



COPPE/UFRJ

APLICAÇÃO DE MODELOS OCULTOS DE MARKOV EM JOGOS DE ESTRATÉGIA
MISTA

Eduardo Ramos Waghabi

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

Orientador: Mário Roberto Folhadela Benevides

Rio de Janeiro
Setembro de 2009

APLICAÇÃO DE MODELOS OCULTOS DE MARKOV EM JOGOS DE ESTRATÉGIA
MISTA

Eduardo Ramos Waghbi

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

Aprovada por:

Prof. Mario Roberto Folhadela Benevides, Ph.D.

Prof. Edward Hermann Haeusler, D.Sc.

Prof. Daniel Ratton Figueiredo, Ph.D.

RIO DE JANEIRO, RJ - BRASIL
SETEMBRO DE 2009

Waghabi, Eduardo Ramos

Aplicação De Modelos Ocultos De Markov Em Jogos De Estratégia Mista/ Eduardo Ramos Waghabi. – Rio de Janeiro: UFRJ/COPPE, 2009.

X, 63 p.: il.; 29,7 cm.

Orientador: Mario Roberto Folhadela Benevides.

Dissertação (mestrado) – UFRJ/ COPPE/ Programa de Engenharia de Sistemas e Computação, 2009.

Referências Bibliográficas: p. 60-63.

1. Modelagem de Jogos de Estratégia Mista. 2. Teoria dos Jogos. 3. Modelos de Markov ocultos. 4. Algoritmos Genéticos. I. Benevides, Mario Roberto Folhadela. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

Agradecimentos

Não foram poucas as vezes em que acreditei não ser possível concluir este trabalho, em todas as suas fases. Enquanto não gostaria de faltar-me com grande parte do esforço, não poderia deixar de lembrar aqui aqueles que foram, mais direta que indiretamente, co-responsáveis por esta prova de crescimento pessoal e profissional. São eles:

Mário Benevides, meu orientador, que me resgatou de uma situação difícil e colocou-me no prumo, fornecendo a quantia exata de motivação quando se fez necessário. Calma, paciência, conhecimento, experiência, simplicidade e um admirável bom humor permanente fazem de Mário o orientador utópico que eu imaginava, quando iniciei esta caminhada.

Daniel Ratton, membro da banca, que foi muito importante para a forma da versão final deste trabalho.

Minha esposa Danielle e meus filhos, Pedro e Helena, por terem suportado meus destemperos, impaciências, e menos do que eu gostaria de ser como marido e pai, durante todo o curso, na esperança de que colheriam os resultados no futuro que se aproxima.

Minha família e amigos, que me motivaram sempre que necessário. Agradeço especialmente àqueles que são também superiores hierárquicos e puderam ser suficientemente generosos para relevar a parte de mim que se dedicava a esta dissertação.

Finalmente, quero agradecer ao corpo docente e funcional da COPPE pelas tantas lições valiosas além de qualquer possibilidade de retribuição.

Obrigado,

Eduardo Ramos Waghbi

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

APLICAÇÃO DE MODELOS OCULTOS DE MARKOV EM JOGOS DE ESTRATÉGIA MISTA

Eduardo Ramos Waghabi

Setembro/2009

Orientador: Mario Benevides

Programa: Engenharia de Sistemas e Computação

Nesta dissertação discutimos possibilidades de aplicação das ferramentas dos Modelos Ocultos de Markov em Jogos de Estratégia Mista, inspirados pelas características em comum de ambos os campos. Para tal, definimos dois tipos de jogos de estratégia mista repetidos entre dois jogadores, com características específicas.

No primeiro jogo, o primeiro jogador alterna entre estratégias mistas à sua discrição. Na segunda espécie de jogo, as alternativas do primeiro jogador não são apenas estratégias mistas, mas jogos estratégicos distintos, com matrizes de recompensas igualmente diferentes. Neste novo jogo, as alternativas do primeiro jogador oferecem equilíbrios diversos, os quais não são do conhecimento do segundo jogador.

A partir destes jogos, queremos responder as seguintes questões:

1. É possível, a partir de informações dos jogos anteriormente disputados, criar um Modelo Oculto de Markov que represente tais jogos?
2. De que forma tal modelo e as ferramentas relacionadas, podem ajudar um jogador a escolher seu próximo lance?
3. Qual é o desempenho deste método em comparação com uma escolha aleatória deste mesmo jogador?

Para tal, apresentamos um método para a modelagem proposta e aplicação de um Modelo Oculto de Markov para auxiliar um dos jogadores. Este método é aplicado em um grande número de cenários, com a preocupação adicional de responder algumas questões relacionadas, como se a semelhança entre o Modelo Oculto de Markov e o jogo real influencia os resultados, e se o aumento de conhecimento do segundo jogador com relação aos detalhes do jogo pode beneficiá-lo, dentro do modelo proposto.

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

HIDDEN MARKOV MODELS IN MIXED STRATEGY GAMES

Eduardo Ramos Waghbi

September/2009

Advisor: Mario Benevides

Department: Systems Engineering and Computer Science

In the present dissertation we discuss possibilities of applications of Hidden Markov Models in Mixed Strategy Games, inspired by the stochastic nature of both fields. For such, we define two types of repeated mixed strategy games between two players, with specific characteristics.

In the first game, a player changes between mixed strategies at will. In the second game, the alternatives of the first player are not only mixed strategies but, by changing the payoff functions, distinct strategic games which feature distinct equilibria that are not known by the other player.

From these games, we want to answer the following questions:

1. It is possible, using information from previously disputed games, to create Hidden Markov Models that represents such games?
2. In which ways such models and related tools can help a player to choose its next move?
3. What is the performance of such method in comparison with a random move from this same player?

For such, we present a method to assist one of the players. This method is applied in a wide number of setups, with an additional concern to answer some related questions, as if the proximity between the Hidden Markov Model and the original game can influence the results, and if the player we are trying to help can benefit from the increase of knowledge about details of the game.

Sumário

1	Introdução.....	1
2	Teoria dos Jogos	3
2.1	Conceitos básicos	4
2.2	Equilíbrio de Nash	7
2.3	Estratégias Mistas	9
2.3.1	Cálculo do Equilíbrio de Nash.....	10
2.3.2	Indiferença e Punição	13
3	Modelos Ocultos de Markov.....	14
3.1	Conceitos básicos	15
3.2	Elementos de um Modelo Oculto de Markov	18
3.2.1	Cálculo de Probabilidades com o algoritmo <i>forward</i>	19
3.2.2	Codificação com o algoritmo de <i>Viterbi</i>	20
3.2.3	Treinamento por Maximização de Expectativa.....	21
3.2.4	Treinamento com Algoritmos Genéticos	22
3.3	Semelhança entre Modelos Ocultos de Markov.....	23
4	Aplicações de Modelos Ocultos de Markov em Jogos de Estratégia Mista	24
4.1	Modelagem dos Jogos	26
4.2	Obtenção do Próximo Lance	30
5	Descrição do Método e Apresentação dos Resultados Obtidos.....	34
5.1	Cenários Apresentados	36

5.1.1 Cenário 1	36
5.1.2 Cenário 2	39
5.1.3 Cenário 3	41
5.1.4 Cenário 4	42
6 Conclusão e Trabalhos Futuros	45
Apêndice A – Resultados dos Cenários	47
A.1 Cenários de 1 a 6	47
A.1 Cenários de 7 a 12	47
A.3 Cenários de 13 a 21	47
A.3 Cenários de 22 a 27	48
Apêndice B – Exemplo de Código-fonte	49
B.1 Estratégia de Tênis	49
B.1.1 Classe estrategiaTennis.java	49
B.1.2 Exemplo de saída da execução	56
Referências Bibliográficas	60

Lista de Figuras

Figura 2.1 – Exemplo de Forma Extensiva de Jogo.....	4
Figura 3.1 – Cadeia de Markov para previsão do tempo.....	16
Figura 3.2 – Modelo Oculto de Markov para previsão do tempo.....	17
Figura 4.1 – Diagrama do jogo de tênis.....	25
Figura 4.2 – Diagrama do jogo de tênis com elementos de A e B representados.....	27
Figura 4.3 – Modelo genérico para um jogo de estratégias mistas.....	28
Figura 4.4 – Modelo genérico para um jogo de estratégias mistas com e variações estratégicas.....	29
Figura 5.1.1.1 – Modelo do Cenário 1.....	37
Figura 5.1.1.2 – Distância <i>versus</i> Recompensa para seqüências de 10 observações	37
Figura 5.1.1.3 – Distância <i>versus</i> Recompensa para seqüências de 20 observações	38
Figura 5.1.1.4 – Distância <i>versus</i> Recompensa para seqüências de 50 observações	38
Figura 5.1.2.1 – Modelo do Cenário 2.....	40
Figura 5.1.4.1 – Modelo do Cenário 4.....	43
Figura 5.1.4.2 – Tendência da recompensa para a melhora da premissa.....	44

Lista de Tabelas

Tabela 2.1 – Exemplo de uma Matriz de Recompensas	5
Tabela 2.2 – Matriz de recompensas do jogo 2.1.....	6
Tabela 2.3 – Matriz de recompensas do jogo Guerra dos Sexos	8
Tabela 2.4 – Matriz de recompensas do jogo Par ou Ímpar	9
Tabela 2.5 - Matriz de recompensas do jogo de tênis	11
Tabela 3.1 – Matriz A do modelo de previsão do tempo	18
Tabela 3.2 – Matriz B do modelo de previsão do tempo	19
Tabela 4.1 – Matriz de recompensas do jogo de Tênis	25
Tabela 4.1 – Matriz A do modelo de previsão do tempo	31
Tabela 5.1.1.1 – Resultados do Cenário 1	37
Tabela 5.1.1.2 – Resultados do Cenário 1 com aumento de informação	39
Tabela 5.1.2.1 – Matriz de recompensas do jogo com três observações	39
Tabela 5.1.2.2 – Resultados do Cenário 2	40
Tabela 5.1.3.1 – Matriz de Recompensas para o perfil Agressivo do Cenário 3.....	41
Tabela 5.1.3.2 – Matriz de Recompensas para o perfil Moderado do Cenário 3	41
Tabela 5.1.3.3 – Matriz de Recompensas para o perfil Defensivo do Cenário 3	41
Tabela 5.1.3.4 – Equilíbrios de Nash para as variações estratégicas do Cenário 3	42
Tabela 5.1.3.5 – Resultados do Cenário 3	42
Tabela 5.1.4.1 – Resultados do Cenário 4	43
Tabela 5.1.4.2 – Resultados do Cenário 4 com variações no treinamento.....	44

Capítulo 1

Introdução

Jogos de estratégia mista vêm sendo utilizados para a modelagem e solução de problemas há tempos, em diversas áreas, muitas das quais apresentam igualmente diversas aplicações para Modelos Ocultos de Markov.

Embora a característica de incerteza apresente-se nas duas áreas de conhecimento, talvez pela pouca idade do desenvolvimento teórico dos Modelos Ocultos de Markov, pode-se perceber a ausência de trabalhos acadêmicos relacionando estes dois conceitos. Embora alguns textos mencionem a idéia (veja MIHOK e TERRY (2005) e FINESSO (2000) para exemplos) não chegam, entretanto, a efetivamente evidenciar as relações.

O presente trabalho objetiva apresentar uma possibilidade de ligação entre estes dois campos de pesquisa.

Queremos, nesta dissertação, verificar se, a partir de um jogo de estratégia mista, repetido, entre dois jogadores, onde o primeiro jogador alterna entre estratégias mistas à sua discrição, é possível treinar um Modelo Oculto de Markov para auxiliar o segundo jogador a escolher seu próximo lance, de forma a maximizar sua recompensa.

Para exemplificar tal jogo, imagine o momento do saque de uma partida de tênis onde, para decidir entre sacar aberto ou no centro da quadra, o jogador sacador alterne entre três perfis distintos: agressivo, moderado e defensivo. O jogador recebedor, por sua vez, conhece os perfis, porém não a freqüência com que o sacador utiliza cada uma delas, embora saiba que esta freqüência é fixa e não se alterará durante as repetições do jogo. Ao jogador recebedor interessa, naturalmente, antecipar corretamente o lance escolhido por seu adversário.

A seguir, introduzimos uma modificação no jogo de tal maneira que as alternativas do primeiro jogador não sejam apenas estratégias mistas, mas, através de matrizes de recompensas diferentes, jogos estratégicos distintos (embora com mesmos jogadores e

estratégias puras). Neste novo jogo, as alternativas do primeiro jogador oferecem equilíbrios diversos, os quais não são do conhecimento do segundo jogador.

Mantendo o exemplo da partida de tênis, o jogo estratégico modificado seria tal que as recompensas seriam medidas como a chance do recebedor retornar a bola adequadamente. Os perfis agressivo, moderado e defensivo do sacador produziriam, através de estilos distintos do saque, chances diferentes do recebedor retornar a bola.

Nestes cenários, queremos responder às seguintes questões:

1. É possível, a partir de informações dos pontos anteriormente disputados, criar um Modelo Oculto de Markov que represente tais jogos?

2. De que forma tal modelo e as ferramentas relacionadas, podem ajudar o jogador recebedor a escolher seu próximo lance?

3. Qual é o desempenho deste método em comparação com uma escolha aleatória do jogador recebedor?

Após a apresentação teórica dos dois assuntos, apresentamos o método de modelagem proposto, a forma de aplicar o Modelo Oculto de Markov para auxiliar um dos jogadores, e em seguida os resultados obtidos com alguns cenários. Incidentemente, nestes cenários, nos preocupamos igualmente com as seguintes questões relacionadas:

a) De que forma a semelhança entre o Modelo Oculto de Markov e o jogo real altera os resultados?

b) Como o aumento de conhecimento do segundo jogador com relação aos detalhes do jogo pode beneficiá-lo, dentro do modelo proposto?

No último capítulo, apresentamos nossas conclusões sobre os resultados e apontamos melhorias a serem realizadas na direção desta pesquisa.

Capítulo 2

Teoria dos Jogos

Embora os conceitos informais das estratégias envolvendo disputas de várias naturezas, por estas serem parte fundamental da condição humana, já estivessem presentes há muito, as sementes da Teoria dos Jogos foram apresentadas em 1928 com a publicação do artigo “Zur Theorie der Gesellschaftsspiele” pelo húngaro de apenas 23 anos John Von Neumann (VON NEUMANN e MORGENSTERN, 1944).

Alguns trabalhos anteriores se destacam estabelecendo alguns pilares, como a solução Minimax para de um jogo de cartas fornecida em 1713 por James Waldegrave, ou o trabalho de Antoine Augustin Cournot “Recherches sur les principes mathématiques de la théorie des richesses” que em 1838 apresentou uma versão restrita do que mais tarde seria conhecido como Equilíbrio de Nash (FUDENBERG e TIROLE, 1994).

Mas o nascimento formal da Teoria dos Jogos deu-se, tal como hoje amplamente reconhecido, da contribuição de Von Neumann com Oskar Morgenstern, no início dos anos 40, através do trabalho referência “Theory of Games and Economic Behavior” (1944) (FUDENBERG e TIROLE, 1994).

Em 1950 a tese de Doutorado de John Forbes Nash Jr. “Non-Cooperative Games”, em conjunto com os três artigos decorrentes dela, provocou grande impacto ao apresentar o hoje ubíquo conceito do Equilíbrio de Nash, uma generalização na obtenção da solução de jogos entre dois oponentes apresentada por Neumann e Morgenstern (NASH, 1950). O Equilíbrio de Nash ampliou as aplicações da teoria de jogos, alcançando campos tão diversos como Economia, Biologia, Filosofia e Inteligência Artificial.

Os conceitos e a nomenclatura da Teoria dos Jogos utilizados para os fins deste trabalho serão apresentados a seguir, de forma resumida.

2.1 Conceitos básicos

A Teoria dos Jogos é uma aplicação matemática para o estudo do comportamento de agentes – ou jogadores – quando em conflito de interesses, parcial ou total. Embora um campo específico da Teoria dos Jogos estude situações onde agentes possam cooperar (veja FUDENBERG e TIROLE (1994)), tal campo não é do interesse deste trabalho e não será abordado.

Um jogo pode ser definido informalmente como uma interação estratégica entre dois ou mais jogadores onde estes, realizando ações (jogadas) alternadamente ou simultaneamente, obtêm como resultado uma recompensa mensurável.

Dos jogadores é esperada uma *função de utilidade* que, quando aplicada aos possíveis resultados da interação, permita ao jogador ordenar perfeitamente suas preferências quanto aos resultados e, conseqüentemente, quanto a suas ações (VON NEUMANN e MORGENSTERN, 1944).

É igualmente esperado dos jogadores um comportamento *racional*, o que significa esperar dos jogadores que suas jogadas sejam invariavelmente no sentido de maximizar suas recompensas (FUDENBERG e TIROLE, 1994).

Finalmente, os jogadores devem conhecer a *forma* do jogo, que especifica quais são as possibilidades de ação em cada momento do jogo, e se suas ações dar-se-ão concomitantemente ou em turnos, em relação às ações de seus oponentes.

Quando os jogadores alternam suas jogadas, o jogo é mais bem representado na *forma extensiva*, através da árvore do jogo, um grafo em forma de uma árvore cujos nós representam as possíveis jogadas dos oponentes, e nas folhas residem as recompensas (GIBBONS, 1992). Podemos ver um exemplo de forma extensiva na figura 2.1, abaixo:

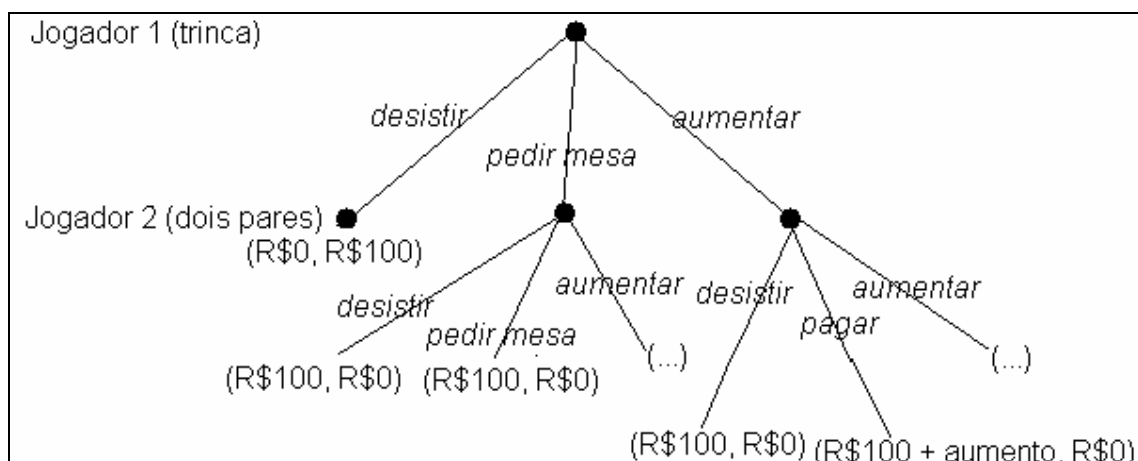


Figura 2.1 – Exemplo de Forma Extensiva de Jogo

Em outra possibilidade, os jogadores tomam suas decisões ao mesmo tempo e sem o conhecimento de outras decisões que não a sua. O jogo é dito então na *forma*

normal ou estratégica e pode ser convenientemente representado através de uma matriz de recompensas (GIBBONS, 1992). Um exemplo desta representação pode ser visto abaixo, na tabela 2.1, onde dois jogadores disputam um clássico jogo conhecido como O *Dilema do Prisioneiro*, e que será esclarecido adiante.

Prisioneiro 2	Prisioneiro 1	
	Calar	Delatar
Calar	-2 anos, -2 anos	-5 anos, -1 ano
Delatar	-1 ano, -5 anos	-4 anos, -4 anos

Tabela 2.1 – Exemplo de uma Matriz de Recompensas

Embora aparentemente diferentes, as duas formas de representação são equivalentes e intercambiáveis, sendo utilizadas segundo sua conveniência (FUDENBERG e TIROLE, 1994).

Os jogadores, de posse da forma do jogo, escolhem suas ações que por fim determinam a solução ou valor do jogo. O conjunto de ações tomadas por um jogador é dito *estratégia* do jogador, e o conjunto de estratégias de todos os jogadores envolvidos é dito *perfil estratégico* do jogo.

Trataremos aqui, por facilidade de representação, apenas de jogos na forma normal ou estratégica. Podemos então definir formalmente um jogo estratégico como (FUDENBERG e TIROLE, 1994):

Definição 2.1 *Um jogo estratégico γ é composto por:*

- *Um conjunto J finito de jogadores*
- *Um conjunto S_i de estratégias puras para cada jogador i , contendo todas as ações que possam ser realizadas pelo jogador; e*
- *Um conjunto u_i de funções utilidade $u_i(s)$ que, para cada jogador i , fornecem a recompensa recebida a partir de um possível perfil estratégico do jogo.*

e pode ser notado como $\gamma = \{J, S_i, u_i\}$

Incidentemente, o conceito de *perfil estratégico* aparece, podendo ser definido como um conjunto que reúne uma possível escolha de cada jogador, moldando uma possibilidade de resultado para o jogo. As escolhas dos jogadores podem ser *puras*, ou seja, limitando-se estritamente a uma estratégia pura, ou *mistas*, quando uma ou mais estratégias puras podem ser escolhidas com uma chance associada (FUDENBERG e TIROLE, 1994). Primeiramente nos ateremos aos perfis estratégicos de estratégias puras:

Definição 2.2 Um perfil estratégico s de estratégias puras para um jogo $\gamma = \{J, S_i, u_i\}$ pode ser definido por:

- $s = (s_1, \dots, s_i)$ onde $s_n \in S_n$ para todo $n \in [1, i]$

Para consolidar a definição e introduzir novos conceitos, podemos considerar o clássico exemplo, presente em virtualmente todos os textos sobre teoria dos jogos, do jogo conhecido como *O Dilema do Prisioneiro*:

Exemplo 2.1 Imagine como exemplo um jogo onde dois prisioneiros, acusados de terem cometido um crime juntos, são interrogados em celas separadas. A polícia tem provas suficientes para condenar os prisioneiros por 2 anos de reclusão cada, mas caso um prisioneiro acuse o outro, a pena do prisioneiro acusado aumentará para 5 anos. Para incentivar a delação, a polícia oferece ao delator a redução de 1 ano da pena.

Temos então a seguinte situação: caso ambos se caleem, ficarão presos por 2 anos cada. Se um delatar o outro, mas o outro calar, o delator sairá da cadeia em 1 ano, e o outro prisioneiro será condenado a 5 anos. Caso ambos delatem uns aos outros, receberão a pena de 5 anos reduzida para 4 anos por sua colaboração. As recompensas podem ser vistas na tabela 2.2 abaixo, sendo o primeiro número de cada célula a recompensa do jogador representado na linha da célula:

Prisioneiro 2	Prisioneiro 1	
	Calar	Delatar
Calar	-2 anos, -2 anos	-5 anos, -1 ano
Delatar	-1 ano, -5 anos	-4 anos, -4 anos

Tabela 2.2 – Matriz de recompensas do jogo 2.1

Note que as recompensas foram representadas como números negativos para manter a definição de que os jogadores desejam maximizar suas recompensas.

Neste caso, $J = \{\text{Prisioneiro 1, Prisioneiro 2}\}$; $S_1 = S_2 = \{\text{Calar, Delatar}\}$. O valor das funções utilidade são $u_1(\text{Calar, Calar}) = u_2(\text{Calar, Calar}) = -2$ anos, $u_1(\text{Calar, Delatar}) = u_2(\text{Delatar, Calar}) = -5$ anos, $u_1(\text{Delatar, Calar}) = u_2(\text{Calar, Delatar}) = -1$ ano, e finalmente, $u_1(\text{Delatar, Delatar}) = u_2(\text{Delatar, Delatar}) = -4$ anos..

O dilema de cada prisioneiro é de, devido à impossibilidade de se comunicar com o outro prisioneiro, não ter certeza se deve calar para, contando com o silêncio do outro, garantir uma pena pequena. Tendo isto em vista, a ação racional a tomar, por mais paradoxal que pareça, é delatar o companheiro. Desta forma ele garante uma pena de 4 anos e, caso o outro cale, sai em apenas 1 ano.

A pena de 1 ano é o melhor caso, mas não é atrás dela que o prisioneiro vai quando opta por delatar, e sim da redução de 5 anos para 4 anos, porque ele sabe que o outro jogador é racional e igualmente optará por delatar¹. O perfil estratégico que é a solução do jogo, portanto, consiste de (Delatar, Delatar).

Este exemplo demonstra um conceito conhecido como *dominância*: a opção de Delatar *domina* a opção de Calar para ambos os prisioneiros, pois provê melhores recompensas em relação à outra opção (Calar) não importando as estratégias dos outro prisioneiro. Podemos definir formalmente *dominância* da seguinte forma (GIBBONS, 1992):

Definição 2.3 Dado o jogo $\gamma = \{J, S_i, u_i\}$, sejam s_n' e s_n'' estratégias do jogador n (isto é, s_n' e $s_n'' \in S_n$). A estratégia s_n' é estritamente dominada por s_n'' se, para qualquer combinação possível de estratégias dos demais jogadores, $u_n(s_n', \dots)$ é estritamente menor que $u_n(s_n'', \dots)$.

Como se espera que jogadores racionais não escolham estratégias dominadas, a eliminação iterativa de estratégias dominadas pode (como o fez no Dilema dos Prisioneiros) indicar a solução do jogo, porém por ser um mecanismo dependente da existência de estratégias dominadas, é aplicável somente em um número restrito de casos. Para todos os fins, uma forma mais ampla de determinar a solução do jogo é necessária.

2.2 Equilíbrio de Nash

Em NASH (1950) foi introduzido o importante conceito, então apresentado como *ponto de equilíbrio* de um jogo, mais tarde conhecido como *Equilíbrio de Nash* de um jogo. Na tese, John Nash prova que em um jogo finito não-cooperativo (no sentido em que os jogadores apenas preocupam-se em maximizar suas recompensas) sempre possui pelo menos um ponto de equilíbrio².

¹ A verossimilhança da racionalidade tal qual pregada pela Teoria dos Jogos clássica já foi suficientemente debatida e atacada (veja, por exemplo, GINTIS (2000)). Experimentos (veja RUBINSTEIN ET AL. (1996)) foram realizados com humanos para avaliar a “racionalidade” da escolha em questões semelhantes ao Dilema do Prisioneiro e em diversos casos relatados, as pessoas não acompanham o lance racional previsto pela Teoria dos Jogos. No caso acima, podemos imaginar que, quando a redução de pena por delatar é diminuída, as pessoas possam preferir calar pela chance de que o outro também cale e os dois saiam com penas pequenas.

² Na realidade, o conceito de Nash vale apenas para estratégias mistas, que serão vistas mais adiante.

Um *Equilíbrio de Nash* pode ser definido informalmente como um ponto na matriz de recompensas (ou seja, um conjunto de estratégias, ou *perfil estratégico*) onde nenhum jogador, dado que as opções dos demais jogadores permaneçam constantes, possui vantagens em mudar a sua estratégia. Mais formalmente (FUDENBERG e TIROLE, 1994):

Definição 2.4 Dado o jogo $\gamma = \{J, S_i, u_i\}$, um perfil estratégico $s = (s_1, s_2, \dots, s_i)$ é um equilíbrio de Nash se, para todo jogador n e sua estratégia s_n :

$$u_n(s) \geq u_n(s_1, \dots, s_n, \dots, s_i) \text{ para todo } s_n \in S_n$$

No Dilema dos Prisioneiros é fácil ver que o perfil (Calar, Calar) não é um Equilíbrio de Nash, pois os dois jogadores têm interesse em mudar suas estratégias para Delatar, devido ao incentivo de redução da pena. O único Equilíbrio de Nash do dilema do prisioneiro (consequência do jogo poder ser resolvido por eliminação iterativa de estratégias dominadas (FUDENBERG e TIROLE, 1994)) é o perfil (Delatar, Delatar).

Nada impede que um jogo possua mais de um *Equilíbrio de Nash*, e a *solução* do jogo é o conjunto de todos estes pontos de equilíbrio, ou “boas estratégias” (Nash, 1950). Para um jogo com mais de um *equilíbrio de Nash*, considere o também clássico exemplo conhecido como “A Guerra dos Sexos”:

Exemplo 2.2 Um casal tenta decidir o programa da noite. Ele gostaria de assistir a uma luta de boxe, e ela gostaria de ir à Ópera. Ambos não gostam do programa do outro, mas acima de tudo preferem sair juntos. As recompensas podem ser vistas na matriz abaixo, sendo o primeiro número de cada célula a recompensa do jogador representado na linha da célula:

Ele	Ela	
	Luta de Boxe	Ópera
Luta de Boxe	2, 1	0, 0
Ópera	0, 0	1, 2

Tabela 2.3 – Matriz de recompensas do jogo Guerra dos Sexos

No Exemplo 2.2, os perfis (Luta de Boxe, Ópera) e (Ópera, Luta de Boxe) são ambos *Equilíbrios de Nash* para o jogo, pois nos dois casos um jogador sozinho não consegue melhorar sua recompensa mudando de estratégia.

Agora consideremos o seguinte jogo, derivado do popular “Par ou Ímpar”:

Exemplo 2.3 Dois jogadores decidem jogar uma disputa de par ou ímpar, que implica nos dois mostrando um número de 0 a 5 dedos de uma mão simultaneamente. Um dos

jogadores será o jogador “Par”, ou seja, ele ganha quando a soma dos números das mãos for par. O outro jogador ganha caso contrário. Para simplificar o número de estratégias dos jogadores, considere que para fins do resultado do jogo basta considerarmos se cada jogador mostrou um número ímpar ou par. As recompensas podem ser vistas na matriz abaixo, sendo o primeiro número de cada célula a recompensa do jogador representado na linha da célula:

Jogador ‘Ímpar’	Jogador ‘Par’	
	Par (0, 2 ou 4 dedos)	Ímpar (1, 3 ou 5 dedos)
Par (0, 2 ou 4 dedos)	0, 1	1, 0
Ímpar (1, 3 ou 5 dedos)	1, 0	0, 1

Tabela 2.4 – Matriz de recompensas do jogo Par ou Ímpar

Em pouco tempo verificando a matriz de recompensas podemos afirmar que não há equilíbrios de Nash para este jogo, já que o outro jogador sempre pode melhorar sua recompensa mudando para a estratégia oposta. Isto não vai de encontro à prova de NASH (1950) de que um jogo sempre possui ao menos um ponto de equilíbrio, mas sim nos mostra que a utilização de *Estratégias Puras* é limitada. Para que o conceito de equilíbrio apresentado por Nash se verifique, é preciso considerar as *Estratégias Mistas*.

2.3 Estratégias Mistas

Uma *Estratégia Mista* é uma distribuição probabilística sobre as estratégias puras de cada jogador (FUDENBERG e TIROLE, 1994), indicando a chance de que o jogador escolha cada estratégia pura. No caso do Exemplo 2.1 (Dilema do Prisioneiro), se as estratégias puras do Prisioneiro 1 são $S_1 = \{\text{Calar, Delatar}\}$, uma possível estratégia mista para ele seria $\sigma_1 = \{0,30 ; 0,70\}$, ou seja, o Prisioneiro 1 cala com 30% de chance, e delata com 70% de chance. A recompensa para tal estratégia pode ser calculada como, caso o adversário cale, $0,30 \cdot -2 + 0,70 \cdot -1 = -1,3$ anos, e caso o adversário delate, $0,30 \cdot -5 + 0,70 \cdot -4 = -4,3$ anos. Formalmente:

Definição 2.5 Uma estratégia mista σ_n de um jogador $n = [1, i]$ em um jogo $\gamma = \{J, S_n, u_i\}$ pode ser definida como $\sigma_n = (w_1, \dots, w_{|S_n|})$ onde, para todo $m = [1, |S_n|]$, w_m representa a chance do jogador n escolher a estratégia pura $s_m \in S_n$, sendo que necessariamente:

1. $w_m = [0, 1]$
2. A igualdade $w_1 + \dots + w_m = 1$ se verifica.

Claramente, toda estratégia pura $s_n \in S_n$ pode ser representada como uma estratégia mista na qual a chance da estratégia s_n ser jogada é de 1, sendo as demais chances das outras estratégias necessariamente 0, segundo o critério 1 da Definição 5 acima.

Reconsiderando o exemplo 2.3 à luz das estratégias mistas, não é difícil verificar que a estratégia $\sigma = \{0,50 ; 0,50\}$ é um ponto de equilíbrio Nash para ambos os jogadores, onde a recompensa de cada um é $0,5 \cdot 1 + 0,5 \cdot 0 = 0,5$ caso o adversário mostre um número par ou ímpar de dedos, indiferentemente. Ou seja, a melhor estratégia é “misturar” suas opções, mostrando um número par de dedos com a mesma frequência com que mostra um número ímpar.

Claro que a determinação exata da estratégia de equilíbrio do exemplo 2.3 é trivial em decorrência da simplicidade de sua tabela de recompensas. O cálculo do equilíbrio de Nash pode ser mais complexo em outros casos, e veremos um procedimento para cálculo na seção seguinte.

2.3.1 Cálculo do Equilíbrio de Nash

Considere o início da disputa de um ponto de tênis no exemplo abaixo, adaptado de GINTIS (2000):

Exemplo 2.4 *Dois jogadores estão disputando uma partida de tênis. Cada vez que um ponto é disputado, um dos jogadores realiza o “saque”. Ao sacar, o jogador pode escolher entre direcionar a bola para um dos lados do jogador recebedor.*

Como a raquete de tênis é normalmente segurada com apenas uma das mãos, os jogadores preferem rebater a bola quando a mesma está do mesmo lado da mão que segura a raquete, para todos os fins denominado aqui de forehand. Quando o jogador se vê na situação de rebater uma bola do lado contrário à mão que segura a raquete, diz-se que rebateu com seu backhand³.

No saque, a bola é lançada com uma velocidade superior à que normalmente ocorre no resto do jogo, de forma que o recebedor muitas vezes se antecipa ao lance com o objetivo de aumentar sua chance de devolver corretamente a bola. Quando ele erra a sua antecipação, no entanto, estas mesmas chances se reduzem. Abaixo está a matriz de recompensas do recebedor, indicando as chances do mesmo devolver a bola adequadamente:

³ No Brasil muitas vezes se utiliza o termo “Revés” como tradução do termo inglês *backhand*, concomitante ao termo “Drive” para o termo *forehand*. Como “Drive” é um termo igualmente estrangeiro, preferimos o uso de *Backhand* e *Forehand* por sua internacionalidade.

Sacador	Recebedor	
	Esquerda	Direita
Esquerda	0,75	0,55
Direita	0,40	0,90

Tabela 2.5 - Matriz de recompensas do jogo de tênis

Trata-se de um jogo clássico de soma zero (em jogos de soma zero, a soma das recompensas dos jogadores é constante e, normalizando-as, podemos expressar a recompensa π_{sacador} do sacador como $1 - \pi_{\text{recebedor}}$) onde o não há equilíbrio através das estratégias puras $S = \{\text{esquerda, direita}\}$. O Equilíbrio de Nash é então obtido através de uma *estratégia mista*, definida pela combinação de percentuais de aplicação de cada estratégia pura, neste caso p_{esquerda} e p_{direita} onde ambos são não-negativos e $p_{\text{esquerda}} = 1 - p_{\text{direita}}$. Se $S = \{\text{esquerda, direita}\}$ são as estratégias puras de cada jogador, notamos:

$$\sigma = p_{\text{esquerda}} \cdot \text{esquerda} + p_{\text{direita}} \cdot \text{direita}$$

O equilíbrio pode ser encontrado em termos dos valores da tabela de recompensas acima, aplicando o teorema fundamental do equilíbrio de Nash (NASH, 1950): suponha que o sacador está utilizando a estratégia $\sigma_{\text{sacador}} = p_{\text{esquerda}} \cdot \text{esquerda} + (1 - p_{\text{esquerda}} \cdot \text{direita})$. Se σ_{sacador} é um equilíbrio então deve ser indiferente às estratégias do recebedor, ou seja, as recompensas do recebedor por jogar *esquerda* e *direita* devem ser iguais. Valendo-nos disto, temos:

$$p_{\text{esquerda}} \cdot 0,75 + (1 - p_{\text{esquerda}}) \cdot 0,40 = p_{\text{esquerda}} \cdot 0,55 + (1 - p_{\text{esquerda}}) \cdot 0,90$$

e

$$p_{\text{esquerda}} = (0,90 - 0,40) / (0,90 - 0,40 + 0,75 - 0,55)$$

$$p_{\text{esquerda}} = 0,714$$

E podemos notar $\sigma_{\text{sacador}} = \{0,714 ; 0,286\}$. Desta forma, o sacador deve sacar na esquerda do recebedor 71,4% das vezes, com uma recompensa esperada de:

$$\pi_{\text{sacador}} = 1 - \pi_{\text{recebedor}}$$

$$\pi_{\text{sacador}} = 1 - (0,714 \cdot 0,75 + (1 - 0,714) \cdot 0,40)$$

$$\pi_{\text{sacador}} = 1 - (0,5355 + 0,1144)$$

$$\pi_{\text{sacador}} = 0,3501$$

Note que isto não significa que o sacador está em desvantagem, mas que em 35% das vezes o receptor *sequer retorna a bola adequadamente*. Nas demais 65% o ponto será disputado normalmente.

Da mesma forma, podemos encontrar a estratégia de equilíbrio do receptor $\sigma_{\text{receptor}} = p_{\text{esquerda}} \cdot \text{esquerda} + (1 - p_{\text{esquerda}}) \cdot \text{direita}$: (note que todas as recompensas estão complementadas seguindo a regra do jogo de soma zero normalizado, ou seja, π_{sacador} expressa como $1 - \pi_{\text{receptor}}$).

$$p_{\text{esquerda}} \cdot 0,25 + (1 - p_{\text{esquerda}}) \cdot 0,45 = p_{\text{esquerda}} \cdot 0,60 + (1 - p_{\text{esquerda}}) \cdot 0,10$$

e

$$p_{\text{esquerda}} = (0,10 - 0,45) / (0,10 - 0,45 + 0,25 - 0,60)$$

$$p_{\text{esquerda}} = -0,35 / (-0,35 - 0,35) = 0,35/0,7 = 0,50$$

Notamos $\sigma_{\text{receptor}} = \{ 0,50; 0,50 \}$. Desta forma, o receptor deve antecipar-se indiferentemente para a esquerda ou para a direita (50% das vezes para cada lado), com uma recompensa esperada de:

$$\pi_{\text{receptor}} = 1 - \pi_{\text{sacador}}$$

$$\pi_{\text{receptor}} = 1 - (0,50 \cdot 0,25 + 0,50 \cdot 0,45)$$

$$\pi_{\text{receptor}} = 1 - (0,125 + 0,225)$$

$$\pi_{\text{receptor}} = 0,65$$

Como observou GINTIS (2000), uma melhora na devolução em um dos lados deslocaria o equilíbrio para o *outro* lado, implicando na menor utilização do lado melhorado. Isto pode ser compreendido pelo fato de que a melhora seria percebida pelo jogador sacador que, razoavelmente, realizaria proporcionalmente menos lançamentos na direção, adequando-se ao novo ponto de equilíbrio.

Vale lembrar que, conforme o já demonstrado no exemplo 2.2, um jogo pode apresentar mais de um equilíbrio de Nash, e a verificação da unicidade do Equilíbrio de Nash para um jogo se mostrou ser um problema NP-Completo (para o qual não há algoritmos polinomiais que o resolvam), conforme provado por GILBOA e ZEMEL (1989).

2.3.2 Indiferença e Punição

Duas observações importantes podem ser realizadas com respeito ao Equilíbrio de Nash.

A primeira é o fato de que se um dos jogadores está utilizando a estratégia do Equilíbrio de Nash, sua recompensa é indiferente às alterações da estratégia dos outros jogadores (FUDENBERG e TIROLE, 1994). No caso do exemplo 2.3, se um dos jogadores está variando adequadamente suas escolhas entre par e ímpar, ele irá ganhar o mesmo número de vezes não importa se o outro jogador decidiu escolher apenas par, apenas ímpar, ou variar as duas estratégias por um critério qualquer.

A escolha da estratégia do equilíbrio de Nash protege, por assim dizer, o jogador que a utiliza. Exatamente por isto, o jogador que não a utiliza está desprotegido contra ataques dos demais jogadores, o que nos leva à segunda observação.

Quando um jogador foge do equilíbrio de Nash, em caso de repetição do jogo, pode ser punido pelos demais jogadores através de estratégias que obtenham um melhor resultado do que seria esperado dada a matriz de recompensas do jogo. Mais uma vez utilizando o exemplo 2.3, se um jogador decide jogar sempre a mesma opção, o outro poderá vencer todas as partidas saindo de sua estratégia do Equilíbrio de Nash e escolhendo adequadamente uma estratégia pura (FUDENBERG e TIROLE, 1994).

É importante notar que o conceito de estratégias mistas é muito criticado pela sua dificuldade de interpretação exceto em jogos repetidos. Como pode um jogador praticar um equilíbrio onde ele deve jogar uma estratégia em, digamos, 40% das vezes, *em apenas uma partida*? Uma possibilidade seria utilizar-se de um artifício para obter o lance a ser jogado como, por exemplo, um dado. Outro bom compromisso entre teoria e prática é imaginar uma estratégia mista como estratégias puras sendo jogadas por vários agentes simultaneamente (GINTIS, 2000).

Capítulo 3

Modelos Ocultos de Markov

As sementes dos conceitos por trás dos Modelos Ocultos de Markov datam do início do século 20, no trabalho do russo Andrei Andreyevich Markov, que mais tarde batizaria esta, assim como grande parte das ferramentas criadas para a manipulação e modelagem de processos estocásticos. Fruto central de seu trabalho sobre *Cadeias de Markov*, a *Hipótese de Markov* assume que o estado atual de um modelo dependa apenas de um número finito de estados anteriores (RUSSEL e NORVIG, 2004). Esta aparentemente grande restrição criou um limite necessário para que a teoria de desenvolvesse a largos passos, produzindo a base para, entre outras aplicações, as Redes Bayesianas Dinâmicas (das quais os Modelos Ocultos de Markov podem ser considerados casos especiais, embora tenham sido desenvolvidos antes).

Em meados da década de 1960, alguns importantes artigos de Leonard Baum e outros autores formaram a base da teoria dos Modelos Ocultos de Markov e seus vários algoritmos relacionados (RUSSEL e NORVIG, 2004). Suas diversas aplicações, contudo, demoraram a aparecer, segundo RABINER (1989), em parte porque os artigos foram publicados em revistas e jornais de matemática, pouco lidos por engenheiros trabalhando no problema de reconhecimento de fala. Outro motivo para tal seria a pouco didática linguagem utilizada nos artigos, motivo este que incentivou o próprio Rabiner a publicar artigos e livros com o intuito de esclarecer os conceitos.

Com o desenvolvimento das Redes Bayesianas Dinâmicas na segunda metade da década de 1980, Modelos Ocultos de Markov ficaram caracterizados como um caso especial das mesmas (RUSSEL e NORVIG, 2004).

A mais bem sucedida aplicação dos Modelos Ocultos de Markov encontra-se no campo do processamento de linguagem natural, mais especificamente no reconhecimento de fala, embora também presente em tradução automática, compreensão textual e conversão de texto para discurso (JUFRAKY e MARTIN, 2006). Modelos Ocultos de Markov também são de grande valia em biologia computacional, especificamente para o

seqüenciamento de DNA. Mas não limitado a estes campos, aplicações dos Modelos Ocultos de Markov tem aparecido na literatura de diversas áreas como Economia, Medicina e Engenharia.

Neste capítulo apresentaremos os conceitos necessários ao escopo desta dissertação, bem como a notação a ser utilizada. É importante notar que estaremos aqui trabalhando apenas com Modelos Ocultos de Markov de tempo discreto (aqui, simplesmente Modelos Ocultos de Markov), que é suficiente para todos os fins deste trabalho e evita as dificuldades adicionais da manipulação do tempo contínuo.

3.1 Conceitos básicos

Um Modelo Oculto de Markov é um modelo probabilístico utilizado para aprendizagem de máquina, mas especificamente como um classificador ou, mais especificamente um *classificador de seqüência* (JUFRASKY e MARTIN, 2006).

A idéia principal por trás do Modelo Oculto de Markov é representar uma variável aleatória como uma *Cadeia de Markov* (ou *processo de Markov*), com a propriedade particular de que tal variável não pode ser medida diretamente, mas sim através de observações possíveis a partir de cada estado da variável.

Uma Cadeia de Markov é uma máquina de estados finitos cujo estado corrente é alterado regularmente em espaços discretos de tempo, sendo o futuro estado determinado probabilisticamente apenas a partir do estado presente da máquina, condição esta conhecida como propriedade de Markov (JUFRASKY e MARTIN, 2006):

Definição 3.1 *Dada uma Cadeia de Markov onde q^t representa o estado corrente da cadeia no instante discreto t , a Propriedade de Markov determina que:*

$$P(q^t | q^1, q^2, \dots, q^{t-1}) = P(q^t | q^{t-1})$$

Cadeias de Markov que seguem a propriedade de Markov também são ditas Cadeias de *primeira-ordem*, com o número de ordem crescendo com o número de instantes passados considerados para a determinação do estado seguinte (caso seja necessário uma memória de dois estados anteriores para a determinação do próximo, diz-se cadeia de segunda-ordem, etc.) (JUFRASKY e MARTIN, 2006).

Para ilustrar o conceito de cadeia de Markov e facilitar a introdução aos Modelos Ocultos de Markov, considere uma pequena cadeia de três estados representando um modelo simplificado do clima de alguma região em um período do ano, utilizada para previsão do tempo:

Exemplo 3.1 Modelo Simplificado do clima para previsão do tempo

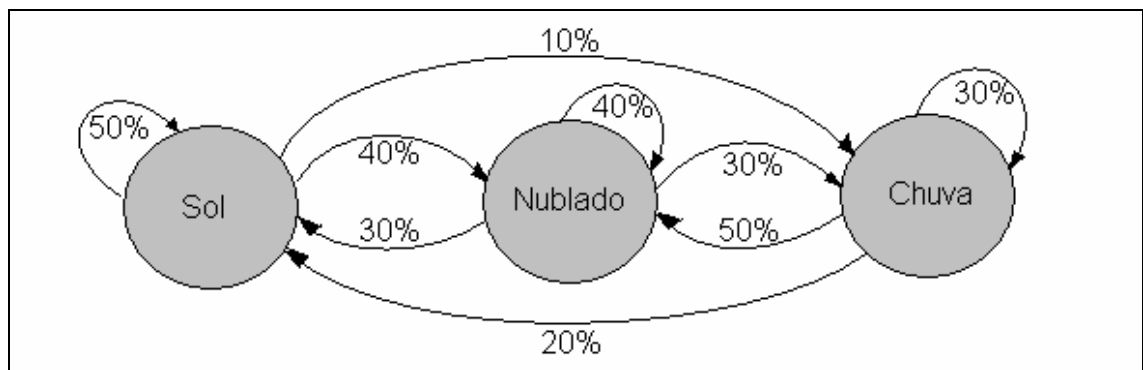


Figura 3.1 – Cadeia de Markov para previsão do tempo

Neste caso, sabendo-se o estado corrente da cadeia (ou o estado inicial), poder-se-ia calcular com facilidade a probabilidade de, por exemplo, a uma seqüência arbitrária de estados ocorrer: Sol, Sol, Nublado, Nublado, Nublado, Chuva. Chamando esta seqüência de observações de $O = (q_1, q_1, q_2, q_2, q_2, q_3)$, e sabendo que a primeira observação representa o clima presente (ou seja, o estado inicial é “sol”) podemos escrever:

$$\begin{aligned} P(O|\text{Modelo}) &= P(q_1|q_1) \cdot P(q_2|q_1) \cdot P(q_2|q_2) \cdot P(q_2|q_2) \cdot P(q_3|q_2) \\ &= (0,50) \cdot (0,40) \cdot (0,40)^2 \cdot (0,30) \\ &= 0,0096 = 0,96\% \end{aligned}$$

Com o exemplo pode-se ver o a restrição imposta pela propriedade de Markov: para se determinar a chance de chover no dia seguinte, não importa por quantos dias o tempo tem estado chuvoso ou nublado, mas apenas o estado do tempo no dia anterior, facilitando muito o trabalho de previsão do tempo.

Até agora, o modelo proposto permite uma observação direta do estado atual (tempo) da cadeia. Para uma maior aplicabilidade, imaginemos que as observações fossem indiretamente ligadas ao estado, intermediadas por uma função probabilística (RABINER, 1989).

Mantendo o exemplo, imaginemos (colocando discretamente a razoabilidade de lado) que o profissional que prevê o tempo não possa observar diretamente o clima. Trabalhando em um ambiente confinado, sua única dica de como está o tempo baseia-se nas vestimentas dos outros companheiros de trabalho. Como nem sempre seus companheiros conseguem prever corretamente o tempo, uma pessoa portando um guarda-chuva não significa *certeza* de chuva, mas tão somente uma probabilidade alta. Da

mesma forma alguém de camiseta poderia ser surpreendido por um mau tempo, mas com menor probabilidade.

Neste cenário ligeiramente estranho, a cadeia de estados do tempo está *oculta* ao meteorologista, e suas aferições têm que se basear em *observações* relacionadas com o tempo através de uma matriz de probabilidades. Em um diagrama:

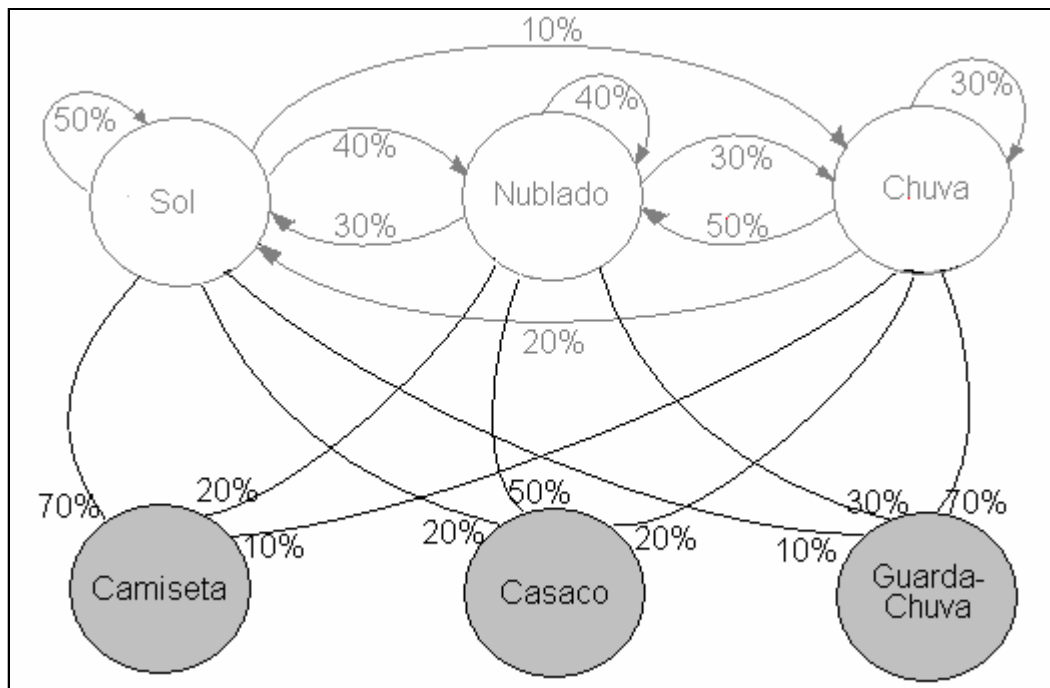


Figura 3.2 – Modelo Oculto de Markov para previsão do tempo

A partir do modelo acima o meteorologista pode inferir, observando o comportamento de seus companheiros durante um tempo determinado (uma semana, digamos), qual foi a seqüência de estados ocultos (o tempo) mais provável de produzir a seqüência de observações, obtendo assim uma expectativa de em qual estado o sistema está no momento atual, e conseqüentemente permitindo a previsão do dia seguinte.

Uma pergunta válida seria questionar como as probabilidades do modelo (os pesos do grafo na figura 3.2) foram obtidas, e como se pode garantir que não mudem. O modelo, como de praxe no campo de aprendizagem de máquina, é *treinado* a partir de um conjunto de dados para estabelecer as probabilidades que permite sua utilização. A resposta à segunda pergunta é que simplesmente não se pode garantir, razão pela qual o sistema deve ser eventualmente treinado de novo.

O exemplo acima é uma boa introdução de um Modelo Oculto de Markov. Na próxima seção definiremos o modelo formalmente.

3.2 Elementos de um Modelo Oculto de Markov

Mantendo nosso exemplo em vista definimos um Modelo Oculto de Markov da seguinte forma (baseado em JUFRASKY e MARTIN (2006)).

Definição 3.1 Um Modelo Oculto de Markov λ é composto por:

- Um conjunto $Q=\{q_i\}$ de estados ocultos do modelo, ou apenas estados.
- Uma matriz de probabilidades de transições $A=\{a_{ij}\}$ entre os estados q_i e q_j , onde $a_{ij} \in [0, 1]$ com $i, j \in [1, |Q|]$, e $\sum a_{ij} = 1$ para um mesmo estado q_i .
- Um conjunto $O=\{o_n\}$ de estados observáveis, ou observações.
- Uma matriz de probabilidades $B=\{b_{in}\}$ indicando a chance do estado q_i produzir a observação o_n , onde $b_{in} \in [0, 1]$ com $i \in [1, |Q|]$, $n \in [1, |O|]$ e $\sum b_{in} = 1$ para um mesmo estado q_i .
- Uma distribuição $\Pi=\{\pi_i\}$ de probabilidades do modelo iniciar no estado q_i , onde $\pi_i \in [0, 1]$ com $i \in [1, |Q|]$.

e pode ser notado como $\lambda =\{A, B, \Pi\}$

Podemos nos deter por um instante para identificar, no exemplo 3.1, os elementos acima. Os conjuntos Q e O são facilmente identificáveis como $Q=\{\text{“sol”}, \text{“nublado”}, \text{“chuva”}\}$ e $O=\{\text{“camiseta”}, \text{“casaco”}, \text{“guarda-chuva”}\}$. As matrizes A e B podem ser igualmente identificadas como:

Matriz de probabilidade de transições A			
	Sol	Nublado	Chuva
Sol	0,50	0,40	0,10
Nublado	0,30	0,40	0,30
Chuva	0,20	0,50	0,30

Tabela 3.1 – Matriz A do modelo de previsão do tempo

Matriz de probabilidade de observações B			
	Camiseta	Casaco	Guarda-Chuva
Sol	0,70	0,20	0,10
Nublado	0,20	0,50	0,30
Chuva	0,10	0,20	0,70

Tabela 3.2 – Matriz B do modelo de previsão do tempo

Note que no exemplo a matriz A não possui valores 0, mas a definição não impede que o sistema não seja ergódico (quando é possível alcançar qualquer estado a partir de qualquer outro estado).

Finalmente, no exemplo não se considerou em qual estado o modelo se iniciaria e, portanto, o vetor Π não está definido.

Em seus influentes artigos do final da década de 80, Lawrence Rabiner destacava como três os problemas fundamentais que os Modelos Ocultos de Markov apresentavam (RABINER, 1989, JUFRAKY e MARTIN, 2006):

Problema 1 – Cálculo de Probabilidade: dada uma seqüência de observações $Ob=\{o^1, o^2, \dots, o^n\}$ e um Modelo Oculto de Markov $\lambda =\{A, B, \Pi\}$, como calcular a probabilidade de que o modelo produza a seqüência observada, ou seja, $P(Ob | \lambda)$?

Problema 2 – Decodificação: dada uma seqüência de observações $Ob=\{o^1, o^2, \dots, o^n\}$ e um Modelo Oculto de Markov $\lambda =\{A, B, \Pi\}$, como calcular qual seria a seqüência de estados ocultos q_i que mais provavelmente produziria a seqüência de observações?

Problema 3 – Treinamento: dada uma seqüência de observações $Ob=\{o^1, o^2, \dots, o^n\}$ e um Modelo Oculto de Markov $\lambda =\{A, B, \Pi\}$, como ajustar os valores de A, B e Π de forma a maximizar a chance de que o modelo produza a seqüência observada, ou seja, maximizar $P(Ob | \lambda)$?

Nas próximas seções veremos como estes problemas foram enfrentados na literatura existente.

3.2.1 Cálculo de Probabilidades com o algoritmo *forward*

O primeiro problema solicita que calculemos, dada uma seqüência de observações $Ob=\{o^1, o^2, \dots, o^n\}$ e um Modelo Oculto de Markov $\lambda =\{A, B, \Pi\}$, a probabilidade de que o modelo produza a seqüência Ob , ou seja, $P(Ob | \lambda)$.

Como mostraram RABINER e JUANG (1986), a solução do problema nº1 pode ser dada pela enumeração de todas as combinações possíveis de tamanho n (o número de observações). No entanto, com cálculos de ordem de i^{2n} , sendo i o número de estados ocultos do modelo, o volume de operações cresce rapidamente, tornando qualquer algoritmo que siga esta idéia virtualmente inútil (RABINER e JUANG, 1986).

A solução mais utilizada nas aplicações de Modelos Ocultos de Markov é a implementação do algoritmo de programação dinâmica conhecido como *forward* (na realidade, o termo mais utilizado é *algoritmo forward-backward*, porém a segunda parte do algoritmo é utilizada apenas em uma solução para o terceiro problema, como veremos adiante).

O algoritmo *forward* calcula uma treliça onde cada célula α representa a probabilidade de estar no estado q_i após as primeiras t observações, dado o Modelo Oculto de Markov λ (JUFRAFSKY e MARTIN, 2006). Formalmente:

$$\alpha_t(i) = P(O_1 O_2 \cdots O_t, q_t = S_i | \lambda)$$

RABINER (1989), entre vários outros autores, demonstra como o algoritmo *forward* funciona demonstrando por indução que a probabilidade $P(\text{Ob} | \lambda)$ pode ser calculada da seguinte forma (em um Modelo Oculto de Markov com N estados ocultos):

Inicialização:	$\alpha_1(i) = \pi_i b_i(O_1),$	$1 \leq i \leq N$
Indução:	$\alpha_{t+1}(j) = \left[\sum_{i=1}^N \alpha_t(i) a_{ij} \right] b_j(O_{t+1}),$	$1 \leq t \leq T - 1; 1 \leq j \leq N$
Terminação	$P(O \lambda) = \sum_{i=1}^N \alpha_T(i).$	

No passo de indução, a célula α do passo seguinte é calculada como a multiplicação da probabilidade do caminho anterior (somatório de $\alpha_t(i)$), pela probabilidade da mudança do estado q_i para o estado q_j (a_{ij}), e pela probabilidade de produzir a observação seguinte ($b_j(O_{t+1})$).

O número de operações do algoritmo é de $i^2 t$ para uma seqüência de t observações em um modelo com i estados ocultos (RABINER, 1989).

3.2.2 Codificação com o algoritmo de *Viterbi*

O segundo problema, conhecido como *decodificação*, pede que calculemos a seqüência de estados ocultos de um Modelo Oculto de Markov $\lambda = \{A, B, \Pi\}$ que mais provavelmente produziria uma seqüência de observações $\text{Ob} = \{o^1, o^2, \dots, o^n\}$.

A solução é dada através de outro algoritmo de programação dinâmica conhecido como *algoritmo de Viterbi*, onde mais uma vez através de indução matemática podemos calcular a variável δ definida como (RABINER, 1989):

$$\delta_t(i) = \max_{q_1, q_2, \dots, q_i} P[q_1 q_2 \dots q_i = i, O_1 O_2 \dots O_t | \lambda]$$

δ pode ser calculada recursivamente através do passo de indução abaixo, que multiplica o caminho de estados com maior probabilidade encontrado ($\max \delta_t(i)$), pela probabilidade de transição do estado i para o estado j (a_{ij}), e pela probabilidade de gerar a observação seguinte ($b_j(O_{t+1})$).

$$\delta_{t+1}(j) = [\max_i \delta_t(i) a_{ij}] \cdot b_j(O_{t+1})$$

O algoritmo de Viterbi é extremamente semelhante ao algoritmo forward, substituindo a soma pela maximização no passo de indução (JUFRAFSKY e MARTIN, 2006). É importante notar que o algoritmo de Viterbi produz a sequência mais provável de estados ocultos capaz de produzir a sequência de observações, o que não implica em obter o estado oculto mais provável de estar após a mesma sequência. Esta distinção será importante para este trabalho, como será evidenciado no capítulo seguinte.

3.2.3 Treinamento por Maximização de Expectativa

O terceiro e mais complexo problema pede que estimemos quais os parâmetros do Modelo Oculto de Markov (ou seja, as matrizes A , B e Π) que maximizam a chance do modelo produzir uma sequência de observações $Ob = \{o^1, o^2, \dots, o^n\}$.

Este problema, também conhecido como *treinamento* do Modelo Oculto de Markov, é o único dos três problemas que até o momento não foi resolvido no sentido amplo da palavra. Todas as soluções apresentadas até o momento contornam sua grande complexidade obtendo máximos locais (RABINER, 1989).

O algoritmo mais utilizado para este propósito é uma variação do algoritmo de Maximização de Expectativa (*Expectation-Maximization* ou *EM*) conhecido como Máxima Verossimilhança (*Maximum Likelihood* ou *ML*) ou, devido a sua introdução por Baum em um artigo de 1967 (veja BAUM e EAGON (1967), e à pesquisa independente de Lloyd Welch (veja WELCH, (2003)) por *algoritmo de Baum-Welch* (COUVREUR, 1996).

Outro algoritmo derivado da Maximização de Expectativa utilizado para o treinamento de Modelos Ocultos de Markov é o K-Means, baseado na técnica de clusterização (PERRONE e CONNELL (2000), FRANCOIS (2006)).

Entretanto, sendo um problema de otimização, diversas outras técnicas vêm sendo empregadas para o treinamento de Modelos Ocultos de Markov, de Estimadores Bayesianos a otimizadores combinatórios como Otimizadores de Enxame de Partículas

(*Particle Swarm Optimization* ou *PSO*) ou Algoritmos Genéticos (AG), a última técnica sendo a utilizada neste trabalho e coberta na seção seguinte.

O algoritmo Baum-Welch está extensamente coberto na literatura de Modelos Ocultos de Markov e, pela nossa preferência ao treinamento com Algoritmos Genéticos (cujos motivos serão apresentados na seção seguinte), não será aqui detalhado. Para maiores informações, consulte RABINER (1989), JUFRAFSKY e MARTIN (2006) ou WELCH (2003).

3.2.4 Treinamento com Algoritmos Genéticos

A vantagem da utilização, no presente trabalho, destes otimizadores combinatórios em relação aos algoritmos convencionais de Maximização de Expectativa como o *Baum-Welch*, independentemente das alegações de melhoria em desempenho (embora existam, como se pode ver nos trabalhos de CHAU, KWONG ET AL (1997) e MANABE, HATANAKA ET AL (2006)) está na facilidade do controle mais minucioso dos parâmetros a serem otimizados. O procedimento ML, em sua forma mais comumente implementada⁴, estima *todos* os parâmetros do modelo, enquanto que na presente aplicação alguns destes são conhecidos. Neste caso, os Algoritmos Genéticos fornecem uma solução para o treinamento do modelo mantendo certos valores constantes e estimando os demais.

A aplicação do AG na estimativa dos parâmetros se dá estabelecendo um cromossoma que contenha os valores desconhecidos das matrizes A , B e Π do modelo (CHAU, KWONG ET AL., 1997). Então se utiliza como função de *fitness* a maximização da chance do modelo – produzido pelos parâmetros contidos no cromossomo, juntamente com eventuais parâmetros já conhecidos – gerar as observações, utilizando para isto algoritmo *forward* descrito nesta seção.

Os operadores usuais de *crossover* e mutação podem ser definidos normalmente como, por exemplo, um ou dois pontos de *crossover* e mutação de um percentual dos valores do cromossomo (CHAU, KWONG ET AL., 1997).

Os resultados de desempenho encontrados pelos pesquisadores são muito positivos: os treinamentos de Modelos Ocultos de Markov com Algoritmos Genéticos têm se mostrado superiores aos tradicionais métodos de Maximização de Expectativa tanto em tempo de convergência quanto em encontrar um máximo local bom, ou mesmo em encontrar o máximo global. É importante lembrar que nenhum dos dois métodos garante a obtenção do máximo global (CHAU, KWONG ET AL., 1997).

Há ainda pesquisa no sentido de uma união das duas técnicas: utilizam-se ciclos de treinamentos com o algoritmo Baum-Welch seguidos de treinamentos por Algoritmos

⁴ Embora não seja impossível adaptar o algoritmo de Baum-Welch para fornecer tal controle, este desenvolvimento estaria além dos objetivos deste trabalho.

Genéticos com o objetivo de aumentar as chances de obtenção do máximo global (MANABE, HATANAKA ET AL.,2006).

3.3 Semelhança entre Modelos Ocultos de Markov

JUANG e RABINER (1985) atacaram o problema de comparar dois Modelos Ocultos de Markov, utilizando para isto uma adaptação do indicador conhecido como Distância de Kullback-Leibler, uma medida de semelhança entre duas funções de distribuição também conhecida como entropia relativa. A Distância de Kullback-Leibler entre duas funções de distribuição P e Q é calculada como:

$$D_{KL}(P \parallel Q) = \int_{-\infty}^{\infty} P \ln \frac{P}{Q} dx$$

Os autores propuseram medir a distância entre dois Modelos Ocultos de Markov λ_1 e λ_2 através da seguinte relação (DUGAD e DESAI, 1996):

$$D(\lambda_1, \lambda_2) = \sum_o P[O | \lambda_1] \ln \left(\frac{P[O | \lambda_1]}{P[O | \lambda_2]} \right)$$

Entretanto, para todos os fins tal cálculo é inviável pela impossibilidade de gerar todas as seqüências possíveis, e ainda pela eventual impossibilidade de um modelo gerar determinadas seqüências que o outro consegue, como nos ensina DUGAD e DESAI (1996). A solução final, inspirada na fórmula acima, para uma medida de equivalência estatística entre dois Modelos Ocultos de Markov λ_1 e λ_2 , utiliza o algoritmo de Viterbi para avaliar o quão bem um modelo consegue gerar observações produzidas pelo outro. A fórmula segue (RABINER, 1989):

$$D(\lambda_1, \lambda_2) = \frac{1}{T} \{ \log P(O^{(2)} | \lambda_1) - \log P(O^{(2)} | \lambda_2) \}$$

Onde $O^{(2)}$ é uma seqüência de observações gerada pelo modelo λ_2 . Como a fórmula acima não é simétrica, isto é, $D(\lambda_1, \lambda_2)$ é diferente de $D(\lambda_2, \lambda_1)$, a versão simétrica da distância é (RABINER, 1989):

$$D_s(\lambda_1, \lambda_2) = \frac{D(\lambda_1, \lambda_2) + D(\lambda_2, \lambda_1)}{2}$$

Capítulo 4

Aplicações de Modelos Ocultos de Markov em Jogos de Estratégia Mista

Jogos de estratégia mista vêm sendo utilizados há tempos para a modelagem e solução de problemas em áreas tão diversas como finanças (veja como exemplo GIBBONS (1992)), militar (SHINAR e LIPMAN, 1995), petrolífera (MEDINA e FERREIRA FILHO, 2003, MEDINA, FERREIRA FILHO ET AL, 2003), comunicação (LIU, MAN ET AL, 2006), teoria da informação (ROST e GERHARD, 2007, além de diversos problemas da informática (MENASCHÉ, 2005; SHEN e FAN, 2007; TU, BI ET AL., 2006; entre muitos outros).

Muitas destas áreas apresentam igualmente diversas aplicações para Modelos Ocultos de Markov como, por exemplo, as diversas aplicações na área financeira (veja diversos artigos em MAMON e ELLIOTT (2007) e informática (MILLER, 1999, CHANG, GIYAN ET AL, 2000, SILVEIRA FILHO, 2006).

Entretanto, como já afirmado na introdução desta dissertação, não obstante a característica de incerteza presente-se em ambas as áreas, não há trabalhos acadêmicos relacionando os dois conceitos. Tal é o objetivo deste trabalho, portanto: apresentar uma possibilidade de ligação entre estes dois campos de pesquisa.

O problema que nos propomos a abordar foi descrito, na introdução, como um jogo de estratégia mista, repetido, entre dois jogadores, onde o primeiro jogador alterna entre estratégias à sua discrição. Resta-nos definir o problema mais formalmente, utilizando a notação apresentada nos capítulos anteriores.

Dado um jogo estratégico $\gamma = \{J, S_i, u_i\}$ de estratégia mista, repetido indefinidamente, entre dois jogadores j_1 e j_2 (ou seja, $J = \{j_1, j_2\}$), onde o jogador j_1 alterna, em frequência pré-estabelecida e fixa ao longo do jogo, entre e estratégias mistas σ_j^1 a $\sigma_j^e = (w_1^e, \dots, w_{|S_j|}^e)$ (distribuições probabilísticas sobre suas estratégias puras, onde w_1 indica a probabilidade de jogar a estratégia pura S_1).

Nem qual a estratégia mista sendo utilizada pelo jogador j_1 em dado momento, nem a distribuição probabilística $(w_1, \dots, w_{|S|})$ de cada estratégia são, a princípio, conhecidas pelo jogador j_2 .

Para exemplificar este jogo, suponha que no momento do saque de uma partida de tênis, para decidir entre sacar aberto ou no centro da quadra, o jogador sacador alterne entre três perfis distintos: agressivo, moderado e defensivo.

Os jogadores estão disputando a devolução adequada da bola, assim interessa ao jogador receptor antecipar o lado corretamente. A matriz de recompensas deste jogo simples está representada na tabela abaixo, pelas recompensas do jogador receptor. Sendo o objeto de disputa uma constante (a possibilidade de devolver a bola), as recompensas do Sacador podem ser inferidas como o complemento de 1:

Jogador Sacador	Jogador Receptor	
	Antecipar Saque Aberto	Antecipar Saque no Centro
Saque Aberto	1	0
Saque no Centro	0	1

Tabela 4.1 – Matriz de recompensas do jogo de Tênis

O jogador receptor, por sua vez, desconhece de que forma cada perfil joga (entre sacar aberto e sacar fechado) e a frequência com que o sacador utiliza cada perfil, embora saiba que esta frequência é fixa e não se alterará durante as repetições do jogo. A figura abaixo ilustra o jogo:

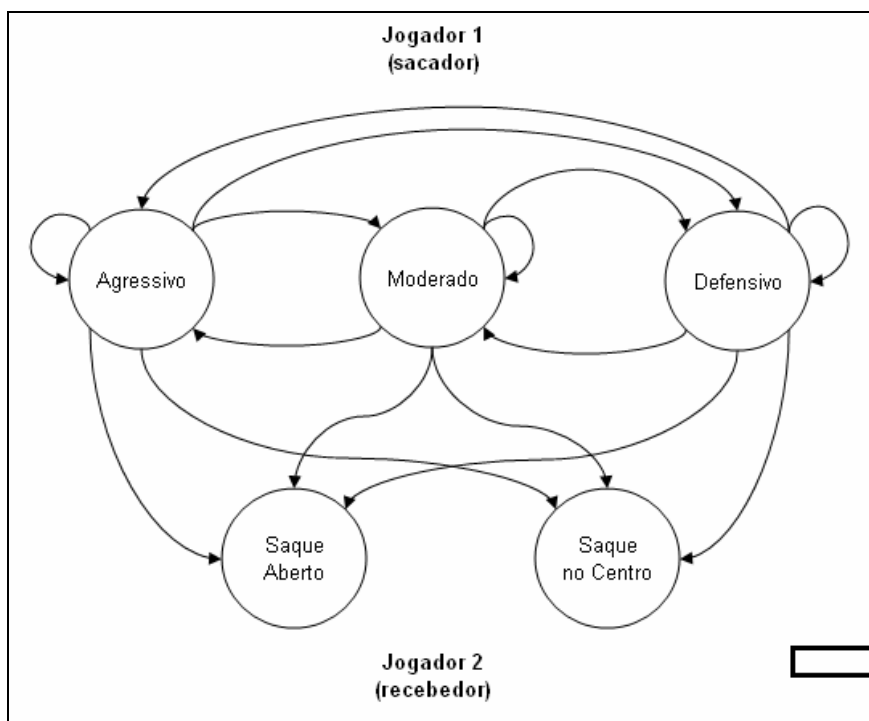


Figura 4.1 – Diagrama do jogo de tênis

Em um segundo momento, sugerimos uma modificação no jogo de tal maneira que as alternativas do primeiro jogador não sejam apenas estratégias mistas, mas jogos estratégicos distintos, com matrizes de recompensas igualmente diferentes. Neste novo jogo, as alternativas do primeiro jogador oferecem equilíbrios diversos, os quais não são do conhecimento do segundo jogador, e podem ou não estar sendo aplicadas.

Mantendo o exemplo da partida de tênis, o jogo estratégico modificado seria tal que as recompensas seriam medidas como a chance do recebedor retornar a bola adequadamente. Os perfis agressivo, moderado e defensivo do sacador produziriam, através de estilos distintos do saque, chances diferentes do recebedor retornar a bola, e conseqüentemente, matrizes de recompensa diferentes.

O desenho deste novo jogo ainda pode ser verificado pela figura 4.1 acima, e o conhecimento do jogador recebedor continua limitado a observar o lance escolhido pelo jogador sacador, ou seja, desconhece igualmente as matrizes de recompensa dos vários perfis a que está submetido.

Queremos, assim, verificar se é possível treinar um Modelo Oculto de Markov $\lambda = \{A, B, \Pi\}$ que represente tal jogo e possa auxiliar o jogador j_2 a escolher seu próximo lance. Ou seja, em passos, verificar se é possível:

1. É possível, a partir de informações dos pontos anteriormente disputados, criar um Modelo Oculto de Markov que represente tais jogos?
2. De que forma tal modelo e as ferramentas relacionadas, podem ajudar o jogador 2 a escolher seu próximo lance?
3. Qual é o desempenho deste método em comparação com uma escolha aleatória do jogador 2, ou ainda uma escolha baseada em contagem simples dos resultados dos jogos anteriores?

Incidentemente, gostaríamos de verificar se o conhecimento, por parte do jogador j_2 , sobre os detalhes das estratégias mistas, ou seja, por exemplo, qual a chance do perfil agressivo sacar no centro, pode influenciar estes três passos. Ainda perguntamos se é possível verificar a semelhança entre o modelo produzido pelo método proposto e o modelo "real", e se esta semelhança influencia no desempenho do modelo.

Vamos abordar estas questões separadamente.

4.1 Modelagem dos Jogos

A questão de criar um Modelo Oculto de Markov que represente o primeiro jogo proposto acima se esclarece com o diagrama 4.1 apresentado. Não é difícil perceber que as estratégias que servem de opção ao jogador j_1 , por alternarem-se de forma fixa e pré-estabelecida, e serem desconhecidas pelo jogador j_2 , formam uma cadeia de Markov

oculta do segundo jogador, para quem apenas as decisões do jogador j_1 , ou seja, o local do saque, são conhecidas. Estas decisões formam, por sua vez, as observações do modelo.

Assim, considerando um modelo $\lambda = \{A, B, \Pi\}$, a matriz A apresenta-se na freqüência com que o jogador j_1 alterna entre os estilos de jogo, e a matriz B é formada pelas estratégias mistas σ_j^1 a $\sigma_j^e = (w_1^e, \dots, w_{|S_j|}^e)$. O vetor Π seria a chance do jogador j_1 iniciar o jogo em uma determinada estratégia.

Para ilustrar estas afirmações, veja na figura 4.2 abaixo o mesmo jogo, porém com os elementos das matrizes A e B representados:

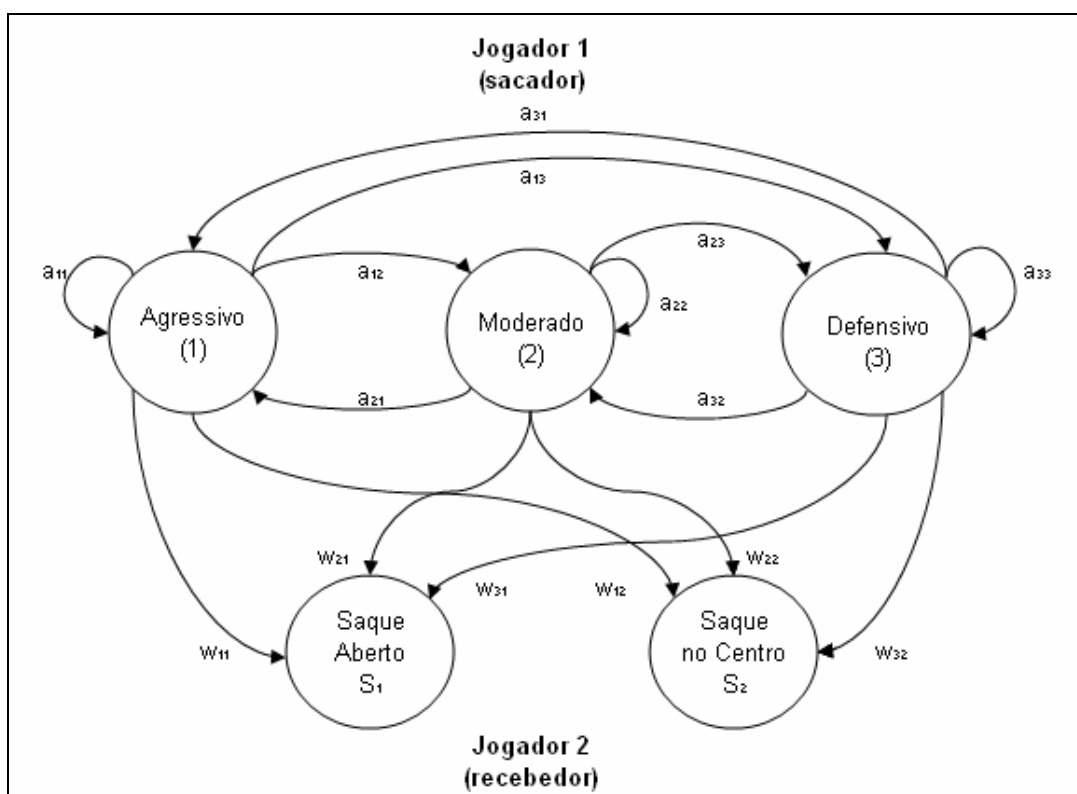


Figura 4.2 – Diagrama do jogo de tênis com elementos de A e B representados

Podemos assim generalizar esta modelagem para um jogo estratégico entre dois jogadores, com um número e qualquer de estratégias mistas entre as quais o primeiro jogador alterna, cada uma sendo uma distribuição probabilística das s estratégias puras do jogo, também em número indefinido.

Em um determinado momento discreto onde a variação estratégica n esteja sendo jogada, o jogo possibilita uma chance a_n^n de, no próximo momento, permanecer na variação n , ou ainda a chance a_m^n de alternar à variação estratégica m , com $m, n = [1, e]$.

Tal cenário poderia ser representado pela figura abaixo:

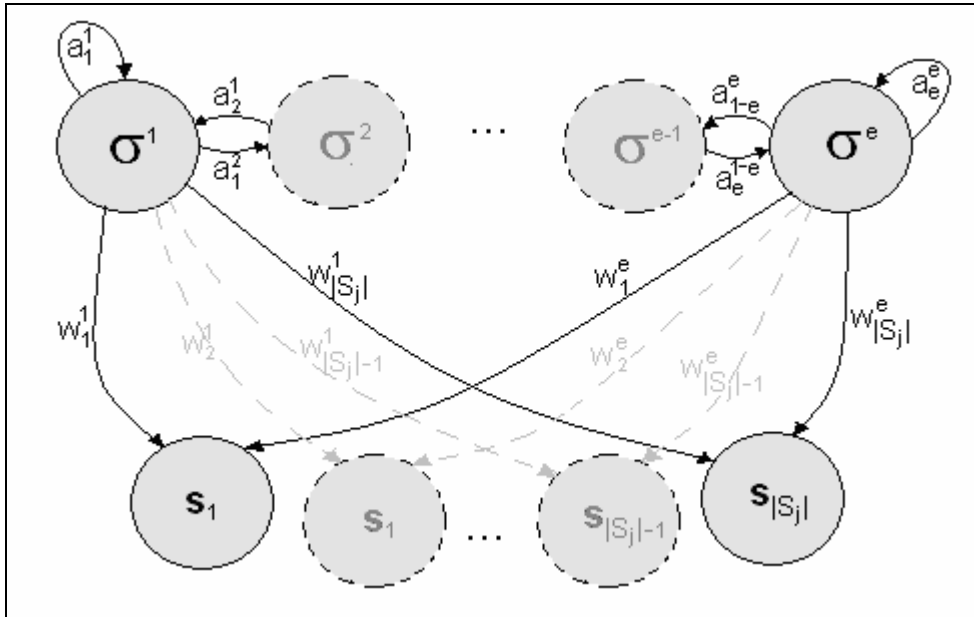


Figura 4.3 – Modelo genérico para um jogo de estratégias mistas

Como já delineado antes, é fácil perceber a aplicação, na figura acima, de um Modelo Oculto de Markov: considere o Modelo Oculto de Markov $\lambda = \{A, B, \Pi\}$ com o conjunto Q de e estados ocultos q_1 a q_e e o conjunto O de $|S_j|$ estados observáveis, onde a matriz de probabilidades de transições $A = \{a_{ij}\}$ entre os estados q_i e q_j e a matriz de probabilidades $B = \{b_{in}\}$ indicando a chance do estado q_i produzir a observação o_n podem ser diretamente deduzidas das transições entre estratégias a e as proporções de soluções w , respectivamente. Resta-nos a definição da distribuição $\Pi = \{\pi_{ij}\}$ que depende diretamente da definição do jogo – a probabilidade de que o jogo se inicie em cada variação estratégica.

O segundo jogo proposto possui modelagem semelhante, pois cada estado oculto agora representa não mais estratégias mistas, mas jogos diferentes (mas semelhantes, no sentido de possuírem os mesmos conjuntos de jogadores e estratégia puras disponíveis), que podem ser representados pela razão de sua diferença: o conjunto de funções utilidade, definidos a partir das matrizes de recompensa. Para auxiliar-nos, chamaremos cada jogo diferente desta forma como uma *variação estratégica*:

Definição 4.1 São definidas como *variações estratégicas* de um $\gamma = \{J, S_i, u_i\}$ os jogos que possuam o conjunto J de jogadores e as estratégias puras S_i iguais, mas diverjam pelo conjunto u_i de funções utilidade.

Desta forma, considere o jogo estratégico $\gamma = \{J, S_i, u_i\}$ com e variações estratégicas discretas identificadas pelas diferentes funções de utilidade u_i^e , cada uma estabelecendo pelo menos um equilíbrio de Nash. Considere ainda que um jogador $j \in J$ possua as

estratégias mistas σ_j^1 a $\sigma_j^e = (w_1^e, \dots, w_{|S_j|}^e)$ que indicam a proporção com que cada solução $s \in S_j$ é jogada em cada equilíbrio de Nash. Porque desejamos auxiliar o jogador 2, apenas as suas funções de utilidade são necessárias, e o modelo pode ser representado desta forma:

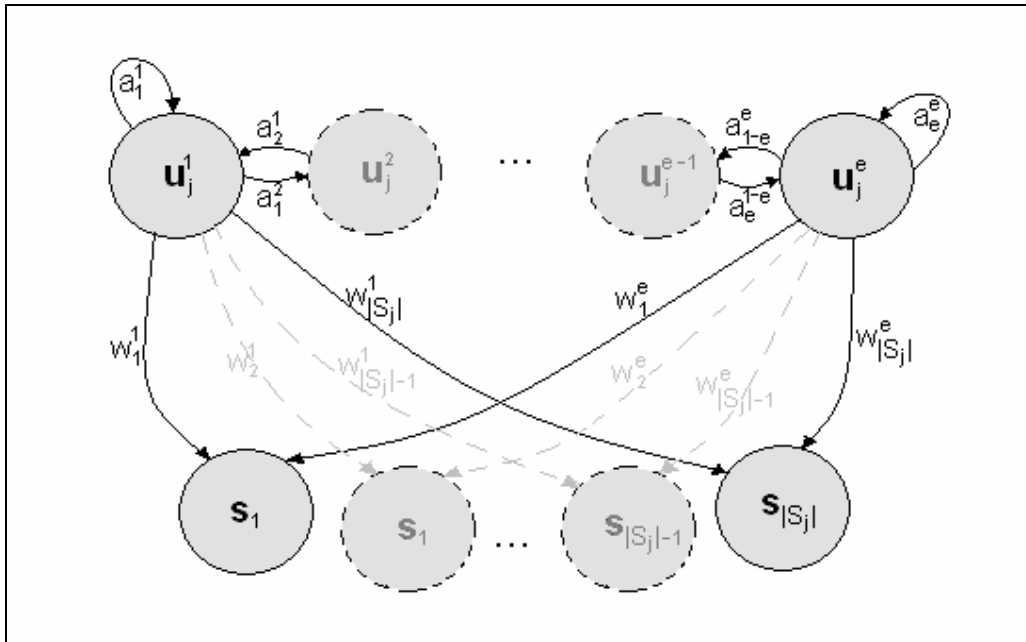


Figura 4.4 – Modelo genérico para um jogo de estratégias mistas com e variações estratégicas

Da mesma forma que no modelo anterior é possível perceber a aplicação de um Modelo Oculto de Markov, mantendo-se as mesmas relações entre os elementos acima esclarecidas.

O treinamento para a obtenção de tal Modelo Oculto de Markov far-se-á da forma mais usual: a partir de um conjunto de seqüências de observações, obtidas através de interações passadas do jogo, que servirão de insumo para os algoritmos de treinamento apresentados no capítulo 3 – de Expectativa de Maximização (Baum-Welch) ou, como utilizado neste trabalho, otimizadores combinatórios como os Algoritmos Genéticos.

A preferência pelo Algoritmo Genético deu-se não pelo seu melhor desempenho, medido pela probabilidade média do Modelo obtido gerar as seqüências que servem de conjunto de treinamento (veja CHAU, KWONG ET AL. (1997)), mas sim pela facilidade com que é possível manusear e fixar determinados valores das matrizes A e B (tal facilidade não se encontra prontamente disponível nas implementações disponíveis de Modelos Ocultos de Markov).

Este manuseio é particularmente útil para a execução dos cenários em que parte do Modelo Oculto de Markov já é conhecida, e não deve ser otimizada, como no caso em que o jogador j_2 conhece as distribuições probabilísticas das observações, ou seja, a matriz B.

O treinamento é, então, realizado utilizando uma estrutura cromossômica semelhante a CHAU, KWONG ET AL. (1997), valendo-se do fato de que algumas das variáveis prescindem de otimização, pois complementam outras em relação à unidade (por exemplo, se existem duas observações, apenas uma das probabilidades é otimizada, a outra sendo necessariamente o complemento para 1).

A função de avaliação (função de *fitness*) deve criar um Modelo Oculto de Markov a partir dos parâmetros contidos nos alelos do indivíduo e utilizar as seqüências de observações do conjunto de treinamento, calculando através do algoritmo *forward* (veja o capítulo 3 desta dissertação) a probabilidade do modelo gerar a seqüência de observações. O valor de *fitness* corresponde à média das probabilidades do Modelo Oculto de Markov gerar as seqüências de observações.

Neste trabalho utilizamos uma biblioteca de funções de Algoritmos Genéticos disponível gratuitamente (JGAP - <http://jgap.sourceforge.net/>), com excelentes resultados mesmo com configurações padrões a solução obtida foi significativamente melhor que as demais. Considerando que desta feita fomos capazes de estabelecer restrições que compatibilizam a solução ao problema, a técnica se mostrou muito eficaz – vale lembrar que o ajuste fino de um algoritmo genético é assunto para diversos livros. Veja, por exemplo, LINDEN (2006).

Uma vez treinado o Modelo Oculto de Markov, podemos prosseguir para o segundo problema, isto é, como utilizar o Modelo para auxiliar o jogador j_2 a escolher seu próximo lance.

4.2 Obtenção do Próximo Lance

Queremos auxiliar o jogador j_2 a escolher seu próximo lance. Obviamente, o jogador j_2 quer jogar o lance que mais provavelmente maximize sua recompensa, ou seja, conhecendo as chances do jogador j_1 jogar cada estratégia pura, escolher sua resposta de acordo com a matriz de recompensas do jogo.

A partir do Modelo Oculto de Markov, supondo que se sabe qual é a estratégia que o jogador j_1 mais recentemente utilizou, é possível calcular, através da matriz de transições de estados A , qual é a chance do Modelo estar em cada estado oculto na próxima iteração. A partir deste próximo estado oculto, por sua vez, é possível calcular qual é a chance do modelo gerar cada observação, ou seja, do jogador j_1 executar cada estratégia pura.

Desta forma, caso saibamos qual é o estado oculto corrente, é possível calcular as chances de cada estratégia pura ser jogada pelo jogador j_1 , segundo Modelo Oculto de

Markov. A chance de cada observação o ocorrer em um modelo com e estados ocultos que se encontra correntemente no estado p é de:

$$\sum_{i=1}^e a_{pi} \cdot b_{io}$$

Ou seja, as chances da transição do estado oculto de p para i , multiplicadas pelas chances deste estado i produzir a observação desejada o . Assim, é possível estabelecer, dado o Modelo Oculto de Markov treinado, o lance mais provável do jogador j_1 para cada estado oculto do modelo que supostamente o modelo esteja, criando uma tabela como a seguinte:

Estado Oculto	Lance Provável
q_1	$\arg \max_o \sum_{i=1}^e a_{1i} \cdot b_{io}$
q_1	$\arg \max_o \sum_{i=1}^e a_{2i} \cdot b_{io}$
...	...
q_e	$\arg \max_o \sum_{i=1}^e a_{ei} \cdot b_{io}$

Tabela 4.1 – Matriz A do modelo de previsão do tempo

De posse destes valores, o jogador j_2 pode, então, definir seu próximo lance, que lhe deve ser fornecido por seu conjunto de funções de utilidade, ou pela matriz de recompensas do jogo. No primeiro jogo definido, com apenas uma matriz de recompensa para todos os estados ocultos, a escolha é imediata, considerando os valores da matriz. No exemplo do jogo de tênis, basta ao jogador recebedor escolher o mesmo lado em que acredita que o sacador jogará.

No segundo modelo proposto, entretanto, como cada estado oculto representa uma variação estratégica, possuidora de sua própria matriz de recompensas, o jogador j_2 deve calcular a chance de transição do estado em que acredita estar o modelo para o próximo estado e avaliar qual o lance que a matriz de recompensas deste estado sugere (isto é, o lance que maximize sua recompensa). Utilizando a notação apresentada, o jogador j_2 deve escolher seu lance $s \in S_2$ calculando:

$$\arg \max_s \sum_{i=1}^e a_{pi} \cdot u_2^i(o, s)$$

Ou seja, o lance cuja recompensa, considerando que o jogador j_i jogará seu lance o (anteriormente calculado) multiplicada pela chance do modelo alterar seu estado de p para o estado possuidor da matriz de recompensas utilizada, seja a maior possível.

A utilização da recompensa, na forma da função de utilidade $u(o,s)$, pode parecer um cuidado desnecessário, pois bastaria utilizar, como no primeiro caso, a matriz B para identificar o lance do jogador sacador. Este cuidado, porém, é importante para casos em que nosso jogo de exemplo não ilustra bem: seria como imaginar que, dentre os perfis possíveis do jogador sacador, haveria um no qual a maior recompensa do jogador receptor estaria em *desviar da bola*, indo para o lado oposto. Assim, o lance do jogador sacador não necessariamente implica em um lance do jogador receptor, e a relação entre estes dois lances está expressa exatamente na matriz de recompensas do jogo.

Entretanto, a premissa utilizada para que estas considerações sejam eficazes, de que conheçamos o estado oculto atual do modelo, não é facilmente obtida. Para alcançar este objetivo, precisaríamos responder a uma quarta pergunta fundamental do Modelo Oculto de Markov: a partir de uma seqüência de observações, qual é a chance $p(q)$ de o modelo encontrar-se em cada estado oculto q ?

A partir deste vetor $p(q)$ de probabilidades seria possível calcular a chance de cada observação ocorrer no instante seguinte, incluindo na equação acima mais um termo, referente à chance do modelo encontrar-se no estado.

A obtenção de tal vetor não é, entretanto, tarefa trivial, e para os objetivos desta dissertação utilizaremos uma simplificação deste problema, através do algoritmo de Viterbi.

Para tal, vale lembrar a aplicação do algoritmo de Viterbi, visto no capítulo 3: obter a seqüência de estados ocultos que mais provavelmente gera uma seqüência de observações. Utilizaremos a premissa de que o Modelo Oculto de Markov encontra-se, necessariamente, no último estado oculto da seqüência obtida pelo algoritmo de Viterbi.

É importante notar a gravidade desta restrição: mesmo a premissa de que o último estado oculto da seqüência obtida pelo algoritmo de Viterbi é o estado oculto mais provável não é garantida. Entendemos que tal premissa faz-se suficientemente precisa para os fins deste presente trabalho. Incidentalmente, durante

as execuções dos cenários a serem apresentados, realizamos uma medição da precisão desta premissa.

Assim, propomos que o algoritmo de Viterbi seja aplicado no Modelo Oculto de Markov obtido, e o último estado da seqüência obtida seja utilizado como se fosse, sem sombra de dúvidas (ou seja, chance igual a 1) o estado atual. A partir deste, aplica-se a equação acima para determinar a chance de cada estratégia pura ser escolhida pelo jogador j_1 , informação suficiente para o jogador j_2 escolher seu lance.

No próximo capítulo descreveremos o método utilizado para a verificação deste procedimento, e os critérios de comparação, bem como os cenários escolhidos para tal.

Capítulo 5

Descrição do Método e Apresentação dos Resultados Obtidos

No capítulo anterior, a partir de um jogo de estratégia mista determinado, apresentamos alguns problemas e descrevemos como o ferramental dos Modelos Ocultos de Markov poderia ser aplicado para solucioná-los.

No presente capítulo objetivamos descrever um método para empregar tais ferramentas e comparar sua eficácia, bem como apresentar cenários diferentes e os resultados deles obtidos.

Devido ao grande número de variáveis deste experimento, não seria viável a apresentação detalhada de cada cenário proposto e executado. Neste capítulo, apresentaremos detalhadamente alguns cenários que julgamos especificamente importantes para as conclusões do experimento, e no apêndice A.1 apresentamos todos os resultados obtidos. Para apreciação, os cenários executados envolvem, em diversas combinações:

1. Jogos com duas ou três observações;
2. Jogos com transições entre estados ocultos em várias configurações diferentes (sem tendência, tendenciosas a um estado particular, tendenciosas a manter-se no mesmo estado, entre outras);
3. Conjuntos de treinamento formados por 10, 25 e 50 seqüências de 10, 20 e 50 observações cada;
4. Modelos treinados considerando nenhuma informação sobre o modelo real, e alguma informação;
5. Para os cenários que envolvem variações estratégicas, combinações diferentes de matrizes de recompensa para provocar diferentes efeitos;
6. Acumulação de resultados considerando a utilização do estado oculto real, para avaliar a premissa introduzida na seção 4.2.

A execução foi realizada segundo o seguinte método proposto, a partir do jogo γ definido no capítulo 4:

1. Cria-se um Modelo Oculto de Markov para o jogo, com todos os elementos de A , B e Π arbitrariamente determinados, servindo de *modelo real*.
2. Gera-se s seqüências de o observações cada, a partir do modelo criado, para servir de conjunto de treinamento.
3. Treina-se um Modelo Oculto de Markov por Algoritmo Genético, a partir das s seqüências, utilizando os pesos já conhecidos, dependendo do cenário, para reduzir as variáveis a serem otimizadas (redução dos graus de liberdade).
4. Para cada seqüências s de o observações:
 - a. Aplica-se o Algoritmo de Viterbi para as $(o - 1)$ primeiras observações da seqüência.
 - b. Assumindo o último estado do caminho obtido no passo 4.a como o estado corrente, determina-se o próximo lance a ser escolhido, através da fórmula fornecida na seção 4.2.
 - c. Compara-se a observação final o da seqüência com o lance escolhido.
 - d. Compara-se a observação final o da seqüência com uma escolha aleatória de lance.
 - e. Acumula-se os resultados ao longo de todas as seqüências disponíveis.

Algumas considerações devem ser feitas sobre estes passos.

A geração das seqüências de observações de nosso conjunto de treinamento é realizada através do Modelo Oculto de Markov criado no primeiro passo, nosso modelo real. Basicamente, itera-se pelo modelo, aplicando-se as chances de mudança de estado e geração de cada observação, pelo número de vezes desejado em cada cenário. Como o conhecimento de qual estado oculto gerou cada observação é importante para a medição do resultado (e da qualidade da premissa definida no capítulo 4), a seqüência “real” de estados ocultos é igualmente armazenada.

O treinamento através de Algoritmos Genéticos possui muitos parâmetros, cujo ajuste fino não foi realizado. Aplicamos a configuração padrão da biblioteca utilizada, à exceção dos parâmetros do número de gerações e do número de indivíduos na população. Para estes dois, executamos cenários com alguns valores diferentes para chegar a uma boa relação entre otimização e tempo de processamento, nominalmente 100 gerações e 180 indivíduos na população.

Para a verificação da efetividade da premissa estabelecida no capítulo 4.2, utilizamos a informação do estado oculto “real”, armazenado durante a geração do conjunto de treinamentos, para, comparada com o estado oculto estabelecido pela aplicação do algoritmo de Viterbi, produzir um índice de acerto da premissa. Este índice foi utilizado para gerar uma distribuição *versus* recompensa, possibilitando a avaliação da influência da premissa nos resultados.

Outra distribuição, entre a medida da distância entre o modelo real e o treinado (medido através da distância de Kullback-Leibler, apresentada na seção 3.3) e a recompensa, foi utilizada para avaliar a influência do treinamento no resultado.

A acumulação de resultados foi realizada de formas diferentes para os lances aleatórios e os lances calculados pela aplicação dos conceitos da seção 4.2. No primeiro caso, para os cenários cujos jogos possuíam distribuição igual entre as observações finais (isto é, aos olhos do jogador j_2 a frequência de observações distribuía-se igualmente), foi gerado um número aleatório para indicar o lance. Quando a distribuição das observações dava-se, por influência de uma transição tendenciosa de estados ocultos, deslocada para alguma das observações, a acumulação foi realizada tanto de forma aleatória como considerando que o jogador j_2 jogaria *sempre* no lance cuja maior frequência se verificou.

Na acumulação dos lances calculados, consideramos também os lances que seriam calculados se soubéssemos o estado oculto que realmente gerou a observação, com o objetivo de criar mais uma forma de verificação da qualidade da premissa da seção 4.2.

5.1 Cenários Apresentados

5.1.1 Cenário 1

Neste cenário estabelecemos um jogo com três estados ocultos e duas observações, tal qual o jogo de tênis descrito na introdução deste trabalho. O jogador j_1 (sacador) alterna entre os perfis agressivo, moderado e defensivo com frequências semelhantes, sem tendências. Por conseguinte, nossa escolha para o comportamento de comparação do jogador j_2 (recebedor) é simplesmente escolher aleatoriamente um lado, com chances iguais.

O cenário foi executado com 50 seqüências de 10, 20 e 50 observações cada com o objetivo de verificar a influência do aumento de observações no método. A distância entre o Modelo Oculto de Markov treinado e o modelo real (utilizado para gerar as observações) foi igualmente medida para avaliar a sua relação com a recompensa obtida. Uma figura do modelo real é apresentada abaixo, para ilustração:

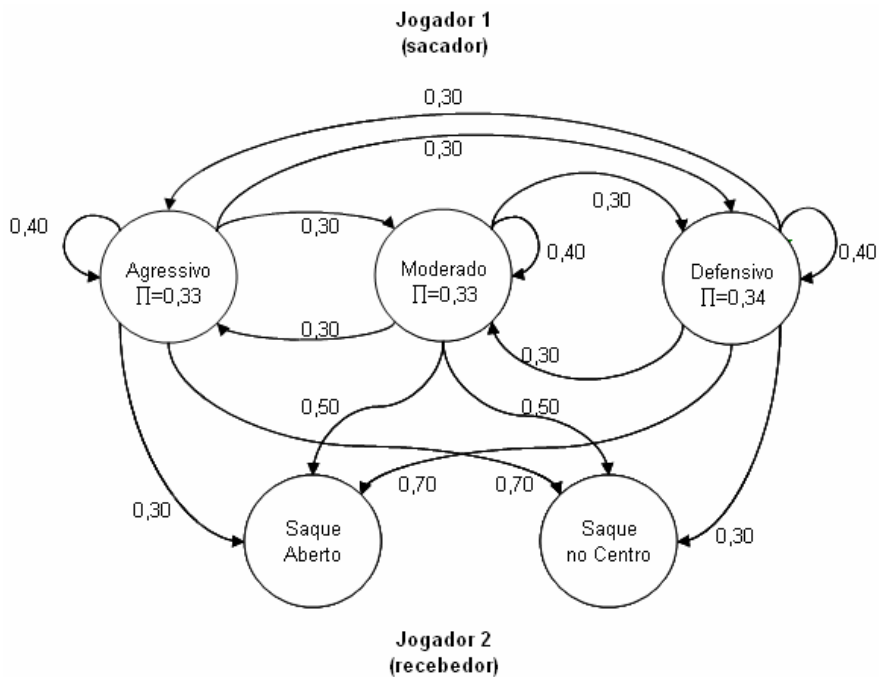


Figura 5.1.1.1 – Modelo do Cenário 1

Os resultados seguem:

Número de Observações	Média da Distância	Recompensa Aleatória	Recompensa Calculada
10	0.9600	24.46 (48.9%)	25.10 (50.2%)
20	0.6157	24.78 (49.6%)	25.61 (51.2%)
50	0.1444	24.97 (49.9%)	24.97 (49.9%)

Tabela 5.1.1.1 – Resultados do Cenário 1⁵

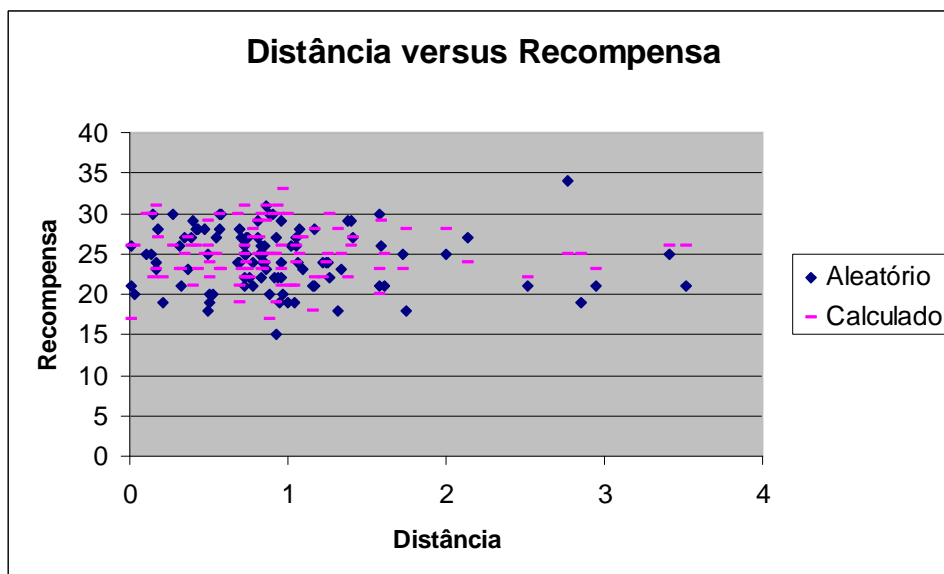


Figura 5.1.1.2 – Distância versus Recompensa para seqüências de 10 observações

⁵ Dado que se trata de seqüências de 50 observações, a recompensa máxima seria de 50.

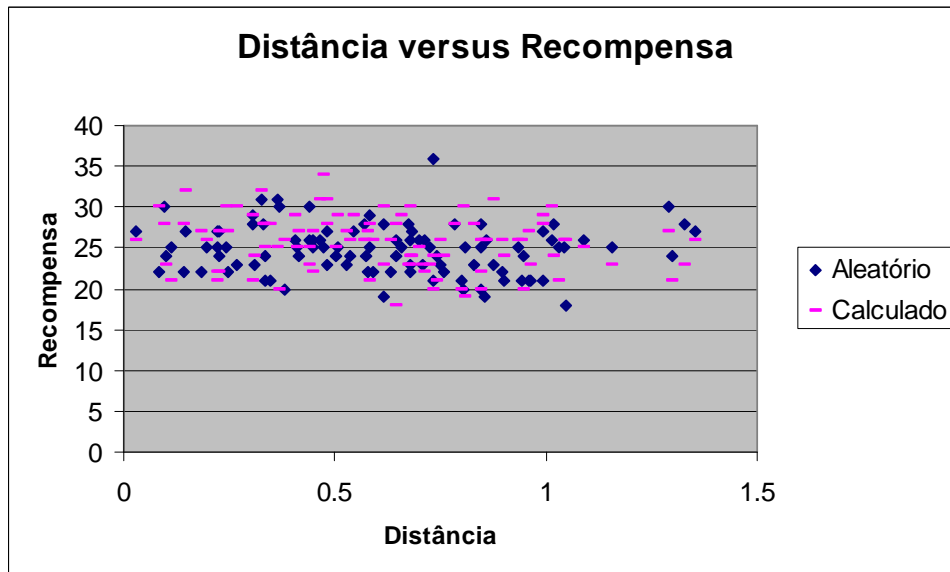


Figura 5.1.1.3 – Distância *versus* Recompensa para seqüências de 20 observações

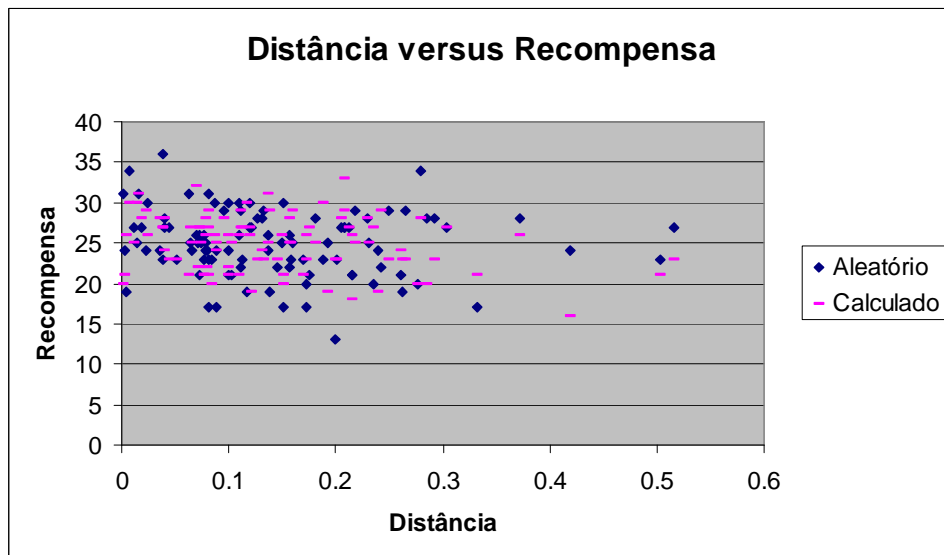


Figura 5.1.1.4 – Distância *versus* Recompensa para seqüências de 50 observações

Pela observação da tabela 5.1.1.1, percebe-se uma vantagem apenas marginal fornecida pelo o método. Ademais, pode-se ver que o aumento de observações reduz a já pequena margem. Já a análise da distância entre os modelos treinados para os modelos reais mostra pouca relação entre a redução da distância (causada pelo melhor treino com seqüência maiores) e a recompensa obtida. Além disto, é possível reparar que, nos gráficos das figuras 5.1.1.2 a 5.1.1.4, não há aparente correlação entre estes dois indicadores, em todos os casos apresentando-se como uma dispersão comum.

Ainda neste cenário, simulamos um aumento de conhecimento do jogador j_2 pelo fornecimento, no Modelo Oculto de Markov treinado, dos valores da matriz B, ou seja, a freqüência com que cada perfil produz cada saque.

Número de Observações	Média da Distância	Recompensa Aleatória	Recompensa Calculada
10	0.0536	25.06 (50.1%)	24.79 (49.6%)
20	0.0471	25.01 (50.0%)	24.87 (49.7%)
50	0.0299	24.86 (49.7%)	25.25 (50.5%)

Tabela 5.1.1.2 – Resultados do Cenário 1 com aumento de informação

Mais uma vez, pela análise da tabela 5.1.1.2, não se pode ver vantagem estatisticamente significativa para qualquer das opções, embora se possa perceber a nítida redução da distância média, se comparada com os valores da tabela 5.1.1.1 anterior.

5.1.2 Cenário 2

Neste cenário estabelecemos um jogo com três estados ocultos e três observações. Um exemplo de tal jogo seria uma disputa de pênalti onde o jogador j_1 (cobrador) alternaria entre os perfis “Zagueiro”, “Meio-Campo” e “Atacante”, cada um dos quais com diferentes chances de cobrar a penalidade no canto oposto (em relação ao pé que desferiu o chute), no centro e no mesmo canto. Mais uma vez, a frequência, arbitrariamente fornecida ao modelo real, de mudança entre os perfis, não possui tendências, e há apenas uma matriz de recompensa para todo o jogo, vista na figura abaixo (Tabela 5.1.2.1), que possui equilíbrio de Nash na estratégia mista (1/3, 1/3, 1/3) para ambos os jogadores, ou seja, ambos devem escolher as três opções com mesma frequência.

Jogador Cobrador	Jogador Goleiro		
	Saltar para o mesmo lado	Ficar no centro	Saltar para o lado oposto
Cobrança fechada	1	0	0
Cobrança no meio	0	1	0
Cobrança aberta	0	0	1

Tabela 5.1.2.1 – Matriz de recompensas do jogo com três observações

Desta forma, a estratégia de comparação para o jogador j_2 (goleiro) consiste em, como no cenário anterior, um lance aleatório entre as três opções. Um diagrama para o jogo pode ser visto na figura abaixo, 5.1.2.1:

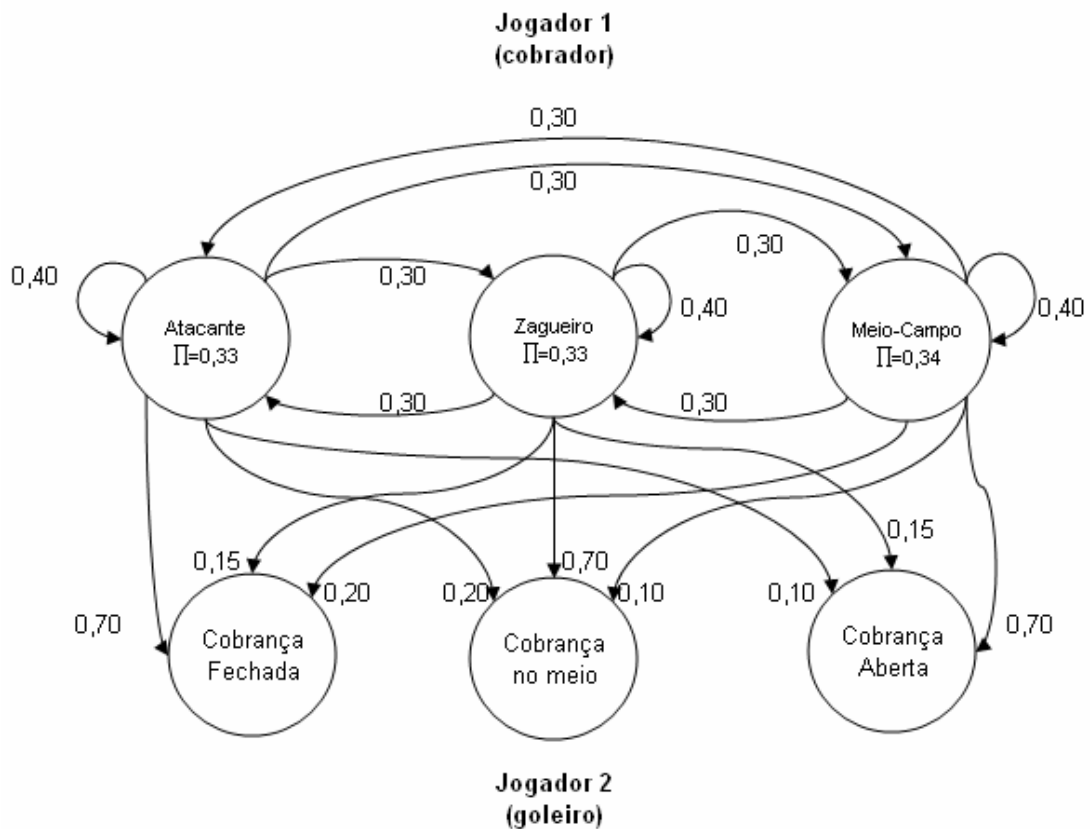


Figura 5.1.2.1 – Modelo do Cenário 2

Neste cenário, o aumento de observações trouxe dificuldades na medida em que as chances de gerar uma seqüência de 50 observações eram calculadas em valores da ordem de 10^{-20} , o que fez com que algumas estruturas das bibliotecas utilizadas não se comportassem da forma esperada. Assim, reduzimos o cenário com o maior número de observações de 50 para 30. Os resultados seguem na tabela abaixo:

Número de Observações	Média da Distância	Recompensa Aleatória	Recompensa Calculada
jogador 2 sem informações sobre os perfis			
10	1.6653	16.76 (33.5%)	17.24 (34.5%)
20	0.7287	17.49 (35,0%)	17.3 (34.6%)
30	0.4675	16.97 (33.9%)	16.91 (33.8%)
jogador 2 com informações sobre os perfis			
10	0.2045	17.31 (34.6%)	17.19 (34.4%)
20	0.1334	17.02 (34.0%)	16.97 (33.9%)
30	0.1127	16.53 (33.1%)	16.82 (33.6%)

Tabela 5.1.2.2 – Resultados do Cenário 2

Mais uma vez, o método proposto não se verificou superior a uma escolha aleatória. Incidentemente, as mesmas observações realizadas no cenário 1 fazem-se pertinentes aqui: o aumento no número de observações não parece influenciar nos

resultados, e o aumento de informação por parte do jogador j_2 igualmente não produz efeitos.

5.1.3 Cenário 3

Neste cenário retornamos ao jogo de tênis, com três estados ocultos e duas observações, mas introduzimos a modificação sugerida na introdução deste trabalho e desenvolvida no capítulo 4: variações estratégicas do mesmo jogo. Neste cenário, cada perfil que serve de opção ao jogador j_1 (sacador) possui uma matriz de recompensa que é aplicada, dado as escolhas de ambos os jogadores, para acumular os resultados. Mantivemos a distribuição igual entre os perfis, entretanto, e a estratégia de comparação mantém-se como escolher aleatoriamente entre os dois lados. O modelo do jogo é exatamente igual ao apresentado no cenário 1, e pode ser visualizado na já apresentada figura 5.1.1.1.

Abaixo, seguem as matrizes de recompensa arbitradas para este cenário, lembrando que o valor da recompensa está expresso para o recebedor, podendo o valor da recompensa do sacador ser obtido como o complemento de 1, que neste caso significa a chance do recebedor retornar a bola adequadamente:

Perfil Agressivo	Jogador Recebedor	
Jogador Sacador	Antecipar Saque Aberto	Antecipar Saque no Centro
Saque Aberto	0,90	0,10
Saque no Centro	0,05	0,80

Tabela 5.1.3.1 – Matriz de Recompensas para o perfil Agressivo do Cenário 3

Perfil Moderado	Jogador Recebedor	
Jogador Sacador	Antecipar Saque Aberto	Antecipar Saque no Centro
Saque Aberto	0,85	0,15
Saque no Centro	0,15	0,85

Tabela 5.1.3.2 – Matriz de Recompensas para o perfil Moderado do Cenário 3

Perfil Defensivo	Jogador Recebedor	
Jogador Sacador	Antecipar Saque Aberto	Antecipar Saque no Centro
Saque Aberto	0,80	0,20
Saque no Centro	0,25	0,90

Tabela 5.1.3.3 – Matriz de Recompensas para o perfil Defensivo do Cenário 3

As matrizes de recompensa acima fornecem equilíbrios de Nash não muito distantes entre si, o que deve ser considerado na análise dos resultados. Os equilíbrios de

Nash destas variações estratégicas podem ser vistos, para o jogador recebedor, na tabela abaixo:

Equilíbrios	Jogador Recebedor	
	Antecipar Saque Aberto	Antecipar Saque no Centro
Perfil Agressivo	0,452	0,548
Perfil Moderado	0,500	0,500
Perfil Defensivo	0,560	0,440

Tabela 5.1.3.4 – Equilíbrios de Nash para as variações estratégicas do Cenário 3

Neste cenário apresentamos duas novas medições, com o objetivo de verificar a eficácia da premissa estabelecida no capítulo 4. A primeira aplica o mesmo método, porém utiliza o estado oculto real (que efetivamente produziu a observação) para aplicar as fórmulas da seção 4.2. A segunda compara o estado oculto obtido pela aplicação do algoritmo de Viterbi com o estado oculto real, calculando a média de acertos. Os resultados podem ser vistos abaixo, na tabela 5.1.3.5:

Número de Observações	Recompensa Aleatória	Recompensa Calculada	Recompensa Calculada(*)	Média de Acerto Premissa
10	22.06 (44,1%)	22.80 (45,6%)	23.19 (46,4%)	33.74%
20	22.02 (44,0%)	21.54 (43,1%)	22.16 (44,3%)	32.32%
50	21.63 (43,3%)	21.20 (42,4%)	21.71 (43,4%)	33.42%

Tabela 5.1.3.5 – Resultados do Cenário 3

(*) calculada utilizando o estado oculto real

Nenhum ganho significativo foi percebido em comparação com a escolha aleatória, porém duas importantes observações podem ser realizadas: primeiro, que a recompensa média calculada utilizando-se o estado oculto real, ao invés do estado oculto obtido pelo algoritmo de Viterbi, demonstra algum ganho. Segundo, que o desempenho da premissa é, sob todos os aspectos, nulo, conforme se pode ver pela coluna “Média de Acerto da Premissa”, equiparando-se a escolher aleatoriamente um estado para ser considerado o estado atual.

5.1.4 Cenário 4

Em mais um cenário do jogo de tênis, com três estados ocultos e três observações, utilizamos as variações estratégicas, mas desta vez desequilibramos a distribuição das chances com que o jogador sacador escolhe os perfis, fazendo-o “preferir” um dos perfis, no caso específico, o perfil Agressivo. O modelo pode ser visto na figura 5.1.4.1, abaixo:

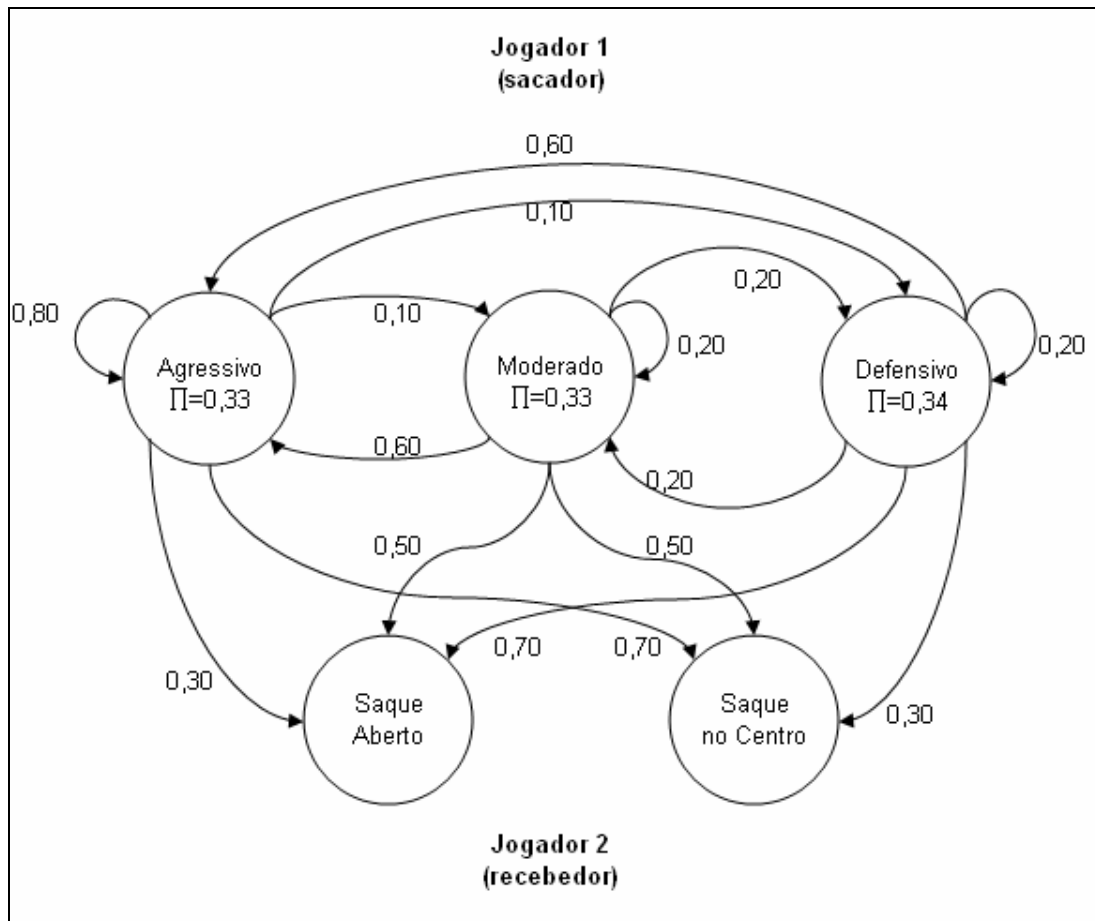


Figura 5.1.4.1 – Modelo do Cenário 4

Neste cenário, a produção de observações de saque no centro é maior que a de saque aberto, uma vez que esta é a tendência do perfil agressivo, e que este perfil foi por nós privilegiado. Desta forma, a estratégia de comparação deve ser contra um jogador que, percebendo a tendência do jogador 1, e na ausência de informações sobre os equilíbrios, intua uma matriz de recompensas simples, como no cenário 1, e jogue sempre para o lado tendencioso, maximizando sua recompensa. As matrizes de recompensa foram mantidas iguais às apresentadas no cenário anterior. O resultado deste cenário segue abaixo:

Número de Observações	Recompensa Aleatória	Recompensa Calculada	Recompensa Calculada(*)	Média de Acerto Premissa
10	22.95 (45,9%)	24.31 (48,6%)	24.35 (48,7%)	34.56%
20	22.34 (44,7%)	24.09 (48,2%)	24.25 (48,5%)	35.32%
50	22.57 (45,1%)	24.04 (48,1%)	24.08 (48,2%)	33.46%

Tabela 5.1.4.1 – Resultados do Cenário 4

(*) calculada utilizando o estado oculto real

Desta feita, o método apresentou melhor desempenho, ainda que não efetivamente significativo. Pode-se, ainda, ver as mesmas tendências, quando à eficácia da premissa, do cenário anterior. Para sublinhar isto, produzimos uma distribuição das recompensas

pelo percentual de acerto da premissa, para os indicadores calculado e aleatório, com suas respectivas linhas de tendências, que pode ser vista na figura 5.4.1.2 abaixo:

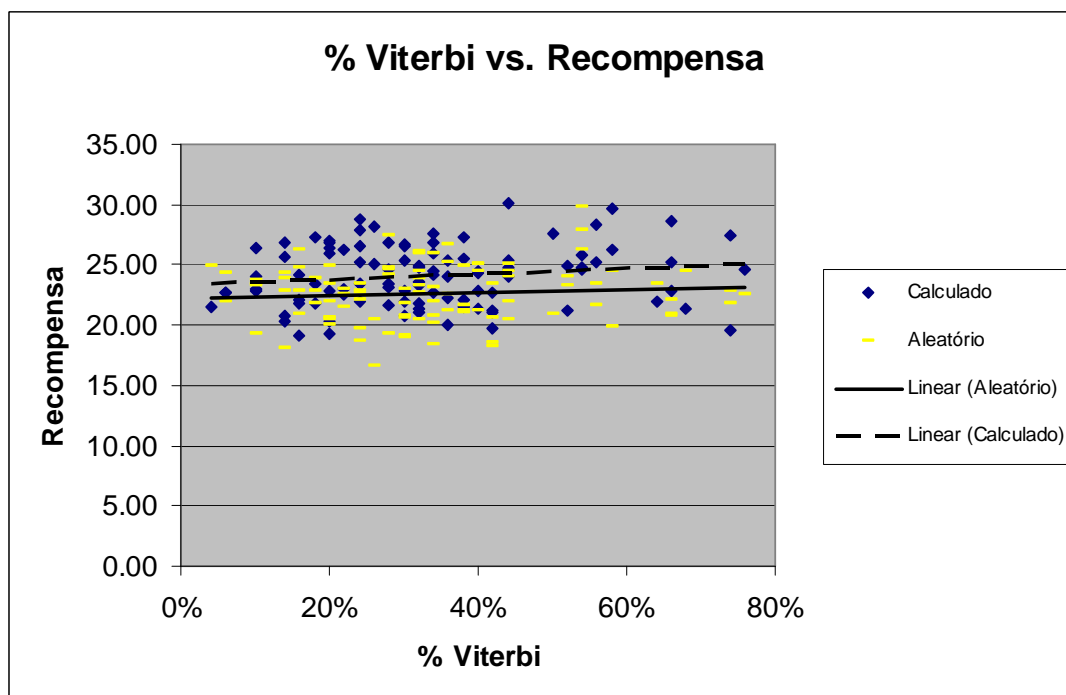


Figura 5.1.4.2 – Tendência da recompensa para a melhora da premissa

Executamos o cenário, ainda, com conjuntos de treinamento contendo diferentes quantidades de seqüências, com o objetivo de avaliar se a sobrecarga de seqüências estaria produzindo um efeito de *overtraining* nos Modelos Ocultos de Markov produzidos. Os resultados estão abaixo:

Número de Sequências	Recompensa Aleatória	Recompensa Calculada	Recompensa Calculada(*)	Média de Acerto Premissa
10	4.39 (43,9%)	4.79 (47,9%)	4.88 (48,8%)	34.56%
25	11.08 (44,3%)	12.05 (48,2%)	12.23 (48,9%)	35.32%
50	22.95 (45,9%)	24.31 (48,6%)	24.35 (48,7%)	34.56%

Tabela 5.1.4.2 – Resultados do Cenário 4 com variações no treinamento

(*) calculada utilizando o estado oculto real

Na verificação dos valores acima não se pode perceber tal ocorrência de *overtraining*, sendo possível perceber até mesmo um efeito no sentido contrário: o treinamento parece melhorar com o aumento de número de seqüência, ainda que timidamente.

No capítulo seguinte, exibiremos conclusões acerca dos cenários executados, e apontaremos direções para novas pesquisas, a partir do aqui exposto.

Capítulo 6

Conclusão e Trabalhos Futuros

Nesta dissertação discutimos possibilidades de aplicação das ferramentas dos Modelos Ocultos de Markov em Jogos de Estratégia Mista. Para tal, definimos dois tipos de jogos de estratégia mista repetidos entre dois jogadores, com características específicas, e avaliamos as seguintes questões:

1. É possível, a partir de informações dos jogos anteriormente disputados, criar um Modelo Oculto de Markov que represente tais jogos?
2. De que forma tal modelo e as ferramentas relacionadas, podem ajudar um jogador a escolher seu próximo lance?
3. Qual é o desempenho deste método em comparação com uma escolha aleatória deste mesmo jogador?

Apresentamos nos capítulos 4 e 5 um método para obter tais respostas, e após avaliação dos diversos cenários, dos quais os mais relevantes foram apresentados no capítulo 5, chegamos às conclusões abaixo expostas.

Em primeiro lugar, acreditamos ter ficado claro que a resposta à primeira questão é afirmativa, seguindo as diretrizes traçadas na seção 4.1 deste trabalho.

A resposta à segunda questão ainda é afirmativa, de acordo com a seção 4.2 da dissertação, porém não sem ressalvas. Para desenvolver o método aqui apresentado, utilizamos uma premissa que mostrou ao mesmo tempo insuficiente e nevrálgica para o sucesso da aplicação, sendo ponto focal para melhorias que novas pesquisas poderão trazer. O acima afirmado baseia-se em observações dos cenários 3 e 4 expostos no capítulo 5, onde foi constatado que a escolha do último estado oculto fornecido pelo algoritmo de Viterbi não é mais eficaz que uma escolha aleatória de um estado, e ainda que a recompensa obtida com a aplicação do método cresce com a utilização de um estado mais preciso.

Finalmente, quanto à terceira questão, podemos afirmar que o desempenho do método, segundo exposto no capítulo anterior, não se mostrou superior a uma escolha. Entretanto, a análise dos cenários executados indicou igualmente espaço para melhora

destes resultados, através da melhora do estado oculto escolhido (como já exposto no parágrafo anterior) e por melhora no método de treinamento do Modelo Oculto de Markov utilizado.

O método utilizado para avaliar a proximidade entre o modelo treinado e o utilizado para gerar o conjunto de treinamento, a distância Kullback-Leibler (apresentada na seção 3.3), como nos ensina DUGAD e DESAI (1996), nem sempre representa bem as diferenças entre os modelos. A isto se deve adicionar que, no jogo com variações estratégicas, as matrizes de recompensa não fazem parte do cálculo de tal distância, embora possam (como detalhado na seção 4.2) influenciar muito a confecção da tabela 4.1, que é fundamental para a eficiência do modelo.

De uma forma geral, o método mostrou melhor desempenho quanto mais erráticas são as escolhas do jogador 1, tendendo para uma ou outra estratégia, detendo-se em algumas destas. Isto se deve ao fato de que um Modelo Oculto de Markov captura a estrutura das tendências do jogo desde que as mesmas existam. Os cenários apresentados no capítulo 5, especialmente os três primeiros, não possuem uma estrutura que difira muito de uma escolha aleatória. No último cenário apresentado, com o estabelecimento arbitrário de uma tendência, o método mostrou ser superior a uma escolha aleatória.

Podemos destacar ainda que a pesquisa deste trabalho apresentou um volume elevado de variáveis cujo controle delicado induziu à redução do número de cenários, por limitações temporais. O estudo mais amplo de todos estes fatores certamente frutificaria em melhora no método ou na aplicação do mesmo.

Diante do acima exposto, podemos indicar como principal direção para trabalhos futuros, um estudo mais detalhado da questão “a partir de uma seqüência de observações, qual é a chance $p(q)$ de o modelo encontrar-se em cada estado oculto q ?”, uma vez que a escolha apresentada na seção 4.2 mostrou-se ineficaz.

Outra possibilidade seria um estudo aprofundado dos métodos de treinamento do Modelo Oculto de Markov, avaliando se, para a presente aplicação, a escolha de Algoritmos Genéticos é efetivamente acertada, uma vez que aqui foi utilizada unicamente pela facilidade na manipulação das variáveis ajustadas, em relação ao mais tradicional Baum-Welch. Caso assim se conclua, mais cuidado no ajuste dos vários parâmetros do algoritmo pode trazer ganhos ao método.

Finalmente, considerando os trabalhos já realizados de aplicações práticas da Teoria dos Jogos, alguns dos quais citados neste documento, se fazem disponíveis possibilidades de trabalhos futuros no estudo de casos particulares de aplicação do modelo nas diversas áreas em que hoje os jogos de estratégia mista são aplicados.

Apêndice A – Resultados dos Cenários

Os cenários abaixo estão agrupados por jogos idênticos. Note que o que nesta seção é chamado de cenário difere do que, ao longo da dissertação, foi assim tratado.

A.1 Cenários de 1 a 6

Número de Observações: 2
Número de Sequências: 100
Gerações do Algoritmo Genético: 100
Número de Iterações: 100
Características do cenário: Matriz única de recompensas, sem tendências entre perfis.

Identificação do Cenário	Tamanho da Sequência	Jogador possui conhecimento?	Média da distância Kullback-Leibler	Média Estratégia Aleatória	Média Estratégia Calculada
1	10	Não	1,1637	48,64%	50,98%
2	20	Não	0,5799	50,96%	51,01%
3	50	Não	0,1690	49,51%	50,00%
4	10	Sim	0,0536	50,12%	49,59%
5	20	Sim	0,0471	50,01%	49,75%
6	50	Sim	0,0299	49,72%	50,49%

A.1 Cenários de 7 a 12

Número de Observações: 2
Número de Sequências: 100
Gerações do Algoritmo Genético: 100
Número de Iterações: 100
Características dos cenários: Matriz única de recompensas, tendência a permanecer no perfil agressivo.

Identificação dos Cenários	Tamanho da Sequência	Jogador possui conhecimento?	Média da distância Kullback-Leibler	Média Estratégia Aleatória	Média Estratégia Calculada
7	10	Não	1,1087	50,14%	56,64%
8	20	Não	0,4914	49,32%	55,40%
9	50	Não	0,1517	49,80%	55,98%
10	10	Sim	0,0536	50,12%	49,59%
11	20	Sim	0,0471	50,01%	49,75%
12	50	Sim	0,0299	49,72%	50,49%

A.3 Cenários de 13 a 21

Número de Observações: 2
Gerações do Algoritmo Genético: 100
Número de Iterações: 100
Características do cenário: Utilização de variações estratégicas, com tendência ao perfil agressivo, jogador 2 sem conhecimento das variações.

Tabelas de Recompensa				
Perfil Agressivo				
	Recebedor		Eq. Nash	
Sacador	Aberto	Fechado	Sacador	Recebedor
Aberto	0,90	0,05	0,475	0,463
Fechado	0,03	0,80	0,525	0,537
Perfil Moderado				
	Recebedor		Eq. Nash	
Sacador	Aberto	Fechado	Sacador	Recebedor
Aberto	0,85	0,15	0,500	0,500
Fechado	0,15	0,85	0,500	0,500
Perfil Conservador				
	Recebedor		Eq. Nash	
Sacador	Aberto	Fechado	Sacador	Recebedor
Aberto	0,75	0,20	0,560	0,560
Fechado	0,20	0,90	0,440	0,440

Identificação do Cenário	Tamanho da Sequência	Número de Sequências	Média Estratégia Aleatória	Média Estratégia Calculada	Média Estratégia Calculada(*)
13	10	50	47,32%	43,84%	43,46%
14	20	50	48,31%	42,96%	42,79%
15	50	50	48,64%	45,33%	44,88%
16	10	25	48,30%	44,92%	44,20%
17	20	25	47,10%	46,10%	45,97%
18	50	25	47,95%	44,07%	42,31%
19	10	10	48,92%	45,77%	42,88%
20	20	10	49,56%	44,82%	44,38%
21	50	10	48,08%	43,53%	42,86%

(*) Utilizando o estado oculto real

A.3 Cenários de 22 a 27

Número de Observações: 2
Número de Sequências: 100
Gerações do Algoritmo Genético: 100
Número de Iterações: 100
Características do cenário: Utilização de variações estratégicas, sem tendência a escolher o perfil

Tabelas de Recompensa				
Perfil Agressivo				
	Recebedor		Eq. Nash	
Sacador	Aberto	Fechado	Sacador	Recebedor
Aberto	0,90	0,08	0,311	0,269
Fechado	0,03	0,40	0,689	0,731
Perfil Moderado				
	Recebedor		Eq. Nash	
Sacador	Aberto	Fechado	Sacador	Recebedor
Aberto	0,85	0,15	0,500	0,500
Fechado	0,15	0,85	0,500	0,500
Perfil Conservador				
	Recebedor		Eq. Nash	
Sacador	Aberto	Fechado	Sacador	Recebedor
Aberto	0,73	0,30	0,368	0,294
Fechado	0,25	0,50	0,632	0,706

Identificação do Cenário	Tamanho da Sequência	Jogador possui conhecimento?	Média Estratégia Aleatória	Média Estratégia Calculada	Média Estratégia Calculada(*)
22	10	Sim	44,13%	38,39%	37,61%
23	20	Sim	44,04%	37,60%	35,09%
24	50	Sim	43,27%	36,32%	34,39%
25	10	Não	42,82%	26,53%	26,53%
26	20	Não	43,44%	26,64%	26,64%
27	50	Não	43,45%	26,19%	26,19%

(*) Utilizando o estado oculto real

Apêndice B – Exemplo de Código-fonte

B.1 Estratégia de Tênis

O código-fonte abaixo, escrito em Java, utiliza as bibliotecas gratuitas JGAP e JAHMM de implementações dos algoritmos de Algoritmos Genéticos e Modelos Ocultos de Markov, respectivamente, para implementar os conceitos descritos nesta dissertação.

B.1.1 Classe estrategiaTennis.java

```
import java.util.ArrayList;
import java.util.List;
import java.util.Random;

import be.ac.ulg.montefiore.run.jahmm.*;
import be.ac.ulg.montefiore.run.jahmm.learn.BaumWelchLearner;
import be.ac.ulg.montefiore.run.jahmm.learn.KMeansLearner;
import be.ac.ulg.montefiore.run.jahmm.toolbox.KullbackLeiblerDistanceCalculator;
import be.ac.ulg.montefiore.run.jahmm.toolbox.MarkovGenerator;

import org.jgap.*;
import org.jgap.impl.*;

public class EstrategiaTennis {

    public static final int glNumEstadosHMM = 3; // Numero de estados ocultos
    public static final int glNumObservacoesHMM = 2; // Numero de observacoes da HMM
    public static final int glNumConjuntos = 50; // Quantidade de sequências geradas
    public static final int glMaxEvolucoes = 10; // numero de evolucoes do AG
    public static final int glIteracoes = 100; // numero total de iteracoes
    public static final boolean glVerbose = false; // habilita saida detalhada
    public static List<List<Integer>> glEstadosObs; // Matriz contendo os estados ocultos das
// observações
    public static List<List<ObservationInteger>> glObservacoes; // Matriz contendo vários conjuntos
// (linhas) de observações (colunas)
    public static final double[][][] matrizRecompensas = { { { 0.90, 0.05 }, { 0.03, 0.8 } },
// { { 0.85, 0.15 }, { 0.85, 0.15 } },
// { { 0.75, 0.20 }, { 0.20, 0.9 } } };

    public static void main(String[] args) throws InvalidConfigurationException {

        Hmm <ObservationInteger > hmmGeracaoObservacoes; // esta hmm é parametrizada para criar
// observações pertinentes
        Hmm <ObservationInteger > hmmTreinada; // esta hmm será treinada por Algoritmos
// Genéticos
        KullbackLeiblerDistanceCalculator klDist; // Algoritmo para calculo da distancia entre HMM
        int[] tamSequencia = {10, 20, 50}; // Tamanho das sequencias
        double[][] estrategia; // estrategia do jogador para cada estado oculto
```

```

double[] acumulado; // resultado obtido pelo jogador

klDist = new KullbackLeiblerDistanceCalculator();

// cria uma hmm de dois estados com observação de inteiros (0 = cara, 1 = coroa)
hmmGeracaoObservacoes = criaHmmEstrategiaTenis(0.33, 0.33, // matriz pi
                                                0.3, 0.5, 0.7, // matriz B
                                                0.8, 0.1, 0.6, 0.2, 0.6, 0.2); // matriz A

glEstadosObs = new ArrayList<List<Integer>>();
glObservacoes = new ArrayList<List<ObservationInteger>>();

// imprime recompensas
if (glVerbose) {
    for (int i = 0; i < glNumEstadosHMM; i++) {
        System.out.println("\nPerfil no." + i + ":");
        System.out.print("j1 \\ j2");
        for (int j = 0; j < glNumObservacoesHMM; j++) System.out.print(" " + j + " ");
        System.out.print("\n----- ");
        for (int j = 0; j < glNumObservacoesHMM; j++) System.out.print("----- ");
        for (int j = 0; j < glNumObservacoesHMM; j++) {
            System.out.print("\n " + j + " | ");
            for (int k = 0; k < glNumObservacoesHMM; k++)
                System.out.print(matrizRecompensas[i][j][k] + " ");
        }
    }
}

for (int j = 0; j < tamSequencia.length; j++) {
    System.out.println("\n\n" + tamSequencia[j] + " observacoes por sequencia ... \n\n");

    // itera para gerar espaço distancia x resultado
    for (int i = 0; i < glIteracoes; i++) {
        // Gera as observacoes
        glEstadosObs.clear();
        glObservacoes.clear();

        GeraConjuntoObservacoes(hmmGeracaoObservacoes, tamSequencia[j]);

        // Utiliza um algoritmo genético para treinar a HMM
        hmmTreinada = TreinaHmmAG();

        if (glVerbose) {
            System.out.print("\n\n ----- Comparação de treinamentos ----- \n\n");
            System.out.print(" HMM original.....: " +
                ProbabilidadeMediaHMM(hmmGeracaoObservacoes, glObservacoes) + "%\n");
            System.out.print(" HMM treinada.....: " + ProbabilidadeMediaHMM(hmmTreinada,
                glObservacoes) + "%\n");
            System.out.print(" Distância.....: " + klDist.distance(hmmGeracaoObservacoes,
                hmmTreinada));
            System.out.print("\n ----- \n\n");
        }

        // calcula as estrategias do jogador
        estrategia = CalculaEstrategiaJogador(hmmTreinada);

        // acumula resultado obtido
        acumulado = AplicaEstrategiaJogador(hmmTreinada, estrategia, tamSequencia[j]);

        System.out.print("\n" + i + ";" + klDist.distance(hmmGeracaoObservacoes, hmmTreinada));
    }
}

```

```

        for (int k = 0; k < acumulado.length; k++) System.out.print("; " + acumulado[k]);

        // libera memoria
        estrategia = null;
        hmmTreinada = null;
    }
}

//*****
// Esta função cria o conjunto de observações
//*****
public static void GeraConjuntoObservacoes(
        Hmm <ObservationInteger > hmmGeracaoObservacoes,
        int tamSequencia) {
    MarkovGenerator <ObservationInteger> mGenerator; // MarkovGenerator é responsável por gerar
                                                    // observações a partir de uma HMM

    List<ObservationInteger> sequenciaObservacoes;
    List<Integer> sequenciaEstados;

    mGenerator = new MarkovGenerator<ObservationInteger>(hmmGeracaoObservacoes);

    // imprime HMM utilizada para a geração das observações
    if (glVerbose) {
        System.out.print(" ---- HMM utilizada para gerar as observações ----\n ");
        System.out.print(hmmGeracaoObservacoes.toString());
        System.out.print(" ---- -----\n ");
    }

    if (glVerbose) {
        System.out.print(" ---- Geração de Observações ----\n");
        System.out.print(" Número de conjuntos.....: " + glNumConjuntos + "\n");
        System.out.print(" Observações por conjunto: " + tamSequencia + "\n ");
    }

    for (int i = 0; i < glNumConjuntos; i++) {
        sequenciaObservacoes = new ArrayList<ObservationInteger>();
        sequenciaEstados = new ArrayList<Integer>();

        for (int j = 0; j < tamSequencia; j++) {
            sequenciaEstados.add(mGenerator.stateNb());
            sequenciaObservacoes.add(mGenerator.observation());
        }
        mGenerator.newSequence();
        glObservacoes.add(sequenciaObservacoes);
        glEstadosObs.add(sequenciaEstados);
        if (glVerbose) System.out.print(glObservacoes.get(i) + "-" + glEstadosObs.get(i) + " \n ");
    }
    if (glVerbose) System.out.print(" ---- -----\n" );
}

// *****
// Esta função treina uma HMM por Algoritmo Genético
// *****
public static Hmm <ObservationInteger > TreinaHmmAG() throws InvalidConfigurationException {
    Hmm <ObservationInteger > hmm; // esta hmm será treinada por Algoritmos Genéticos
    Genotype populacao; // populacao de soluções
}

```

```

Configuration conf;          // parametros do GA
FitnessFunction myFunc;     // função de avaliação
Gene[] geneMoedaViciada;    // gene contendo os parâmetros da HMM
int numGenes = 9;          // Numero de genes do cromossomo

Configuration.reset();
conf = new DefaultConfiguration();          // utilizando configurações padrão
myFunc = new FuncaoFitnessMoedaViciada(glObservacoes); // definindo a função de avaliação
conf.setPreservFittestIndividual(true); // mantém o indivíduo mais adaptado na próxima geração
conf.setFitnessFunction(myFunc);

// criando o cromossomo
geneMoedaViciada = new Gene[numGenes];
for (int i = 0; i < numGenes; i++)
    geneMoedaViciada[i] = new DoubleGene(conf, 0, 1);

conf.setSampleChromosome(new Chromosome(conf, geneMoedaViciada));
conf.setPopulationSize(80);
populacao = Genotype.randomInitialGenotype(conf); // inicializa com genótipos aleatórios

// evolui a população
if (glVerbose) System.out.print(" ---- Treinando HMM ----\n iteração Probabilidade Média\n ---
-----\n");

long startTime = System.currentTimeMillis();
for (int i = 0; i < glMaxEvolucoes * 10; i++) {
    populacao.evolve();
    if ((i % glMaxEvolucoes) == 0 && glVerbose) {
        System.out.println("    " + i + "          " +
            (populacao.getFittestChromosome().getFitnessValue() - 1) + "%");
    }
}
long endTime = System.currentTimeMillis();
if (glVerbose) System.out.println("Tempo evolução: " + (endTime - startTime) * 1000 + "s");

geneMoedaViciada = populacao.getFittestChromosome().getGenes();
hmm = criaHmmEstrategiaTennis(0.33, 0.33,
    ((DoubleGene)geneMoedaViciada[0]).doubleValue() ,
    ((DoubleGene)geneMoedaViciada[1]).doubleValue() ,
    ((DoubleGene)geneMoedaViciada[2]).doubleValue() ,
    ((DoubleGene)geneMoedaViciada[3]).doubleValue() ,
    ((DoubleGene)geneMoedaViciada[4]).doubleValue() ,
    ((DoubleGene)geneMoedaViciada[5]).doubleValue() ,
    ((DoubleGene)geneMoedaViciada[6]).doubleValue() ,
    ((DoubleGene)geneMoedaViciada[7]).doubleValue() ,
    ((DoubleGene)geneMoedaViciada[8]).doubleValue() );

// Imprime HMM final
if (glVerbose) System.out.print(" ---- HMM treinada por GA ----\n" + hmm.toString() + "\n ----
-----\n");

conf = null;
myFunc = null;
geneMoedaViciada = null;

// retorna HMM
return hmm;

```

```

}

// *****
// Esta função calcula a estratégia a ser jogada para cada estado oculto
// *****

public static double[][] CalculaEstrategiaJogador(
    Hmm <ObservationInteger > hmm) {
    ObservationInteger e10;
    double[][] lance;

    if (glVerbose) {
        System.out.print("\n ----- Calculando lance seguinte -----");
        System.out.print("\n Estado oculto   Estrategia ");
        System.out.print("\n ----- \n");
    }

    // calcula, de acordo com a dissertação, qual o próximo lance
    // dado que o HMM esta em p, para cada observacao o, somatório(i=1 a e) de (Api.Bio)
    //
    e10 = new ObservationInteger(0);
    lance = new double[glNumEstadosHMM][glNumObservacoesHMM];

    for (int i = 0; i < glNumEstadosHMM; i++)
        for (int j = 0; j < glNumObservacoesHMM; j++)
            lance[i][j] = 0;

    for (int i = 0; i < glNumEstadosHMM; i++) {
        for (int j = 0; j < glNumObservacoesHMM; j++)
            lance[i][0] += hmm.getAij(i, j) * hmm.getOpdf(j).probability(e10);

        lance[i][1] = (1 - lance[i][0]);
        if (glVerbose) System.out.print("      " + i + "      (" + lance[i][0] +
            ", " + lance[i][1] + ")\n");
    }

    e10 = null;
    return lance;
}

// *****
// Esta função aplica viterbi para determinar o estado oculto atual mais provavel
// e aplica a estratégia do jogador, acumulando o resultado esperado
// *****

public static double[] AplicaEstrategiaJogador(
    Hmm <ObservationInteger > hmm,
    double[][] estrategia,
    int tamSequencia) {

    double[] acumulado = {0, 0, 0, 0, 0, 0, 0}; // acumulador da recompensa do jogador
    Random gerador; // Gerador de numeros aleatorios
    ViterbiCalculator viterbi; // Calculadora do algoritmo de Viterbi
    ObservationInteger observacaoReal; // variavel que mantem a constante 0 para um
    // ObservationInteger

    int ultPerfilReal; // estado oculto real antes da ultima observacao
    int ultPerfilViterbi; // estado oculto antes da ultima observacao segundo Viterbi
    int ultObservacao; // indice da ultima observacao da sequencia

    double maiorChanceCalculada;
    int lanceCalculado;
    int lanceCalculadoSe;

```

```

int lanceAleatorio;

if (glVerbose) System.out.print("\n\n ----- Aplicando Viterbi ----- \n");
observacaoReal = new ObservationInteger(0);
gerador = new Random();

for (int i = 0; i < glNumConjuntos; i++) {
    ultObservacao = tamSequencia - 1;

    // retira última observação da sequencia
    observacaoReal = glObservacoes.get(i).get(ultObservacao);
    glObservacoes.get(i).remove(ultObservacao);
    ultObservacao--;

    // obtem o último perfil realmente jogado na sequencia
    ultPerfilReal = glEstadosObs.get(i).get(ultObservacao);

    // cacula algoritmo de viterbi para a sequencia reduzida e obtem o ultimo estado
    viterbi = new ViterbiCalculator(glObservacoes.get(i), hmm);
    ultPerfilViterbi = viterbi.stateSequence()[ultObservacao];

    // verifica os lances escolhidos
    lanceCalculado = 0;
    maiorChanceCalculada = 0;
    for (int j = 0; j < estrategia[0].length; j++) {
        if (estrategia[ultPerfilViterbi][j] > maiorChanceCalculada) {
            maiorChanceCalculada = estrategia[ultPerfilViterbi][j];
            lanceCalculado = j;
        }
    }

    lanceCalculadoSe = 0;
    maiorChanceCalculada = 0;
    for (int j = 0; j < estrategia[0].length; j++) {
        if (estrategia[ultPerfilReal][j] > maiorChanceCalculada) {
            maiorChanceCalculada = estrategia[ultPerfilViterbi][j];
            lanceCalculadoSe = j;
        }
    }

    lanceAleatorio = 0;
    //if (gerador.nextDouble() > 0.5) lanceAleatorio = 1;
    if (gerador.nextDouble() > 1) lanceAleatorio = 1;
    // acumula resultados dos métodos

    // inteiros para jogo com recompensas fixas
    if (lanceAleatorio == observacaoReal.value) acumulado[0]++;
    if (lanceCalculado == observacaoReal.value) acumulado[1]++;
    if (lanceCalculadoSe == observacaoReal.value) acumulado[2]++;

    // inteiros para jogo com recompensas variaveis
    acumulado[3] += matrizRecompensas[ultPerfilReal][observacaoReal.value][lanceCalculadoSe];
    acumulado[4] += matrizRecompensas[ultPerfilReal][observacaoReal.value][lanceAleatorio];
    acumulado[5] += matrizRecompensas[ultPerfilReal][observacaoReal.value][lanceCalculado];

    // acumula eficiência da premissa
    if (ultPerfilReal == ultPerfilViterbi) acumulado[6]++;
    if (glVerbose) {

```



```

        System.out.print(glObservacoes.get(i) + ";" + observacaoReal.value + ";" +
            ultPerfilReal + ";" + ultPerfilViterbi + ";" + lanceAleatorio + ";" +
            lanceCalculado + ";" + acumulado[4] + ";" + acumulado[5]+ "\n");
    }

    viterbi = null;
}

observacaoReal = null;
gerador = null;

return acumulado;
}

// *****
// Esta função calcula a probabilidade média da geração de uma sequencia de observações
// por uma HMM
// *****
public static double ProbabilidadeMediaHMM(Hmm<ObservationInteger> HMMTreinada,
    List<List<ObservationInteger>> Observacoes) {
    ForwardBackwardCalculator calculadora;
    double somaProbabilidades;

    somaProbabilidades = 0;
    for (int i = 0; i < Observacoes.size();i++) {
        calculadora = new ForwardBackwardCalculator(Observacoes.get(i), HMMTreinada);
        somaProbabilidades += calculadora.probability();
        calculadora = null;
    }

    return somaProbabilidades / Observacoes.size();
}

// *****
// esta função cria uma hmm de acordo com os parâmetros informados
// *****
public static Hmm <ObservationInteger > criaHmmEstrategiaTenis (
    double pi1, double pi2,
    double pB11, double pB21, double pB31,
    double pA11, double pA12, double pA21, double pA22, double pA31, double pA32) {
    Hmm <ObservationInteger > hmm; // esta será a hmm retornada

    // cria uma hmm de dois estados com observação de inteiros (0=saque aberto, 1 = saque fechado)
    hmm = new Hmm <ObservationInteger >(3, new OpdfIntegerFactory (2));

    // *****
    // estabelece os demais parametros da hmm
    // matriz Pi = parametros iniciais
    hmm.setPi (0, pi1);
    hmm.setPi (1, pi2);
    hmm.setPi (2, 1 - (pi1 + pi2));

    // matriz B = probabilidade de cada estado gerar as observações
    // fixa prob. de geracao da observacao, simulando o conhecimento de um jogador
    //pB11 = 0.30;
    //pB21 = 0.50;
    //pB31 = 0.7;
    hmm.setOpdf (0, new OpdfInteger (new double [] {pB11 , (1 - pB11)}));
}

```

```

hmm.setOpdf (1, new OpdfInteger (new double [] {pB21 , (1 - pB21)}));
hmm.setOpdf (2, new OpdfInteger (new double [] {pB31 , (1 - pB31)}));

// matriz A = probabilidades de transição de estados
hmm.setAij (0, 0, pA11);
hmm.setAij (0, 1, pA12);
hmm.setAij (0, 2, 1 - (pA11 + pA12));
hmm.setAij (1, 0, pA21);
hmm.setAij (1, 1, pA22);
hmm.setAij (1, 2, 1 - (pA21 + pA22));
hmm.setAij (2, 0, pA31);
hmm.setAij (2, 1, pA32);
hmm.setAij (2, 2, 1 - (pA31 + pA32));

return hmm;
}
}

```

B.1.2 Exemplo de saída da execução

```

Perfil no.0:
j1 \ j2    0    1
-----
  0 | 0.9  0.05
  1 | 0.03  0.8
Perfil no.1:
j1 \ j2    0    1
-----
  0 | 0.85  0.15
  1 | 0.85  0.15
Perfil no.2:
j1 \ j2    0    1
-----
  0 | 0.75  0.2
  1 | 0.2   0.9

10 observacoes por sequencia ...

---- HMM utilizada para gerar as observações ----
HMM with 3 state(s)

State 0
  Pi: 0.33
  Aij: 0,8 0,1 0,1
  Opdf: Integer distribution --- 0,3 0,7

State 1
  Pi: 0.33
  Aij: 0,6 0,2 0,2
  Opdf: Integer distribution --- 0,5 0,5

State 2
  Pi: 0.33999999999999997
  Aij: 0,6 0,2 0,2

```

Opdf: Integer distribution --- 0,7 0,3

---- Geração de Observações ----

Número de conjuntos.....: 50

Observações por conjunto: 10

[0, 0, 1, 1, 1, 1, 1, 1, 1, 1] - [2, 2, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 1, 0, 1, 1, 0, 0, 1, 1] - [2, 1, 0, 0, 0, 0, 2, 1, 2, 1]
[0, 1, 1, 0, 1, 1, 1, 1, 0, 1] - [0, 0, 0, 0, 0, 0, 0, 0, 2, 2]
[1, 1, 0, 1, 0, 0, 0, 1, 0, 1] - [2, 0, 0, 2, 1, 0, 0, 0, 2, 1]
[0, 1, 1, 1, 0, 1, 0, 1, 0, 1] - [1, 0, 0, 2, 0, 0, 2, 0, 0, 0]
[0, 1, 0, 1, 1, 0, 1, 0, 1, 0] - [0, 0, 0, 0, 0, 0, 0, 0, 0, 1]
[1, 1, 0, 1, 1, 0, 1, 1, 1, 1] - [0, 2, 0, 0, 0, 0, 0, 0, 0, 0]
[1, 1, 1, 0, 0, 1, 0, 0, 1, 1] - [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[1, 1, 1, 1, 0, 1, 1, 0, 1, 1] - [0, 0, 0, 0, 2, 0, 0, 0, 0, 0]
[0, 0, 0, 1, 1, 0, 0, 0, 0, 0] - [1, 2, 2, 1, 0, 2, 1, 2, 0, 2]
[0, 1, 1, 1, 1, 0, 0, 1, 1, 1] - [2, 1, 0, 0, 0, 0, 0, 0, 0, 0]
[1, 1, 0, 1, 1, 0, 1, 0, 0, 1] - [1, 0, 2, 1, 2, 2, 0, 0, 0, 0]
[0, 1, 1, 0, 0, 1, 1, 0, 1, 0] - [2, 0, 0, 0, 2, 0, 0, 0, 0, 1]
[0, 1, 1, 0, 1, 1, 1, 1, 1, 1] - [0, 0, 0, 1, 0, 0, 0, 2, 0, 0]
[1, 1, 1, 0, 1, 0, 1, 0, 1, 1] - [2, 2, 0, 1, 2, 2, 2, 0, 0, 0]
[1, 1, 1, 1, 1, 0, 1, 0, 1, 0] - [1, 0, 2, 0, 0, 2, 0, 0, 0, 2]
[1, 0, 1, 1, 1, 1, 1, 1, 0, 0] - [2, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[1, 1, 1, 0, 1, 1, 1, 0, 0, 0] - [0, 0, 0, 2, 1, 0, 0, 0, 0, 0]
[1, 1, 1, 1, 1, 1, 1, 1, 1, 0] - [0, 0, 2, 0, 0, 0, 1, 0, 0, 2]
[1, 1, 0, 1, 1, 0, 1, 1, 1, 0] - [1, 1, 1, 0, 0, 0, 0, 1, 2, 1]
[0, 1, 1, 0, 0, 1, 1, 0, 0, 1] - [0, 0, 1, 0, 0, 0, 0, 1, 0, 0]
[0, 0, 1, 1, 0, 0, 0, 0, 1, 1] - [2, 0, 0, 0, 1, 2, 2, 0, 0, 0]
[1, 0, 1, 1, 0, 1, 1, 0, 0, 1] - [0, 0, 0, 0, 0, 1, 1, 0, 1, 0]
[1, 0, 0, 0, 1, 0, 0, 1, 0, 0] - [1, 1, 2, 0, 0, 0, 0, 0, 0, 0]
[1, 1, 0, 0, 0, 0, 1, 1, 0, 1] - [0, 0, 0, 2, 1, 1, 0, 2, 0, 0]
[1, 1, 0, 1, 1, 1, 1, 1, 0, 1] - [0, 0, 0, 0, 0, 0, 2, 0, 0, 1]
[0, 0, 0, 1, 0, 1, 1, 1, 1, 0] - [2, 2, 0, 0, 1, 0, 1, 1, 0, 0]
[1, 0, 1, 1, 0, 1, 1, 0, 1, 1] - [1, 1, 0, 0, 0, 0, 0, 1, 0, 0]
[1, 1, 0, 1, 1, 1, 1, 1, 0, 1] - [0, 0, 0, 0, 0, 0, 0, 0, 1, 1]
[0, 1, 1, 1, 1, 0, 1, 1, 1, 0] - [1, 0, 0, 0, 2, 0, 0, 0, 0, 2]
[0, 1, 0, 1, 0, 1, 1, 0, 0, 1] - [0, 0, 0, 0, 0, 0, 2, 2, 0, 1]
[0, 0, 1, 1, 0, 1, 0, 0, 0, 1] - [2, 1, 0, 0, 2, 0, 0, 0, 0, 0]
[1, 0, 0, 1, 0, 1, 0, 1, 0, 1] - [0, 0, 0, 0, 1, 2, 2, 0, 0, 0]
[1, 1, 1, 1, 0, 0, 1, 0, 0, 1] - [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[1, 1, 0, 1, 1, 1, 1, 1, 0, 0] - [0, 2, 2, 0, 1, 0, 0, 1, 0, 0]
[0, 1, 0, 1, 1, 1, 1, 1, 1, 1] - [0, 2, 0, 0, 1, 0, 0, 0, 0, 1]
[1, 1, 1, 1, 1, 0, 1, 1, 1, 0] - [2, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[1, 0, 1, 1, 1, 0, 1, 1, 0, 0] - [2, 0, 0, 0, 0, 1, 0, 0, 0, 2]
[1, 1, 1, 0, 1, 0, 1, 0, 0, 1] - [2, 0, 1, 0, 1, 0, 0, 2, 0, 0]
[1, 1, 0, 1, 1, 0, 0, 1, 1, 0] - [0, 0, 1, 0, 0, 2, 2, 0, 0, 0]
[0, 1, 1, 1, 1, 1, 0, 1, 0, 1] - [0, 0, 0, 0, 0, 0, 0, 0, 1, 0]
[0, 0, 0, 1, 1, 1, 0, 0, 0, 1] - [2, 0, 2, 2, 0, 0, 1, 2, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 1, 1, 1] - [2, 0, 0, 0, 1, 0, 0, 0, 0, 0]
[0, 0, 0, 1, 1, 1, 1, 0, 0, 0] - [1, 2, 2, 0, 0, 0, 0, 0, 2, 0]
[0, 1, 0, 0, 0, 1, 0, 0, 0, 1] - [2, 0, 0, 0, 0, 0, 2, 0, 0, 0]
[1, 1, 0, 1, 0, 0, 1, 1, 1, 1] - [1, 0, 2, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 1, 0, 0, 1, 0, 1, 1] - [1, 2, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 1, 1, 0, 1, 1, 1, 1, 1, 0] - [1, 1, 0, 0, 1, 1, 0, 0, 1, 2]
[1, 0, 0, 0, 0, 1, 1, 1, 0, 0] - [2, 0, 0, 0, 1, 1, 0, 0, 2, 0]
[1, 1, 1, 0, 1, 1, 0, 0, 1, 0] - [0, 0, 0, 0, 2, 0, 0, 2, 0, 0]

---- Treinando HMM ----

iteração Probabilidade Média

```

0      0.0021291679357005044%
10     0.00257885067245911%
20     0.0038125957559762558%
30     0.0042011808327790945%
40     0.004960888336224656%
50     0.0051901734246726505%
60     0.005604366537066419%
70     0.005729967853083329%
80     0.00597564893142577%
90     0.005976344762666397%
Tempo total de evolução: 3328000s
---- HMM treinada por GA ----
HMM with 3 state(s)

State 0
  Pi: 0.33
  Aij: 0,165 0,833 0,002
  Opdf: Integer distribution --- 0,003 0,997

State 1
  Pi: 0.33
  Aij: 0,002 0,003 0,995
  Opdf: Integer distribution --- 0,004 0,996

State 2
  Pi: 0.33999999999999997
  Aij: 0,994 0,001 0,005
  Opdf: Integer distribution --- 0,854 0,146

-----

----- Comparação de treinamentos -----
HMM original.....: 0.0014968769093930995%
HMM treinada.....: 0.005976344762666483%
Distância.....: 0.5416755492261867
-----

----- Calculando lance seguinte -----
Estado oculto  Estratégia
-----
0      (0.00414221200441966, 0.9958577879955803)
1      (2.0138179924712027E-5, 0.9999798618200753)
2      (0.0025294340324752396, 0.9974705659675248)

----- Aplicando Viterbi -----
[0, 0, 1, 1, 1, 1, 1, 1, 1, 1];1;0;0;0;1;0.03;0.8
[0, 0, 1, 0, 1, 1, 0, 0, 1];1;2;0;0;1;0.23;1.7000000000000002
[0, 1, 1, 0, 1, 1, 1, 1, 0];1;2;2;0;1;0.43000000000000005;2.6
[1, 1, 0, 1, 0, 0, 0, 1, 0];1;2;2;0;1;0.6300000000000001;3.5
[0, 1, 1, 1, 0, 1, 0, 1, 0];1;0;2;0;1;0.6600000000000001;4.3
[0, 1, 0, 1, 1, 0, 1, 0, 1];0;0;0;0;1;1.56;4.35
[1, 1, 0, 1, 1, 0, 1, 1, 1];1;0;1;0;1;1.59;5.1499999999999995
[1, 1, 1, 0, 0, 1, 0, 0, 1];1;0;0;0;1;1.62;5.949999999999999
[1, 1, 1, 1, 0, 1, 1, 0, 1];1;0;0;0;1;1.6500000000000001;6.749999999999999
[0, 0, 0, 1, 1, 0, 0, 0, 0];0;0;2;0;1;2.5500000000000003;6.799999999999999
[0, 1, 1, 1, 1, 0, 0, 1, 1];1;0;1;0;1;2.58;7.599999999999999
[1, 1, 0, 1, 1, 0, 1, 0, 0];1;0;2;0;1;2.61;8.399999999999999

```

[0, 1, 1, 0, 0, 1, 1, 0, 1];0;1;0;0;1;3.46;8.549999999999999
[0, 1, 1, 0, 1, 1, 1, 1, 1];1;0;1;0;1;3.4899999999999998;9.35
[1, 1, 1, 0, 1, 0, 1, 0, 1];1;0;0;0;1;3.5199999999999996;10.15
[1, 1, 1, 1, 1, 0, 1, 0, 1];0;0;0;0;1;4.42;10.200000000000001
[1, 0, 1, 1, 1, 1, 1, 1, 0];0;0;2;0;1;5.32;10.250000000000002
[1, 1, 1, 0, 1, 1, 1, 0, 0];0;0;2;0;1;6.2200000000000001;10.300000000000002
[1, 1, 1, 1, 1, 1, 1, 1, 1];0;0;0;0;1;7.1200000000000001;10.350000000000003
[1, 1, 0, 1, 1, 0, 1, 1, 1];0;2;1;0;1;7.8700000000000001;10.550000000000002
[0, 1, 1, 0, 0, 1, 1, 0, 0];1;0;2;0;1;7.9000000000000001;11.350000000000003
[0, 0, 1, 1, 0, 0, 0, 0, 1];1;0;0;0;1;7.93000000000000015;12.150000000000004
[1, 0, 1, 1, 0, 1, 1, 0, 0];1;1;2;0;1;8.7800000000000001;12.300000000000004
[1, 0, 0, 0, 1, 0, 0, 1, 0];0;0;2;0;1;9.6800000000000001;12.350000000000005
[1, 1, 0, 0, 0, 0, 1, 1, 0];1;0;2;0;1;9.71;13.150000000000006
[1, 1, 0, 1, 1, 1, 1, 1, 0];1;0;2;0;1;9.74;13.950000000000006
[0, 0, 0, 1, 0, 1, 1, 1, 1];0;0;0;0;1;10.64;14.000000000000007
[1, 0, 1, 1, 0, 1, 1, 0, 1];1;0;0;0;1;10.67;14.800000000000008
[1, 1, 0, 1, 1, 1, 1, 1, 0];1;1;2;0;1;11.52;14.950000000000008
[0, 1, 1, 1, 1, 0, 1, 1, 1];0;0;1;0;1;12.42;15.000000000000009
[0, 1, 0, 1, 0, 1, 1, 0, 0];1;0;2;0;1;12.45;15.800000000000001
[0, 0, 1, 1, 0, 1, 0, 0, 0];1;0;2;0;1;12.479999999999999;16.600000000000001
[1, 0, 0, 1, 0, 1, 0, 1, 0];1;0;2;0;1;12.509999999999998;17.400000000000001
[1, 1, 1, 1, 0, 0, 1, 0, 0];1;0;2;0;1;12.539999999999997;18.200000000000001
[1, 1, 0, 1, 1, 1, 1, 1, 0];0;0;2;0;1;13.439999999999998;18.250000000000001
[0, 1, 0, 1, 1, 1, 1, 1, 1];1;0;1;0;1;13.469999999999997;19.050000000000001
[1, 1, 1, 1, 1, 0, 1, 1, 1];0;0;1;0;1;14.369999999999997;19.100000000000012
[1, 0, 1, 1, 1, 0, 1, 1, 0];0;0;2;0;1;15.269999999999998;19.150000000000013
[1, 1, 1, 0, 1, 0, 1, 0, 0];1;0;2;0;1;15.299999999999997;19.950000000000014
[1, 1, 0, 1, 1, 0, 0, 1, 1];0;0;1;0;1;16.199999999999996;20.000000000000014
[0, 1, 1, 1, 1, 1, 0, 1, 0];1;1;1;0;1;17.049999999999997;20.150000000000013
[0, 0, 0, 1, 1, 1, 0, 0, 0];1;0;2;0;1;17.08;20.950000000000014
[0, 0, 0, 0, 0, 0, 0, 1, 1];1;0;1;0;1;17.11;21.750000000000014
[0, 0, 0, 1, 1, 1, 1, 0, 0];0;2;2;0;1;17.86;21.950000000000014
[0, 1, 0, 0, 0, 1, 0, 0, 0];1;0;2;0;1;17.89;22.750000000000014
[1, 1, 0, 1, 0, 0, 1, 1, 1];1;0;1;0;1;17.92;23.550000000000015
[0, 0, 0, 1, 0, 0, 1, 0, 1];1;0;0;0;1;17.950000000000003;24.350000000000016
[0, 1, 1, 0, 1, 1, 1, 1, 1];0;1;1;0;1;18.800000000000004;24.500000000000014
[1, 0, 0, 0, 0, 1, 1, 1, 0];0;2;2;0;1;19.550000000000004;24.700000000000014
[1, 1, 1, 0, 1, 1, 0, 0, 1];0;0;0;0;1;20.450000000000003;24.750000000000014

0;0.5406168879939739;19.0;31.0;31.0;24.750000000000014;20.450000000000003;24.750000000000014;18.0

Capítulo 8

Referências Bibliográficas

BAUM, L., EAGON, J.A., 1967, *An Inequality with applications to statistical estimation for probabilistic functions of Markov processes and to a model for ecology*, Boletim da Sociedade Matemática Americana nº 73, pág.360-363.

CHANG, H.S., GIYAN, R. CHONG, E.K.P, 2000, On-line Scheduling via Sampling, Proceedings of the 5th International Conference on Artificial Intelligence Planning and Scheduling, Colorado, EUA.

CHAU, C. W. , KWONG, S. , DIU, C.K. ET AL., 1997, *Optimization Of HMM By A Genetic Algorithm*, Conferência Internacional IEEE sobre Acústica, Fala e Processamento de Sinal, Munique, Alemanha.

COUVREUR, C, 1996, *Hidden Markov Models and their Mixtures*. Tese de Pós-Doutorado (*Habilitation à diriger des recherches*), Universidade Católica de Louvan, França

DUGAD, R., DESAI, U.B., 1996, *A Tutorial On Hidden Markov Models*, Relatório nº SPANN-96-1. Instituto de Tecnologia Indiano, India

FINESSO, L., 2000, *The Complexity of Hidden Markov Models*, ERCIM News No.40/Jan 2000

FRANCOIS, J.M., 2006, *Java implementation of Hidden Markov Model (HMM) related algorithms*, Página WWW disponível em <http://code.google.com/p/jahmm/> em 13/3/2009

FUDENBERG, D., TIROLE, J., 1994, *Game Theory*, MIT Press, Cambridge.

GIBBONS, R., 1992, *Game Theory For Applied Economists*, Princeton University Press, New Jersey, EUA.

GILBOA, I., ZEMEL, E., 1989, *Nash and Correlated Equilibria: Some Complexity Considerations*, Games and Economic Behavior, vol 1, páginas 80-93, Elsevier, Amsterdam, Holanda.

GINTIS, H., 2000, *Game Theory Evolving*, Princeton University Press, New Jersey, EUA.

JUANG, B.H., RABINER, L.R., 1985, *A Probabilistic Distance Measure for Hidden Markov Models*, AT&T Technical Journal, vol 64:22, 391-408.

JURAFSKY, D., MARTIN, J. H., 2006, *Speech and Language Processing: An introduction to natural language processing*, Prentice Hall, New Jersey, EUA, 1st ed.

LIU, Y., MAN, H., COMANICIU, C., 2006, *A Game Theoretic Approach to Efficient Mixed Strategies for Intrusion Detection*, IEEE International Conference on Communications 2006 (ICC 2006), Istambul, Turquia.

LINDEN, R., 2006, *Algoritmos Genéticos*, Brasport, Rio de Janeiro, Brasil

MAMON, R., ELLIOTT, R., 2007, *Hidden Markov Models in Finance*, Springer Science, New York, EUA.

MANABE, S., HATANAKA, T., UOSAKI, K. ET AL., 2006, *Training Hidden Markov Model Structure with Genetic Algorithm for Human Motion Pattern Classification*, Conferência Internacional SICE-ICASE, Bexco, Coréia do Sul.

MEDINA F., FERREIRA FILHO, V. J. M., 2003, *Um exemplo de aplicação de Teoria dos Jogos em leilões de blocos exploratórios*, 2º Congresso Brasileiro de P&D em Petróleo e Gás, Rio de Janeiro, Brasil.

MEDINA F., FERREIRA FILHO, V. J. M., MOTTA, R. R. ET AL., 2003, *Equilíbrio Entre Decisões Ótimas Para Um Jogo Estratégico Em Petróleo*, XXXV Simpósio Brasileiro de Pesquisa Operacional, Natal, Brasil.

MENASCHÉ, D. S., 2005, *Controle de Congestionamento: Uma Abordagem Dinâmico-Populacional via Teoria dos Jogos Evolucionários*, Dissertação de Mestrado, COPPE, Rio de Janeiro.

MIHOK, B., TERRY, M., 2005, *Statistical Learning and Inference Methods for Reasoning in Games*, conteúdo de palestra ministrada em 30/Mar/2005 no Massachusetts Institute of Technology, EUA, disponível em http://web.mit.edu/16.412j/www/html/Advanced%20lectures/Slides/Mihok_Terry.pdf

MILLER, D.R.H, LEEK, T., SCHWARTZ, R.M., 1999, *A Hidden Markov Model Information Retrieval System*, 22nd International Conference on Research and Development in Information Retrieval (SIGIR'99), Califórnia, EUA.

NASH, J. F., 1950, *Non-Cooperative Games*, Dissertação de Doutorado, Universidade de Princeton, New Jersey, EUA.

PERRONE, M. P., CONNELL, S. D., 2000, *K-Means Clustering for Hidden Markov Models*, in Proc. of 7th Int'l Workshop on Frontiers in Handwriting Recognition, 2000

RABINER, L., 1989, *A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition*, Proceedings of the IEEE, 77 (2), p. 257–286.

RABINER, L., E JUANG, B. H., 1986, *An introduction to hidden Markov models*, Revista IEEE ASSP, pp. 4-16.

RASMUSEN, E., 2006, *Games and Information: An Introduction to Game Theory*, Wiley-Blackwell, EUA.

ROST, P., GERHARD, F., 2007, *A Generalized Mixed Strategy for Multiterminal Relay Networks*, IEEE Information Theory Workshop 2007 (ITW 2007), Califórnia, EUA.

RUBINSTEIN, A., TVERSKY, A., HELLER, D., 1996, *Naive Strategies in Competitive Games*, Understanding Strategic Interaction - Essays in Honor of Reinhard Selten, W.Guth et al., Springer-Verlag, 394-402.

RUSSELL, S., NORVIG, P., 2004, *Inteligência Artificial*, Elsevier Editora, São Paulo, Brasil, tradução da Segunda Edição.

SHEN, Y., FAN, Y., 2007, *Cooperative Mixed Strategy for Service Selection in Service Oriented Architecture*, IEEE Systems, Man and Cybernetics 2007 (SMC 2007), Quebec, Canadá.

SHINAR, J.; LIPMAN, Y.; ZARKH, M., 1995, *Mixed strategies in missile versus missile interception scenarios*, Conferência Americana de Controle, pág 4116 - 4120 vol.6.

SILVEIRA FILHO, F.J., 2006, *Previsão De Estatísticas De Perdas De Pacotes Usando Modelos De Markov Ocultos*, Dissertação de Mestrado, COPPE, Rio de Janeiro.

TU, K. BI, K., GU, N. ET AL., 2006, *Mac Address Assignment In Wireless Sensor Networks: A Mixed Strategy Game Approach*, International Conference on Systems and Networks Communication 2006 (ICSNC'06), Tahiti, Polinésia Francesa.

VON NEUMANN, J., MORGENSTERN, O., 1944, *Theory of Games and Economic Behavior*, Princeton University Press, New Jersey, EUA, 60.ed.

WELCH, L., 2003, *Hidden Markov Models and the Baum-Welch Algorithm*, IEEE Information Theory Society Newsletter, Dezembro de 2003.