quanto à generalização com relação aos outros algoritmos estudados.

Fizemos também uma comparação entre a performance de PM e BKL onde utilizamos os números $T = 2$, $N_\tau = 150$, $n = 2$, $m = 3$ e novamente um aluno inicial simétrico. A figura 5.7 mostra o resultado para um total de 300 seqüências. O gráfico mostra que a curva de aprendizado dos dois é praticamente igual, com pequenas variações que, no gráfico, são quase imperceptíveis. Dada a diferença muito grande no tempo de computação, aproximadamente 5 segundos para rodar PM e 340 minutos para BKL, PM torna-se um algoritmo muito melhor que BKL do ponto de vista prático.

Curva de Aprendizado *Off-line*

A figura 5.8 mostra o resultado da aplicação do algoritmo PM a uma situação onde tem-se apenas uma seqüência de tamanho T a ser aprendida. Os parâmetros utilizados na simulação foram $n = 2$, $m = 3$ e $N_\tau = 100$. As curvas de aprendizado mostram o resultado para quatro tamanhos diferentes de subseqüências: 2, 4, 6 e 8. Podemos ver claramente que, quanto maior o tamanho da subseqüência utilizada, menos informação o algoritmo consegue tirar dos dados e pior a sua generalização. Em todas as curvas a medida de generalização utilizada foi a distância euclidiana d_E devido ao enorme tempo de computação requerido para o cálculo de d_{KL} .

5.3 Estudo Teórico: A Moeda Simples

É interessante analisarmos o comportamento dos dois algoritmos bayesianos, PM e BKL, em uma situação simplificada. Para isso, consideraremos o aprendizado de um HMM que

Figura 5.5: **Posterior Média:** a) Comparação entre PM e BC. BC foi rodado com $\eta_{BC} = 0.5$ e para 5 diferentes valores de λ indicados junto às respectivas curvas. b) Comparação entre PM e BWO. BWO foi rodado para 4 diferentes valores de η_{BW} indicados junto às respectivas curvas. Para os dois gráficos: $n = 2$, $m = 3$, $T = 2$ e $N_\tau = 500$ e a linha inferior corresponde à curva de aprendizado de PM.

Figura 5.6: **Posterior Média:** a) Comparação entre PM e BC. BC foi rodado com $\eta_{BC} = 0.5$ e para 5 diferentes valores de λ indicados junto às respectivas curvas. Escalas em log-log. b) Comparação entre PM e BWO. BWO foi rodado para 4 diferentes valores de η_{BW} indicados junto às respectivas curvas. Para os dois gráficos: $n = 2$, $m = 3$, $T = 2$ e $N_\tau = 500$ e a linha inferior corresponde à curva de aprendizado de PM. Escalas em log-log.

Figura 5.7: **Posterior Média:** a) Gráfico comparativo da performance de PM versus BKL. Ambos foram rodados para os seguintes valores: $n = 2$, $m = 3$, $T = 2$ e $N_\tau = 150$, para um total de 300 seqüências. Os pequenos círculos mostram o resultado de BKL, enquanto a linha tracejada mostra o resultado de PM; b) mesmo gráfico em escala log-log.

Figura 5.8: **Posterior Média:** a) Gráfico mostrando as curvas de aprendizado para uma situação onde há apenas uma única sequência de tamanho T a ser aprendida dividida em diferentes tamanhos de subsequências indicados na legenda; b) mesmo gráfico em escala log-log.

corresponde ao lançamento de uma única moeda simples, que pode ou não estar viciada. Este é um caso particular de HMM onde temos apenas 1 estado escondido ($n = 1$) e 2 estados observados ($m = 2$), que chamaremos de H (*head* ou cara) e T (*tail* ou coroa). Neste caso, então, apenas dois parâmetros serão aprendidos: a probabilidade do estado H ser emitido, que simbolizaremos por ξ_H , e a probabilidade do estado T ser emitido, simbolizada por ξ_T .

Considere, agora, o evento no qual lançamos essa moeda e obtemos o estado H. Vejamos como os parâmetros mudam segundo cada um dos algoritmos.

5.3.1 BKL

A nossa distribuição *a priori* para os parâmetros do HMM será uma Dirichlet de duas variáveis (também conhecida como **distribuição Beta**)

$$P_0(\xi_H, \xi_T) = \frac{\Gamma(u_H + u_T)}{\Gamma(u_H)\Gamma(u_T)} (\xi_H)^{u_H-1} (\xi_T)^{u_T-1}. \quad (5.39)$$

Sendo o resultado do lançamento igual ao estado H e simbolizando os novos parâmetros após o aprendizado por $\hat{u}$, obteremos o seguinte sistema digama de equações

$$\begin{aligned} \psi(\hat{u}_H) - \psi(\hat{u}_H + \hat{u}_T) &= \mu_H, \\ \psi(\hat{u}_T) - \psi(\hat{u}_H + \hat{u}_T) &= \mu_T, \end{aligned} \quad (5.40)$$

com

$$\begin{aligned} \mu_H &= \psi(u_H + 1) - \psi(u_H + u_T + 1), \\ \mu_T &= \psi(u_T) - \psi(u_H + u_T + 1). \end{aligned} \quad (5.41)$$

A solução é facilmente obtida por inspeção e resulta nas seguintes fórmulas de atualização para os dois parâmetros da distribuição :

$$\begin{aligned} \hat{u}_H &= u_H + 1, \\ \hat{u}_T &= u_T. \end{aligned} \quad (5.42)$$

Se os valores iniciais de u_H e u_T forem colocados como sendo iguais ao número já observado de caras e coroas, respectivamente, em uma sequência anterior ao lançamento, o resultado é que a estimativa para as probabilidades será simplesmente a sua frequência relativa, o que é exatamente o resultado mostrado no Capítulo 2 de [95].

5.3.2 PM

Utilizando a mesma *priori* do algoritmo BKL, teremos que a distribuição posterior, após a obtenção de uma cara, será dada por

$$Q(\xi_H, \xi_T) = \frac{1}{Z_Q} \xi_H P_0(\xi_H, \xi_T), \quad (5.43)$$

$$Z_Q = \langle \xi_H \rangle, \quad (5.44)$$

onde a média é calculada na distribuição *a priori*. Sendo os novos parâmetros dados por $\hat{u}$, temos

$$\hat{u}_0 = \hat{u}_H + \hat{u}_T = \frac{\langle \xi_H \rangle_Q - \langle (\xi_H)^2 \rangle_Q}{\langle (\xi_H)^2 \rangle_Q - (\langle \xi_H \rangle_Q)^2}, \quad (5.45)$$

$$\langle \xi_H \rangle_Q = \frac{\langle (\xi_H)^2 \rangle}{\langle \xi_H \rangle} = \frac{u_H + 1}{u_H + u_T + 1}, \quad (5.46)$$

$$\langle (\xi_H)^2 \rangle_Q = \frac{\langle (\xi_H)^3 \rangle}{\langle \xi_H \rangle} = \frac{(u_H + 1)(u_H + 2)}{(u_H + u_T + 1)(u_H + u_T + 2)}, \quad (5.47)$$

e, portanto,

$$\hat{u}_0 = \hat{u}_H + \hat{u}_T = u_H + u_T + 1, \quad (5.48)$$

resultando em

$$\hat{u}_H = u_H + 1, \quad (5.49)$$

$$\hat{u}_T = u_T,$$

que coincide com o resultado do algoritmo anterior.

A aplicação dos dois algoritmos bayesianos tratados nesta tese (BKL e PM) a este problema simples deixa claro que as equações obtidas através deles fazem sentido intuitivamente, embora em situações mais complicadas elas introduzam refinamentos que podem ser considerados como correções aos algoritmos mais básicos.

5.4 Conceitos Variáveis

Existem situações de aprendizado onde o professor (ou a regra a ser aprendida) pode mudar com o tempo. Um caso especial é o chamado **Teste de Wisconsin** [37, 101]. Originalmente criado para tentar detectar lesões no lobo pré-frontal de pacientes, o teste mede a capacidade de uma pessoa de inferir mudanças de regra. Daremos, a seguir, uma descrição resumida desse teste.

Consideremos um baralho onde as cartas contém um determinado conjunto de figuras desenhadas de tal maneira que as cartas possam ser arranjadas em diferentes categorias de maneiras variadas. Em um baralho comum, por exemplo, poderíamos classificar as cartas pelas suas cores (vermelhas/pretas), por seus números (pares/ímpares) etc. O examinador, isto é, a pessoa que irá aplicar o teste, escolhe então uma regra de classificação e pede para o paciente tentar inferí-la e aplicá-la a uma seqüência de cartas que será mostrada. A cada carta mostrada, o examinador diz ao paciente se ele a classificou corretamente ou não. O examinador então, a qualquer momento, escolhe uma nova regra de classificação e mede o número de exemplos depois do qual o paciente é capaz de inferir a nova regra.

Utilizamos uma forma adaptada desse teste para observar o comportamento dos dois principais algoritmos *on-line* estudados em nosso trabalho: BC e PM. Traçamos a curva de aprendizado para os dois algoritmos com os parâmetros $N_T = 200$, $n = 2$, $m = 3$ e com alunos iniciais simétricos para um total de 3000 seqüências. Para o algoritmo BC, utilizamos $\lambda = 0.01$ e $\eta_{BC} = 10.0$. A cada 500 seqüências, o professor inicial é reescolhido aleatoriamente. O gráfico resultante está mostrado na figura 5.9.

Podemos observar que o algoritmo BC é surpreendentemente adaptativo quanto à mudança de regra, o que já não ocorre com PM. A cada mudança de professor, BC consegue rapidamente reduzir a distância entre o aluno e o professor quase ao mesmo valor do instante anterior à mudança na regra. Já PM mostra um efeito de persistência na regra antiga, apresentando um aprendizado cada vez mais lento após cada mudança, um efeito já observado em percéptrons na mesma situação [101]. A explicação desse efeito para o algoritmo bayesiano PM é de que a covariância da distribuição posterior projetada diminui ao longo do aprendizado de forma rápida (um dos motivos pelo qual o seu aprendizado é melhor) e, no momento de uma mudança, ela já está estreita demais para conseguir se adaptar aos novos dados. O algoritmo BC, embora não tenha uma covariância explícita, provavelmente possui uma covariância efetiva que não estreita tanto e tão rápido como a de PM, resultando numa aprendizagem inferior em situações estáticas, mas em uma melhor adaptatividade em casos onde a regra varia.

A figura 5.10 mostra o resultado da comparação entre PM e BC para o caso onde, a cada seqüência, o professor recebe um pequeno acréscimo aleatório. O acréscimo é feito somando ao professor um outro professor gerado aleatoriamente multiplicado por um fator que, para esta simulação, fixamos em 10^{-5} . Nota-se, novamente, que BC é muito mais adaptativo que PM, cuja distância ao professor cresce muito rapidamente.

Tanto a figura 5.9 quanto a figura 5.10 são qualitativamente idênticas a estudos similares feitos para percéptrons [50, 101], mesmo sendo modelos de aprendizado completamente diferentes, o que sugere que esses possam ser comportamentos universais indepen-

Figura 5.9: **Teste de Wisconsin para HMMs:** Gráfico comparativo da performance de PM versus BC. A linha contínua representa PM e a linha tracejada representa BC. O professor é mudado aleatoriamente a cada 500 seqüências.

dentes do modelo estudado. No entanto, para confirmar essa especulação seria necessária uma demonstração analítica embora testes numéricos com outros modelos possam ser realizados para verificar sua plausibilidade.

5.5 Aplicação : Previsão de Índices

Como aplicação do algoritmo PM que desenvolvemos neste trabalho, consideraremos uma versão simplificada de aplicação em bolsa de valores utilizando dados reais do índice da Bolsa de Valores de São Paulo (IBOVESPA).

Utilizaremos uma versão deflacionada do valor de fechamento desse índice obtida pelos autores de [102] e utilizada no artigo citado. Usamos as cotas referentes ao período desde janeiro de 1990 a janeiro de 2005.

Figura 5.10: **Mudança Aleatória Contínua:** Gráfico comparativo da performance de PM versus BC. A linha contínua representa PM e a linha tracejada representa BC. O professor sofre uma mudança aleatória pequena a cada sequência.

A situação analisada será bastante simplificada e idealizada. O objetivo é apenas estudar a performance dos HMMs quando utilizados para aprender uma série temporal real. Modelos mais realistas podem ser obtidos com as modificações das regras que usaremos a seguir.

O procedimento consiste em:

- 1) O **investidor** inicia com um montante M de dinheiro.
- 2) Dada a série histórica do fechamento da bolsa até o dia t , o investidor deve escolher que posição irá tomar: se ficará comprado (C), vendido (V) ou fora do mercado (F) no dia

$t + 1$.

3) Ao final do dia, o investidor realiza seus lucros de acordo com o valor de fechamento e soma o resultado ao seu montante original M .

4) O investidor prossegue a partir do passo 2 indefinidamente, sempre tendo à sua disposição o montante M .

O passo 2 do procedimento acima descrito requer uma explicação mais detalhada. O investidor decide qual das três posições (C, V ou F) irá tomar de acordo com a sua previsão do índice dada por uma determinada estratégia $\mathcal{E}$ à sua escolha.

Para que essa escolha seja tomada, a série temporal do índice analisado é transformada em uma série composta por esses três estados, tendo como significado a posição correta a ser tomada. Isso é feito calculando-se a variação percentual do índice para cada dia dada por

$$\Delta(t) = \frac{I(t+1) - I(t)}{I(t)} = \frac{I(t+1)}{I(t)} - 1, \quad (5.50)$$

onde $I(t)$ é o valor do índice no dia t . Um limiar para a variação percentual, que chamaremos de $\delta > 0$, é definido de tal forma que cada estado é definido comparando seu valor com o valor de Δ :

C: se $\Delta > \delta$, isso significa que a variação percentual do índice é maior que o limiar definido, ou seja, houve uma alta percentual no índice maior que δ . Quando um investidor assume essa posição, ele usa todo o dinheiro acumulado para comprar cotas do índice pelo preço de fechamento do dia t e as vende pelo valor de fechamento do dia $t + 1$.

V: se $\Delta < -\delta$, isso significa que a queda percentual do índice ultrapassa o limiar definido. Se assumir esta posição, o investidor fará dois acordos futuros que permitirão que ele use todo o seu dinheiro acumulado para comprar o índice pelo fechamento do dia $t + 1$ e vendê-lo pelo fechamento do dia t .

F: se $-\delta < \Delta < \delta$, a variação percentual do índice, tanto para cima quanto para baixo, não será suficiente para justificar o risco de investir. Nessa posição, investidor permanecerá fora do mercado, isto é, não tomará ação nenhuma.

Analisando a informação contida na série temporal transformada nessa série de estados, o investidor utilizará a sua estratégia para decidir qual deles assumirá em cada instante t .

Essas situações são altamente idealizadas. Assumimos total liquidez, isto é, sempre que queremos comprar ou vender há uma outra parte que irá fechar o negócio e que os acordos de compra e venda futuros são fechados sem nenhum custo. Assumimos, também, o valor de abertura de um dia como sendo o valor de fechamento do dia anterior. Além disso, não fazemos nenhuma aplicação com o dinheiro nos períodos em que ficamos fora do mercado.

A figura 5.11 mostra o resultado da simulação de 4 diferentes estratégias utilizadas para a previsão do índice. As duas primeiras, descritas abaixo, são extremamente ingênuas e o objetivo é utilizá-las para comparação com as duas seguintes, baseadas no aprendizado de HMMs. As duas estratégias são:

a) Aleatória: nesta estratégia, uma das três posições (C, V ou F) é simplesmente escolhida aleatoriamente a cada dia, sem nenhuma preocupação com o valor do índice ou de δ . O gráfico mostra a média dos resultados obtidos em 27000 rodadas com os mesmos dados.

A média mostrada no gráfico pode ser calculada analiticamente. Simbolizemos por $G(t)$ o ganho do investidor no dia t . Caso ele fique fora do mercado (posição F), temos que $G(t) = 0$. Caso ele fique comprado (posição C) temos

$$G(t) = \frac{M}{I(t-1)} I(t) - M = M \frac{\Delta I(t)}{I(t-1)}, \quad (5.51)$$

onde $\Delta I(t) = I(t) - I(t-1)$. Por fim, caso o investidor fique vendido (posição V), seu ganho será

$$G(t) = \frac{M}{I(t)} I(t-1) - M = -M \frac{\Delta I(t)}{I(t)}. \quad (5.52)$$

Como qualquer uma das três posições pode ser assumida aleatoriamente com probabilidade $1/3$, o ganho médio em t é facilmente obtido

$$\begin{aligned} \langle G(t) \rangle &= \frac{1}{3} \frac{M \Delta I(t)}{I(t-1)} - \frac{1}{3} \frac{M \Delta I(t)}{I(t)} \\ &= \frac{M \Delta I(t)}{3} \left(\frac{1}{I(t-1)} - \frac{1}{I(t)} \right) \\ &= \frac{M [\Delta I(t)]^2}{3 I(t) I(t-1)}. \end{aligned} \quad (5.53)$$

Se traçarmos a curva correspondente, ela coincide com a apresentada no gráfico.

b) Dia Anterior: essa estratégia supõe que a variação do índice em $t + 1$ será exatamente igual à variação em t e uma posição é tomada comparando essa variação com δ da maneira já explicada.

As outras duas estratégias consistem em treinar um HMM extremamente simples, com dois estados escondidos ($n = 2$), que podem ser interpretados como duas tendências distintas do mercado (otimista ou *bullish* e pessimista ou *bearish*), e três estados observados correspondentes às três posições C, V e F. O HMM é treinado de forma *on-line* utilizando os valores de fechamento do IBOVESPA sendo que uma das estratégias consiste em treiná-lo com o algoritmo BC (seção 4.5) e a outra com o algoritmo PM (seção 5.2).

Pela seção anterior (seção 5.4), sabemos que BC se adapta melhor que PM a situações com professor variável. Devido ao fato do índice da bolsa estar sujeito a um número muito grande de influências e que essas influências podem mudar com o tempo, esperamos que BC forneça melhores resultados que PM nesse caso. Por exemplo, o comportamento do índice pode ser afetado por mudanças na política econômica ou alterações na sua carteira.

A previsão com os HMMs é feita da seguinte forma: para que os algoritmos *on-line* possam ser aplicados, quebramos a série de cotas em subsequências de tamanho T_s . Assim, a última subsequência aprendida sempre corresponde às T_s últimas cotas antes do dia da previsão. Após o HMM ter aprendido até o dia da previsão, calculamos a probabilidade $\mathcal{P}(y_{T_s+1}|y_1^{T_s})$ do próximo estado observado dada a última subsequência. Isso pode ser obtido facilmente através de

$$\mathcal{P}(y_1^{T_s+1}) = \mathcal{P}(y_{T_s+1}|y_1^{T_s})\mathcal{P}(y_1^{T_s}), \quad (5.54)$$

e, portanto,

$$\mathcal{P}(y_{T_s+1}|y_1^{T_s}) = \frac{\mathcal{P}(y_1^{T_s+1})}{\mathcal{P}(y_1^{T_s})}. \quad (5.55)$$

Calculadas essas probabilidades para cada um dos três estados observados, a nossa previsão será o estado de maior probabilidade, isto é, denominando a previsão de y^* ,

$$y^* = \arg \max_o \mathcal{P}(y_{T_s+1} = o|y_1^{T_s}). \quad (5.56)$$

Em todas as simulações o investidor começa com uma quantia $M = 10000$ unidades monetárias e o tamanho das subsequências dos HMMs utilizado foi $T_s = 2$. Pode-se observar que durante aproximadamente os primeiros 11 anos, a estratégia de usar o resultado do dia anterior é melhor que qualquer outra, mas após esse período, os HMMs começam a aprender melhor e ultrapassam essa estratégia. Como previsto, BC se adapta antes de PM. A estratégia aleatória é, como esperado, a pior de todas.

Figura 5.11: **Previsão do Ibovespa:** a) Gráfico comparativo da performance de 4 diferentes estratégias: apostando aleatoriamente (Aleat.), apostando de acordo com o resultado do dia anterior (Dia Ant.), e usando os HMMs treinados pelos algoritmos PM e BC com subsequências de tamanho 2. b) Gráfico mostrando a evolução no tempo do índice Ibovespa deflacionado.

Simulações para tamanhos diferentes de subsequências nos algoritmos BC e PM indicam pouca variação nas conclusões: BC continua tendo um desempenho melhor que PM. Observa-se que o gráfico de PM é praticamente o mesmo independentemente do tamanho da subsequência, enquanto BC apresenta variações um pouco maiores, porém pouco importantes na prática.

5.6 Aprendizado e Quebra de Simetria

As simulações que apresentamos durante a análise dos algoritmos de aprendizado mostraram valores médios da distância d_{KL} entre o HMM aluno e o HMM professor durante o processo de aprendizado para um número N_τ de professores gerados aleatoriamente e, em alguns casos, também para um número N_σ de alunos gerados aleatoriamente. Tais médias têm como objetivo a obtenção de curvas de aprendizado suaves, isto é, que não apresentem quedas abruptas na generalização.

Embora tenhamos estudado médias com a finalidade dessas quedas não aparecerem em nossos gráficos, a existência delas está associada a um fenômeno fundamental no aprendizado, a quebra de simetria nos parâmetros do aluno, e que é um objeto importante de estudo no aprendizado *on-line* [41].

As partes das curvas de aprendizado que são praticamente planas antes ou depois de uma queda são chamadas de **platôs** e correspondem a uma situação onde é difícil quebrar a simetria entre os parâmetros do aluno durante o aprendizado.

A seguir, para visualizarmos tais quebras de simetria, mostraremos simulações dos algoritmos estudados nesta tese com um aluno inicial simétrico aprendendo de apenas um único professor. Mostramos, então, sua curva de aprendizado e os valores de seus parâmetros à medida que o aprendizado acontece. Os gráficos mostram claramente que os pontos onde acontece um salto na generalização do aluno são exatamente os momentos onde ocorre uma quebra de simetria em seus parâmetros: eles deixam de ter o mesmo valor para adquirirem valores diferentes entre si.

Em todos os casos, o HMM professor possui $n = 2$ e $m = 3$ com parâmetros dados por

$$\pi = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad A = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad B = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad (5.57)$$

e o tamanho das seqüências é $T = 2$, de tal forma que o professor gera apenas a seqüência $y = \{1, 3\}$.

O gráfico 5.12 mostra o resultado para o aprendizado de 3000 seqüências através do algoritmo BC com parâmetros $\lambda = 0.01$ e $\eta_{BC} = 1.0$. O gráfico mostra duas grandes

Figura 5.12: **Quebra de Simetria:** Gráfico mostrando a distância entre aluno e professor e os valores das entradas dos parâmetros do HMM aluno em função do número de seqüências aprendidas para o algoritmo BC.

quedas em d_{KL} : uma logo no início do aprendizado e uma em torno de 1000 seqüências. Os gráficos de π e A só apresentam quebra de simetria nos parâmetros na segunda queda, já o gráfico de B , onde é mostrada apenas uma das linhas dessa matriz, pois a outra só difere por uma permutação, nota-se claramente que cada queda em d_{KL} representa uma quebra de simetria em suas entradas e que as regiões de platô representam intervalos de tempo em que o aluno não consegue aumentar a separação entre os parâmetros restantes que, embora já sejam diferentes, continuam muito próximos até a próxima quebra de simetria. No gráfico 5.13, que mostra a mesma situação de aprendizado para o algoritmo

Figura 5.13: **Quebra de Simetria:** Gráfico mostrando a distância entre aluno e professor e os valores das entradas dos parâmetros do HMM aluno em função do número de seqüências aprendidas para o algoritmo PM.

PM, a segunda queda é muito mais acentuada e nota-se que a quebra de simetria não ocorre apenas em B , mas também em A . Para fins de comparação, o gráfico 5.14 mostra a mesma situação para o algoritmo BWO com uma taxa de aprendizado $\eta_{BW} = 0.01$. Nesse algoritmo, diferentemente dos outros dois, só ocorre quebra de simetria no começo do aprendizado e apenas na matriz B , sendo que as matrizes π e A continuam com o mesmo valor ao longo de todo o aprendizado.

Podemos notar, comparando os três algoritmos acima, que quanto mais simetrias são quebradas, melhor é a generalização do algoritmo.

Figura 5.14: **Quebra de Simetria:** Gráfico mostrando a distância entre aluno e professor e os valores das entradas dos parâmetros do HMM aluno em função do número de seqüências aprendidas para o algoritmo BWO.

Através desse processo de quebra de simetria, podemos também explicar o porquê da curva de aprendizado da média nos alunos de BW ficar acima do aluno inicial simétrico e a de BWO ficar abaixo. Em BW, a simetria das entradas das matrizes é quebrada nas iterações do algoritmo para cada exemplo, o que faz com que o aluno simétrico não seja necessariamente pior que os outros não simétricos. Já em BWO, a simetria demora mais a ser quebrada, o que faz com que o aluno simétrico fique em desvantagem. Isso significa que, quanto mais o algoritmo demora para quebrar a simetria, pior é o resultado do aluno inicial simétrico em comparação com os outros e quanto antes o algoritmo quebrar a

simetria, menos relevante é a simetria do aluno inicial.

Capítulo 6

Conclusões e Perspectivas

6.1 Conclusões

Apresentamos, neste trabalho, um estudo comparativo da performance de cinco diferentes algoritmos de aprendizado para HMMs, dois deles já existentes na literatura (BW e BC) e três deles sugeridos por nós (BWO, BKL e PM). Esses estudos foram feitos utilizando-se duas diferentes medidas de distâncias entre o HMM professor e o HMM aluno durante o aprendizado, a saber, d_{KL} e d_E .

6.1.1 Distâncias entre HMMs

Com relação às distâncias utilizadas, podemos dizer que d_{KL} é a mais apropriada para as análises devido à sua base teórica mais firme. Podemos dizer que a distância d_E é uma boa aproximação para ser usada quando queremos comparar a performance de algoritmos diferentes, principalmente em casos onde d_{KL} é muito custosa computacionalmente ou quando não fornece a informação desejada devido ao fato das distribuições comparadas não serem absolutamente contínuas uma em relação à outra. No entanto, as simulações mostraram que precisamos ter cautela com a interpretação dos resultados de d_E pois, no caso do algoritmo PM, por exemplo, o resultado assintótico dado por d_E é completamente diferente daquele dado por d_{KL} . A nossa sugestão é que d_E , com os cuidados descritos acima, seja usada quando d_{KL} é inviável de ser calculada, mas que esta última sempre tenha preferência para análises mais precisas como comportamentos assintóticos do erro de generalização.

No entanto, ambas as distâncias citadas só são úteis quando temos acesso ao professor. Durante o nosso trabalho observamos que talvez seja possível utilizar as variações nas

matrizes de emissão do aluno como uma aproximação para essas medidas, mas poucos testes foram feitos e precisamos trabalhar mais nesse ponto para podermos fazer afirmações mais concretas.

6.1.2 Algoritmos

Quanto aos algoritmos analisados, concluímos que, apesar de mais custoso computacionalmente que os algoritmos não bayesianos, o algoritmo bayesiano aproximado PM leva a uma melhor generalização em uma situação estática e possui uma performance praticamente igual, com um tempo de computação muito inferior, ao algoritmo bayesiano BKL.

Em situações onde existe uma mudança na regra a ser aprendida durante a fase de aprendizado, o algoritmo bayesiano PM tem uma performance inferior ao algoritmo BC. O primeiro demora a se adaptar à nova regra, o que o segundo faz com uma grande rapidez para mudanças bruscas e, embora não consiga seguir o professor no caso de mudanças contínuas, se afasta dele mais lentamente que PM.

Comparando os gráficos obtidos nesta tese com resultados de estudos análogos feitos para percéptrons com camadas escondidas [83, 104, 110], tanto em situações estáticas quanto em situações onde o professor varia, notamos uma semelhança qualitativa entre as curvas de aprendizado nos dois modelos. Levando em conta a grande diferença entre percéptrons e HMMs, esses resultados parecem sugerir que o comportamento qualitativo das curvas de aprendizado apresenta um caráter geral que parece independe do particular modelo utilizado estando relacionado mais diretamente com a situação e o algoritmo utilizado para o aprendizado. No entanto, esta afirmação precisa de muito mais testes e frisamos que é apenas uma especulação baseada nas nossas simulações aqui apresentadas.

6.2 Perspectivas

Durante o trabalho tivemos idéias para vários estudos e desenvolvimentos futuros com os quais, por motivos diversos, não tivemos tempo de prosseguir. A seguir, apresentamos uma lista deles com alguns comentários.

6.2.1 Estudos Teóricos

Generalizações

- Levando-se em conta que HMMs são casos particulares, existe a possibilidade dos algoritmos desenvolvidos nesta tese poderem ser generalizados de tal forma que possam ser aplicados a Modelos Gráficos gerais.

Análise dos Algoritmos

- Uma possibilidade que nos ocorreu inicialmente, mas que postergamos com o intuito de fazer um estudo computacional mais detalhado dos algoritmos, foi a de utilizar técnicas de réplicas. Isso é uma possibilidade interessante para um trabalho futuro.
- Um outro estudo interessante é sobre a possibilidade de encontrarmos um fator de escala, a exemplo do α em redes neurais, nas curvas de aprendizado. Como em redes neurais temos que $p = \alpha N$, onde p é o número de exemplos e N a dimensão dos vetores de entrada, esse fator nos HMMs deve estar relacionado ao número de seqüências, ao tamanho delas e ao número de estados escondidos e observados. Não fizemos testes a esse respeito ainda, mas pretendemos estudar essa possibilidade no futuro.
- Gostaríamos de entender melhor o que acontece com a média nos alunos para o algoritmo BW para podermos compreender o porquê de um aluno inicial simétrico ser sempre a melhor escolha.
- Os algoritmos estudados envolvem uma série de parâmetros, por exemplo, em BWO temos η_{BW} , em BC temos λ e η_{BC} e em PM temos a escolha de qual covariância usar na distribuição projetada. Gostaríamos de estudar mais a fundo a influência desses parâmetros em cada algoritmo não só do ponto de vista numérico como também, se possível, analiticamente.
- Pretendemos estudar com mais detalhes como os algoritmos se comportam quando variamos n , m e o tamanho das seqüências aprendidas. As simulações parecem mostrar que os aspectos qualitativos não mudam, mas acreditamos que um estudo mais rigoroso se faz necessário.
- A relação entre aprendizado e quebra de simetria parece ter implicações profundas e acreditamos que resultados gerais, aplicáveis a qualquer espécie de máquina,

possam ser obtidos com um estudo mais aprofundado. Esse é um tema ao qual pretendemos estender nossas pesquisas futuramente.

6.2.2 Aplicações

Bioinformática

Apesar de termos estudado a possibilidade, não tivemos tempo suficiente para aplicar os algoritmos aqui estudados em problemas de análise de DNA e proteínas. Como essas aplicações têm um interesse tecnológico muito grande, pretendemos começar os testes com elas assim que possível.

Computação

HMMs podem ser utilizados para geração de programas usando uma representação para estes análoga à utilizada em [72] se observarmos que a sequência de instruções que forma um programa em LISP pode ser interpretada como uma sequência de estados.

Econometria

Observamos que a aplicação de HMMs à previsão de um índice econômico leva a resultados interessantes. Uma direção futura seria aplicar modelos mais complexos de HMMs treinados com os algoritmos apresentados para tentar prever situações um pouco mais complexas e realistas.

Existe, também, um controverso “método” para previsão em economia chamado de “Elliot Waves” [71], que é baseado na observação de que certos padrões se repetem nos gráficos dos índices. Pode-se pensar em utilizar HMMs para verificar se esses padrões realmente existem e utilizá-los de forma mais eficiente.

Linguística

A medida de quão fonética é uma língua talvez possa ser obtida utilizando-se a topologia do HMM aluno após o aprendizado.

Apêndice A

O Método de Descida pelo Gradiente

O método de **descida pelo gradiente**, também chamado de método do máximo declive ou pelos nomes em inglês *steepest descent method* e *gradient descent* [79], é um método numérico geral para a obtenção de um mínimo local de uma função $f(\mathbf{x})$ (e, portanto, do máximo de $-f(\mathbf{x})$). Sua aplicação na área de aprendizado em máquinas se faz como um algoritmo para implementação de técnicas que visam a extremização de alguma função de custo, como mínimos quadrados ou máxima verossimilhança.

Considere uma função $f : \mathbb{R}^n \mapsto \mathbb{R}$. O gradiente dessa função, dado por

$$\nabla f(\mathbf{x}) = \left(\frac{\partial f(\mathbf{x})}{\partial x_1}, \frac{\partial f(\mathbf{x})}{\partial x_2}, \dots, \frac{\partial f(\mathbf{x})}{\partial x_n} \right), \quad (\text{A.1})$$

indica a direção na qual a função cresce mais rápido.

A descida pelo gradiente é um método iterativo que visa ir caminhando, a cada passo de iteração na direção contrária ao gradiente calculado no ponto em questão até que um mínimo local seja atingido, ou seja, a cada passo temos

$$\mathbf{x}^{p+1} = \mathbf{x}^p - \eta \nabla f(\mathbf{x}^p), \quad (\text{A.2})$$

onde $\eta > 0$ é geralmente chamada de **taxa de aprendizado** na maioria das aplicações em aprendizado de máquinas.

Manter η constante ao longo da minimização geralmente não é uma estratégia muito eficiente e existem métodos denominados de **métodos de gradiente conjugado** [11, 79] que resultam em algoritmos mais eficientes computacionalmente. No entanto, tais métodos não serão por nós utilizados e uma descrição detalhada está fora do objetivo principal deste trabalho.

Apêndice B

Distribuições de Dirichlet

B.1 Definições

A Distribuição de Dirichlet para um vetor n -dimensional $x = (x_1, x_2, \dots, x_n)$ que obedeça aos vínculos

$$0 \leq x_i \leq 1, \sum_{i=1}^n x_i = 1, \quad (\text{B.1})$$

é dada por

$$\mathcal{D}(x|u) = \frac{1}{Z(u)} \prod_{i=1}^n x_i^{u_i-1}, \quad u_i > 0, \quad (\text{B.2})$$

onde a normalização é dada por

$$Z(u) = \frac{\prod_{i=1}^n \Gamma(u_i)}{\Gamma(u_0)}, \quad (\text{B.3})$$

com

$$u_0 = \sum_{i=1}^n u_i. \quad (\text{B.4})$$

B.2 Normalização

A seguir, mostraremos como é obtida a normalização da Distribuição de Dirichlet. Procedemos integrando a distribuição não normalizada com a medida correta, isto é, obser-

vando os vínculos aos quais o vetor x deve obedecer

$$\begin{aligned} Z(u) &= \int dx \delta \left(\sum_i x_i - 1 \right) \prod_i \theta(x_i) x_i^{u_i-1} = \\ &= \int \frac{dk}{2\pi} e^{ik} \prod_i \int dx_i \theta(x_i) e^{-ikx_i} x_i^{u_i-1} = \\ &= \int \frac{dk}{2\pi} e^{ik} \prod_i I(u_i), \end{aligned} \quad (\text{B.5})$$

onde θ é a função degrau de Heaviside

$$\theta(x) = \begin{cases} 1 & x \geq 0, \\ 0 & x < 0, \end{cases} \quad (\text{B.6})$$

e

$$I(u, k) = \int dx \theta(x) e^{-ikx} x^{u-1} = \int_0^\infty dx e^{-ikx} x^{u-1}. \quad (\text{B.7})$$

Fazendo a mudança de variáveis $y = ikx$, a integral se torna

$$I(u, k) = \frac{1}{(ik)^u} \int_0^{ik\infty} dy e^{-y} y^{u-1}. \quad (\text{B.8})$$

Precisamos analisar separadamente os casos $k > 0$ e $k < 0$. Para $k > 0$, utilizaremos o contorno de integração no plano complexo indicado na figura B.1 formado pelas três partes C_1 , C_2 e C_3 . A função a ser integrada é

$$f(z) = e^{-z} z^{u-1}, \quad (\text{B.9})$$

que não possui pólos no interior do contorno completo, que chamaremos de $C = C_1 + C_2 + C_3$, e, portanto, tem como resultado o valor zero

$$\int_C f(z) dz = \int_{C_1} f(z) dz + \int_{C_2} f(z) dz + \int_{C_3} f(z) dz = 0. \quad (\text{B.10})$$

Fazendo-se $R \rightarrow \infty$, a integral em C_3 tende a zero. A integral em C_2 é feita em $\mathbb{R}$ e nada mais é que a representação integral da função gama com um sinal negativo devido à orientação do contorno. Assim, temos que

$$I(u, k) = \frac{1}{(ik)^u} \int_0^{ik\infty} dy e^{-y} y^{u-1} = \frac{\Gamma(u)}{(ik)^u}, \quad (\text{B.11})$$

para $k > 0$. O caso $k < 0$ pode ser analisado analogamente simplesmente utilizando o contorno mostrado na figura B.2, com o mesmo resultado. Substituindo em $Z(u)$, obtemos

$$Z(u) = \frac{\prod_{i=1}^n \Gamma(u_i)}{2\pi i^{u_0}} A(u_0), \quad (\text{B.12})$$

Figura B.1: **Normalização da Distribuição de Dirichlet:** Caminho de integração no plano complexo utilizado para o caso $k > 0$.

onde

$$A(x) \equiv \int_{-\infty}^{\infty} dk \frac{e^{ik}}{k^x}. \quad (\text{B.13})$$

Usando o fato de que

$$\frac{1}{k^x} = \frac{-1}{x-1} \frac{d}{dk} \frac{1}{k^{x-1}}, \quad (\text{B.14})$$

podemos escrever

$$A(x) = \frac{-1}{x-1} \int_{-\infty}^{\infty} dk e^{ik} \frac{d}{dk} \frac{1}{k^{x-1}} = \frac{i}{x-1} \int_{-\infty}^{\infty} dk e^{ik} \frac{1}{k^{x-1}}, \quad (\text{B.15})$$

onde, na última passagem, fizemos uma integração por partes. Portanto, podemos observar que $A(x)$ satisfaz à relação de recorrência

$$A(x) = \frac{i}{x-1} A(x-1), \quad (\text{B.16})$$

e, portanto, temos que

$$A(x) = \frac{c i^x}{\Gamma(x)}. \quad (\text{B.17})$$

O valor da constante c pode ser obtido observando-se que $A(1) = ic$. A integral em $A(1)$ é imediata e tem como resultado o valor $2\pi i$. Assim, chegamos ao resultado final

$$A(x) = \frac{2\pi i^x}{\Gamma(x)}. \quad (\text{B.18})$$

Figura B.2: **Normalização da Distribuição de Dirichlet:** Caminho de integração no plano complexo utilizado para o caso $k < 0$.

Substituindo em $Z(u)$, reproduzimos o resultado conhecido,

$$Z(u) = \frac{\prod_{i=1}^n \Gamma(u_i)}{\Gamma(u_0)}. \quad (\text{B.19})$$

B.3 Momentos

Os momentos da Distribuição de Dirichlet são facilmente calculados. Consideremos o caso geral

$$\left\langle \prod_i x_i^{\alpha_i} \right\rangle = \int \frac{dx}{Z(u)} \delta \left(\sum_i x_i - 1 \right) \prod_i \theta(x_i) x_i^{u_i-1+\alpha_i}, \quad (\text{B.20})$$

que, dada a integral calculada na seção anterior, nada mais é que

$$\left\langle \prod_i x_i^{\alpha_i} \right\rangle = \frac{\prod_i \Gamma(u_i + \alpha_i)}{\Gamma(u_0 + \alpha_0)} \frac{\Gamma(u_0)}{\prod_i \Gamma(u_i)}, \quad (\text{B.21})$$

onde $\alpha_0 = \sum_i \alpha_i$.

Por meio dessa fórmula, podemos obter as médias

$$\begin{aligned} \langle x_i \rangle &= \left[\prod_{j \neq i} \Gamma(u_j) \right] \frac{\Gamma(u_i + 1)}{\Gamma(u_0 + 1)} \frac{\Gamma(u_0)}{\prod_j \Gamma(u_j)} = \\ &= \frac{\Gamma(u_i + 1)}{\Gamma(u_i)} \frac{\Gamma(u_0)}{\Gamma(u_0 + 1)}, \end{aligned} \quad (\text{B.22})$$

resultando em

$$\langle x_i \rangle = \frac{u_i}{u_0}, \quad (\text{B.23})$$

que, obviamente, satisfazem à relação

$$\sum_i \langle x_i \rangle = 1, \quad (\text{B.24})$$

como esperaríamos

As variâncias são dadas por

$$\text{Var}(x_i) = \langle x_i^2 \rangle - \langle x_i \rangle^2. \quad (\text{B.25})$$

Calculando

$$\begin{aligned} \langle x_i^2 \rangle &= \frac{\Gamma(u_i + 2)}{\Gamma(u_i)} \frac{\Gamma(u_0)}{\Gamma(u_0 + 1)} = \\ &= \frac{u_i(u_i + 1)}{u_0(u_0 + 1)}, \end{aligned} \quad (\text{B.26})$$

temos, então

$$\begin{aligned} \text{Var}(x_i) &= \frac{u_i(u_i + 1)}{u_0(u_0 + 1)} - \frac{u_i^2}{u_0^2} = \\ &= \frac{u_i^2 u_0 + u_i u_0 - u_i^2 u_0 - u_i^2}{u_0^2(u_0 + 1)}, \end{aligned} \quad (\text{B.27})$$

resultando em

$$\text{Var}(x_i) = \frac{u_i(u_0 - u_i)}{u_0^2(u_0 + 1)}. \quad (\text{B.28})$$

Um outro momento muito útil de ser calculado é dado pela fórmula geral

$$l_i = \left\langle \left(\prod_j x_j^{\alpha_j} \right) \ln x_i \right\rangle, \quad (\text{B.29})$$

que pode ser calculado recorrendo-se a uma aplicação extremamente elementar do truque de réplicas (ou simplesmente uma continuação analítica)

$$\begin{aligned} l_i &= \left[\frac{\partial}{\partial n} \left\langle \left(\prod_{j \neq i} x_j^{\alpha_j} \right) x_i^{\alpha_i + n} \right\rangle \right]_{n=0} = \\ &= \left\{ \frac{\partial}{\partial n} \left[\frac{\prod_{j \neq i} \Gamma(u_j + \alpha_j) \Gamma(u_i + \alpha_i + n)}{\Gamma(u_0 + \alpha_0 + n)} \frac{\Gamma(u_0)}{\prod_j \Gamma(u_j)} \right] \right\}_{n=0} = \\ &= \frac{\Gamma(u_0)}{\prod_j \Gamma(u_j)} \prod_{j \neq i} \Gamma(u_j + \alpha_j) \left[\frac{\Gamma'(u_i + \alpha_i + n)}{\Gamma(u_i + \alpha_i + n)} - \frac{\Gamma(u_i + \alpha_i + n) \Gamma'(u_0 + \alpha_0 + n)}{\Gamma^2(u_0 + \alpha_0 + n)} \right]_{n=0} = \\ &= \frac{\Gamma(u_0)}{\prod_j \Gamma(u_j)} \frac{\prod_j \Gamma(u_j + \alpha_j)}{\Gamma(u_0 + \alpha_0)} \left[\frac{\Gamma'(u_i + \alpha_i)}{\Gamma(u_i + \alpha_i)} - \frac{\Gamma'(u_0 + \alpha_0)}{\Gamma(u_0 + \alpha_0)} \right], \end{aligned} \quad (\text{B.30})$$

e, portanto

$$\left\langle \left(\prod_j x_j^{\alpha_j} \right) \ln x_i \right\rangle = \left\langle \prod_j x_j^{\alpha_j} \right\rangle [\psi(u_i + \alpha_i) - \psi(u_0 + \alpha_0)], \quad (\text{B.31})$$

que serão usadas na seção 5.1 para encontrarmos a projeção da distribuição posterior na família de Dirichlets e dará origem ao sistema digama (apêndice C) de equações.

B.4 Obtenção pelo Princípio da Máxima Entropia

A Distribuição de Dirichlet pode ser obtida como a distribuição $\mathcal{D}$ que maximiza a entropia e que obedece aos seguintes vínculos ¹

$$0 \leq x_i \leq 1, \quad \sum_i x_i = 1, \quad (\text{B.32})$$

e

$$\int d\mu \mathcal{D}(x) = 1, \quad \int d\mu \mathcal{D}(x) \ln x_i = \alpha_i, \quad (\text{B.33})$$

onde a medida $d\mu$ é definida como

$$d\mu \equiv \delta \left(\sum_i x_i - 1 \right) \prod_i \theta(x_i) dx_i, \quad (\text{B.34})$$

para que as integrações sejam feitas nos conjuntos corretos obedecendo aos vínculos para x . Os vínculos para $\mathcal{D}(x)$ representam a sua condição de normalização e a fixação das médias dos logaritmos. A função entropia a ser maximizada fica então

$$\mathcal{L} = \int d\mu \mathcal{D} \ln \mathcal{D} + \lambda \left(\int d\mu \mathcal{D} - 1 \right) + \sum_i \lambda_i \left(\int d\mu \mathcal{D} \ln x_i - \alpha_i \right). \quad (\text{B.35})$$

A condição de máximo $\delta \mathcal{L} / \delta \mathcal{D} = 0$ resulta em

$$\delta \left(\sum_i x_i - 1 \right) \left[\prod_i \theta(x_i) \right] \left(1 + \ln \mathcal{D} + \lambda + \sum_i \lambda_i \ln x_i \right) = 0, \quad (\text{B.36})$$

cujas solução é dada por

$$\mathcal{D}(x) = \exp(-\lambda - 1) \prod_i x_i^{-\lambda_i}, \quad (\text{B.37})$$

que, identificando $Z = \exp(\lambda + 1)$ e $u_i = 1 - \lambda_i$, é a distribuição de Dirichlet na forma (B.2).

¹Após o término desta tese, encontramos na literatura o trabalho [107] onde uma derivação das distribuições de Dirichlet via Máxima Entropia como a nossa é apresentada. No entanto, frizamos que obtivemos a nossa dedução de forma independente sem prévio conhecimento do artigo citado.

B.5 Mistura de Dirichlets

Uma mistura de Dirichlets é uma combinação linear de distribuições de Dirichlet

$$\mathcal{Q}(x) = \frac{1}{Z_{\mathcal{Q}}} \sum_i \alpha_i \mathcal{D}_i(x). \quad (\text{B.38})$$

com

$$Z_{\mathcal{Q}} \equiv \int d\mu \mathcal{Q}(x), \quad d\mu \equiv \delta \left(\sum_i x_i - 1 \right) \prod_i \theta(x_i) dx_i. \quad (\text{B.39})$$

Considerando que as distribuições de Dirichlet $\mathcal{D}_i$ estão normalizadas, temos que

$$Z_{\mathcal{Q}} = \sum_i \alpha_i \int d\mu \mathcal{D}_i(x) = \sum_i \alpha_i. \quad (\text{B.40})$$

Note que se os coeficientes α_i da mistura somarem 1, então $Z_{\mathcal{Q}}$ também se torna 1 e a normalização é desnecessária. Consideremos o valor da soma das médias da mistura

$$\begin{aligned} \sum_j \langle x_j \rangle_{\mathcal{Q}} &= \sum_j \frac{1}{(\sum_i \alpha_i)} \sum_i \alpha_i \int d\mu x_j \mathcal{D}_i(x) = \\ &= \frac{1}{(\sum_i \alpha_i)} \sum_i \alpha_i \sum_j \langle x_j \rangle_i = \\ &= 1, \end{aligned} \quad (\text{B.41})$$

pois a soma das médias de cada Dirichlet vale 1.

Apêndice C

O Sistema Digama

O sistema de equações que chamamos de **sistema digama** aparece na obtenção do algoritmo bayesiano *on-line* para HMMs que derivamos na seção 5.1, equação (5.12) e é dado, de forma geral, por

$$\psi(u_i) - \psi\left(\sum_{j=1}^n u_j\right) = \mu_i, \quad i = 1, \dots, n, \quad (\text{C.1})$$

onde $u_i > 0$ são as incógnitas e μ_i são constantes dadas.

A solução desse sistema pode ser obtida transformando-o em um problema de encontrar um ponto fixo da forma que explicaremos a seguir. Primeiramente, definimos $u_0 = \sum_i u_i$ e isolamos u_i em cada uma das equações para obtermos

$$u_i = \psi^{-1}(\mu_i + \psi(u_0)). \quad (\text{C.2})$$

Somamos, então, sobre i

$$u_0 = \sum_i \psi^{-1}(\mu_i + \psi(u_0)), \quad (\text{C.3})$$

que é equivalente ao problema de achar o ponto fixo da função g dada por

$$g(x) = \sum_i \psi^{-1}(\mu_i + \psi(x)). \quad (\text{C.4})$$

Para achar as soluções desse problema, utilizamos a estratégia de iterar o mapa

$$x_{t+1} = \sum_i \psi^{-1}(\mu_i + \psi(x_t)), \quad (\text{C.5})$$

o que converge rapidamente para o ponto fixo.

Apêndice D

Projeção de uma Distribuição em uma Família Paramétrica

Neste apêndice apresentaremos o resultado geral para a projeção de uma distribuição qualquer em uma distribuição paramétrica que minimize a distância de Kullback-Leibler entre as duas. Para o resultado que apresentaremos, a distribuição paramétrica pertencerá a uma família que possa ser obtida através do Princípio da Máxima Entropia.

D.1 Obtenção da Família Paramétrica via Máxima Entropia

Simbolizando um membro geral da família paramétrica que queremos obter por $P(x)$, o nosso objetivo é deduzir a forma funcional das distribuições dessa família através do Princípio da Máxima Entropia. Assumiremos que a família é definida através de um conjunto de n vínculos dados por

$$\langle f_i(x) \rangle_P \equiv \int dx f_i(x) P(x) = \mu_i. \quad (\text{D.1})$$

O Princípio da Máxima Entropia diz que a forma funcional procurada é a solução da extremização da lagrangeana

$$\mathcal{L} = \int dx P \log P + \lambda \left(\int dx P - 1 \right) + \sum_i \lambda_i \left(\int dx f_i P - \mu_i \right). \quad (\text{D.2})$$

Essa extremização tem como resultado

$$P(x) = \exp \left[-(\lambda + 1) - \sum_i \lambda_i f_i(x) \right]. \quad (\text{D.3})$$

A distribuição ainda pode ser escrita como

$$P(x) = \frac{1}{Z} e^{-\sum_i \lambda_i f_i(x)}, \quad (\text{D.4})$$

com

$$Z = e^{\lambda+1}, \quad (\text{D.5})$$

que pode ser obtido usando-se a condição de normalização resultando em

$$Z = \int dx e^{-\sum_i \lambda_i f_i(x)}. \quad (\text{D.6})$$

D.2 Projeção de uma Distribuição na Família Paramétrica

Considere a distribuição de probabilidades arbitrária $Q(x)$. Queremos aproximá-la por uma distribuição $P(x)$ pertencente à família paramétrica obtida na seção anterior minimizando a divergência de Kullback-Leibler entre as duas

$$D_{KL}(Q||P) = \int dx Q(x) \log \frac{Q(x)}{P(x)}. \quad (\text{D.7})$$

A distribuição $P(x)$ é caracterizada pelas constantes λ_i e para as encontrarmos de tal forma que D_{KL} seja minimizada, temos que extremizá-la com respeito a essas constantes

$$\begin{aligned} \frac{\partial}{\partial \lambda_j} D_{KL} &= -\frac{\partial}{\partial \lambda_j} \int dx Q \log P \\ &= -\int dx Q \frac{\partial}{\partial \lambda_j} \log P \\ &= -\left\langle \frac{\partial}{\partial \lambda_j} \log P \right\rangle_Q = 0. \end{aligned} \quad (\text{D.8})$$

O logaritmo da distribuição paramétrica é obtido através de sua forma funcional dada pela equação (D.4) e nos dá

$$\log P = -\log Z - \sum_i \lambda_i f_i, \quad (\text{D.9})$$

cuja derivada é

$$\frac{\partial}{\partial \lambda_j} \log P = -\frac{1}{Z} \frac{\partial Z}{\partial \lambda_j} - f_j. \quad (\text{D.10})$$

A derivada de Z tem como resultado

$$-\frac{1}{Z} \frac{\partial Z}{\partial \lambda_j} = \langle f_j \rangle_P, \quad (\text{D.11})$$

e, substituindo-o na equação (D.10), obtemos para a derivada logarítmica de P

$$\frac{\partial}{\partial \lambda_j} \log P = \langle f_j \rangle_P - f_j, \quad (\text{D.12})$$

onde, como na seção anterior, o índice P na média indica que ela é tomada com relação à distribuição $P(x)$.

Usando esse resultado na equação (D.8), obtemos como solução do problema da projeção as condições

$$\langle f_j \rangle_P = \langle f_j \rangle_Q, \quad (\text{D.13})$$

com o índice Q indicando que a média correspondente deve ser tomada com relação à distribuição $Q(x)$. As constantes λ_i são obtidas através da solução dessas equações que podem ser diretas, como no caso da projeção em uma gaussiana como no trabalho de Oppen [74], ou requererem uma solução numérica como no caso de nossa projeção em distribuições de Dirichlet que resultam no sistema digama (ver apêndice C).

Apêndice E

Notação

E.1 Abreviações

A seguir, estão tabeladas as siglas utilizadas neste trabalho.

BC	Algoritmo de Baldi-Chauvin
BKL	Algoritmo Bayesiano <i>On-line</i>
BW	Algoritmo <i>off-line</i> de Baum-Welch
BWO	Algoritmo Baum-Welch <i>On-line</i>
GP	Gaussian Process - Processo Gaussiano
HMM	Hidden Markov Model - Cadeias de Markov com Variáveis de Estado Escondidas
MAP	Maximum a Posteriori - Posterior Máxima
ML	Maximum Likelihood - Máxima Verossimilhança
PM	Algoritmo Posterior Média
RNA	Rede Neural Artificial

E.2 Símbolos

A seguir, listamos os símbolos mais comuns que aparecem ao longo desta tese.

d	Nos gráficos, simboliza a distância entre o HMM professor e o HMM aluno
d_E	Distância Euclideana entre HMMs
d_{KL}	Distância de Kullback-Leibler entre HMMs
m	Número de estados observados de um HMM
n	Número de estados escondidos de um HMM
q	Estados escondidos de um HMM ou estados de uma cadeia de Markov
y	Estados observados de um HMM
η_{BC}	Taxa de aprendizado para o algoritmo BC
η_{BW}	Taxa de aprendizado para o algoritmo BWO
λ	Temperatura efetiva no algoritmo BC
π	Vetor de probabilidades iniciais em um HMM ou em uma cadeia de Markov
θ	Conjunto de parâmetros de uma distribuição de probabilidades
A	Matriz de transição de um HMM ou de uma cadeia de Markov
B	Matriz de emissão de um HMM
D_P	Conjunto dos P dados observados em um determinado experimento
D_{KL}	Distância de Kulback-Leibler entre duas distribuições de probabilidades.
M	Número de variáveis escondidas em um modelo
N	Número de variáveis observadas em um modelo
N_σ	Número de alunos em uma simulação
N_τ	Número de professores em uma simulação
O	Conjunto dos estados observados de um HMM
P	Número de dados para um determinado aprendizado
S	Conjunto dos estados escondidos de um HMM ou de uma cadeia de Markov
T	Tamanho da sequência observada
$\Pi(V)$	Pais do vértice V
$\mathbb{N}$	Conjunto dos números naturais
$\mathbb{R}$	Conjunto dos números reais
$\mathcal{P}$	Operador probabilidade, isto é, $\mathcal{P}(a)$ indica a probabilidade de ocorrência de a
$\mathcal{Q}$	Conjunto das variáveis escondidas de um modelo, mistura de Dirichlets
$\mathcal{X}$	Conjunto de todas as variáveis de um modelo
$\mathcal{Y}$	Conjunto das variáveis observadas de um modelo

Referências Bibliográficas

- [1] Aji, S.M., McEliece, R.J. - *The Generalized DIstributive Law* - IEEE Trans. on Information Theory **46**, 325-343 (2000)
- [2] Bach, F.R., Jordan, M.I. - *Learning Graphical Models with Mercer Kernels* - Preprint, 1-8 (2002)
- [3] Baggenstoss, P.M. - *A Modified Baum-Welch Algorithm for Hidden Markov Models with Multiple Observation Spaces* - IEEE Trans. on Speech and Audio Process. **9**, 411-416 (2001)
- [4] Bahlmann, C., Burkhardt, H. - *Measuring HMM Similarity with the Bayes Probability of Error and its Application on Online Handwriting Recognition* - Proc. of the 6th ICDAR, 406-411 (2001)
- [5] Baldi, P., Chauvin, Y. - *Smooth On-Line Learning Algorithms for Hidden Markov Models* - Neural Computation **6**, 307-318 (1994)
- [6] Baldi, P., Brunak, S. - *Bioinformatics: The Machine Learning Approach* - Second Edition, MIT Press (2001)
- [7] Beal, M.J., Ghahramani, Z., Rasmussen, C.E. - *The Infinite Hidden Markov Model* - Neural Information Processing Systems **14**, 577-585 (2002)
- [8] Bengio, Y. - *Markovian Models for Sequential Data* - Neural Computing Surveys **2**, 129-162 (1999)
- [9] Bengio, Y., Frasconi, P. - *Input/Output HMMs for Sequence Training* - IEEE Trans. on Neural Networks **7**, 1231-1249 (1996)
- [10] Bilmes, J.A. - *A Gentle Tutorial of the EM Algorithm and its Application to Parameter Estimation for Gaussian Mixture and Hidden Markov Models* - Tech. Rep. International Computer Science Institute, Berkeley, TR-97-021, 1-15 (1998)

- [11] Bishop, C. - *Neural Networks for Pattern Recognition* - Oxford University Press (1995)
- [12] Boyd, S., Vandenberghe, L. - *Convex Optimization* - Cambridge University Press (2004)
- [13] Brémaud, P. - *Markov Chains, Gibbs Fields, Monte Carlo Simulation, and Queues* - Springer-Verlag New York (1999)
- [14] Cappé, O. - *Ten years of HMMs* - Preprint, 1-23 (2001) - Disponível na Internet no endereço: <http://www.tsi.enst.fr/~cappe/docs/hmmbib.html>
- [15] Caticha, A., Preuss, R. - *Maximum Entropy and Bayesian Analysis: Entropic Priors* - physics/0307055, 1-23 (2003)
- [16] Chai, W., Vercoe, B. - *Folk Music Classification Using Hidden Markov Models* - Proceedings of International Conference on Artificial Intelligence (2001)
- [17] Chien, J.T. - *Online Hierarchical Transformation of Hidden Markov Models for Speech Recognition* - IEE Trans. on Speech and Audio Process. **7**, 656-667 (1999)
- [18] Cook, S.A. - *The Complexity of Theorem-Proving Procedures* - Proceedings of the 3rd ACM Symposium on Theory of Computing, 151-158 (1971)
- [19] Cover, T., Thomas, J. - *Elements of Information Theory* - John Wiley & Sons, New York (1991)
- [20] de Moraes, R.M., Machado, L.S. - *Hidden Markov Models for Learning Evaluation in Virtual Reality Simulators* - International Journal of Computers & Applications **25**, 212-215 (2003)
- [21] Davis, R.I.A., Lovell, B.C. - *Comparing and evaluating HMM ensemble training algorithms using train and test and condition number criteria* - Pattern Anal. Applic. **6**, 327-336 (2003)
- [22] Dempster, A., Laird, N., Rubin, D. - *Maximum likelihood from incomplete data via the EM algorithm* - J. Royal Stat. Soc., Series B, **39**, 1-38 (1977)
- [23] Dugad, R., Desai, U.B. - *A Tutorial on Hidden Markov Models* - Technical Report No. SPANN-96.1, 1-16 (1996)

- [24] Durbin, R., Eddy, S., Krogh, A., Mitchison, G. - *Biological sequence analysis: Probabilistic models of proteins and nucleic acids* - Cambridge University Press, Cambridge (1998)
- [25] Engel, A., Van den Broeck, C.P.L. - *Statistical Mechanics of Learning* - Cambridge University Press (2001)
- [26] Ephraim, Y., Merhav, N. - *Hidden Markov Processes* - IEEE Trans. Inf. Theory **48**, 1518-1569 (2002)
- [27] Ephraim, Y., Dembo, A., Rabiner, L.R. - *A Minimum Discrimination Information Approach for Hidden Markov Modeling* - IEEE Trans. Inf. Theory **35**, 1001-1013 (1989)
- [28] Falkhausen, M., Reininger, H., Wolf, D. - *Calculation of Distance Measures Between Hidden Markov Models* - em *Proceedings of Eurospeech '95*, 1487-1490 (1995)
- [29] Franz, S., Leone, M., Ricci-Tersenghi, F., Zecchina, R. - *Exact solutions for diluted spin glasses and optimization problems* - Phys. Rev. Lett. **87**, 127209-127212 (2001)
- [30] Frasconi, P., Soda, G., Vullo, A. - *Hidden Markov Models for Text Categorization in Multi-Page Documents* - J. Intell. Inf. Systems **18**, 195-217 (2002)
- [31] Gales, M.J.F. - *Connectionist Probability Estimators in HMM Speech Recognition* - Tech. Report CUED/FINFENG/TR291, Cambridge Univ., 1-20 (1997)
- [32] Garey, M., Johnson, D.S. - *Computers and Intractability: A Guide to the Theory of NP-completeness* - W.H. Freeman and Co., San Francisco (1979)
- [33] Ghahramani, Z., Hinton, G.E. - *Variational Learning for Switching State-Space Models* - Neural Computation **12**, 831-864 (2000)
- [34] Ghahramani, Z., Jordan, M.I. - *Factorial Hidden Markov Models* - Machine Learning **29**, 245-273 (1997)
- [35] Hartmanis, J., Hopcroft, J.E. - *An Overview of the Theory of Computational Complexity* - J. of the Assoc. for Comp. Machin. **18**, 444-475 (1971)
- [36] Hastie, T., Tibshirani, R., Friedman, J.H. - *The Elements of Statistical Learning* - Springer (2001)

- [37] Heaton, R.K., Chelune, G.J., Talley, J.L., Kay, G.G., Curtis, G. - *Wisconsin Card Sorting Test Manual: Revised and Expanded* - Psychological Assessment Resources Inc., Odessa (1993)
- [38] Heckerman, D. - *A Tutorial on Learning Bayesian Networks* - Microsoft Research, Technical Report MSR-TR-95-06, 1-40 (1995)
- [39] Henderson, J., Salzberg, S., Fasman, K.H. - *Finding genes in DNA with a Hidden Markov Model* - J. Comp. Biol. **4**, 127-141 (1997)
- [40] Hertz, J., Krogh, A., Palmer, R.G. - *Introduction to the Theory of Neural Computation* - Addison-Wesley Publishing Company (1992)
- [41] Heskes, T., Wiegerinck, W. - *On-line Learning with Time-Correlated Examples* - em *On-line Learning in Neural Networks*, 251-278, edited by David Saad, Cambridge University Press, Cambridge (1998)
- [42] Iusem, A. - *$P = NP$ ou as Sutilezas da Complexidade Computacional* - Matemática Universitária **5**, 33-60 (1987)
- [43] Jaynes, E.T. - *Probability Theory: The Logic of Science* - Cambridge University Press (2003)
- [44] Jordan, M.I., Sejnowski, T.J. - *Graphical Models: Foundations of Neural Computation* - The MIT Press (2001)
- [45] Jordan, M.I. - *Learning in Graphical Models* - MIT Press (1998)
- [46] Juang, B.H., Rabiner, L.R. - *A Probabilistic Distance measure between HMMs* - AT&T Technical Journal **64**, 391-408 (1985)
- [47] Juang, B.H., Rabiner, L.R. - *The segmental k-means algorithm for estimating the parameters of hidden Markov models* - IEEE Trans. Acoust., Speech. Signal Processing **ASSP-38**, 1639-1641 (1990)
- [48] Kalman, R.E. - *A New Approach to Linear Filtering and Prediction Problems* - Trans. of the ASME, J. Basic Engin. **82**, 35-45 (1960)
- [49] Kehagias, A. - *Bayesian Classification of Hidden Markov Models* - Math. and Comp. Modelling **23**, 25-43 (1996)

- [50] Kinouchi, O., Caticha, N. - *Optimal generalization in perceptrons* - J. Phys. A: Math. Gen. **25**, 6243-6250 (1992)
- [51] Kirkpatrick, S., Selman, B. - *Critical Behavior in the Satisfiability of Random Boolean Expression* - Science **264**, 1297-1301 (1994)
- [52] Krogh, A., Brown, M., Mian, I.S., Sjolander, K., Haussler, D. - *Hidden Markov models in computational biology: Applications to protein modeling* - J. Mol. Biol. **235**, 1501-1531 (1994)
- [53] Krogh, A., Mian, I.S., Haussler, D. - *A Hidden Markov Model that finds Genes in Escherichia coli DNA* - Nucleic Acids Research **22**, 4768-4778 (1994)
- [54] Krogh, A. - *Using database matches with HMMGene for automated gene detection in Drosophila* - Genome Research **10**, 523-528 (2000)
- [55] Kullback, S., Leibler, R.A. - *On information and sufficiency* - Ann. Math. Statist. **22**, 79-86 (1951)
- [56] Kullback, S. - *Information Theory and Statistics* - Dover, New York (1997)
- [57] Lauritzen, S.L. - *Graphical Models* - Oxford University Press (1996)
- [58] Leek, T.R. - *Information Extraction Using Hidden Markov Models* - Tese de Mestrado, Universidade da Califórnia, San Diego (1997)
- [59] MacKay, D.J.C. - *Ensemble Learning for Hidden Markov Models* - Preprint, 1-7 (1997)
- [60] Mackay, D.J.C. - *Choice of Basis for Laplace Approximation* - Machine Learning **33**, 77-86 (1998)
- [61] MacKay, D.J. - *Information Theory, Inference and Learning Algorithms* - Cambridge University Press (2002)
- [62] McCulloch, W.S., Pitts, W. - *A Logical Calculus of Ideas Immanent in Nervous Activity* - Bulletin of Mathematical Biophysics **5**, 115-133 (1943)
- [63] Mertens, S. - *Phase Transitions in the Number Partitioning Problem* - Phys. Rev. Lett. **81**, 4281-4284 (1998)
- [64] Mertens, S. - *Computational Complexity for Physicists* - Computing in Science and Engineering, May/June, 31-47 (2002)

- [65] Mézard, M., Parisi, G., Zecchina, R. - *Analytic and Algorithmic Solution of Random Satisfiability Problems* - Science **297**, 812-815 (2002)
- [66] Monasson, R., Zecchina, R. - *Entropy of the K-Satisfiability Problem* - Phys. Rev. Lett. **76**, 3881-3885 (1996)
- [67] Monasson, R., Zecchina, R. - *Statistical mechanics of the random K-satisfiability model* - Phys. Rev. E **56**, 1357-1370 (1997)
- [68] Moon, T.K. - *The Expectation-Maximization Algorithm* - IEEE Signal Process. Mag., 47-60 (1996)
- [69] Müller, K.-R., Mika, S., Rätsch, G., Tsuda, K., Schölkopf, B. - *An Introduction to Kernel-Based Learning Algorithms* - IEEE Trans. on Neural Networks **12**, 181-201 (2001)
- [70] Murphy, K.P. - *An Introduction to Graphical Models* - Preprint, 1-19 (2001)
- [71] Neely, G., Hall, E. - *Mastering Elliot Wave: Presenting the Neely Method: The First Scientific, Objective Approach to Market Forecasting with the Elliott Wave Theory (version 2)* - Windsor Books (1990)
- [72] Neirotti, J.P., Caticha, N. - *Dynamics of the evolution of learning algorithms by selection* - Phys. Rev. E **67**, 041912 (2003)
- [73] Nielsen, M.A., Chuang, I.L. - *Quantum Computation and Information*
- [74] Oppor, M. - *A Bayesian Approach to On-line Learning* - em *On-line learning in Neural Networks*, 363-378, edited by David Saad, Cambridge University Press, Cambridge (1998)
- [75] Österreicher, F. - *Csiszár's f-divergences - Basic Properties* - Preprint, 1-13 (2002) - <http://matilda.vu.edu.au/~rgmia/databasepapers/Vortrag1.pdf>
- [76] Pardo, B., Shifrin, J., Birmingham, W. - *Name that tune: a pilot study in finding a melody from a sung query* - J. Am. Soc. for Information, Scien. and Technol. **55**, 283-300 (2004)
- [77] Pearl, J. - *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference* - Morgan Kaufman (1998)

- [78] Pfam: **Protein families database of alignments and HMMs** - <http://www.sanger.ac.uk/Software/Pfam/>
- [79] Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P. - *Numerical Recipes in C* - Segunda Edição, Cambridge (1997)
- [80] PROSITE: Database of protein families and domains - <http://ca.expasy.org/prosite/>
- [81] Rabiner, L.R., Juang, B.H. - *An Introduction to Hidden Markov Models* - IEEE ASSP Magazine, Jan., 4-16 (1986)
- [82] Rabiner, L.R. - *A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition* - Proc. IEEE **77**, 257-286 (1989)
- [83] Rattray, M., Saad, D. - *Analysis of on-line training with optimal learning rates* - Phys. Rev. E **58**, 6379-6391 (1998)
- [84] Renals, S., Morgan, N., Bourlard, H., Cohen, M., Franco, H.- *Connectionist Probability Estimators in HMM Speech Recognition* - IEEE Trans. on Speech and Audio Processing **2**, 161-174 (1994)
- [85] Reza, F.M. - *An Introduction to Information Theory* - Dover (1994)
- [86] Riegler, P., Caticha, N. - *MaxEnt Queries and Sequential Sampling* - em *Bayesian Inference and Maximum Entropy Methods in Science and Engineering*, editado por A.Mohammad-Djafari, AIP Press (2001)
- [87] Rosenblatt, F. - *The Perceptron: A probabilistic model for information storage and organization in the brain* - Psychological Review **65**, 386-408 (1958).
- [88] Saad, D., Opper, M. - *Advanced Mean Field Methods, Theory and Practice* - MIT Press (2001)
- [89] Saad, D. - *On-line Learning in Neural Networks* - Cambridge University Press (1999)
- [90] Schalkoff, R. - *Pattern Recognition: Statistical, Structural and Neural Approaches* - John Wiley and Sons, Inc. (1992)
- [91] Shannon, C.E. - *A mathematical theory of communication* - Bell System Technical Journal **27**, 379-423 e 623-656 (1948)

- [92] Shatkay, H., Kaelbling, L.P. - *Learning Hidden Markov Models with Geometric Information* - Technical Report CS-97-04, Brown University, 1-16 (1997)
- [93] Shawe-Taylor, J., Cristianini, N. - *Kernel Methods for Pattern Analysis* - Cambridge University Press (2004)
- [94] Silva, J., Narayanan, S. - *A Statistical Discrimination Measure for Hidden Markov Models based on Divergence* - em Proc. of ICLSP 2004
- [95] Sivia, D.S. - *Data Analysis: A Bayesian Tutorial* - Claredon Press, Oxford (2000)
- [96] Smyth, P. - *Belief networks, hidden Markov models and Markov random fields: A unifying view* - Patt. Rec. Lett. **18**, 1261-1268 (1997)
- [97] Solla, S.A., Winther, O. - *Optimal Perceptron Learning: an Online Bayesian Approach* - em On-line learning in Neural Networks, 363-378, edited by David Saad, Cambridge University Press, Cambridge (1998)
- [98] Stiller, J.C., Radons, G. - *Online Estimation of Hidden Markov Models* - IEEE Sign. Process. Lett. **6**, 213-215 (1999)
- [99] Stolcke, A. - *Bayesian Learning of Probabilistic Language Models* - PhD. Thesis, University of California at Berkeley (1994)
- [100] Vapnik, V. - *The Nature of Statistical Learning Theory*
- [101] Vicente, R., Kinouchi, O., Caticha, N. - *Statistical Mechanics of Online Learning of Drifting Concepts: A Variational Approach* - Machine Learning **32**, 179-201 (1998)
- [102] Vicente, R., de Toledo, C.M., Leite, V.B.P., Caticha, N. - *Underlying Dynamics of Typical Fluctuations of an Emerging Market Price Index: The Heston Model from Minutes to Months* - Preprint, 1-20 (2005)
- [103] Vicente, R., Saad, D., Kabashima, Y. - *Low-Density Parity Check Error-Correcting Codes: A Statistical Physics Approach* - Advances in Imaging and Electron Physics **125**, 231-353 (2002)
- [104] Vicente, R., Caticha, N. - *Functional optimization of online algorithms in multi-layer neural networks* - J. Phys. A: Math. and Gen. **30**, L599-L605 (1997)

- [105] Vihola, M., Harju, M., Salmela, P., Suontausta, J., Savela, J. - *Two Dissimilarity Measures for HMMs and Their Application in Phoneme Model Clustering*- Proc. ICASSP 2002, Vol. I, pp. 933-936 (2002)
- [106] Vihola, M. - *Dissimilarity Measures for Hidden Markov Models and Their Application in Multilingual Speech Recognition* - Tese de Mestrado, Tampere University of Technology (2001)
- [107] Vlad, M.O., Tsuchiya, M., Oefner, P., Ross, J. - *Bayesian analysis of systems with random chemical composition: Renormalization-group approach to Dirichlet distributions and the statistical theory of dilution* - Phys. Rev. E **65**, 011112(1)-01112(8) (2001)
- [108] Walder, C.J., Kootsookos, P.J., Lovell, B.C. - *Towards a Maximum Entropy Method for Estimating HMM Parameters* - Preprint, 45-49
- [109] Weigt, M., Hartmann, A.K. - *Number of Guards Needed by a Museum: A Phase Transition in Vertex Covering of Random Graphs* - Phys. Rev. Lett. **84**, 6118-6121 (2000)
- [110] West, A.H.L., Saad, D. - *On-line learning with adaptive back propagation in two-layer networks* - Phys. Rev. E **56**, 3426-3445 (1997)
- [111] Williams, C.K.I. - *Prediction with Gaussian Processes: From Linear Regression to Linear Prediction and beyond* - NCRG, Technical Report NCGR/97/012, 1-19 (1997)

Índice Remissivo

algoritmo

- bayesiano *on-line*, 29

- HMM, 80

- definição , 6

- eficiência, 6

- posterior média, 85

- trivial, 6

annealing, 79

aprendizado em máquinas, 24

backward procedure, 38

Baldi-Chauvin

- algoritmo de, 67

Baum-Welch

- algoritmo *on-line*, 58

- fórmulas de reestimação de, 51

cadeia de Markov, 10

campos aleatórios de Markov, 22

cláusula, 8

classe de complexidade, 7

- NP, 8

- P, 7

complexidade computacional, 5

Cook

- teorema de, 8

curva de aprendizado, 28

descida pelo gradiente, 115

digamma

- sistema, 123

Dirichlet, distribuição de

- definição , 116

- normalização , 116

distância euclideana, 50

distribuição *a priori*, 17

distribuição paramétrica, 14

eficiência

- de um algoritmo, 6

EM

- algoritmo, 16

erro

- de generalização , 28

- de treinamento, 28

fórmula, 8

forward procedure, 36

generalização , 28

gradient descent, 115

grafo de transição , 10

HMM, 32

- aprendizado, 45

- topologia, 35

IBOVESPA, 100

inferência bayesiana, 17

inferência estatística, 14

instância

- de um problema, 5

- tamanho, 6

- Kullback-Leibler
 - divergência de, 47
- máxima entropia
 - método, 19
- máxima verossimilhança, 15
- máximo *a posteriori*, 18
- matriz de emissão, 34
- matriz de transição, 10
 - HMM, 34
- modelo de aprendizado, 25
- modelo de Ising, 23
- modelos gráficos, 20
 - direcionados, 21
 - não-direcionados, 22
- perceptron booleano, 25
- posterior média
 - estimativa, 18
- problemas difíceis, 8
- problemas fáceis, 7
- processo markoviano, 10
- redes de crenças, 21
- satisfatibilidade
 - problema da, 8
- softmax
 - representação, 67
- steepest descent, 115
- variáveis escondidas, 14
- variáveis observadas, 14
- verossimilhança, 15
 - logarítmica, 15
- Viterbi
 - algoritmo de, 39
- Wisconsin
- teste de, 98