Aplicação de Modelos de Markov Ocultos na Obtenção de Taxas de Mortalidade das Larvas do Mosquito da Dengue

Kleber Padovani de Souza

Dissertação de Mestrado apresentada à
Faculdade de Computação do
Centro de Ciências Exatas e Tecnologia da
Universidade Federal de Mato Grosso do Sul

Orientador: Prof. Dr. Hemerson Pistori

O autor recebeu apoio financeiro da CAPES durante o desenvolvimento deste trabalho, que contribui com o projeto de pesquisa denominado LARVIC, criado e mantido pela Universidade Católica Dom Bosco em parceria com a empresa Tec-Sinapse Tecnologia da Informação Ltda. sob apoio das agências de fomento CNPq e FUNDECT.

Fevereiro de 2010

Aplicação de Modelos de Markov Ocultos na Obtenção de Taxas de Mortalidade das Larvas do Mosquito da Dengue

Este exemplar corresponde à redação
final da dissertação devidamente corrigida
e defendida por Kleber Padovani de Souza
e aprovada pela Comissão Julgadora em 26/02/2010.

Banca Examinadora:

- Prof. Dr. Hemerson Pistori - UCDB
- Profa. Dra. Valguima Victoria Viana Aguiar Odakura - UFGD
- Profa. Dra. Maria Bernadete Zanusso - FACOM/UFMS

Resumo

Com o intuito de combater a proliferação do mosquito transmissor do vírus da dengue e, consequentemente, diminuir os casos de infecção humana pelo vírus, diversos larvicidas vêm sendo estudados recentemente por profissionais das áreas da saúde e afins. Nos últimos anos, pesquisadores da Universidade Católica Dom Bosco avaliam a eficácia de substâncias extraídas de plantas com potencial larvicida no combate às larvas desse inseto. Dentre os critérios utilizados na avaliação de uma substância, encontra-se a taxa de mortalidade das larvas. Essa taxa pode ser obtida por meio de observação visual periódica das larvas sujeitas à substância. No entanto, características humanas, como suscetibilidade à exaustão e subjetividade, podem comprometer a precisão dessa taxa, o que influencia na avaliação final do larvicida em questão. Por conta desse e de outros fatores, torna-se interessante a construção de sistemas computacionais para observação automática das larvas. Por se tratar de uma tarefa realizada com base em aspectos visuais, a aplicação de técnicas de visão computacional (VC) associadas a outras técnicas possivelmente é adequada ao desenvolvimento de tais sistemas. Nesse contexto, o grupo de pesquisadores em VC da universidade citada criou um projeto de pesquisa, denominado LARVIC, que visa automatizar a obtenção dessas taxas com auxílio da VC. Com o intuito de contribuir com o sistema em desenvolvimento nesse projeto, este trabalho investiga a aplicação de modelos de Markov ocultos como mecanismo de classificação de comportamentos de larvas. Como resultado, por meio da identificação da falência de larvas, pode-se obter, dentre outras informações, a taxa de mortalidade em cada experimento. Para isso, sequências de imagens de larvas vivas e mortas contidas em recipientes apropriados foram capturadas em laboratório e processadas por algoritmos de VC e áreas relacionadas para aquisição de informações que viabilizaram a análise da aplicação proposta. A investigação se fundamenta em três perspectivas, a saber: 1) as probabilidades iniciais dos modelos, utilizadas como base para o reajuste de probabilidades, 2) o critério de parada para a etapa de reajuste e 3) o modo de utilização dos modelos para classificação dos comportamentos. Foram consideradas probabilidades iniciais parcialmente aleatórias e pré-computadas manual e automaticamente. Para avaliação do critério de parada padrão utilizado nos experimentos, foram examinados outros critérios mais robustos. Como alternativa ao modo de classificação tradicionalmente utilizado, os modelos foram combinados com técnicas de aprendizagem de máquina, com o intuito de ampliar o desempenho dos classificadores. Para analisar cada perspectiva, foram executados diferentes experimentos através do uso de algoritmos provenientes de bibliotecas específicas e outras implementações aqui desenvolvidas. Por meio de análise baseada em taxas de acerto e gráficos sobre o desempenho, foi possível concluir que o desempenho dos classificadores baseados apenas em modelos de Markov ocultos, embora inferior ao de outros classificadores, foi considerável para a aplicação estudada.

Palavras-chave: Modelos de Markov ocultos, comportamento animal, reconhecimento de padrões.

Abstract

In order to prevent the proliferation of Dengue transmitter – scientifically named as *Aedes aegypti* – and therefore decrease human contamination by such insect, many larvaecides have been developed recently. Researchers from Dom Bosco Catholic University evaluate the effectiveness of vegetal-derived substances capable to combat such animal larvae. Death rate is among many evatuation criteria of these substances. Such rate may be obtained by periodic visual observation of larvae subjected to some substance. However, human limitations, such as relative conclusions and exhaustion susceptibility, can damage a larvaecide investigation. Thus, a computer system which can automatically observe larvae while experiments are performed is a great support for science. For this type of systems, computer vision (CV) algorithms may be appropriated once larva observation is a visual-based task. In addition, such systems can be substantially improved when using techniques from other areas, specially pattern recognition and machine learning. In this context, the research group of CV from the aforementioned university created a project, named LARVIC, so as to develop an automatic system based on CV. In order to collaborate with this project, the present work investigate the applicability of Hidden Markov Models in larvae behaviour classification. For this purpose, image sequences of dead and alive larvae inside a specific recipient were captured and then processed by algorithms of CV and related areas, producing relevant data which allowed the analysis of the proposed application. Three aspects were considered during the examination, namely: 1) initial probabilities of models, which are base for parameters reestimation, 2) stop criterion of reestimation stage, and 3) how to use the models as a classifier. Partly random and previously computed initial probabilities were used. Stronger criteria were examinated to analyse standard criterion. A combination with models and machine learning techniques was built to increase the perfomance of classifiers based on the major technique of this work. External libraries and additional implementations made experiments possible. Based on error rate and performance graphs, we conclude with this work that the performance of classifiers based on Hidden Markov Models with no combination, though inferior to some classifiers, was substantial for the approached application.

Keywords: Hidden Markov models, animal behavior, pattern recognition.

Lista de Figuras

2.1 Exemplo de segmentação por subtração de fundo. 5
2.2 Exemplo de segmentação baseada em atributos de cores. 6
2.3 Ilustração de um mesmo objeto em posições diferentes. 8
2.4 Ilustração de um mesmo objeto em ângulos e tamanhos diferentes. 9
2.5 Exemplo de seleção de pontos de contorno para cálculo de curvaturas. 10
2.6 Exemplo de construção de vetores e ilustração de ângulos entre eles. 11
2.7 Imagens de exemplo das quais serão extraídas suas características. 14
2.8 Exemplo de curva de ROC. 17
2.9 Espaço ROC. 21

3.1 Exemplos de cadeias de Markov. 23
3.2 Ilustração da cadeia de Markov para modelagem de comportamento climático com base nas observações de um conjunto de amostras. 23
3.3 Treliça referente ao exemplo dado. 29
3.4 Treliça de possíveis estados para a observação O^1. 33
3.5 Treliças representando sequências de estados possíveis para determinadas sequências de observações. 33
3.6 Treliças representando sequências de estados possíveis para determinadas sequências de observações. 34
3.7 Treliças representando valores das variáveis α e β em um HMM com 4 estados e uma sequência de observações com 7 símbolos. 36
3.8 Representação da probabilidade de transição do estado 2 para o estado 4 e geração do quinto símbolo da sequência de observações. 36
3.9 Exemplo de HMM completamente conectado. 38
3.10 Exemplos de HMM's de 4 estados com topologias diferentes. 38

4.1 Diagrama de implementações. 48

5.1 Ilustração de segmentação de imagem utilizada nos experimentos por meio da combinação de segmentações baseadas em subtração de fundo e em atributos de cores. 50

5.2 Imagens de larvas em diferentes posturas. 53

6.1 Curvas ROC para experimento com HMM's com probabilidades iniciais aleatórias. 57

6.2 Curvas ROC para experimento com HMM's com probabilidades iniciais pré-computadas manualmente. 57

6.3 Curvas ROC para experimento com HMM's com probabilidades iniciais pré-computadas automaticamente com 2 estados. 57

6.4 Curvas ROC para experimento com HMM's com probabilidades iniciais pré-computadas automaticamente com 3 e 4 estados. 58

6.5 Melhores curvas ROC entre algoritmos de aprendizagem de máquina (utilizando 2 características). 61

6.6 Piores curvas ROC entre algoritmos de aprendizagem de máquina (utilizando 2 características). 61

6.7 Melhores curvas ROC entre algoritmos de aprendizagem de máquina (utilizando 4 características). 61

6.8 Piores curvas ROC entre algoritmos de aprendizagem de máquina (utilizando 4 características). 62

6.9 Melhores curvas ROC entre algoritmos de aprendizagem de máquina (utilizando 1 característica). 62

6.10 Piores curvas ROC entre algoritmos de aprendizagem de máquina (utilizando 1 característica). 62

A.1 Treliça representando os valores de α nas possíveis sequências de estados geradoras de uma sequência de observações de 5 símbolos em um HMM com 3 estados. . . 65

A.2 Treliça representando os valores de β nas possíveis sequências de estados. Note que o círculo que contém o valor $\beta_3(1)$ está tracejado, pois a probabilidade de geração do terceiro símbolo não está agregada no valor da variável. 66

B.1 Curvas ROC para algoritmo IBk (2 características). 67

B.2 Curvas ROC para algoritmo J48 (2 características). 67

B.3 Curvas ROC para algoritmo SMO (2 características). 68

B.4 Curvas ROC para algoritmo MLP (2 características). 68

B.5 Curvas ROC para algoritmo IBk (4 características). 68

B.6 Curvas ROC para algoritmo J48 (4 características). 69

B.7 Curvas ROC para algoritmo SMO (4 características). 69

B.8 Curvas ROC para algoritmo MLP (4 características). 69

B.9 Curvas ROC para algoritmo IBk (1 característica). 69
B.10 Curvas ROC para algoritmo J48 (1 característica). 70
B.11 Curvas ROC para algoritmo SMO (1 característica). 70
B.12 Curvas ROC para algoritmo MLP (1 característica). 70

Lista de Tabelas

2.1 Grupos de ângulos. 11

2.2 Histogramas dos dois objetos dados como exemplo considerando 6 grupos de ângulos determinados arbitrariamente. Os valores para os dois objetos em cada intervalo foram obtidos por meio de contagem simples de ocorrência. 11

2.3 Objetos com os valores extraídos das respectivas características. 14

2.4 Lista de objetos para análise de determinado classificador. A segunda coluna apresenta a classe real do objeto e a classe predita pelo classificador é exibida na terceira coluna. 15

2.5 Exemplo de matriz de confusão. 16

2.6 Matriz de confusão para duas classes. 16

2.7 Lista de objetos para análise de determinado classificador baseado em pontuação. A segunda coluna apresenta a classe real do objeto; a terceira apresenta a pontuação produzida para o classificador para cada objeto; a quarta e a quinta apresentam a classe predita pelo classificador utilizando limiares de 0.8 e 0.7, respectivamente. 18

3.1 Probabilidades de ocorrência de chuva no dia seguinte. 24

3.2 Exemplo de matriz de transição de estados do primeiro grupo. 25

3.3 Exemplo de matriz de transição de estados do segundo grupo. 25

3.4 Exemplo de matriz de emissão de símbolos dos estados do primeiro grupo. 26

3.5 Exemplo de matriz de emissão de símbolos dos estados do segundo grupo. 26

3.6 Vetor de probabilidades de ocorrência inicial dos estados do HMM do primeiro grupo. 26

3.7 Vetor de probabilidades de ocorrência inicial dos estados do HMM do segundo grupo. 26

3.8 Sequências de possíveis estados geradores da sequência de observações $O = v_1, v_2, v_1$. 29

3.9 Probabilidades de ocorrências iniciais, emissões de símbolos e transições de estados, respectivamente. 32

3.10 Sequências de estados possíveis no modelo e probabilidades de geração da sequência de observações $O = \{A, A, B\}$. 32

6.1 Matrizes de confusão dos experimentos com probabilidades iniciais variadas para os HMM's. O primeiro par de colunas apresenta a matriz de confusão considerando probabilidades aleatórias; o par seguinte considera probabilidades pré-computadas manualmente; e os outros 3 últimos pares consideram, nesta ordem, probabilidades pré-computadas automaticamente com 2, 3 e 4 estados. 56

6.2 Matrizes de confusão para o segundo e terceiro critérios de parada. No segundo critério, são considerados diferentes limiares. No terceiro, todas as variações de iterações (de 50 a 1000) apresentaram os mesmos resultados. 59

6.3 Matrizes de confusão para modelos com probabilidades iniciais pré-computadas automaticamente e diferentes critérios de parada. 59

6.4 Matrizes de confusão dos experimentos com algoritmos de aprendizagem de máquina (2 características). 60

6.5 Matrizes de confusão dos experimentos com algoritmos de aprendizagem de máquina (4 características). 60

6.6 Matrizes de confusão dos experimentos com algoritmos de aprendizagem de máquina (1 característica). 60

C.1 Exemplo de produtos a serem agrupados. 71

C.2 Primeira iteração do algoritmo para o exemplo dado. 72

C.3 Segunda iteração do algoritmo para o exemplo dado. 73

C.4 Terceira iteração do algoritmo para o exemplo dado. 73

Sumário

1 Introdução **1**

2 Fundamentação teórica **4**

2.1 Visão Computacional 4

2.1.1 Segmentação 5

2.1.2 Descrição de objetos 7

2.2 Classificação automática 13

2.2.1 Reconhecimento de padrões 13

2.2.2 Avaliação de classificadores 15

3 Modelos de Markov Ocultos **22**

3.1 Elementos do HMM 25

3.2 Incógnitas principais 27

3.2.1 Avaliação da observação 27

3.2.2 Melhor sequência de estados 32

3.2.3 Treinamento do modelo 35

3.3 Tipos de HMM's 37

3.4 Mudança de escala 38

3.4.1 Probabilidade da sequência de observações 39

3.4.2 Sequência de estados ótima 40

3.5 Conjunto de símbolos contínuo 41

3.6 Combinação de HMM's com técnicas de aprendizagem 41

4 Implementações **44**

5 Experimentos **49**

5.1 Aquisição e tratamento de imagens 49

5.2 Metodologia para treinamento e classificação 51

5.3 Configuração dos experimentos 54

6 Resultados e análise **56**

6.1 Experimentos do grupo 1 56

6.2 Experimentos do grupo 2 58

6.3 Experimentos do grupo 3 59

6.4 Experimentos do grupo 4 60

7 Considerações finais **63**

Apêndices **65**

A Conceitos suplementares **65**

A.1 Algoritmos Forward e Backward 65

B Curvas ROC adicionais **67**

B.1 Pré-processamento com 2 características 67

B.2 Pré-processamento com 4 características 68

B.3 Pré-processamento com 1 característica 68

C Algoritmo K-Means **71**

Referências Bibliográficas **74**

Capítulo 1

Introdução

As epidemias de dengue ocorridas nos últimos anos despertaram o país para a necessidade de se buscar formas mais eficazes de combate ao seu mosquito transmissor, o *Aedes aegypti*. Experimentos com novos larvicidas envolvem a análise, em várias repetições, da mortalidade de larvas utilizando diversas combinações de concentrações e larvicidas diferentes. Centenas e até milhares de experimentos precisam ser realizados até se encontrar uma substância que produza o efeito desejado. O Mato Grosso do Sul se destaca pela busca de novos larvicidas feitos a partir de extratos de sua diversificada flora pantaneira e de cerrado [31, 33, 34, 41].

Na Universidade Católica Dom Bosco (UCDB), um grupo de pesquisadores desenvolve e avalia a eficácia desses larvicidas para o combate às larvas do mosquito. Após o desenvolvimento do larvicida, são realizados testes laboratoriais que analisam os efeitos dessa substância. A contagem de mortalidade de larvas é um desses testes. Ela é feita por análise visual de recipientes contendo larvas expostas durante 24 horas às substâncias em teste. Em instantes definidos, um especialista observa os recipientes, identifica e registra a quantidade de larvas vivas e mortas. Ao final do experimento, com base nesses registros, são alcançadas algumas conclusões a respeito do efeito do larvicida analisado.

Como essa contagem é realizada geralmente por humanos, que naturalmente possuem suas limitações, alguns erros são possíveis. Por exemplo, por se tratar de uma tarefa minuciosa e que exige muita atenção, é possível que o observador fique exausto e isso, obviamente, pode comprometer a qualidade dos registros de observação. Outro exemplo cabível é a subjetividade e a inconsistência. Em alguns momentos, a identificação do comportamento das larvas não é simples. Por exemplo, antes da larva morrer, seus movimentos característicos de locomoção ficam levemente modificados. Assim, um observador com pouca experiência pode registrar erroneamente que uma determinada larva em processo de falência está viva. Portanto, nesse cenário, dois observadores poderiam registrar comportamentos diferentes para uma mesma larva em um mesmo instante de tempo.

Nesse contexto, a contagem automática de mortalidade de larvas pode ser útil para realização dessa tarefa eliminando alguns dos problemas inerentes à natureza humana. A utilização de visão computacional nesses problemas é uma das alternativas para automatização dessa contagem. Basicamente, a visão computacional busca extrair informações de imagens para identificar e classificar objetos presentes nelas [33]. Desse modo, pode-se construir um sistema de visão computacional que, em momentos determinados, processa imagens de recipientes com larvas e, em seguida, classifica e registra os comportamentos de cada uma dessas larvas. A criação desse

sistema é o objetivo do projeto denominado LARVIC [31], criado e mantido pelo INOVISAO – Grupo de Pesquisa, Desenvolvimento e Inovação em Visão Computacional da UCDB.

Através do projeto LARVIC, a contagem de mortalidade está sendo automatizada. As imagens são capturadas através de uma câmera posicionada acima dos recipientes contendo as larvas [37]. Posteriormente, essas imagens são processadas por meio de algoritmos de processamento de imagens e visão computacional. Em seguida, com base nas informações provenientes do processamento anterior, os comportamentos das larvas são classificados e registrados. O objetivo deste trabalho é analisar a aplicação de um modelo estatístico, denominado Modelo de Markov Oculto (ou HMM, do inglês *Hidden Markov Model*), nessa etapa de classificação de comportamentos com o intuito de contribuir com esse projeto.

Para tal objetivo, foram realizados diferentes experimentos com imagens laboratoriais reais de larvas. De modo resumido, primeiramente, foram capturadas as imagens das larvas. Em seguida, foram extraídos atributos relevantes dessas imagens, que foram posteriormente adequados para serem utilizados pelos HMM's. Com esses atributos, foi possível a realização de diversos experimentos, explorando particularidades desses modelos, combinando-os com técnicas de aprendizagem de máquina e comparando seus resultados com os resultados dessas técnicas. De modo sucinto, foram analisados classificadores compostos por HMM's considerando diferentes probabilidades iniciais, modos de treinamento e classificação.

As probabilidades iniciais utilizadas para os modelos foram probabilidades uniformes em conjunto com probabilidades aleatórias e probabilidades pré-computadas de forma manual e automática. Os modos de treinamento consideraram diferentes critérios de parada, o que interfere diretamente no número de iterações do algoritmo utilizado. Por fim, foram utilizados dois modos de classificação: o tradicional – em que são consideradas as probabilidades de geração de símbolos dos modelos – e o combinado – em que são embutidas técnicas de aprendizagem de máquina no classificador baseado em HMM's como auxílio na classificação de entradas.

Por meio dessa abordagem combinada, é possível obter melhores classificações em determinadas aplicações [19, 52, 54, 55], comparando com as classificações obtidas apenas com HMM's. Essa melhoria pode ser observada especialmente em aplicações cujas classificações realizadas por meio dos modelos apresentam grande incerteza. Para melhor compreensão, suponha a existência de um classificador responsável pela identificação de uma anomalia em pacientes por meio de sequências de DNA. Portanto, dada uma sequência de DNA de um paciente qualquer, o classificador deve informar se esse paciente é portador (ou não) de tal anomalia.

Dadas as sequências de DNA de dois pacientes, com base no treinamento realizado previamente, o mecanismo utilizado pelo classificador informa que a probabilidade do primeiro paciente ser portador é de 90% e que a probabilidade de não ser é de 9%. Já para o segundo paciente, o mecanismo informa as probabilidades de 96% de ser portador e de 93% de não ser. Comparando probabilidades, o classificador pode inferir que ambos os pacientes são portadores da anomalia. No entanto, com base nas probabilidades, nota-se que o resultado dado ao segundo paciente é pouco confiável, pois as probabilidades de ser portador e de não ser são muito próximas. Logo, essa classificação possui alto grau de incerteza. Em situações como essa, pode ser útil a utilização da abordagem combinada. Nesses casos, uma alternativa seria recorrer a um classificador auxiliar para tomada da decisão final. Dessa forma, resultados mais consistentes podem ser obtidos.

Analogamente, esse foi o critério utilizado neste trabalho nos experimentos realizados com classificadores combinados. Primeiramente, o classificador encontra a classe mais provável. Pos-

teriormente, são calculadas individualmente as distâncias entre a probabilidade dessa classe e as probabilidades de todas as outras classes. Se a proporção de alguma dessas distâncias com relação à maior probabilidade não for maior que o limiar definido, o classificador auxiliar toma a decisão final. Caso contrário, nada é feito e a classificação segue de forma tradicional, baseada na classe de maior probabilidade.

Considerando novamente o exemplo da análise de DNA e um limiar de 0.1, para o primeiro paciente, teríamos a classificação tradicional, pois a distância entre a probabilidade da classe mais provável e a outra classe é igual a 81 ($90 - 9 = 81$) e, consequentemente, a proporção dessa distância com relação à maior probabilidade é 0.9 ($81/90 = 0.9$), que é maior que o limiar. Em contrapartida, para o segundo paciente, a proporção da distância – que corresponde a aproximadamente 0.03 – é menor que o limiar, tornando necessário o auxílio do classificador auxiliar.

Para construção dos classificadores baseados em HMM's, foi utilizada uma biblioteca, chamada JAHMM, que possui implementações abertas e gratuitas (na linguagem Java) de algoritmos de HMM's e relacionados. Porém, foram necessárias adaptações nessa biblioteca para que todos os experimentos pudessem ser realizados. Foi criado um classificador baseado em HMM's, que, basicamente, recebe sequências de observações como entrada e as classificam com base em um conjunto de classes conhecido. Além da classificação tradicional, foi implementada a classificação combinada nesse classificador. Como essa classificação pode utilizar algoritmos de aprendizagem de máquina, foi implementada uma interface de comunicação com a ferramenta Weka, que possui diversos algoritmos implementados de inteligência artificial.

Após a realização dos experimentos, foram utilizadas as taxas de acerto dos classificadores e suas respectivas curvas ROC na análise dos resultados. As curvas ROC são gráficos que, dentre outras finalidades, podem ser utilizados para visualização de informações sobre o desempenho de classificadores. Nessa análise, resumidamente, foi observado que o desempenho dos classificadores baseados em HMM's, embora inferior ao desempenho de outros classificadores, foi considerável para esta aplicação. Os resultados obtidos com esses classificadores são comparáveis aos resultados obtidos com classificadores baseados em algoritmos renomados de aprendizagem de máquina, como redes neurais e máquinas de vetor de suporte. Além disso, foi notado que, para a aplicação estudada, a combinação entre HMM's com outros algoritmos não foi favorável.

Este trabalho está organizado em 7 capítulos, sendo este o primeiro. No segundo capítulo, é apresentada uma fundamentação teórica para que o leitor possa relembrar/aprender conceitos teóricos que foram utilizados neste trabalho, o que facilitará a compreensão dos capítulos seguintes. O Capítulo 3 apresenta a teoria da técnica principal deste trabalho – o HMM – e assuntos relacionados a ela. No Capítulo 4, são descritas as implementações necessárias para a realização dos experimentos propostas, apresentados no quinto capítulo. Os resultados desses experimentos são expostos no Capítulo 6. Além disso, é exibida uma análise sobre esses resultados, discorrendo sobre o desempenho dos classificadores e apontando possíveis justificativas para comportamentos relevantes. Finalmente, o último capítulo conclui o trabalho apresentando um breve resumo das atividades executadas, resultados obtidos e sugestões de trabalhos futuros que contribuirão com este.

Capítulo 2

Fundamentação teórica

Neste capítulo, serão fornecidos alguns conhecimentos teóricos necessários para a compreensão dos capítulos posteriores. Serão abordados alguns conceitos da área de visão computacional que tornaram possível a execução dos experimentos aqui realizados. No entanto, espera-se o conhecimento básico por parte do leitor dos conceitos referentes a imagens digitais coloridas e monocromáticas.

2.1 Visão Computacional

A visão computacional, ou visão de máquina, é uma área de conhecimento que se dedica a desenvolver teorias e métodos voltados à extração automática de informações "úteis" contidas em imagens, sendo que a utilidade dessa informação é altamente relacionada à aplicação. Por exemplo, para uma indústria de manufatura, uma informação útil pode estar relacionada com alguma diferença na cor (ou na forma) padrão esperada para um produto sem defeitos.

Sistemas de visão computacional trabalham com imagens – geralmente capturadas por dispositivos ótico-eletrônicos, como câmeras e filmadoras digitais – e buscam produzir descrições úteis das informações contidas nas imagens. Essas descrições podem ser utilizadas, por exemplo, na classificação de objetos ou no controle automático de algum dispositivo atuador, como um braço robótico ou uma rede de esteiras rolantes em uma linha de montagem. Contudo, tarefas relativamente simples de serem realizadas por seres humanos, como diferenciar chaves de fenda simples de chaves de fenda em cruz[1] em uma linha de montagem, apresentam-se como grandes desafios para sistemas automáticos de visão computacional.

Em alguns desses sistemas, é necessária a utilização de algoritmos para a segmentação e extração de características em imagens, como ocorre neste trabalho. Resumidamente, em visão computacional, a segmentação separa as informações (objetos, pessoas, animais, etc.) contidas em imagens e a extração de características obtém informações relevantes dos objetos contidos nas imagens para utilizá-las posteriormente para algum fim específico. As subseções seguintes abordam cada uma dessas áreas de forma mais detalhada.

[1] Popularmente conhecidas como chaves Philips.

2.1.1 Segmentação

A segmentação é um processo de considerável importância em trabalhos que envolvem a análise de imagens, tais como os sistemas baseados em visão computacional [53]. Nesse processo, os objetos de interesse para a aplicação são separados dos demais objetos, para que sejam posteriormente processados. Logo, a existência de erros nessa etapa pode certamente propagá-los para as etapas seguintes, interferindo negativamente na eficácia do sistema.

Existem diversos algoritmos para segmentação de imagens que se baseiam em diferentes características discriminantes. Dentre eles, temos as segmentações realizadas pixel a pixel, que avaliam cada pixel da imagem isoladamente utilizando determinado critério para classificá-lo. As técnicas de segmentação por subtração de fundo e baseadas em aprendizagem supervisionada utilizam essa abordagem, como descrito a seguir.

Segmentação por subtração de fundo

A subtração de fundo é uma técnica para segmentação de imagens em que os objetos de interesse são encontrados nas imagens a partir de uma comparação entre essas imagens e uma imagem de referência, que corresponde à imagem do ambiente sem a presença de tais objetos [26]. Desse modo, as regiões da imagem que apresentam uma distância significativa da imagem de referência são consideradas como constituintes de objetos de interesse.

Essa técnica é geralmente empregada na segmentação de imagens capturadas por um dispositivo fisicamente fixado, cujas imagens produzidas são obtidas de um mesmo ponto de observação (ou ponto de fuga). As Figuras 2.1(a) e 2.1(b) ilustram um exemplo de imagens obtidas de uma câmera fixa e a Figura 2.1(c) representa a segmentação da Figura 2.1(b) por subtração de fundo, utilizando a Figura 2.1(a) como imagem de referência. Nesse exemplo, o intuito era identificar novos objetos inseridos em um ambiente fechado (Figura 2.1(a)). Note que, de fato, o objeto – uma maleta – inserido no ambiente (Figura 2.1(b)) foi corretamente segmentado (Figura 2.1(c)).

(a) Imagem de referência (b) Imagem a segmentar (c) Imagem segmentada

Figura 2.1: Exemplo de segmentação por subtração de fundo.

Uma das formas mais simples para subtrair o fundo de imagens é através da utilização de um limiar de distância entre um pixel da imagem a ser segmentada e o respectivo pixel da imagem de referência, como utilizado em [18]. Nesse contexto, são considerados como partes de objetos os pixels da imagem que obedecerem a condição expressa pela Equação 2.1.1, em que x e y formam a coordenada do pixel, *threshold* é um limiar previamente definido e $I(x, y)$ e $R(x, y)$ representam a intensidade do pixel localizado na coordenada (x, y) da imagem a ser segmentada e da imagem de referência, respectivamente.

$$|I(x, y) - R(x, y)| > threshold \tag{2.1.1}$$

Essa classificação é realizada considerando a utilização de imagens em tons de cinza. Logo, para imagens coloridas, é necessária uma adaptação. Uma alternativa simples para resolver esse problema é converter a imagem colorida em uma nova imagem em tons de cinza e, posteriormente, utilizar o mesmo critério de avaliação de pixels acima citado. No entanto, existem técnicas mais sofisticadas para subtração de fundo em imagens coloridas [7, 20, 58].

Segmentação baseada em aprendizagem supervisionada

Outra maneira de segmentar imagens coloridas e monocromáticas é através da utilização de aprendizagem supervisionada, como utilizado e/ou mencionado em [2, 14, 21, 48]. Nesses trabalhos, os pixels da imagem foram avaliados de acordo com as características de suas cores, que são obtidas através dos valores das componentes do espaço de cores utilizado. A Figura 2.2 mostra um exemplo de segmentação de pele baseada em cores, em que a imagem exibida na Figura 2.2(a) foi segmentada utilizando amostras de cores de pele, produzindo a imagem representada pela Figura 2.2(b).

(a) Imagem a segmentar (b) Imagem segmentada

Figura 2.2: Exemplo de segmentação baseada em atributos de cores.

Um segmentador de imagens simples e eficiente para algumas aplicações baseado em aprendizagem estatística pode ser desenvolvido utilizando uma distribuição normal [14, 15]. As variáveis da distribuição poderiam ser os próprios valores das componentes do espaço de cores utilizado (RGB, HSV, YCbCr, etc.). Como exemplo, poderíamos utilizar as componentes R e G do espaço de cores RGB e, nesse caso, trabalharíamos com uma distribuição com duas variáveis.

Inicialmente, os parâmetros dessa distribuição (média e variância) devem ser estimados. Essa estimação pode ser realizada utilizando amostras (fragmentos de imagens) que apresentam a cor desejada. No caso da Figura 2.2, calcularíamos a média e a variância dos valores dos pixels da cor da pele e, consequentemente, obteríamos a distribuição.

Por fim, dada a existência da distribuição, podemos construir o segmentador estatístico. Para classificar os pixels de uma nova imagem, podemos utilizar o valor da função de densidade de probabilidade (FDP) dos mesmos. Nesse sentido, estabelecemos um limiar e, caso o valor da FDP do pixel seja superior a esse limiar, o classificamos como objeto; caso contrário, o classificamos como fundo.

Outro segmentador de imagens similar também baseado em aprendizagem supervisionada, no entanto mais robusto, pode ser construído utilizando algoritmos de aprendizagem de máquina [2, 9, 17]. Nessa abordagem, os atributos de cores são utilizados como dados de entrada para os algoritmos, que, por sua vez, classificam essa entrada como pertencente a uma das classes conhecidas. Como exemplo, podemos ter duas classes, uma associada aos objetos de interesse e outra ao fundo.

2.1.2 Descrição de objetos

A descrição de objetos contidos em imagens pode ser realizada através de suas características internas e externas, obtidas pelo processo de extração de características. As características internas correspondem aos pixels que compõem o objeto em si, enquanto as características externas referem-se à fronteira dele. Os descritores baseados nas características internas e externas podem ser chamados de descritores de região e contorno, respectivamente [46].

Dentre os descritores de região, encontram-se os momentos estatísticos – aqui denominados momentos estatísticos simples –, os momentos estatísticos centralizados e os momentos estatísticos centrais normalizados. Cada momento, independentemente de sua categoria, pertence a uma determinada ordem e produz um valor real para uma dada entrada. Em visão computacional, essa entrada geralmente trata-se de uma imagem digital [29, 49].

A Equação 2.1.2 define a fórmula geral para o cálculo dos momentos estatísticos simples de uma imagem, em que $I(x, y)$ corresponde à intensidade do pixel localizado na coordenada (x, y) da imagem e W e H correspondem, nesta ordem, à largura e à altura da imagem.

$$m_{pq} = \sum_{x=1}^{W} \sum_{y=1}^{H} x^p y^q I(x, y) \tag{2.1.2}$$

A ordem do momento estatístico corresponde à soma de seus parâmetros p e q. Por exemplo, os momentos m_{12} e m_{30} pertencem à ordem 3, enquanto os momentos m_{22} e m_{13} à ordem 4. As Equações 2.1.3, 2.1.4 e 2.1.5 ilustram todos os momentos de ordens 0, 1 e 2, respectivamente.

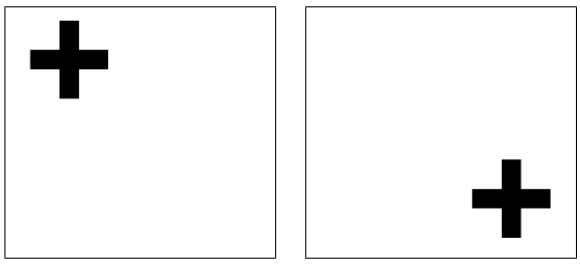
$$m_{00} = \sum_{x=1}^{W} \sum_{y=1}^{H} I(x, y) \tag{2.1.3}$$

$$\begin{aligned} m_{10} &= \sum_{x=1}^{W} \sum_{y=1}^{H} x I(x, y) \\ m_{01} &= \sum_{x=1}^{W} \sum_{y=1}^{H} y I(x, y) \end{aligned} \tag{2.1.4}$$

$$\begin{aligned} m_{20} &= \sum_{x=1}^{W} \sum_{y=1}^{H} x^2 I(x, y) \\ m_{11} &= \sum_{x=1}^{W} \sum_{y=1}^{H} xy I(x, y) \\ m_{02} &= \sum_{x=1}^{W} \sum_{y=1}^{H} y^2 I(x, y) \end{aligned} \tag{2.1.5}$$

Embora sejam úteis em várias aplicações, os momentos estatísticos simples não são invariantes à translação. Ou seja, o valor produzido por um momento ao se utilizar uma imagem com

apenas um objeto pode ser diferente do valor produzido por esse mesmo momento utilizando outra imagem com o mesmo objeto em posição diferente. Por exemplo, observe que as Figuras 2.3(a) e 2.3(b) contêm o mesmo objeto, mas em posições diferentes. No entanto, o valor produzido pelo momento m_{11} é 145.370 para a primeira imagem e 586.290 para a segunda.



(a) Imagem original (b) Imagem transladada

Figura 2.3: Ilustração de um mesmo objeto em posições diferentes.

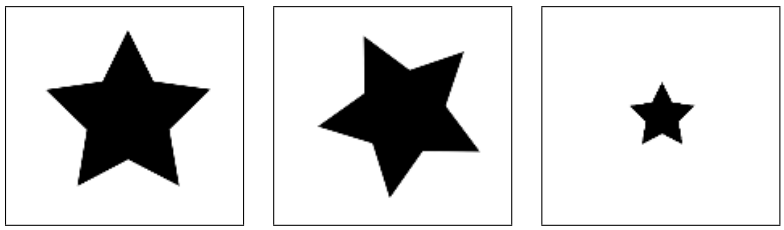
Como o enfoque é a descrição de objetos, idealmente, os valores de um momento para um mesmo objeto em posições diferentes deveriam ser iguais. Em outras palavras, os momentos devem produzir valores invariantes à translação. Em contrapartida aos momentos estatísticos simples, os momentos estatísticos centralizados apresentam essa invariabilidade e podem ser definidos de acordo com a Equação 2.1.6. Como exemplo, a Equação 2.1.7 define os momentos estatísticos centralizados de ordem 1.

$$\mu_{pq} = \sum_{x=1}^{W}\sum_{y=1}^{H}(x-\bar{x})^p(y-\bar{y})^q I(x,y) \tag{2.1.6}$$

$$\begin{aligned}\mu_{10} &= \sum_{x=1}^{W}\sum_{y=1}^{H}(x-\bar{x})I(x,y) \\ \mu_{01} &= \sum_{x=1}^{W}\sum_{y=1}^{H}(y-\bar{y})I(x,y)\end{aligned} \tag{2.1.7}$$

Para o exemplo acima citado, como temos invariância nos momentos estatísticos centralizados, os valores produzidos por μ_{10} tanto para a primeira quanto para a segunda imagem são zero. Contudo, assim como os momentos estatísticos simples, esses momentos não são invariantes a mudanças de rotação e escala. Como exemplo, considere as imagens exibidas nas Figuras 2.4(a), 2.4(b) e 2.4(c). Embora essas imagens contenham o mesmo objeto, o valor produzido por determinado momento μ_{xy} para esse objeto na primeira imagem pode ser diferente dos valores produzidos por μ_{xy} para esse mesmo objeto nas outras imagens.

Os momentos estatísticos centrais normalizados, definidos na Equação 2.1.8, são invariantes à escala e, com base neles, é constituído um conjunto especial composto de sete momentos – denominados *momentos invariantes* ou *momentos de Hu* – que, além de serem invariantes à escala, são invariantes à rotação. A Equação 2.1.9 exibe o primeiro momento de Hu (φ_1). Mais detalhes sobre esses sete momentos podem ser encontrados em [29]. A Equação 2.1.10 exibe os momentos estatísticos centrais normalizados de ordem 2.



(a) Imagem original (b) Imagem rotacionada (c) Imagem reduzida

Figura 2.4: Ilustração de um mesmo objeto em ângulos e tamanhos diferentes.

$$\eta_{pq} = \frac{\mu_{pq}}{\mu_{00}^{\frac{p+q}{2}+1}} \tag{2.1.8}$$

$$\varphi_1 = \eta_{20} + \eta_{02} \tag{2.1.9}$$

$$\begin{aligned} \eta_{20} &= \frac{\mu_{20}}{\mu_{00}^2} \\ \eta_{11} &= \frac{\mu_{11}}{\mu_{00}^2} \\ \eta_{02} &= \frac{\mu_{02}}{\mu_{00}^2} \end{aligned} \tag{2.1.10}$$

Com base nesses momentos, podemos encontrar alguns valores estatísticos referentes à imagem, que podem ser chamados de funções definidas por momentos [49], que podem ser utilizados como valores discriminantes. Dentre eles, para imagens binárias, temos o centro de massa – definido como as coordenadas x e y (x_c,y_c) na Equação 2.1.11 –, a variância nos eixos x e y (Equação 2.1.12), a excentricidade da elipse correspondente à imagem e a orientação dessa elipse, que são definidas na Equação 2.1.13 como ϵ e θ, respectivamente.

$$\begin{aligned} x_c &= \frac{m_{10}}{m_{00}} \\ y_c &= \frac{m_{01}}{m_{00}} \end{aligned} \tag{2.1.11}$$

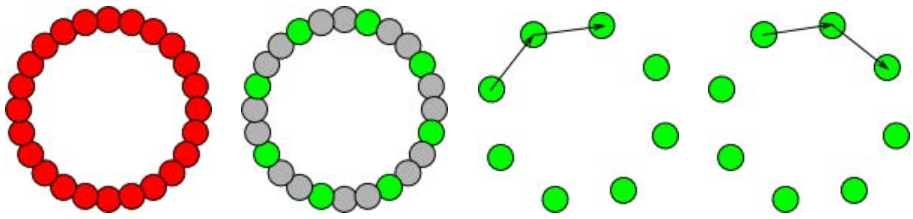
$$\begin{aligned} \sigma_x^2 &= \frac{\mu_{20}}{m_{00}} \\ \sigma_y^2 &= \frac{\mu_{02}}{m_{00}} \end{aligned} \tag{2.1.12}$$

$$\begin{aligned} \tan\theta &= \frac{\mu02 - \mu20 - 2\mu11 + \lambda}{\mu02 - \mu20 + 2\mu11 - \lambda} \\ \epsilon^2 &= \frac{\mu02 + \mu20 + \lambda}{\mu02 + \mu20 - \lambda} \\ \lambda &= \sqrt{(\mu20 - \mu02)^2 + 4\mu_{11}^2} \end{aligned} \tag{2.1.13}$$

Outra forma de descrever objetos em imagens é através do seu contorno, como referido anteriormente. De acordo com a denominação de Truyenque em [51], a k-curvatura é um cálculo de curvatura que leva em conta apenas alguns pontos do contorno do objeto e pode ser utilizado na sua descrição. O cálculo considera apenas alguns pontos com o intuito de minimizar a influência dos ruídos na descrição dos objetos. De forma geral, o contorno do objeto é descrito através de um conjunto de ângulos formados por certos pontos do contorno.

Para melhor compreensão, considere o conjunto completo de pontos do contorno como $P = \{p_1, p_2, p_3, ..., p_n\}$, em que $p_i = (x_i, y_i)$. Utilizando-se três pontos do contorno, são criados dois vetores e calculado o ângulo formado por eles. Partindo de um ponto j, outros dois pontos são selecionados, sendo eles p_j, p_{j+k} e p_{j+2k}, utilizando uma constante inteira k previamente definida.

Por exemplo, suponha que a Figura 2.5(a) exibe os pontos do contorno de determinada imagem. Tomando o valor de k como 3, selecionamos o ponto inicial do contorno e os pontos que estão a k pontos de distância dos outros, como ilustrado na Figura 2.5(b), em que os pontos em destaque correspondem aos pontos selecionados. Por fim, constroem-se dois vetores para cada trio de pontos consecutivos, como demonstrado nas Figuras 2.5(c) e 2.5(d), em que são mostrados os dois vetores do primeiro e do segundo trio, nessa ordem.



(a) Conjunto completo de pontos do contorno

(b) Pontos selecionados utilizando $k = 3$

(c) Exemplo de vetores para três pontos selecionados

(d) Exemplo de vetores para outros três pontos selecionados

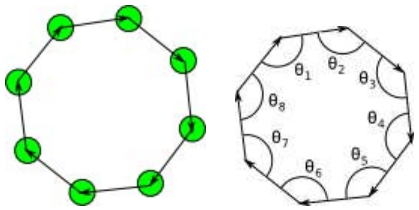
Figura 2.5: Exemplo de seleção de pontos de contorno para cálculo de curvaturas.

Após a construção dos pares de vetores, são calculados os ângulos formados entre os pares. Esses ângulos são obtidos através dos valores de θ na Equação 2.1.14, em que v e w correspondem aos dois vetores construídos. Mais detalhes relacionados à realização desses cálculos podem ser encontrados em [46]. A Figura 2.6 apresenta os vetores para os pontos selecionados do contorno e os ângulos para cada par de vetores, denominados como θ_i, sendo $i = \{1, 2, ..., 8\}$. Logo, representaríamos esse objeto com essa técnica através dos valores desses oito ângulos.

$$\theta = \cos^{-1}\left(\frac{vw}{|v||w|}\right) \tag{2.1.14}$$

Como os tamanhos dos objetos podem ser variáveis em determinadas aplicações, a quantidade de ângulos produzidos torna-se também variável. No exemplo anterior, o tamanho do objeto interferiu no número de ângulos produzidos, pois, se o objeto fosse maior, provavelmente teríamos mais de 8 ângulos. Nesses casos, para uniformizar o número de características produzidas, pode-se criar um histograma de ângulos. Desse modo, independentemente do número de ângulos produzidos pelo objeto, sempre teremos o mesmo número de características.

Para construir esse histograma, devemos criar grupos com um ou mais ângulos e, em seguida,



(a) Conjunto de vetores (b) Ângulos entre os pares de vetores

Figura 2.6: Exemplo de construção de vetores e ilustração de ângulos entre eles.

registrar a frequência de cada ângulo no seu respectivo grupo. Como estamos lidando com ângulos, sempre teremos valores não negativos menores que 360. Nesse caso, podemos, por exemplo, criar empiricamente 6 grupos, em que cada grupo contém 60 ângulos. A Tabela 2.1 apresenta esses 6 grupos com os respectivos intervalos de ângulos.

	Ângulos
1	[0, 60[
2	[60, 120[
3	[120, 180[
4	[180, 240[
5	[240, 300[
6	[300, 360[

Tabela 2.1: Grupos de ângulos.

Portanto, dada a existência de um histograma de ângulos, podemos padronizar o número de características de modo independente da quantidade de ângulos existentes. Por exemplo, dois objetos cujos ângulos são (20, 110, 8, 300 e 180) e (100, 180, 12, 315, 90, 85, 226, 94, 60 e 47) podem ser representados por meio de 6 características, como mostra a Tabela 2.2.

	Ângulos	**Objeto 1**	**Objeto 2**
1	[0, 60[	2	1
2	[60, 120[	1	6
3	[120, 180[	0	0
4	[180, 240[	1	2
5	[240, 300[	0	0
6	[300, 360[	1	1

Tabela 2.2: Histogramas dos dois objetos dados como exemplo considerando 6 grupos de ângulos determinados arbitrariamente. Os valores para os dois objetos em cada intervalo foram obtidos por meio de contagem simples de ocorrência.

Adicionalmente, existem outras características relacionadas a forma que auxiliam na descrição de objetos em imagens. Dentre elas, temos o fator de forma (*form factor*), a circularidade (*roundness*), a relação entre eixos (*aspect ratio*) e a densidade (*compactness*), que também serão utilizadas neste trabalho. Segue abaixo um breve descritivo sobre cada uma dessas caracterísi-

cas. Maiores detalhes e outras características podem ser encontrados em [5, 16, 42].

O fator de forma de um objeto corresponde à divisão da área desse objeto pela área de uma circunferência cujo perímetro corresponde ao perímetro do objeto. A área do objeto pode ser encontrada computacionalmente na imagem. Para encontrar a área da circunferência, é necessário encontrar seu raio. Dessa circunferência, é conhecido apenas seu perímetro, que é o mesmo do objeto e que também pode ser encontrado na imagem. Por meio desse perímetro[2], podemos encontrar o raio da circunferência e, consequentemente, calcular sua área. Desse modo, temos que o raio da circunferência é igual a $\frac{P}{2\times\pi}$, em que P é o perímetro do objeto. Com isso, encontramos a área da circunferência e, por fim, o fator de forma, como apresentado, nessa ordem, nas Equações 2.1.15 e 2.1.16.

$$A_{circ} = \pi \times R^2 = \pi \times \left(\frac{P}{2 \times \pi}\right)^2 = \frac{\pi \times P^2}{4 \times \pi^2} = \frac{P^2}{4 \times \pi} \tag{2.1.15}$$

$$FF = \frac{A_{obj}}{A_{circ}} = \frac{A_{obj}}{\frac{P^2}{4\times\pi}} = \frac{A_{obj} \times 4 \times \pi}{P^2} \tag{2.1.16}$$

A circularidade mede o alongamento do objeto. Podemos encontrá-la de forma similar ao cálculo do fator de forma. No entanto, divide-se a área do objeto pela área de uma circunferência cujo diâmetro é igual ao comprimento do objeto. A definição de comprimento para um objeto não é única. Aqui, para encontrar esse comprimento, primeiramente, encontramos uma elipse com o mesmo momento de segunda ordem do objeto. O maior eixo dessa elipse é o comprimento do objeto. Portanto, a dificuldade ao encontrar a circularidade de um objeto está em encontrar seu comprimento, pois, através dele, encontramos o raio da circunferência, calculamos sua área e, posteriormente, sua circularidade. Felizmente, esse comprimento também pode ser encontrado de forma computacional [49]. Assim, a circularidade pode ser obtida por meio da Equação 2.1.17.

$$C = \frac{A_{obj}}{A_{circ}} = \frac{A_{obj}}{\pi \times R^2} = \frac{A_{obj}}{\pi \times (\frac{comprimento}{2})^2} = \frac{A_{obj}}{\frac{\pi \times comprimento^2}{4}} = \frac{4 \times A_{obj}}{\pi \times comprimento^2} \tag{2.1.17}$$

Similarmente à circularidade, a relação entre eixos também mede o alongamento do objeto, porém, por meio de outras medidas do objeto. Para calculá-la, encontram-se os dois eixos da elipse correspondente ao objeto e divide-se o maior pelo menor. Assim como o maior eixo da elipse, o menor eixo também pode ser encontrado. Apenas para formalização, a Equação 2.1.18 define a relação aqui descrita, em que $largura$ corresponde ao menor eixo da elipse, e a Equação 2.1.19 formaliza a densidade do objeto, em que C corresponde à circularidade desse objeto.

$$AR = \frac{comprimento}{largura} \tag{2.1.18}$$

$$D = \sqrt{C} \tag{2.1.19}$$

[2] O perímetro da circunferência corresponde a $2 \times \pi \times R$, em que R é o raio.

2.2 Classificação automática

As informações provenientes da etapa de extração de características podem ser utilizadas na classificação automática de objetos com auxílio do reconhecimento de padrões, que se trata de uma disciplina científica cujo objetivo é a criação de teorias e técnicas que permitam a classificação de objetos, ou padrões, dentre um conjunto de categorias ou classes [50]. Dependendo da aplicação, esses objetos podem ser imagens, sequências de caracteres, sons ou qualquer outro tipo de sinal (geralmente digitalizado) capturado através dos sensores de um sistema computacional.

No caso deste projeto, o reconhecimento de padrões será utilizado para classificar automaticamente determinados comportamentos animais em sequências de imagens, como será descrito posteriormente. Contudo, por se tratar de classificações automáticas realizadas sem participação humana, torna-se necessária a avaliação dessa classificação, a fim de se estimar sua qualidade. Existem métodos eficazes para essa avaliação, como as taxas de erro e as curvas ROC (*Receiver Operating Characteristics*). Segue abaixo uma breve explanação sobre reconhecimento de padrões e avaliação de classificadores para contextualização do leitor.

2.2.1 Reconhecimento de padrões

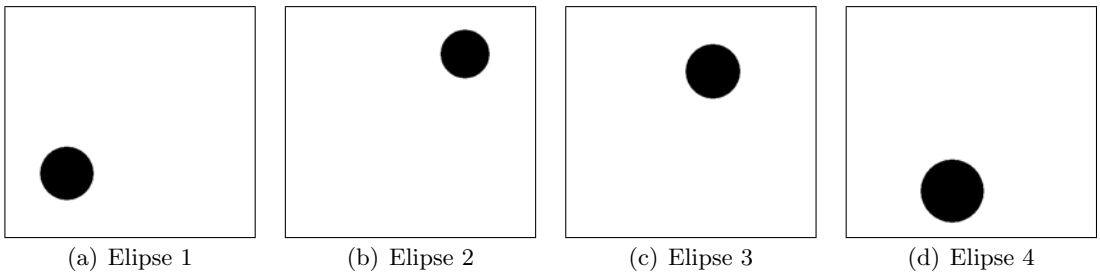
Embora o reconhecimento de padrões possa ser aplicado a problemas sem qualquer relação com imagens e visão, existe uma rica interseção entre essa área e a área da visão computacional, uma vez que o reconhecimento de padrões é uma importante etapa em boa parte dos problemas de visão computacional. Este trabalho se encaixa nessa interseção, em que temos a utilização dos conceitos de visão computacional – que basicamente nos auxiliam na compreensão do conteúdo das imagens – e a aplicação dos modelos de Markov ocultos (descritos no próximo capítulo) como técnica principal para o reconhecimento de padrões.

Com base no reconhecimento de padrões, podemos classificar algumas características – geralmente extraídas através de sensores – entre uma das classes conhecidas. No caso da visão computacional, esses sensores podem ser câmeras e as características a se classificar podem ser os valores obtidos na descrição de objetos. Por exemplo, considere que, de cada uma das imagens exibidas na Figura 2.7, foram extraídos os momentos m_{00} e μ_{11}. A Tabela 2.3 exibe os resultados para cada um dos objetos.

A Tabela 2.3 exibe oito conjuntos de características, sendo 4 deles referentes às elipses e outros 4 aos retângulos. A cada um desses conjuntos é atribuído o nome de instância, ou seja, temos 8 instâncias nesse exemplo. Para esse exemplo, essa coleção de instâncias poderia ser utilizada para treinar o sistema. Nesse caso, essa coleção seria chamada de conjunto de treinamento e suas instâncias seriam instâncias de treinamento.

No problema acima citado, temos duas classes: *Elipse* e *Retângulo*. Analisando a Tabela 2.3, podemos notar que é possível associar cada uma das oito instâncias a uma das duas classes por meio dos valores de suas características extraídas previamente. Observe que os valores da característica m_{00} de todas as instâncias da classe *Elipse* são menores que 11000. Analogamente, os valores da característica μ_{11} também são menores que 265.00. Logo, por meio de análise empírica, descobrimos dois padrões no conjunto de treinamento. Em outras palavras, essa foi a etapa de reconhecimento de padrões.

Como dito, com base nesse reconhecimento, podemos classificar novas instâncias observando apenas os valores de suas características. Como exemplo, considere a existência de uma fi-



(a) Elipse 1 (b) Elipse 2 (c) Elipse 3 (d) Elipse 4

(e) Retângulo 1 (f) Retângulo 2 (g) Retângulo 3 (h) Retângulo 4

Figura 2.7: Imagens de exemplo das quais serão extraídas suas características.

Objeto	**Características**		**Classe**
	m_{00}	μ_{11}	
Elipse 1	7796	222.75	Elipse
Elipse 2	6445	180.00	Elipse
Elipse 3	8106	227.31	Elipse
Elipse 4	10848	263.25	Elipse
Retângulo 1	20000	4925.25	Retângulo
Retângulo 2	20000	4925.25	Retângulo
Retângulo 3	17563	4320.00	Retângulo
Retângulo 4	15400	3781.00	Retângulo

Tabela 2.3: Objetos com os valores extraídos das respectivas características.

gura qualquer. Dessa figura, sabe-se apenas que o valor das características m_{00} e μ_{11}, que são 18963 e 4681.15, respectivamente. Com base nos valores dessas características e no conjunto de treinamento supracitado, podemos inferir que essa figura é um retângulo, mesmo sem observá-la visualmente. A essa etapa pode-se atribuir o nome de classificação.

Em classificadores denominados supervisionados, temos a etapa de treinamento, em que o classificador "aprende" a identificar os padrões com base em um conjunto de instâncias (com suas características e as respectivas classes). Após essa etapa, idealmente, o classificador deve estar apto a classificar corretamente as informações de entrada, mesmo que essa entrada não pertença ao conjunto de instâncias utilizado no treinamento. Logo, torna-se importante para as aplicações de modo geral que o classificador tenha uma capacidade de generalizacão adequada.

No entanto, geralmente quando há treinamento excessivo do classificador ou quando algumas amostras de treinamento não são acessíveis, essa generalização não ocorre. Com isso, o desempenho do classificador tende a ser alto para instâncias do conjunto de treinamento e baixo para instâncias desconhecidas. A essa situação, em que o classificador é ajustado demasiadamente a

um conjunto específico de instâncias, é dado o nome *overfitting*.

2.2.2 Avaliação de classificadores

Como realizado anteriormente, com base no reconhecimento de padrões, é possível realizar a classificação automática de determinadas instâncias entre diversas classes previamente conhecidas. Embora essa classificação possa ser bastante útil em muitas aplicações, é importante que o classificador que as produziu seja avaliado, para que se possa identificar sua precisão e, consequentemente, prever suas possíveis falhas dentro da aplicação onde ele foi empregado. Essa avaliação pode ser realizada por meio do estudo das matrizes de confusão derivadas dos resultados produzidos pelo classificador em análise [35].

Uma matriz de confusão (também conhecida como matriz de contingência) apresenta de forma resumida os resultados das classificações realizadas por um classificador. Por exemplo, considere a existência de 10 objetos geométricos cujas classes reais são conhecidas. Desses objetos, são extraídas suas características, que são posteriormente enviadas a um classificador. Com base nessas características, o classificador predirá a qual classe as características pertencem (ou mais se assemelham). A Tabela 2.4 mostra cada um dos objetos com sua respectiva classe real e a classe predita pelo classificador. Por exemplo, o primeiro objeto é um triângulo, mas foi classificado como um retângulo; o segundo é uma elipse classificada como triângulo; e o terceiro é um retângulo corretamente classificado.

Objeto	**Real**	**Predita**
1	Triângulo	Retângulo
2	Elipse	Triângulo
3	Retângulo	Retângulo
4	Triângulo	Triângulo
5	Triângulo	Elipse
6	Elipse	Elipse
7	Retângulo	Elipse
8	Retângulo	Retângulo
9	Elipse	Elipse
10	Retângulo	Retângulo

Tabela 2.4: Lista de objetos para análise de determinado classificador. A segunda coluna apresenta a classe real do objeto e a classe predita pelo classificador é exibida na terceira coluna.

A Tabela 2.5 exibe a matriz de confusão para o exemplo acima, em que as colunas representam as classes preditas pelo classificador, e as linhas as classes reais. Como exemplo, note na Tabela 2.4 que, dentre os 3 triângulos, apenas um foi classificado corretamente, pois um foi classificado como uma elipse e outro como um retângulo. Essa informação pode ser observada resumidamente na primeira linha da matriz de confusão. Da mesma forma, o objeto 2 – que foi classificado incorretamente – está na primeira coluna da segunda linha da matriz de confusão, já os objetos 6 e 9 na segunda coluna.

Na última coluna da matriz de confusão, são apresentados os números reais de instâncias de cada classe. No exemplo anterior, de acordo com a Tabela 2.4, tínhamos 3 triângulos, 3 elipses e 4 retângulos. Essa informação também pode ser conferida na última coluna da respectiva matriz

	Triângulo	Elipse	Retângulo	
Triângulo	1	1	1	3
Elipse	1	2	0	3
Retângulo	0	1	3	4
	2	4	4	10

Tabela 2.5: Exemplo de matriz de confusão.

de confusão. Por outro lado, a última linha apresenta a quantidade de objetos classificados para cada classe. Por exemplo, dentre todos os objetos, dois foram classificados como triângulo, quatro como elipse e os outros quatro como retângulo. Por fim, o elemento da última linha e coluna da matriz se refere ao número total de instâncias classificadas.

Sem perda de generalidade, trabalhando apenas com duas classes, podemos visualizar na matriz de confusão as mesmas informações com denominações diferentes, sendo elas os números de falsos-positivos (FP), verdadeiros-positivos (TP), falsos-negativos (FN) e verdadeiros-negativos (TN). Para isso, consideramos uma classe como positivo e a outra como negativo.

Um TP é uma instância cuja classe real é positivo e a classe predita pelo classificador também é positivo, enquanto um FN é uma instância da classe positivo que fora classificada como negativo. Analogamente, um TN e um FP ocorrem, respectivamente, quando uma instância da classe negativo é corretamente ou incorretamente classificada. A Tabela 2.6 exibe uma matriz de confusão para duas classes, indicando cada uma dessas informações. Adicionalmente, nesta ordem, POS e NEG representam o número de instâncias realmente positivas e negativas; PP e PN o número de instâncias classificadas como positivas e negativas; e N o número total de instâncias.

	Positivo	Negativo	
Positivo	TP	FN	POS
Negativo	FP	TN	NEG
	PP	PF	N

Tabela 2.6: Matriz de confusão para duas classes.

Com base nas quantidades de TP, TN, FP e FN de uma matriz de confusão, é possível derivar valores que podem auxiliam na comparação entre classificadores. Dentre eles, temos duas taxas de erro e acerto, que são equivalentes e correspondem, respectivamente, às proporções de instâncias incorretamente classificadas com relação ao conjunto completo $\left(\frac{FP+FN}{N}\right)$ e corretamente classificadas $\left(\frac{TP+TN}{N}\right)$. Embora bastante utilizadas, as taxas de erro e acerto podem não ser adequadas na comparação de classificadores para determinadas aplicações [35].

Por exemplo, considerando uma aplicação de reconhecimento de larvas vivas e mortas em que a classe morta é considerada como positiva, suponha que as taxas de FP e FN de dois classificadores diferentes após a análise de larvas submetidas a um determinado larvicida em estudo sejam respectivamente $FP_1 = 99$, $FN_1 = 1$, $FP_2 = 2$ e $FN_2 = 98$. Com essas taxas, a taxa de erro de ambos os classificadores é a mesma, independentemente do número de instâncias do conjunto completo. Utilizando a taxa de erro como critério de avaliação, podemos inferir que os desempenhos desses classificadores são equivalentes. No entanto, para a aplicação em

questão, provavelmente isso não é válido.

Como um dos objetivos da aplicação citada como exemplo é analisar a eficácia de larvicidas com base nas taxas de mortalidade das larvas, provavelmente, é preferível que um larvicida eficaz seja classificado como ineficaz que um larvicida ineficaz seja classificado como eficaz, pois, nesse último caso, larvicidas reconhecidamente eficazes podem ser substituídos pelo novo larvicida erroneamente classificado como eficaz, prejudicando o controle do mosquito transmissor da doença.

Note que o primeiro classificador classificou erroneamente 99 larvas vivas como mortas e 1 larva morta como viva. Já o segundo classificou 2 larvas vivas como mortas e 98 mortas como vivas. Assim, a probabilidade de um larvicida ineficaz ser aprovado é maior se o primeiro classificador for utilizado, pois, em seus testes, muitas larvas vivas foram consideradas como mortas (que é o esperado de larvicidas eficazes). Logo, em casos de erro, nesse exemplo, falsos-negativos são preferíveis a falsos-positivos. Isso nos faz questionar a conclusão anterior baseada apenas nas taxas de erro, que indicou a equivalência entre os classificadores.

Alternativamente, podemos utilizar representações gráficas dos resultados de classificadores, o que permite uma melhor visualização do comportamento dos classificadores. Os gráficos ROC possibilitam essa visualização. As primeiras utilizações dos gráficos ROC em aprendizagem de máquina ocorreram em meados dos anos 90, embora eles já tenham sido utilizados anteriormente em outras áreas [12]. Esses gráficos viabilizam a visualização, organização e seleção de classificadores com base nos seus desempenhos.

A Figura 2.8 ilustra um gráfico de ROC referente aos resultados de classificação de um determinado classificador. Esse gráfico é bidimensional, sendo que os eixos X e Y apresentam as taxas de falsos-positivos e verdadeiros-positivos, respectivamente. Em seguida, serão apresentados detalhes sobre a construção e a análise de curvas de ROC aplicadas na avaliação de classificadores. Maiores detalhes sobre esse assunto podem ser encontrados em [3, 12, 35].

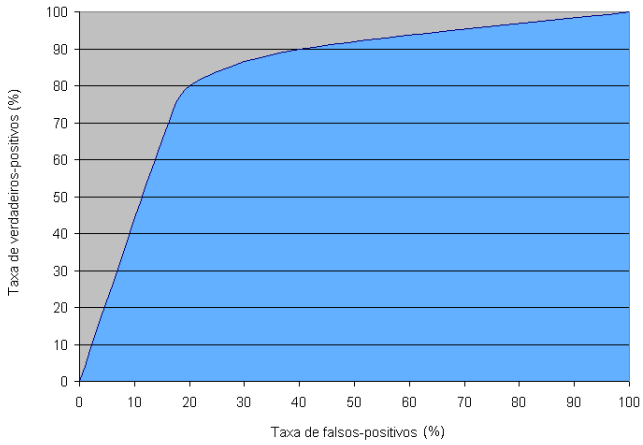


Figura 2.8: Exemplo de curva de ROC.

Ao avaliar um conjunto de características, alguns classificadores produzem uma pontuação (comumente referida como *score*) e, sobre essa pontuação, é aplicado um limiar que permite a classificação da entrada dentre um conjunto de classes. Por exemplo, na Tabela 2.4, foram exibidas as classes reais dos objetos e as classes preditas pelo classificador utilizado. Nesse caso, em vez de produzir a classe, o classificador pode produzir apenas uma pontuação – como ocorre

nas redes neurais –, que é posteriormente processada e, consequentemente, produz a classificação da entrada.

Abaixo, na Tabela 2.7, apresentamos 10 instâncias e, para cada uma, é exibida sua classe real (positivo ou negativo) e o valor da pontuação produzida pelo classificador ao processá-la. Com base nessa pontuação, são apresentadas duas classificações, sendo que cada uma utiliza um valor de limiar (λ). Todos os valores abaixo do limiar são classificados como negativos e o restante como positivo. Note que, se assumirmos um limiar de 0.8, teremos 2 classificações incorretas (objetos 3 e 8), que não ocorrem ao utilizarmos o limiar de 0.7.

Instância	**Real**	**Pontuação**	$\lambda = 0.8$	$\lambda = 0.7$
1	Positivo	0.81	Positivo	Positivo
2	Positivo	0.93	Positivo	Positivo
3	Positivo	0.75	Negativo	Positivo
4	Negativo	0.54	Negativo	Negativo
5	Negativo	0.43	Negativo	Negativo
6	Positivo	0.89	Positivo	Positivo
7	Positivo	0.98	Positivo	Positivo
8	Positivo	0.74	Negativo	Positivo
9	Negativo	0.20	Negativo	Negativo
10	Negativo	0.37	Negativo	Negativo

Tabela 2.7: Lista de objetos para análise de determinado classificador baseado em pontuação. A segunda coluna apresenta a classe real do objeto; a terceira apresenta a pontuação produzida para o classificador para cada objeto; a quarta e a quinta apresentam a classe predita pelo classificador utilizando limiares de 0.8 e 0.7, respectivamente.

O código abaixo descreve um método para construção da curva de ROC classificadores considerando um conjunto de instâncias.

Algoritmo 1: Algoritmo para criação de pontos da curva ROC.

Entrada: L = conjunto de exemplos; $f(i)$ = probabilidade do i-ésimo exemplo ser positivo; P = número de exemplos positivos; N = número de exemplos negativos
Saída : R = Lista de pontos ROC

```
ORDENADO ← ordenarDecrescentemente(L, f) ;
FP ← TP ← 0 ;
R ← () ;
F_ant ← -∞ ;
i ← 1 ;
while i ≤ |ORDENADO| do
    if f(i) ≠ F_ant then
        x ← FP/N ;
        y ← TP/P ;
        adicionarPonto(x, y, R) ;
        F_ant ← f(i) ;
    end
    if exemploPositivo(ORDENADO[i]) then
        TP ← TP + 1 ;
    else
        FP ← FP + 1 ;
    end
    i ← i + 1 ;
end
adicionarPonto(1, 1, R) ;
```

Dadas as curvas ROC de dois classificadores para um mesmo conjunto de instâncias, torna-se interessante compará-las, a fim de identificar qual deles é mais adequado para o problema. Uma forma de resumir uma curva de ROC a um único valor é através do cálculo da área abaixo da curva, chamado de AUC (*Area Under Curve*) [12]. Essa área pode ser observada em destaque na Figura 2.8. O método para obter essa área é similar ao método de construção da curva acima descrito e é apresentado abaixo.

Algoritmo 2: Algoritmo para cálculo da área da curva ROC.

Entrada: L = conjunto de exemplos; $f(i)$ = probabilidade do i-ésimo exemplo ser positivo; P = número de exemplos positivos; N = número de exemplos negativos
Saída : A = Área da curva ROC

```
ORDENADO ← ordenarDecrescentemente(L, f) ;
FP ← TP ← 0 ;
FP_ant ← TP_ant ← 0 ;
A ← 0 ;
F_ant ← -∞ ;
i ← 1 ;
while i ≤ |ORDENADO| do
    if f(i) ≠ F_ant then
        A ← A + areaTrapezio(FP, FP_ant, TP, TP_ant) ;
        F_ant ← f(i) ;
        FP_ant ← FP ;
        TP_ant ← TP ;
    end
    if exemploPositivo(ORDENADO[i]) then
        TP ← TP + 1 ;
    else
        FP ← FP + 1 ;
    end
    i ← i + 1 ;
end
A ← A + areaTrapezio(N, FP_ant, N, TP_ant) ;
A ← A / (P+N) ;
```

Paralelamente, outra forma de se comparar dois classificadores no espaço ROC é baseado em suas posições dentro desse espaço. Ao se analisar os resultados de classificação de determinado classificador, é possível obter as taxas de verdadeiros-positivos e falsos-positivos. Logo, é possível projetá-lo dentro do espaço ROC. Dessa forma, ao se comparar dois classificadores, teremos dois pontos nesse espaços. A Figura 2.9 mostra o espaço ROC com 5 pontos, representando 5 classificadores diferentes.

Alguns pontos do espaço ROC são dignos de destaque, como os pontos A, B e C. O ponto A representa um classificador que nunca classifica uma entrada como positivo, enquanto o ponto C é um classificador que sempre classifica as entradas como positivo. Por fim, o ponto B representa o classificador perfeito, em que todas as entradas foram corretamente classificadas. Os classificadores cujos pontos estão sobre a linha diagonal são considerados classificadores aleatórios. Aqueles que estão à direita e à esquerda dessa linha são, nesta ordem, piores e melhores que os classificadores aleatórios.

Ao se comparar dois classificadores dessa forma, pode se dizer que um é melhor que o outro se ele estiver acima e à esquerda do outro [35], como ocorre na Figura 2.9. Nela, o classificador representado pelo ponto D pode ser considerado como melhor que o classificador representado pelo ponto E, devido à sua posição no espaço. Contudo, existem várias formas de se analisar o espaço (e as curvas) de ROC. Além disso, é válido observar que a utilização de valores resumidos – como a AUC e a localização do respectivo ponto referente ao classificador no espaço ROC –, assim como a taxa de erro, oculta informações sobre o desempenho do classificador que, dependendo da

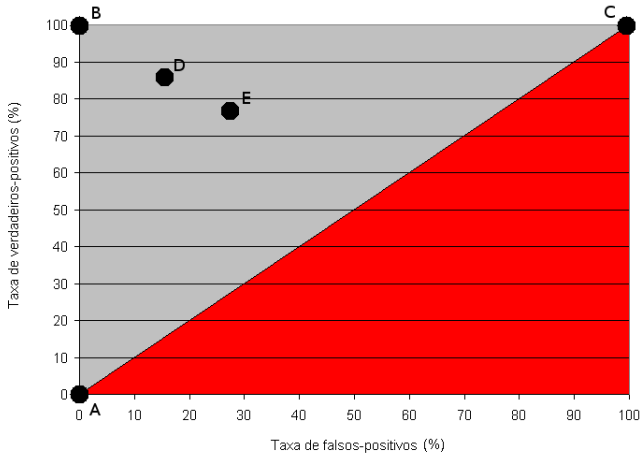


Figura 2.9: Espaço ROC.

aplicação, podem ser relevantes. Maiores detalhes sobre comparações de classificadores podem ser encontrados nas referências anteriormente citadas.

Em problemas com mais de duas classes, temos matrizes de confusão com dimensões superiores, em vez de matrizes 2×2 apresentadas recentemente. Para lidar com várias classes, podemos criar vários gráficos ROC, sendo um gráfico para cada classe, gerando várias matrizes de confusão bidimensionais [12]. Para isso, a classe em destaque é considerada como positivo e as outras classes como negativos. Outra possibilidade é combinar as classes aos pares considerando uma como positivo e a outra como negativo.

Para essa última abordagem, devido à existência de vários gráficos, o cálculo da área abaixo da curva é redefinido de acordo com a Equação 2.2.1, em que C corresponde ao conjunto de classes, c_i refere-se à i-ésima classe do conjunto C e $AUC(x, y)$ trata-se da AUC considerando as classes x e y.

$$AUC_{total} = \frac{2}{|C|(|C|-1)} \sum_{\{c_p, c_q\} \in C} AUC(c_p, c_q) \tag{2.2.1}$$

Capítulo 3

Modelos de Markov Ocultos

Antes de descrever o modelo de Markov oculto, é interessante conhecer sua origem, os processos de Markov. Em meados de 1907, Markov definiu e investigou algumas propriedades que hoje são conhecidas como processos de Markov [6]. A principal característica dos processos de Markov é a definição que toda a história passada está resumida no valor atual do processo. Em outras palavras, apenas o estado atual do processo influencia na próxima mudança de estado desse processo.

Essa teoria pode ser utilizada em estimações para determinados padrões em que existe uma estrutura que influencia a probabilidade do próximo evento ocorrer. Por exemplo, em alguns idiomas – como no Português e no Inglês – a probabilidade de se encontrar a letra u após a detecção da letra q é muito alta, considerando que praticamente sempre após a letra q é encontrada a letra u.

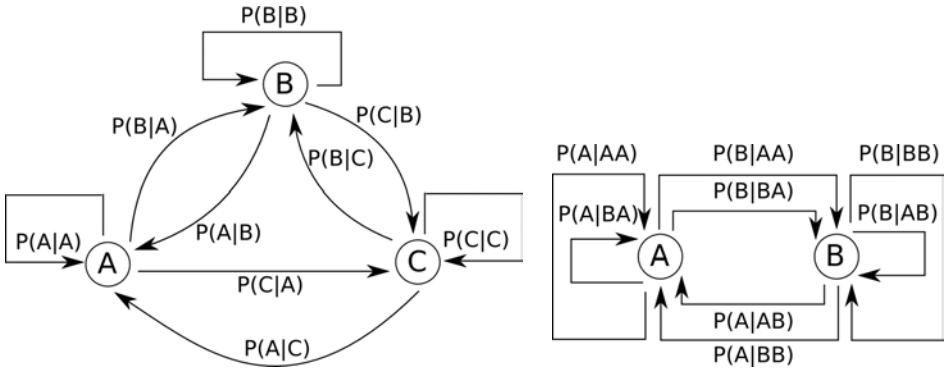
Com base nisso, podemos definir a cadeia de Markov, que é um processo de Markov cujo espaço de estados é discreto. As cadeias de Markov podem ser representadas graficamente como um grafo direcionado, em que os vértices são os estados e as arestas são as probabilidades de transição entre esses estados. Diz-se que, em uma cadeia de Markov de ordem j, a probabilidade do próximo evento ocorrer depende apenas dos j estados visitados anteriormente em um espaço de tempo discreto.

As Figuras 3.1(a) e 3.1(b) representam, respectivamente, cadeias de Markov de primeira e segunda ordem, em que $P(X|Y)$ é a probabilidade de ocorrência de X (estado futuro) dada a ocorrência de Y (conjunto de estados percorridos anteriormente).

Na Figura 3.1(a), existe apenas uma transição com origem em A e destino em B, pois a cadeia é de ordem 1, o que implica que apenas o estado anterior (A) interfere na ocorrência do próximo estado (B), desconsiderando o passado mais antigo. Associada a essa transição, temos $P(B|A)$, que é a probabilidade de ocorrência de B dada a ocorrência de A. Em nosso caso, $P(B|A)$ representa a probabilidade de transição do estado A para o estado B.

Já na Figura 3.1(b), como temos uma cadeia de ordem 2, existem duas transições de A para B. Associadas a essas transições, temos $P(B|AA)$ e $P(B|BA)$, que, analogamente, são as probabilidades de ocorrência de B dada a ocorrência de AA e BA, respectivamente. No nosso caso, ambas correspondem à probabilidade de transição do estado A para o estado B, mas $P(B|BA)$ considera a proveniência de B e $P(B|AA)$ a proveniência de A.

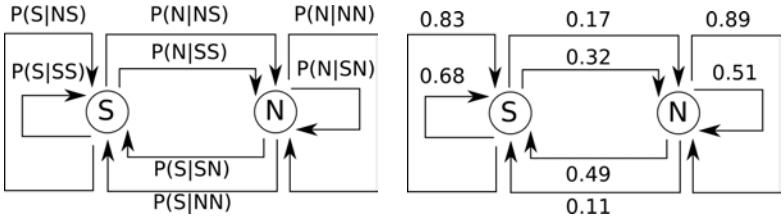
As sequências de observações em uma cadeia de Markov são conjuntos ordenados cujos ele-



(a) Cadeia de ordem 1 com 3 estados. (b) Cadeia de ordem 2 com 2 estados.

Figura 3.1: Exemplos de cadeias de Markov.

mentos são os próprios estados da cadeia. Por exemplo, suponha que, com base em conjuntos de observações do clima, foi observado que a probabilidade de ocorrência de chuva em determinado dia estava condicionada ao clima observado nos dois dias anteriores. Dessa forma, pode-se utilizar uma cadeia de Markov de segunda ordem para representar esse comportamento, como ilustra a Figura 3.2, que apresenta uma cadeia com um estado (S) indicando a ocorrência de chuva no dia e outro (N) indicando a não-ocorrência.



(a) Cadeia com possíveis transições. (b) Cadeia com probabilidades de transição.

Figura 3.2: Ilustração da cadeia de Markov para modelagem de comportamento climático com base nas observações de um conjunto de amostras.

Na Figura 3.2(a), cada transição entre os estados representa um evento possível. Por exemplo, a transição com origem no estado N e destino e S, assinalada com a probabilidade $P(S|SN)$, representa a probabilidade de chover em determinado dia, dado que no dia anterior não tenha chovido e que no dia anterior ao anterior tenha. A Figura 3.2(b) exibe a mesma cadeia, porém, com os valores das respectivas probabilidades que foram obtidas através da análise das amostras.

Assim, podemos calcular a probabilidade de ocorrência de chuva em determinado dia com base em observações referentes ao dia atual e ao anterior, como demonstrado na Tabela 3.1. A tabela exibe o estado observado no dia anterior (ontem), o estado atual (hoje) e as probabilidades de se observar S ou N no dia seguinte (amanhã), sendo que S significa a ocorrência de chuva e N o contrário.

Um modelo de Markov oculto é similar a uma Cadeia de Markov, porém, os elementos dos conjuntos de observações não pertencem ao conjunto de estados, mas a outro conjunto de símbolos, tornando a sequência de estados percorrida oculta ao observador. Esses símbolos são gerados pelos estados em cada instante de tempo, contudo, o estado gerador não é conhecido.

Ontem	**Hoje**	**Amanhã**	
		S	**N**
S	S	0.68	0.17
S	N	0.49	0.51
N	S	0.83	0.32
N	N	0.11	0.89

Tabela 3.1: Probabilidades de ocorrência de chuva no dia seguinte.

Logo, existem duas camadas estocásticas nos modelos de Markov ocultos.

A primeira camada estocástica é uma cadeia de Markov de primeira ordem, porém, não é diretamente observável como nas cadeias de Markov, em que cada estado é uma possível observação[1]. A segunda camada estocástica é um conjunto de probabilidades que indica, para cada estado, as probabilidades de emissão de cada símbolo do modelo.

Como citado, a sequência de estados percorrida em um modelo, dada uma sequência de observações, é oculta ao observador. Ou seja, tendo uma sequência de observações (conjunto ordenado de símbolos), não se sabe a sequência de estados percorrida pelo modelo para geração dessa sequência, mas somente um valor probabilístico desse caminho, e, por isso, o modelo é chamado de *oculto*.

Para melhor compreensão dessa diferença e de outros conceitos que serão expostos ao longo do texto, considere o seguinte exemplo. Suponha dois grupos mistos de pessoas, compostos por indivíduos que conseguem assobiar e outros que não conseguem. Contudo, todos esses indivíduos estão aptos a baterem palmas. No primeiro grupo, temos 3 pessoas que assobiam e 1 que não assobia e, no segundo, 2 assobiam e 2 não. Desses dois grupos são gravadas diversas sequências de assobios e palmas, sendo que a cada instante uma pessoa assobia ou bate palma e, em seguida, escolhe aleatoriamente outra pessoa (ou a si próprio) para fazer o mesmo.

Nesse contexto, dada uma sequência qualquer com palmas e assobios, quais as probabilidades dela ter sido gerada por cada um dos grupos? Esse problema poderia ser modelado com HMM's, de forma que cada grupo seria um HMM, os indivíduos seriam estados do respectivo modelo e os símbolos seriam "assobio" e "palma". Todos os estados (pessoas) possuiriam uma probabilidade de geração de cada símbolo.

Como exemplo, para os indivíduos que sabem assobiar, a probabilidade de geração de cada símbolo é de 50%, enquanto que, para os outros, a probabilidade de geração de uma palma é de 100% e de um assobio é, obviamente, 0%. As sequências de observações seriam conjuntos ordenados de símbolos (palmas e assobios), e não estados (pessoas). Dessa forma, podemos observar que, quando se utilizam HMM's, os estados percorridos não são conhecidos, pois não são mais observações, como ocorre nas cadeias de Markov. Para um símbolo qualquer emitido (palma ou assobio), não sabemos qual seu estado emissor (pessoa), apenas uma probabilidade para cada estado.

[1] O termo "observações" recebe diferentes significados quando aplicados a processos de Markov e a modelos de Markov ocultos. No primeiro, essas observações são os estados percorridos pelo modelo, enquanto que nos modelos de Markov ocultos são os símbolos gerados nos estados do modelo, porém, a sequência de estados não é conhecida.

3.1 Elementos do HMM

Um modelo de Markov oculto geralmente é definido por uma tripla, $\lambda = (A, B, \pi)$, em que A é o conjunto de probabilidades de transições de estados, B é o conjunto de probabilidades de emissão de símbolos para cada estado e π é o conjunto de probabilidades de cada estado iniciar a geração de símbolos. Os conjuntos A e B também podem ser representados como matrizes e o conjunto π como um vetor.

Para compreender a matriz de transição de estados (A), considere o exemplo anterior e, suponha ainda, as seguintes restrições. No primeiro grupo, apenas um dos indivíduos aptos a assobiar (denominado $a2$) pode escolher o indivíduo inapto para ser o próximo gerador do símbolo e, no segundo grupo, os indivíduos que não sabem assobiar (denominados $i2$ e $i3$) devem escolher sempre um indivíduo que consiga assobiar para gerar o próximo símbolo.

Assim, a Tabela 3.2 exibe os valores da matriz de probabilidades das transições de estados do primeiro grupo, em que as linhas e as colunas representam os estados de origem e destino das transições, respectivamente. Por exemplo, a probabilidade do indivíduo $a1$ escolher a si mesmo é de 33.33% e a probabilidade dele escolher o indivíduo $i1$ é nula. Da mesma forma, a Tabela 3.3 exibe as probabilidades do segundo grupo.

	a1	**a2**	**a3**	**i1**
a1	0.3333	0.3333	0.3333	0.00
a2	0.25	0.25	0.25	0.25
a3	0.3333	0.3333	0.3333	0.00
i1	0.25	0.25	0.25	0.25

Tabela 3.2: Exemplo de matriz de transição de estados do primeiro grupo.

	a4	**a5**	**i2**	**i3**
a4	0.25	0.25	0.25	0.25
a5	0.25	0.25	0.25	0.25
i2	0.5	0.50	0.0	0.0
i3	0.5	0.50	0.0	0.0

Tabela 3.3: Exemplo de matriz de transição de estados do segundo grupo.

A matriz B indica as probabilidades de emissão de símbolos para cada estado do modelo. A Tabela 3.4 apresenta os valores das probabilidades de B para o HMM referente ao primeiro grupo enquanto a matriz do segundo grupo é descrita na Tabela 3.5, lembrando que os indivíduos que sabem assobiar também conseguem bater palmas e escolhem de maneira aleatória qual desses dois sons irão emitir. Em ambas as tabelas, as linhas são os estados do modelo e as colunas indicam os símbolos.

Finalmente, o conjunto π define as probabilidades de cada estado ser o gerador do primeiro símbolo das sequências de observações, ou seja, a probabilidade de determinado estado iniciar as sequências de observações. Ainda no exemplo anterior, considere que apenas o indivíduo inapto a assobiar ($i1$) pode iniciar uma sequência de observações no primeiro grupo e, no segundo grupo, para cada sequência de observações, apenas um dos indivíduos aptos ($a4$ ou $a5$) – escolhido

	ASSOBIO	**PALMA**
a1	0.5	0.5
a2	0.5	0.5
a3	0.5	0.5
i1	0.0	1.0

Tabela 3.4: Exemplo de matriz de emissão de símbolos dos estados do primeiro grupo.

	ASSOBIO	**PALMA**
a4	0.5	0.5
a5	0.5	0.5
i2	0.0	1.0
i3	0.0	1.0

Tabela 3.5: Exemplo de matriz de emissão de símbolos dos estados do segundo grupo.

aleatoriamente – pode ser o gerador inicial. Assim, as probabilidades de ocorrência inicial dos estados do primeiro e do segundo HMM são descritas respectivamente nas Tabelas 3.6 e 3.7.

Estado	**Probabilidade**
a1	0.0
a2	0.0
a3	0.0
i1	1.0

Tabela 3.6: Vetor de probabilidades de ocorrência inicial dos estados do HMM do primeiro grupo.

Estado	**Probabilidade**
a4	0.5
a5	0.5
i2	0.0
i3	0.0

Tabela 3.7: Vetor de probabilidades de ocorrência inicial dos estados do HMM do segundo grupo.

A seguir, serão descritos alguns termos adotados que são comumente encontrados em textos da área e ao longo deste trabalho.

1. O HMM é representado pelo símbolo λ;
2. Os estados do modelo são denotados pelo conjunto $S = \{s_1, s_2, ..., s_N\}$, de tamanho N;
3. Os símbolos admitidos pelo modelo estão contidos no conjunto $V = \{v_1, v_2, ..., v_M\}$, de tamanho M, também conhecido como alfabeto do modelo;

4. Uma sequência de observações é denotada pelo conjunto ordenado $O = \{o_1, o_2, ..., o_T\}$, composto de T elementos quaisquer do conjunto V, em que T e o_t representam, respectivamente, o tamanho da sequência e o símbolo observado no instante t da sequência, tal que $1 \leq t \leq T$;

5. Quando conhecida, uma sequência de estados para determinada sequência observação é representada pelo conjunto $Q = \{q_1, q_2, ..., q_T\}$, composto por T elementos de S, em que q_t representa o estado gerador do t-ésimo símbolo da sequência de observações de tamanho T;

6. O vetor $\pi = \{\pi_1, \pi_2, ..., \pi_N\}$ apresenta um valor probabilístico para cada um dos N estados do modelo, sendo π_{estado} a probabilidade de *estado* ocorrer, dado que *estado* pode ser o índice do estado ou ele próprio. Por exemplo, a probabilidade do estado s_i ser inicial, considerando $S = \{s_1, s_2, ..., s_N\}$, pode ser descrita como π_i ou π_{s_i};

7. A matriz de transições de estados é representada por A_{NxN} e seus elementos são referenciados na forma $a_{(origem,destino)}$ (ou $a_{origem,destino}$), em que *origem* e *destino* correspondem nessa ordem aos estados de origem e destino da transição (ou aos índices desses estados).

8. A matriz de probabilidades de emissão de símbolos é representada por B_{NxM} e seus elementos são escritos como $b_{estado}(simbolo)$. As alternativas notacionais descritas nos dois itens anteriores também são válidas neste para *estado* e *simbolo*;

9. A probabilidade da sequência de observações O ter sido gerada pelo modelo λ é representada por $P(O|\lambda)$.

3.2 Incógnitas principais

Existem três incógnitas implícitas nos HMM's, cujas soluções contribuem para o funcionamento eficaz das aplicações que os utilizam [28], sendo elas a avaliação da observação, a melhor sequência de estados da observação e o treinamento dos modelos[2].

3.2.1 Avaliação da observação

O primeiro problema se refere à descoberta da probabilidade da sequência de observações O ter sido gerada por um determinado modelo λ. Esse tipo de situação pode ser muito frequente nas aplicações com HMM's. Como exemplo, no reconhecimento de voz, ao se produzir um fonema qualquer, essa entrada pode ser classificada como pertencente ao modelo que indicar a maior probabilidade ($P(\lambda|O)$); outro exemplo, mais relacionado com este trabalho, seria no reconhecimento de comportamentos animais, em que um comportamento é classificado de acordo com o modelo que apresentar a maior probabilidade, considerando a existência de um modelo para cada comportamento.

A maneira mais trivial de se calcular a probabilidade de determinada sequência de observações ter sido gerada por um modelo é através da verificação de todas as sequências de estados

[2] As três incógnitas descritas são citadas por alguns autores como os três problemas básicos dos HMM's, como o problema da avaliação, o problema do melhor caminho e o problema do treinamento.

de tamanho T possíveis, e, posteriormente, calcular a probabilidade de O ter sido gerada pelos respectivos estados dessas sequências (Q). Assim, a probabilidade da sequência O ter sido gerada pelos estados da sequência Q no modelo λ é descrita na Equação 3.2.1.

$$P(O|Q,\lambda) = \prod_{i=1}^{T} b_{q_i}(o_i) \tag{3.2.1}$$

Adicionalmente, a probabilidade da sequência de estados Q ocorrer em λ é a probabilidade do estado q_i ser inicial multiplicada às probabilidades de transição entre os estados da sequência, como mostra a Equação 3.2.2.

$$P(Q|\lambda) = \pi_{q_1} . \prod_{i=1}^{T-1} a_{(q_i,q_{i+1})} \tag{3.2.2}$$

Assim, a probabilidade da sequência Q ocorrer e gerar a observação O pode ser escrita como a probabilidade conjunta dos eventos descritos pelas Equações 3.2.1 e 3.2.2, como é demonstrado na Equação 3.2.3.

$$P(O,Q|\lambda) = P(Q|\lambda).P(O|Q,\lambda) = \pi_{q_1} . \prod_{i=1}^{T-1} a_{(i,i+1)} . \prod_{i=1}^{T} b_{q_i}(o_i) \tag{3.2.3}$$

Por fim, a probabilidade de O ter sido gerada pelo modelo é a soma da probabilidade acima para todas as sequências de estados possíveis em λ, da seguinte forma:

$$P(O|\lambda) = \sum_{\text{cada } Q} P(O|Q,\lambda).P(Q|\lambda) \tag{3.2.4}$$

Como o número total de arranjos de tamanho K (com repetição) utilizando x elementos é x^K, o número de possíveis sequências de estados de tamanho T utilizando os N estados é N^T. Para auxílio na observação do custo computacional do cálculo dessa probabilidade, a Equação 3.2.3 é reescrita na Equação 3.2.5.

$$\pi_{q_1} . \prod_{i=1}^{T-1} a_{(q_i,q_{i+1})} . \left(\prod_{i=1}^{T-1} b_{q_i}(o_i) \right) . b_{q_T}(o_T) \tag{3.2.5}$$

Dado que cada produtório $\prod_{i=1}^{T-1}$ realiza implicitamente $T-2$ multiplicações, temos que o cálculo da Equação 3.2.5 resulta em $1 + (T-2) + 1 + (T-2) + 1 = 2T - 1$ multiplicações para cada sequência de estados. Dessa forma, para realizar o cálculo de $P(O|\lambda)$ através das equações anteriores são necessárias $N^T.(2T-1)$ multiplicações e $N^T - 1$ somas, tornando-o um processo impraticável computacionalmente para alguns casos.

Para solução desse problema de maneira mais eficaz, existem dois métodos independentes, chamados *forward* e *backward* [38], que utilizam a técnica de programação dinâmica para minimização do custo. Antes de explicar o funcionamento algorítmico do *forward*, vamos inicialmente analisar um outro exemplo.

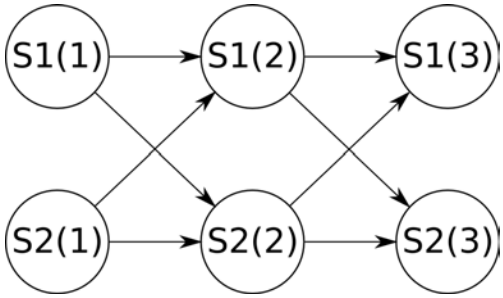


Figura 3.3: Treliça referente ao exemplo dado.

Suponha um HMM qualquer, λ, formado por 2 estados ($S = \{s_1, s_2\}$), 2 símbolos ($V = \{v_1, v_2\}$) e uma sequência de observações de tamanho 3 ($O = \{v_1, v_2, v_1\}$). Para o cálculo de $P(O|\lambda)$ devemos obter a probabilidade de O ser gerada por todas as sequências de estados possíveis. A Tabela 3.8 exibe as sequências possíveis para esse caso.

Sequência	$o_1 = v_1$	$o_2 = v_2$	$o_3 = v_1$
Q_1	s_1	s_1	s_1
Q_2	s_1	s_1	s_2
Q_3	s_1	s_2	s_1
Q_4	s_1	s_2	s_2
Q_5	s_2	s_1	s_1
Q_6	s_2	s_1	s_2
Q_7	s_2	s_2	s_1
Q_8	s_2	s_2	s_2

Tabela 3.8: Sequências de possíveis estados geradores da sequência de observações $O = v_1, v_2, v_1$.

A treliça (grafo direcionado) representada pela Figura 3.3 apresenta as sequências de estados como possíveis caminhos, em que os vértices são os estados na forma $s_x(y)$, sendo x a identificação do estado e y o instante de tempo, e as arestas são as transições de estados. Por exemplo, a sequência Q_4 pode ser obtida na treliça percorrendo-se os estados $s_1(1)$, $s_2(2)$ e $s_2(3)$ e a sequência Q_7 através dos estados $s_2(1)$, $s_2(2)$ e $s_1(3)$.

Para calcular a probabilidade $P(O|\lambda)$, devemos somar as probabilidades de todas as sequências de estados que terminam em s_1 e s_2 após a geração do último símbolo da sequência de observações (($s_1(3)$ e $s_2(3)$), ou seja, todas as sequências exibidas na Tabela 3.8.

O cálculo de $P(O|\lambda)$ será particionado e realizado de forma inversa, sendo que, em vez de se calcular a probabilidade de geração do primeiro símbolo da sequência de observações (o_1), será calculada a probabilidade de geração do último símbolo (o_3), em seguida, do penúltimo (o_2) e assim sucessivamente.

Definiremos a probabilidade do estado s_1 gerar o último símbolo (o_3) da sequência de observações O como sendo a probabilidade do estado gerador do símbolo anterior (o_2) ter sido s_1 ou s_2 e, em seguida, transitar para s_1, gerando o último símbolo. A Equação 3.2.6 exibe esse cálculo, em que $\alpha_t(state) = P(state = q_t|o_1, ..., o_t, \lambda)$, sendo q_t o estado gerador de o_t.

$$\alpha_3(s_1) = \alpha_2(s_1).a_{s_1,s_1}.b_{s_1}(o_3) + \alpha_2(s_2).a_{s_2,s_1}.b_{s_1}(o_3) = \\ (\alpha_2(s_1).a_{s_1,s_1} + \alpha_2(s_2).a_{s_2,s_1}).b_{s_1}(o_3) \tag{3.2.6}$$

Como mencionado, para se obter $P(O|\lambda)$, devemos somar $\alpha_T(s_x)$, em que $1 \leq x \leq N$. Nesse contexto, devemos somar $\alpha_3(s_1)$ e $\alpha_3(s_2)$ para o exemplo dado. Analogamente, podemos calcular $\alpha_3(s_2)$, $\alpha_2(s_1)$ e $\alpha_2(s_2)$, como exibe a Equação 3.2.7.

$$\begin{aligned} \alpha_3(s_2) &= (\alpha_2(s_1).a_{s_1,s_2} + \alpha_2(s_2).a_{s_2,s_2}).b_{s_2}(o_3) \\ \alpha_2(s_1) &= (\alpha_1(s_1).a_{s_1,s_1} + \alpha_1(s_2).a_{s_2,s_1}).b_{s_1}(o_2) \\ \alpha_2(s_2) &= (\alpha_1(s_1).a_{s_1,s_2} + \alpha_1(s_2).a_{s_2,s_2}).b_{s_2}(o_2) \end{aligned} \tag{3.2.7}$$

Por fim, o cálculo de $\alpha_1(s_x)$, ou seja, a probabilidade de cada estado gerar a primeira observação da sequência de observações, é descrito na Equação 3.2.8, em que $1 \leq x \leq N$.

$$\alpha_1(s_x) = \pi_{s_x}.b_{s_x}(o_1) \tag{3.2.8}$$

Para o exemplo dado, temos $\alpha_1(s_1)$ e $\alpha_1(s_2)$ demonstrados na Equação 3.2.9.

$$\begin{aligned} \alpha_1(s_1) &= \pi_{s_1}.b_{s_1}(o_1) \\ \alpha_1(s_2) &= \pi_{s_2}.b_{s_2}(o_1) \end{aligned} \tag{3.2.9}$$

Realizando o cálculo dessa maneira, é possível observar que alguns valores parciais obtidos são reutilizados em outros cálculos, como ocorre ao se computar os valores de $\alpha_3(s_1)$ e $\alpha_3(s_2)$ em que em ambos utilizamos os valores de $\alpha_2(s_1)$ e $\alpha_2(s_2)$.

Generalizando os passos executados, temos que $P(O|\lambda)$ corresponde ao somatório de $\alpha_T(s_x)$, , tais que $O = \{o_1, ..., o_T\}$ e $\lambda = (\pi_N, A_{N\times N}, B_{N\times M})$, como demonstrado na Equação 3.2.10.

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(s_i) \tag{3.2.10}$$

Seguindo a generalização, podemos escrever o cálculo de $\alpha_t(s_x)$ quando $2 \leq t \leq T$ de acordo com a Equação 3.2.11 e $\alpha_1(s_x)$ de acordo com a Equação 3.2.12.

$$\alpha_t(s_x) = \left[\sum_{i=1}^{N} \alpha_{t-1}(s_i).a_{s_i,s_x}\right].b_{s_x}(o_t) \tag{3.2.11}$$

$$\alpha_1(s_x) = \pi_{s_x}.b_{s_x}(o_1) \tag{3.2.12}$$

Seguindo esse raciocínio, temos o método *forward*, em que se aplica a técnica de programação dinâmica armazenando-se os valores de $\alpha_t(s_x)$ após seus cálculos, evitando a repetição do cálculo, como descrito abaixo.

1. Inicialização
$$\alpha_1(i) = \pi_i.b_i(o_1), 1 \leq i \leq N \tag{3.2.13}$$

2. Indução
$$\alpha_t(j) = \left[\sum_{i=1}^{N} \alpha_{t-1}(i).a_{ij}\right].b_j(o_t), 2 \leq t \leq T \ \ e \ \ 1 \leq j \leq N \tag{3.2.14}$$

3. Terminação
$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i) \tag{3.2.15}$$

Alternativamente, como mencionado, existe outra maneira de se obter a probabilidade desejada, que é através do procedimento *backward*, representado pela variável β (mais detalhes no Apêndice A.1 deste trabalho). O algoritmo *backward*, assim como o *forward*, também utiliza a estratégia de programação dinâmica para minimizar o custo computacional do cálculo. Contudo, vale lembrar que os procedimentos *forward* e *backward* são independentes e apenas uma das variáveis (α ou β) é necessária para solucionar a incógnita da avaliação.

A variável $\beta_t(i)$ é a probabilidade da sequência de observações de $t+1$ até T, dado o estado S_i no tempo t e o modelo λ. Dessa forma, β pode ser calculada da seguinte maneira:

1. Inicialização
$$\beta_T(i) = 1, 1 \leq i \leq N. \tag{3.2.16}$$

2. Indução
$$\beta_t(i) = \sum_{j=1}^{N} a_{ij}.b_j(o_{t+1}).\beta_{t+1}(j), t = T-1, T-2, .., 1, \ \ 1 \leq i \leq N. \tag{3.2.17}$$

3. Terminação
$$P(O|\lambda) = \sum_{i=1}^{N} \pi_i.b_i(o_1).\beta_1(i) \tag{3.2.18}$$

No entanto, como inicialmente citado, de fato, a informação desejada geralmente é a probabilidade de cada modelo dada uma observação ($P(\lambda|O)$), em vez da probabilidade da observação dado o modelo ($P(O|\lambda)$). Com base no teorema de Bayes, podemos reescrever essa probabilidade na forma exibida pela Equação 3.2.19.

$$P(\lambda|O) = \frac{P(\lambda, O)}{P(O)} = \frac{P(O|\lambda)P(\lambda)}{P(O)} \tag{3.2.19}$$

A probabilidade de ocorrência de determinado modelo ($P(\lambda)$) geralmente pode ser estimada matematicamente através do conhecimento *a priori* sobre aplicação em questão. Já a probabilidade de ocorrência da sequência de observações ($P(O)$) pode não ser tão simples de ser

encontrada. Contudo, caso o objetivo da busca por $P(\lambda|O)$ seja a comparação entre as probabilidades de geração de uma mesma sequência de observações O por modelos diferentes, podemos substituir o denominador da Equação 3.2.19 por uma constante qualquer.

3.2.2 Melhor sequência de estados

A segunda incógnita é encontrar a melhor sequência de estados percorrida pelo modelo para uma determinada sequência de observações. Ou seja, procura-se localizar a sequência de estados cuja probabilidade final seja a maior dentre todas as sequências possíveis em determinado modelo. Essa incógnita pode ser resolvida utilizando o algoritmo de Viterbi [28], que, dado um modelo λ e uma sequência de observações $O = \{o_1, o_2, ..., o_T\}$, localiza uma sequência de estados $Q = \{q_1, q_2, ..., q_T\}$ cuja probabilidade de geração da sequência de observações dada seja a maior possível.

Por exemplo, suponha um modelo $\lambda = (A, B, \pi)$, cujas probabilidades são expostas na Tabela 3.9. Considerando a sequência de observações $O = \{A, A, B\}$, podemos observar na Tabela 3.10 as probabilidades de geração dessa sequência no modelo, em que são apresentadas as sequências de estados geradoras e suas respectivas probabilidades.

π	
1	0.10
2	0.90

B	a	b
1	0.80	0.20
2	0.25	0.75

A	1	2
1	0.40	0.60
2	0.30	0.70

Tabela 3.9: Probabilidades de ocorrências iniciais, emissões de símbolos e transições de estados, respectivamente.

Estados	**Probabilidade**
1-1-1	$0.10 \times 0.80 \times 0.40 \times 0.80 \times 0.40 \times 0.20 \approx 2.05 \times 10^{-3}$
1-1-2	$0.10 \times 0.80 \times 0.40 \times 0.80 \times 0.60 \times 0.75 \approx 11.5 \times 10^{-3}$
1-2-1	$0.10 \times 0.80 \times 0.60 \times 0.25 \times 0.30 \times 0.20 = 0.72 \times 10^{-3}$
1-2-2	$0.10 \times 0.80 \times 0.60 \times 0.25 \times 0.70 \times 0.75 = 6.30 \times 10^{-3}$
2-1-1	$0.90 \times 0.25 \times 0.30 \times 0.80 \times 0.40 \times 0.20 = 4.32 \times 10^{-3}$
2-1-2	$0.90 \times 0.25 \times 0.30 \times 0.80 \times 0.60 \times 0.75 = 24.3 \times 10^{-3}$
2-2-1	$0.90 \times 0.25 \times 0.70 \times 0.25 \times 0.30 \times 0.20 \approx 2.36 \times 10^{-3}$
2-2-2	$0.90 \times 0.25 \times 0.70 \times 0.25 \times 0.70 \times 0.75 \approx 20.7 \times 10^{-3}$

Tabela 3.10: Sequências de estados possíveis no modelo e probabilidades de geração da sequência de observações $O = \{A, A, B\}$.

Note que a sequência de estados composta ordenadamente pelos estados 2, 1 e 2 (em negrito) produz a maior probabilidade para a sequência de observações dada. Assim, para esse caso, essa é a sequência procurada, pois sua probabilidade de geração da sequência de observações é a máxima possível no modelo.

Como observado na incógnita anterior, percorrer todas as sequências de estados possíveis no modelo não é a alternativa mais inteligente devido ao custo computacional dessa busca exaustiva. O algoritmo de Viterbi propõe uma alternativa para solução dessa incógnita que também utiliza

programação dinâmica, a fim de minimizar esse custo. Esse algoritmo é similar ao algoritmo *forward*, descrito anteriormente, porém, em vez de encontrar a probabilidade de geração de uma sequência de observações em todas as sequências de estados possíveis, seu foco é localizar a sequência de estados que produz a maior probabilidade no modelo.

Se em vez de termos 3 elementos na sequência de observações $O = \{A, A, B\}$, tivéssemos apenas o primeiro elemento, ou seja, $O^1 = \{A\}$, teríamos duas probabilidades possíveis – a probabilidade de gerar o símbolo no estado 1 e a probabilidade de gerá-lo no estado 2, como representado na Figura 3.4. Assim, nesse caso, a melhor sequência de estados é $Q^1 = \{2\}$, pois é a sequência de estados que produz a maior probabilidade para gerar a observação O^1.

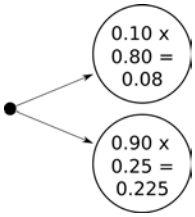
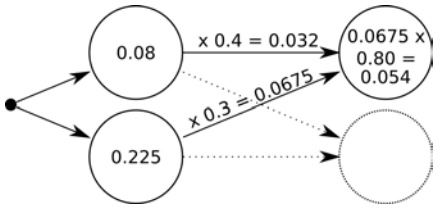
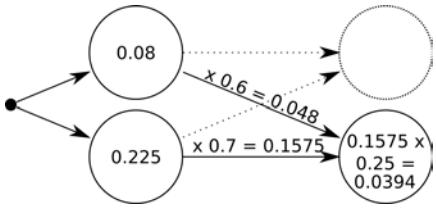


Figura 3.4: Treliça de possíveis estados para a observação O^1.

Considerando agora apenas os dois primeiros símbolos de O, temos $O^2 = \{A, B\}$. Para descobrirmos qual sequência de estados produz a maior probabilidade de geração de O^2, podemos utilizar a informação oriunda da análise anterior de O^1, pois, podemos reutilizar esses valores para encontrarmos as probabilidades de geração do segundo símbolo em cada um dos estados do modelo, como ilustra a Figura 3.5.



(a) Geração do segundo símbolo da sequência de observações no estado 1.



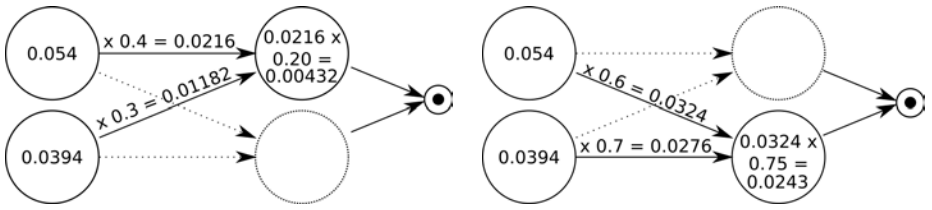
(b) Geração do segundo símbolo da sequência de observações no estado 2.

Figura 3.5: Treliças representando sequências de estados possíveis para determinadas sequências de observações.

A Figura 3.5(a) representa a geração do segundo símbolo no estado 1, em que as probabilidades de geração do primeiro símbolo em cada estado e a probabilidade de transição desse estado para o estado 1 são comparadas, multiplicando-se, por fim, a maior delas pela probabilidade de geração do segundo símbolo no estado 1. Assim, a maior probabilidade possível para se gerar o segundo símbolo da sequência de observações no estado 1 é de 5.4%, gerando o primeiro símbolo no estado 2; do mesmo modo, a maior probabilidade possível para se gerar o segundo símbolo no estado 2 é de 3.94%, também gerando o primeiro símbolo no estado 2.

Por fim, como sabemos as probabilidades máximas de geração do segundo símbolo da sequência de observações em cada um dos estados, podemos encontrar a sequência de estados que produz a maior probabilidade de geração da sequência O.

Se considerarmos a geração do terceiro símbolo da sequência O pelo estado 1, o estado anterior deve ser o estado 1, pois a probabilidade de geração do símbolo anterior juntamente



(a) Geração do terceiro símbolo da sequência de observações no estado 1.

(b) Geração do terceiro símbolo da sequência de observações no estado 2.

Figura 3.6: Treliças representando sequências de estados possíveis para determinadas sequências de observações.

com a probabilidade de transição do estado 1 para ele mesmo totalizam 2.16%, enquanto que, se considerarmos a geração do segundo símbolo no estado 2, temos a probabilidade de 1.182%. O mesmo é válido se considerarmos que o terceiro símbolo é gerado pelo estado 2, no entanto, o estado anterior deve ser o estado 1.

Como não existem mais símbolos a serem gerados na sequência de observações O, podemos encontrar a sequência de estados ótima a partir do estado que apresenta a maior probabilidade após a geração do último símbolo de O. No exemplo, esse estado é o 2, que possui probabilidade de 2.43%, enquanto que a probabilidade no estado 2 é de 0.432%. Como visto, se o terceiro símbolo da sequência O for gerado pelo estado 2, o símbolo anterior deve ser gerado pelo estado 1 e, consequentemente, o primeiro deve ser gerado pelo estado 2. Assim, temos a sequência 2-1-2 como sequência ótima.

O algoritmo de Viterbi utiliza o mesmo raciocínio utilizado para encontrar a sequência de estados ótima, armazenando as probabilidades máximas em cada estado a cada instante de tempo e os respectivos estados anteriores que produzem as probabilidades máximas. Para armazenar as probabilidades, utilizaremos as variáveis $\delta_t(j)$, que armazena a probabilidade máxima de se gerar o t-ésimo símbolo da sequência de observações no estado j, e $\psi_t(j)$, que armazena qual o estado gerador do $(t-1)$-ésimo símbolo da sequência de observações que maximiza a probabilidade do estado j gerar o t-ésimo símbolo.

$$\delta_t(j) = \begin{cases} \pi_j.b_j(o_t), & t = 1 \\ \max_{1\leq j\leq N}[\delta_{t-1}(i).a_{ij}].b_j(o_t), & 2 \leq t \leq T \end{cases} \quad 1 \leq j \leq N \tag{3.2.20}$$

$$\psi_t(j) = \begin{cases} 0, & t = 1 \\ arg\,max_{1\leq j\leq N}[\delta_{t-1}(i).a_{ij}], & 2 \leq t \leq T \end{cases} \quad 1 \leq j \leq N \tag{3.2.21}$$

Através das variáveis definidas, podemos obter a sequência de estados ótima e a sua probabilidade de geração da sequência de observações, como descrevem as Equações 3.2.22 e 3.2.23, respectivamente. Note que a sequência de estados ótima é obtida de forma regressiva, ou seja, do último estado gerador ao primeiro.

$$P^* = \max_{1\leq i\leq N}[\delta_T(i)] \tag{3.2.22}$$

$$q_t^* = \begin{cases} arg\,max_{1\leq i\leq N}[\delta_t(i)], & t = T \\ \psi_{t+1}(q_{t+1}^*), & t = T-1, T-2, ..., 1 \end{cases} \tag{3.2.23}$$

3.2.3 Treinamento do modelo

A última incógnita é a configuração dos parâmetros (π, A e B) do HMM com base em uma ou mais sequências de observações, que são chamadas de sequências de treinamento. Não existe uma maneira conhecida de realizar analiticamente esse ajuste para encontrar o modelo que maximize a probabilidade das sequências de treinamento quando não se conhecem os estados geradores das observações, porém, é possível escolher o modelo que sua probabilidade seja localmente maximizada usando um procedimento iterativo.

A existência de um processo que ajusta automaticamente os parâmetros do modelo com base nas sequências de observações é a principal força dos HMM's [11]. Esse algoritmo é chamado Baum-Welch [38] e trata-se de uma especialização do algoritmo EM – *Expectation-Maximization* [8, 36] – aplicada aos HMM's.

Antes de descrever o método de reestimação, é interessante que sejam definidas algumas variáveis que serão utilizadas ao longo do processo. Primeiramente, define-se a variável $\xi_t(i,j)$ como sendo a probabilidade de estar no estado s_i no momento t e transitar para o estado s_j no momento seguinte, dado o modelo e a observação. Dessa forma, a Equação 3.2.24 descreve $\xi_t(i,j)$, considerando $Q = q_1, q_2, ..., q_T$ a sequência de estados e $S = s_1, s_2, ..., s_N$ o conjunto de estados do modelo.

$$\xi_t(i,j) = P(q_t = s_i, q_{t+1} = s_j | O, \lambda), \tag{3.2.24}$$

Dados que $\alpha_t(i)$ é a probabilidade de estar no estado s_i no tempo t desde o início da observação e que $\beta_t(i)$ é a probabilidade de geração da sequência no modelo do instante $t+1$ ao fim, estando no estado s_i no tempo t, então $\xi_t(i,j)$ pode ser reescrito com o auxílio das variáveis α e β, descritas na Seção 3.2.1, da seguinte forma:

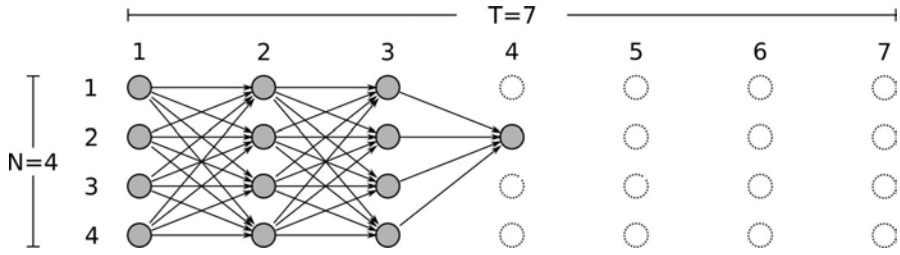
$$\xi_t(i,j) = \frac{\alpha_t(i).a_{ij}.b_j(o_{t+1}).\beta_{t+1}(j)}{P(O|\lambda)} = \frac{\alpha_t(i).a_{ij}.b_j(O_{t+1}).\beta_{t+1}(j)}{\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_t(i).a_{ij}.b_j(o_{t+1}).\beta_{t+1}(j)} \tag{3.2.25}$$

Por exemplo, considerando um HMM com 4 estados (N=4) e uma sequência de observações com 7 símbolos (T=7), podemos observar as variáveis $\alpha_4(2)$ e $\beta_5(4)$ nas Figuras 3.7(a) e 3.7(b), respectivamente. Observe que a variável $\beta_5(4)$ não contém a probabilidade de geração do quinto símbolo da sequência de observações do estado 4, por esse motivo, a respectiva circunferência está tracejada na Figura 3.7(b).

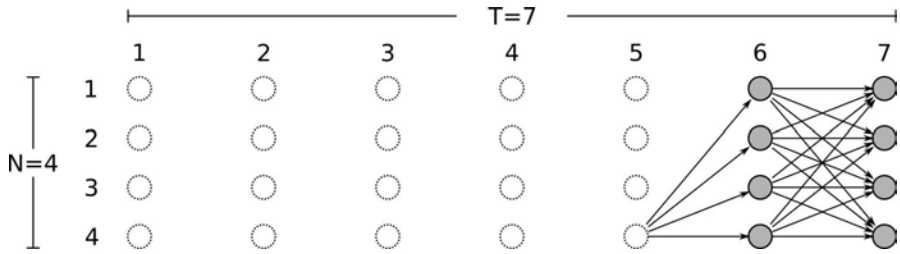
O valor da variável $\xi_4(2,4)$ nesse HMM pode ser obtido por meio da "união" das variáveis $\alpha_4(2)$ e $\beta_5(4)$ com a probabilidade conjunta da probabilidade de transição do estado 2 para o estado 4 e da probabilidade de geração do quinto símbolo pelo estado 4, conforme ilustra a Figura 3.8. Por fim, divide-se esse valor pela probabilidade de todos os casos possíveis ($P(O|\lambda)$).

Outra variável a ser compreendida é $\gamma_t(i)$, que representa a probabilidade de estar no estado s_i no instante de tempo t, como descreve a equação abaixo:

$$\gamma_t(i) = P(q_t = s_i | O, \lambda) \tag{3.2.26}$$



(a) Representação gráfica do valor da variável $\alpha_4(2)$.



(b) Representação gráfica do valor da variável $\beta_5(4)$.

Figura 3.7: Treliças representando valores das variáveis α e β em um HMM com 4 estados e uma sequência de observações com 7 símbolos.

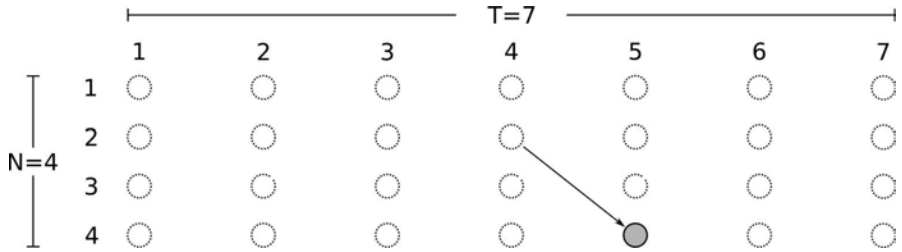


Figura 3.8: Representação da probabilidade de transição do estado 2 para o estado 4 e geração do quinto símbolo da sequência de observações.

Da mesma forma, deduz-se $\gamma_t(i)$ através de α e β da seguinte maneira:

$$\gamma_t(i) = \frac{\alpha_t(i).\beta_t(i)}{\sum_{i=1}^{N} \alpha_t(i).\beta_t(i)} \tag{3.2.27}$$

Pode-se ainda relacionar as variáveis ξ e γ da seguinte forma:

$$\gamma_t(i) = \sum_{j=1}^{N} \xi_t(i,j) \tag{3.2.28}$$

Nesse sentido, observa-se que a probabilidade de partir do estado s_i para s_j na sequência de observações é o somatório das probabilidades de estar em cada instante da observação no estado s_i e transitar para ao estado s_j. Ou seja,

$$\sum_{t=1}^{T-1} \xi_t(i,j) \tag{3.2.29}$$

Já a probabilidade de estar no estado s_i e partir para um estado qualquer, pode ser escrita através da Equação 3.2.30:

$$\sum_{t=1}^{T-1} \gamma_t(i) \tag{3.2.30}$$

Com as definições acima temos os seguintes resultados:

$$\bar{\pi}_i = \text{probabilidade de estar no estado } s_i \text{ no instante } 1 = \gamma_1(i) \tag{3.2.31}$$

$$\bar{a}_{ij} = \frac{\text{número de transições do estado } s_i \text{ para o estado } s_j}{\text{número de transições do estado } s_i \text{ para qualquer estado}} \tag{3.2.32}$$

$$\bar{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)} \tag{3.2.33}$$

$$\bar{b}_i(k) = \frac{\text{número de vezes que foi observado o símbolo } o_k \text{ no estado } s_i}{\text{número de vezes no estado } s_i} \tag{3.2.34}$$

$$\bar{b}_i(k) = \frac{\overset{o_t=v_k}{\sum_{t=1}^{T}} \gamma_t(i)}{\sum_{t=1}^{T} \gamma_t(i)} \tag{3.2.35}$$

Dessa forma, temos todos os parâmetros para a reestimação do modelo com base na observação.

O algoritmo Baum-Welch realiza diversas iterações sobre a sequência de observações dada, com o intuito de reestimar os parâmetros do HMM com base nessa entrada. Para encerrar a reestimação, deve ser estabelecido um critério de parada para o algoritmo, como a estabilização dos valores dos parâmetros de uma iteração para a outra, por exemplo.

Como o algoritmo Baum-Welch não pode garantir a localização da configuração que produz a probabilidade máxima global, um detalhe importante em sua utilização para a reestimação dos parâmetros do HMM é a configuração inicial do modelo, ou seja, o ponto de partida do algoritmo. Uma escolha inicial inadequada pode resultar em um máximo local muito distante do máximo global e, consequentemente, um modelo incoerente [22].

3.3 Tipos de HMM's

Os HMM's podem ser classificados de acordo com a estrutura de suas matrizes de transição de estados (matriz A) [39]. Dentre os diversos tipos (topologias) diferentes existentes, temos os HMM's completamente conectados (*fully connected* ou *ergodic*) e os HMM's esquerda-para-direita (*left-to-right*).

Nos HMM's completamente conectados, aqui chamados de HMM-CC, a matriz de transição de estados possui apenas elementos positivos, ou seja, $a_{i,j} > 0$, tal que $1 \leq i, j \leq N$. Em outras palavras, significa que é possível transitar de um estado para todos os outros estados do modelo. A Figura 3.9 ilustra um HMM de 4 estados com essa topologia.

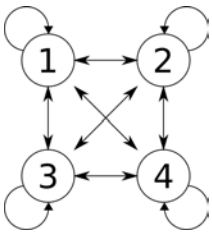
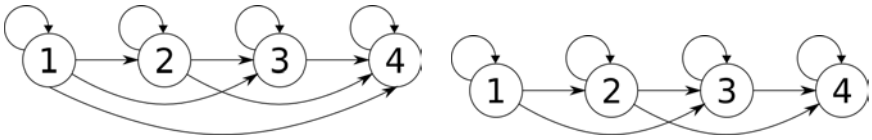


Figura 3.9: Exemplo de HMM completamente conectado.

Os HHMs de topologia esquerda-para-direita (HMM-ED) permitem probabilidades de transições positivas apenas para os estados com índices superiores (ou igual) ao estado origem, ou seja, $a_{ij} = 0$ quando $j < i$. A Figura 3.10(a) ilustra um exemplo de HMM com essa topologia. Geralmente quando se utilizam HMM's-ED, adicionam-se algumas restrições para assegurar que algumas transições não serão permitidas, como é o caso do HMM ilustrado pela Figura 3.10(b), em que existem transições apenas para os dois estados seguintes ao estado origem.



(a) Exemplo de HMM esquerda-direita convencional.

(b) Exemplo de HMM esquerda-direita com restrição $a_{ij} = 0$ quando $j > i + \Delta i$.

Figura 3.10: Exemplos de HMM's de 4 estados com topologias diferentes.

3.4 Mudança de escala

Devido ao fato de lidarmos com probabilidades nos HMM's, em algumas circunstâncias, temos sucessivas multiplicações de valores reais entre 0 e 1, como na obtenção da probabilidade de determinada sequência de observações (incógnita 1) ou sequência de estados ótima em um modelo (incógnita 2). Os números reais menores que 1 e maiores que 0 podem ser escritos na forma $A \times 10^{-p}$, tal que $p \in \mathbb{N}^+$. Assim, nota-se que T multiplicações sucessivas de valores dessa natureza resultam em valores muito próximos de 0 à medida que se aumenta o número de observações, pois teríamos

$$\prod_{i=1}^{T} A_i \times 10^{\sum_{i=1}^{T} -p_i}.$$

Logo, devido às limitações dos computadores, precisamos de alguma alternativa para evitar que os valores das probabilidades necessitem de tão alta precisão numérica. Para isso, existem

algumas técnicas que possibilitam alterar a escala dos valores, mantendo-os em determinada faixa de valores de representação computacional possível. Dentre tais técnicas, duas são brevemente descritas nas seções seguintes e maiores detalhes a respeito dessas e outras técnicas podem ser encontrados em [39].

3.4.1 Probabilidade da sequência de observações

Como citado anteriormente, uma das formas de calcularmos o valor da probabilidade de dada sequência de observações O em um modelo λ é através do algoritmo *forward*, conforme definição na Seção 3.2.1. Para representar os valores das variáveis $\alpha_t(i)$ escaladas, tais que $1 \leq i \leq N$ e $1 \leq t \leq T$, utilizaremos a notação $A_t(i)$. O escalamento ocorre através da multiplicação do valor original por um coeficiente, de forma que a proporção dos valores se mantenha.

Assim, inicialmente, obtemos os valores de $A_1(i)$, de acordo com a Equação 3.4.1.

$$A_1(i) = c_1.\alpha_1(i), 1 \leq i \leq N \qquad (3.4.1)$$
$$c_1 = \frac{1}{\sum_{i=1}^{N} \alpha_1(i)}$$

Posteriormente, obtemos as variáveis seguintes $A_t(i)$, tal que $2 \leq t \leq T$, como é exibido na Equação 3.4.2.

$$A_t(i) = c_t.[\sum_{g=1}^{N} A_{t-1}(g).a_{gi}].b_i(o_t), 1 \leq i \leq N \qquad (3.4.2)$$
$$c_t = \frac{1}{\sum_{i=1}^{N} A_t(i)}$$

Assim, normalizamos os valores de $\alpha_t(i)$, eliminando os valores muito pequeno. No entanto, segundo o algoritmo *forward*, para obtermos a probabilidade de determinada sequência de observações no modelo ($P(O|\lambda)$), somamos os valores de $\alpha_T(i)$. Porém, os valores de $\alpha_T(i)$ não são mais conhecidos, tendo agora apenas os valores de $A_T(i)$, tal que $\sum_{i=1}^{N} A_T(i) = 1$ independente do valor de $P(O|\lambda)$. Para recuperar essa informação, seguimos os seguintes passos.

Os valores escalados podem ser escritos como

$$A_t(i) = (\prod_{m=1}^{t} c_m)\alpha_t(i)$$

Dessa forma,

$$A_T(i) = (\prod_{m=1}^{T} c_m)\alpha_T(i)$$

Somando-se todos os valores de $A_T(i)$ podemos alcançar o valor de $P(O|\lambda)$, como descrevem as Equações 3.4.3 e 3.4.4.

$$\sum_{i=1}^{N} A_T(i) = \sum_{i=1}^{N}(\prod_{m=1}^{T} c_m)\alpha_T(i) = \prod_{m=1}^{T} c_m \sum_{i=1}^{N} \alpha_T(i) = \prod_{m=1}^{T} c_m P(O|\lambda) \tag{3.4.3}$$

Como normalizamos os valores a cada instante de tempo em relação à soma de todos os valores, temos que $\sum_{i=1}^{N} A_t(i) = 1$, tal que $1 \leq t \leq T$, e, logo, $\sum_{i=1}^{N} A_T(i) = 1$. De acordo com essa afirmação e com a Equação 3.4.3, temos

$$\begin{aligned} \sum_{i=1}^{N} A_T(i) = \prod_{m=1}^{T} c_m P(O|\lambda) = 1 \\ P(O|\lambda) = \frac{1}{\prod_{m=1}^{T} c_m} \end{aligned} \tag{3.4.4}$$

A fim de eliminar as multiplicações existentes na Equação 3.4.4, podemos obter o logaritmo natural do valor de $P(O|\lambda)$, como descreve a Equação 3.4.5.

$$\ln P(O|\lambda) = \ln \frac{1}{\prod_{m=1}^{T} c_m} = \ln 1 - \ln \prod_{m=1}^{T} c_m = -\sum_{m=1}^{T} \ln c_m \tag{3.4.5}$$

Logo,

$$P(O|\lambda) = e^{-\sum_{m=1}^{T} \ln c_m} \tag{3.4.6}$$

3.4.2 Sequência de estados ótima

O algoritmo de Viterbi, utilizado como solução para a segunda incógnita, descrita anteriormente, pode ser matematicamente adaptado a fim de se eliminar as multiplicações existentes na sua equação original. Para isso, em vez de calcularmos os valores de $\delta_t(j)$ utilizando os valores de π_i, a_{ij} e $b_i(o_t)$, utilizamos $\tilde{\pi}_i$, $\tilde{a}_{ij}$ e $\tilde{b}_i(o_t)$, tais que

$$\begin{aligned} \tilde{\pi}_i &= \log \pi_i \\ \tilde{a}_{ij} &= \log a_{ij} \\ \tilde{b}_i(o_t) &= \log b_i(o_t) \\ \tilde{\delta}_t(i) &= \log \delta_t(i) \\ \tilde{P}^* &= \log P^* \end{aligned} \tag{3.4.7}$$

Dados dois números reais x e y, tal que $x > y$, temos que $\log x > \log y$. Com base nessa afirmação e na Equação 3.2.23, podemos chegar às seguintes conclusões.

$$q_t^* = \begin{cases} arg\,max_{1 \leq i \leq N}[\tilde{\delta}_t(i)], & t = T \\ \psi_{t+1}(q_{t+1}^*), & t = T-1, T-2, ..., 1 \end{cases} \tag{3.4.8}$$

$$\tilde{\delta}_t(j) = \begin{cases} \tilde{\pi}_j + \tilde{b}_j(o_t), & t = 1 \\ \max_{1 \leq j \leq N}[\tilde{\delta}_{t-1}(i) + \tilde{a}_{ij}] + \tilde{b}_j(o_t), & 2 \leq t \leq T \end{cases} \quad 1 \leq j \leq N \tag{3.4.9}$$

$$\psi_t(j) = \begin{cases} 0, & t = 1 \\ arg\, max_{1 \leq j \leq N}[\tilde{\delta}_{t-1}(i) + \tilde{a}_{ij}], & 2 \leq t \leq T \end{cases} \quad 1 \leq j \leq N \tag{3.4.10}$$

$$\tilde{P}^* = \max_{1 \leq i \leq N}[\tilde{\delta}_T(i)] \tag{3.4.11}$$

Dessa forma, reduzimos os valores das probabilidades, além de eliminarmos as multiplicações da formula original.

3.5 Conjunto de símbolos contínuo

Os HMM's citados até o momento consideravam que os símbolos dos modelos eram discretos e únicos. Ou seja, todos os símbolos das sequências de observações estavam contidos no alfabeto do modelo e cada um deles correspondia a um único valor. No entanto, isso pode ser um empecilho para as aplicações em que as observações são contínuas [39] ou quando a observação pode ser composta por mais de um valor.

Uma alternativa para adaptar as aplicações que trabalham com observações contínuas aos modelos com observações discretas é através da discretização dos sinais observados. Porém, essa alternativa pode trazer consigo perda de informação ao converter o sinal contínuo em discreto. Em algumas aplicações, essa perda pode ser significante e interferir nos resultados. Como exemplo dessas técnicas, temos a quantização escalar – Scalar Quantization [40, 44] – e a quantização vetorial – Vector Quantization [1].

Outra alternativa é alterar o modelo de probabilidade de emissão de símbolos dos estados por uma função de densidade de probabilidade. Assim, elimina-se a perda de informação existente na discretização. Como exemplo, pode-se utilizar a função de densidade de probabilidade de gaussianas multi-dimensionais ou de modelos de mistura de gaussianas. Nesse último caso, pode ainda ser utilizado um modelo de mistura de gaussianas multivariável, que também oferece suporte aos casos em que a observação é composta.

3.6 Combinação de HMM's com técnicas de aprendizagem

Os HMM's são modelos estatísticos amplamente aplicados na modelagem de problemas com variações temporais. A popularidade da técnica é devida ao sucesso obtido em diversas aplicações que utilizaram-na no reconhecimento de fala. No entanto, nos últimos anos, a técnica vem sendo empregada com sucesso em diversas aplicações de visão computacional que envolvem reconhecimento de padrões, como no reconhecimento de letras manuscritas [30, 47, 59], no reconhecimento de gestos [10, 23, 27, 57] e na classificação de comportamentos animais [13, 25, 45, 60].

A classificação com os HMM's geralmente ocorre através da associação de um HMM para cada classe do problema. Para se classificar uma determinada amostra dentre as classes conhecidas, é calculada a probabilidade de geração da respectiva amostra por cada um dos HMM's

(geralmente, através do algoritmo *forward*). Por fim, a amostra é classificada como pertencente à classe associada ao HMM que apresentar a maior probabilidade de geração (HMM mais provável).

Embora as taxas de acerto dessas aplicações sejam consideráveis, os HMM's não são suficientes para o reconhecimento de padrões em algumas aplicações. Um dos principais atrativos dos HMM's é sua habilidade de lidar com a variabilidade das informações ao longo do tempo. Contudo, em alguns casos, a discriminação entre observações de classes distintas não é bem sucedida. Nesses casos, uma alternativa para se obter taxas de acerto mais altas é a utilização dos HMM's em combinação com outras técnicas de aprendizagem de máquina.

Em [52], Wang e Ju descrevem uma abordagem de classificação que utiliza a combinação de HMM com a técnica do vizinho mais próximo (KNN - *K Nearest Neighbor*) com o intuito de minimizar os erros de classificação presentes na abordagem tradicional (descrita a seguir), que aparecem principalmente quando a diferença entre os valores das probabilidades dos dois HMM's mais prováveis é pequena. Com base nisso, eles propõem a utilização de um classificador KNN em uma segunda fase de classificação para os casos em que a diferença citada anteriormente não é considerável. Segundo os autores, essa combinação adiciona pouco custo computacional e eleva a taxa de acerto do classificador, que aumentou de 82.1% para 88.3% em aplicação voltada ao reconhecimento de expressões faciais.

De forma similar, Jianjun Ye *et al.* apresentam a utilização de um classificador baseado em HMM e SVM para o reconhecimento de sinais da língua de sinais Chinesa [55]. O classificador utilizado é dividido em dois estágios. No primeiro, o sistema tenta realizar a classificação da amostra com um classificador baseado apenas em HMM's. Caso seja identificada uma possível confusão nessa classificação, a amostra é submetida ao segundo estágio, em que um classificador SVM toma a decisão final. O motivo da utilização dessa abordagem é que, segundo os autores, os HMM's apresentam bom desempenho para lidar com dados dinâmicos, mas têm alguns problemas quando existem poucas amostras da classe; por outro lado, as SVM apresentam boa generalização, o que pode suprir a deficiência dos HMM's.

Outro trabalho desenvolvido que utiliza a combinação de HMM com SVM é apresentado em [19]. Nele, o classificador HMM/SVM é aplicado no reconhecimento de símbolos manuscritos da seguinte forma. Cada classe possui uma etiqueta de identificação e um conjunto de HMM's. Para cada amostra de treinamento, é obtido um vetor de características. Esse vetor é composto da etiqueta da classe, de informações globais da amostra – como centro de massa, tamanho, etc. – e do valor normalizado da probabilidade de emissão do HMM mais provável para essa amostra dentre todos os HMM's da classe em questão. Por fim, o conjunto de vetores de características da base de treinamento é utilizado para treinar as SVM's.

Ainda nesse trabalho, para cada par de classes é criado um classificador que possui uma função discriminante que determina se a amostra observada pertence a uma classe ou à outra. Na etapa de reconhecimento, para o símbolo desconhecido a ser classificado, primeiramente, é obtido seu respectivo vetor de características. Esse vetor é enviado a cada um dos classificadores. Caso a função discriminante determine que a amostra pertence a uma classe, essa classe recebe um voto. Caso contrário, a outra classe recebe o voto. Ao fim, o símbolo é classificado como pertencente à classe que obteve o maior número de votos.

Neste trabalho, será utilizada uma abordagem de combinação de HMM com técnicas de aprendizagem de máquina similar à utilizada em [52]. No entanto, aliados à técnica de classificação com HMM, serão utilizados diversos algoritmos de aprendizagem de máquina, tais

como árvores de decisão, máquinas de vetor de suporte e redes neurais [4, 43]. Detalhes dessa abordagem serão fornecidos posteriormente.

Capítulo 4

Implementações

Para a realização dos experimentos – descritos posteriormente –, foi necessária a utilização (e/ou adaptação) de alguns recursos auxiliares, sendo eles a plataforma SIGUS – que, dentre outras bibliotecas, utiliza as funcionalidades da ImageJ – e as ferramentas Weka e JAHMM. Todos esses recursos foram implementados na linguagem de programação Java e possuem códigos-fonte abertos e gratuitos.

Resumidamente, a SIGUS [32] é um conjunto de implementações de algoritmos voltados à área de visão computacional; a ImageJ é um pacote para processamento de imagens; a Weka contém uma coleção de algoritmos de aprendizagem de máquina para o reconhecimento de padrões; e a JAHMM é uma biblioteca de códigos de apoio à utilização de modelos de Markov ocultos[1].

Da plataforma SIGUS, foram utilizados algoritmos para segmentação das imagens e para extração de características dos objetos contidos nelas. Os algoritmos de segmentação utilizados são baseados na técnica de subtração de fundo e em aprendizagem supervisionada. Para segmentação das imagens com aprendizagem supervisionada, foi necessário o auxílio de uma implementação do algoritmo de árvores de decisão (J48) da ferramenta Weka, embutido na SIGUS. Para se extrair as características relevantes dos objetos da imagem, foram utilizados algoritmos de momentos da imagem, k-curvatura e descritores de forma.

A Weka fornece suporte à utilização de técnicas de aprendizagem de máquina através de um ambiente gráfico próprio ou por meio de seus códigos em Java. Utilizando a aprendizagem de máquina, podemos realizar classificações de instâncias, como realizado na Seção 2.2.1, no entanto, de forma muito mais sofisticada. Por meio da Weka, é possível aplicar facilmente diversos algoritmos de inteligência artificial em conjuntos de dados e, adicionalmente, coletar informações estatísticas para análise das classificações realizadas.

Utilizando a interface gráfica da Weka, podemos realizar classificações de instâncias através de arquivos próprios da ferramenta. Esses arquivos possuem o formato denominado ARFF (*Attribute-Relation File Format*). Basicamente, a estrutura desse arquivo é composta de um cabeçalho e do conjunto de instâncias. No cabeçalho, são definidos os nomes da coleção de instâncias e das características que compõem cada instância. Além disso, são definidos os tipos de dados que cada característica suporta (como dados numéricos, textuais, entre outros).

[1] As ferramentas ImageJ, Weka e JAHMM (e informações referentes a elas) podem ser encontradas atualmente na internet, pelos endereços `http://rsbweb.nih.gov/ij`, `http://www.cs.waikato.ac.nz/ml/weka` e `http://jahmm.googlecode.com/`, respectivamente.

O Código 4.1, que será explanado a seguir, ilustra uma coleção de instâncias referente ao exemplo da Seção 2.2.1 (Tabela 2.7) em um arquivo no formato ARFF. Inicialmente, a coleção de instâncias se inicia com a palavra-chave @relation, que segue com o nome da coleção – para o exemplo, nomeamos a coleção de *FormasGeometricas*. Em seguida, nomeamos as características da coleção com a palavra-chave @attribute, como observado no código, em que temos duas características ($m00$ e $u11$).

Note também que as classes existentes na coleção são mapeadas no formato ARFF como uma característica. A essa característica, demos o nome de *classe* no código. Por fim, após a palavra-chave @data, apresentamos as instâncias da coleção, sendo que elas são compostas dos valores de cada uma das características definidas no cabeçalho, incluindo a classe à qual a instância pertence. No código, de acordo com o exemplo, temos 8 instâncias, compostas de dois valores (correspondentes aos valores de m_{00} e μ_{11}) e a respectiva classe.

```
@relation FormasGeometricas

@attribute m00 numeric
@attribute u11 numeric
@attribute classe {elipse,retangulo}

@data
 7796.0, 222.75,elipse
 6445.0, 180.00,elipse
 8106.0, 227.31,elipse
10848.0, 263.25,elipse
20000.0,4925.25,retangulo
20000.0,4925.25,retangulo
17563.0,4320.00,retangulo
15400.0,3781.00,retangulo
```

Código 4.1: Exemplo de arquivo de formato ARFF

Os arquivos em formato ARFF são processados pela Weka para treinamentos de seus algoritmos de aprendizagem e para classificações de instâncias. Alternativamente, para os mesmos fins, os algoritmos da Weka também reconhecem coleções de instâncias via códigos implementados em Java. De forma resumida, cada instância é mapeada como um objeto de uma classe denominada *Instance* e pertence a uma coleção, que é mapeada como um objeto da classe *Instances*.

Para implementação dos HMM's e técnicas relacionadas, foi utilizada a ferramenta JAHMM, que apresenta suporte para diversas estruturas de HMM. No entanto, foram necessárias algumas adaptações na ferramenta para que algumas funcionalidades necessárias fossem embutidas a ela. Segue abaixo uma breve explicação técnica sobre os recursos básicos oferecidos pela JAHMM.

Cada modelo é representado como um objeto da classe *Hmm*. Nele, os estados são representados por números inteiros e, para cada estado, existe uma função de densidade de probabilidade (FDP) de observação. Existem quatro tipos de observações possíveis na ferramenta: discreta, inteira, decimal e conjunto de decimais. No conjunto de símbolos discreto/inteiro, todos os símbolos são previamente informados, diferentemente do que ocorre nos outros tipos, que não são finitos.

Para cada tipo de observação, existe uma classe que possui uma FDP que recebe observações desse tipo como entrada. Quando se lida com observações do tipo discreto ou inteiro, a

ferramenta manipula uma probabilidade para cada símbolo existente. No caso de observações contínuas, utilizam-se distribuições de probabilidade. As observações decimais são tratadas com modelo de mistura gaussiana e os conjuntos de decimais com uma gaussiana multivariada.

Além da estrutura dos HMM's, a ferramenta disponibiliza outros recursos que auxiliam na utilização dos HMM's e também oferece as implementações dos algoritmos *Forward*, *Backward*, *Viterbi* e *Baum-Welch*. Dentre esses, apenas o *Viterbi* não possui uma versão alternativa com mudança de escala.

Mesmo existindo as implementações dos algoritmos que indicam a probabilidade de determinada sequência de observações ser gerada por um modelo, a ferramenta não realiza classificação de sequências de observações com base nas probabilidades de alguns modelos. Dessa forma, um classificador baseado em HMM's integrado a alguns recursos da ferramenta Weka foi acrescentado à ferramenta, permitindo diversas formas de classificação de sequências de observações, que serão descritas a seguir.

Com as modificações anexadas à ferramenta, são possíveis três tipos diferentes de classificação de sequências de observações baseados em HMM's. O primeiro é a classificação tradicionalmente utilizada, em que é construído (e treinado apropriadamente) um modelo para cada classe existente. Por fim, uma determinada sequência de observações é classificada como pertencente à classe cujo HMM apresentar a maior probabilidade de geração para a sequência. Já o segundo e o terceiro acrescentam mais uma etapa nessa classificação, com o intuito de melhorar os resultados de classificação, como descrito abaixo.

A etapa de teste (ou classificação) implementada é dividida em duas fases, sendo elas a classificação primária obrigatória (baseada em HMM's) e a secundária opcional (baseada em outras técnicas). A etapa primária corresponde à tradicional forma de classificação com HMM's, em que, para cada classe, é construído um HMM. As amostras a serem classificadas são convertidas em sequências de observações. Por fim, cada amostra é classificada como pertencente à classe cujo HMM apresenta maior probabilidade de geração da respectiva sequência de observação.

Como mencionado, a abordagem acima descrita apresenta algumas limitações quando os HMM's mais prováveis apresentam probabilidades próximas. A fim de melhorar os resultados do classificador nessas situações, as sequências de observações podem ser submetidas à segunda etapa de classificação. Ao avaliar uma sequência de observações, é calculada a probabilidade de geração dessa sequência por cada um dos HMM's existentes. Caso a diferença entre a maior probabilidade e qualquer outra probabilidade seja inferior a um determinado limiar, o classificador solicita a classificação secundária. Caso contrário, ele assume que a amostra pertence à classe do HMM que produziu a maior probabilidade para sequência de observações em análise. Essa diferença é denominada grau de diferença [54] *apud* [52].

A classificação secundária implementada pode ser executada em dois modos diferentes: abrangente e seletivo. Em ambos os modos, as probabilidades de geração da sequência de observações pelos HMM's são utilizadas como características. No entanto, o primeiro modo utiliza todo o conjunto de treinamento como base para a classificação – ou seja, todas as classes conhecidas são compreendidas –, enquanto, no seletivo, apenas a classe referente ao HMM que apresentou a maior probabilidade e as classes referentes aos HMM's com probabilidades próximas a essa são incluídas no treinamento. Note que, no segundo modo, apenas as classes mais prováveis podem ser escolhidas na classificação secundária, o que não ocorre no primeiro.

Por exemplo, suponha que um determinado comportamento animal registrado em imagens deve ser classificado por um sistema. Durante o treinamento desse sistema, foi construído um

HMM para cada comportamento possível. Considere nesse exemplo a existência de 3 comportamentos (ócio, alimentação e ruminação). Inicialmente, esse comportamento (amostra) deve ser convertido em uma sequência de observações que é reconhecida por todos os HMM's do sistema – que pode ser uma série de informações referentes aos quadros do conjunto de imagens (ou vídeo). Essa sequência de observações é enviada para cada HMM, que, por sua vez, informa sua respectiva probabilidade de geração para essa sequência.

Supondo utopicamente que o HMM referente à classe ócio (denominado HMM1) informou a probabilidade de 90% para a sequência recebida, o HMM da classe alimentação (HMM2) informou 42% e o HMM da classe ruminação (HMM3) informou 92%. Nessa circunstância, a classificação primária consideraria que o comportamento em questão corresponde a ruminação, já que o HMM mais provável é o HMM3. Caso esse sistema assuma um grau de diferença de 1%, a classificação é encerrada, ignorando a classificação secundária – dado que as diferenças entre as probabilidades do HMM3 e os outros são maiores que o grau de diferença. Porém, se considerarmos que o grau de diferença seja de 5%, teremos que realizar a classificação secundária.

Na classificação secundária, como citado, são utilizados os valores das probabilidades de cada HMM como características. Nesse sentido, no exemplo anterior, teremos 3 características compondo um padrão a ser classificado. Caso o modo de classificação escolhido seja o seletivo, consideraremos apenas as classes ócio e ruminação, desconsiderando a classe alimentação. Logo, o padrão pertencerá a uma dessas duas classes. Em contrapartida, caso o modo de classificação seja o abrangente, todas as classes serão consideradas. Nesse último modo, existe a possibilidade do comportamento ser classificado como alimentação, embora o HMM dessa classe tenha apresentado baixa probabilidade na classificação primária.

Neste trabalho, as implementações dos algoritmos utilizados na classificação secundária são provenientes da ferramenta Weka. Para integrar as duas ferramentas (JAHMM e Weka), foi adicionada uma camada intermediária que possibilita a comunicação entre elas. Essa camada é responsável por adaptar a utilização dos algoritmos de inteligência artificial de acordo com a proposta aqui apresentada. Em contrapartida, os dados produzidos por meio da JAHMM podem ser exportados para arquivos no formato ARFF e podem ser posteriormente utilizados dentro da Weka.

Para melhor compreensão, segue abaixo, na Figura 4.1, um diagrama comprimido para visualização das implementações realizadas. Os círculos representam as bibliotecas já implementadas, as setas indicam o fornecimento de serviços ou informações e os retângulos as implementações novas. Recapitulando, para confecção do classificador, foram utilizadas as implementações dos HMM's da biblioteca JAHMM. Por conta da abordagem combinada, foi necessária a utilização dos algoritmos da Weka no classificador. Para análise dos dados dentro da Weka, o classificador exporta seus dados em arquivos ARFF. Por fim, os experimentos são realizados no ambiente experimental, que utiliza a biblioteca Weka e o classificador para o reconhecimento de padrões e a SIGUS para processamento das imagens.

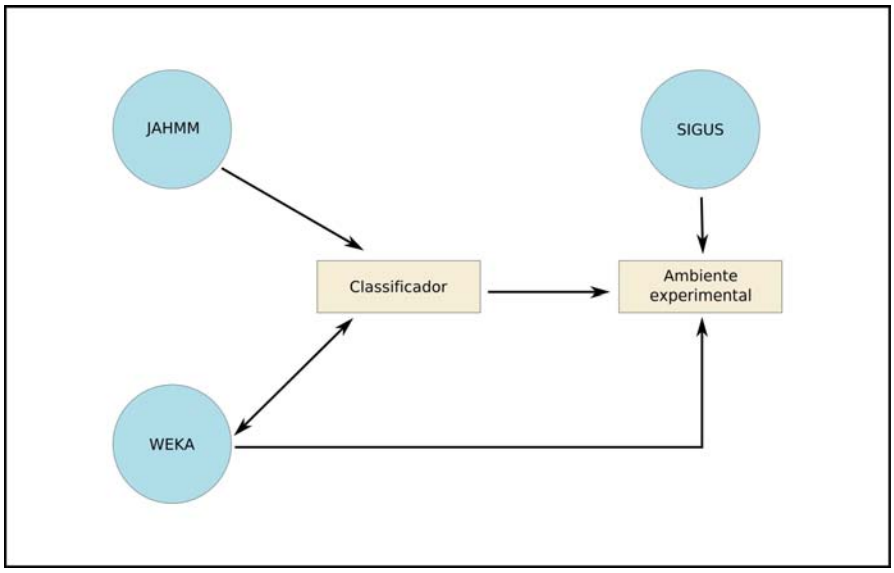


Figura 4.1: Diagrama de implementações.

Capítulo 5

Experimentos

Os experimentos deste trabalho foram realizados utilizando sequências de imagens de larvas em pequenos recipientes. Resumidamente, o objetivo desses experimentos é identificar automaticamente certos comportamentos dos animais nessas sequências. Os comportamentos a serem classificados nas larvas são de vitalidade e falência. Segue abaixo uma breve explicação de cada um desses comportamentos e a importância de identificá-los em experimentos biológicos.

As imagens das larvas aqui utilizadas pertencem a um grupo de pesquisadores que analisa a eficiência de larvicidas no combate às larvas do *Aedes aegypti* [34, 41]. Um dos critérios utilizados pelos pesquisadores para avaliar a eficácia dessas substâncias é a taxa de mortalidade das larvas durante um período de tempo determinado. Para isso, em intervalos definidos, um especialista observa as larvas dentro do recipiente e registra o número de larvas vivas e mortas nesses instantes. Ao final do período, com base nesses registros, são obtidas conclusões a respeito do larvicida em questão. Para visualização do ambiente, a Figura 5.1(a) apresenta uma imagem proveniente de alguma subsequência de imagens utilizada nos experimentos.

Os comportamentos analisados nas larvas são fundamentalmente triviais. No entanto, a identificação do momento em que a larva entra em processo de falência não é tão trivial assim. Nesse momento, o movimento característico da larva se modifica, tornando-se ligeiramente menos harmônico e mais lento até que se encerre a movimentação, caracterizando o momento de óbito. Já a vitalidade é mais simples de ser identificada. Quando viva, a larva se movimenta de forma típica, embora ela fique imóvel em determinados momentos, como ocorre durante a respiração.

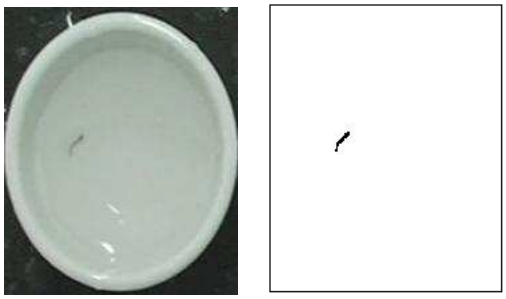
Inicialmente, de acordo com o que será exposto em mais detalhes nas seções seguintes, as imagens das larvas serão capturadas e posteriormente processadas, para que se possam ser extraídas informações relevantes desses animais. Em seguida, essas informações serão fornecidas aos classificadores nas etapas de treinamento e classificação. Dessa forma, podemos analisar o desempenho de classificação desses classificadores para essa aplicação.

5.1 Aquisição e tratamento de imagens

Foram capturados 3 trechos de vídeo com 1300 imagens cada contendo apenas uma larva, sendo que 2 deles correspondem a imagens de uma larva morta (ou morrendo) – inserida em um recipiente com água e larvicida – e o outro de um larva viva – em um recipiente com água sem larvicida. Essas imagens foram capturadas dentro do laboratório por meio de uma

câmera filmadora com auxílio de um tripé. Como existem várias ocorrências dos comportamentos analisados nos 3 trechos, dos dois trechos de larva morta, foram extraídas 24 sequências menores e, do outro trecho, foram extraídas 10 sequências. Essas sequências não possuem um tamanho fixo, mas cada uma contém aproximadamente 100 imagens.

Após a aquisição das sequências, as imagens foram processadas, com o intuito de auxiliar na classificação automática dos comportamentos abordados. Essas imagens foram segmentadas com um algoritmo de segmentação baseado na combinação entre a técnica de subtração de fundo e segmentação baseada em atributos de cores. Primeiramente, foi subtraído o fundo da imagem e, em seguida, retirados os ruídos por meio da segmentação por cores. Essa eliminação de ruídos foi aplicada para reduzir os erros provocados por eventuais alterações no ambiente, como mudanças na iluminação. Por fim, em algumas imagens, foi realizada segmentação manual, para retirada de partículas fora do interesse da aplicação. A Figura 5.1 ilustra o resultado da segmentação de uma das imagens utilizadas nos experimentos, em que a imagem à esquerda corresponde à imagem original da larva e a imagem à direita à imagem segmentada por esses algoritmos.



(a) Imagem original da larva no recipiente utilizado para experimentos biológicos.

(b) Resultado da segmentação da imagem da larva.

Figura 5.1: Ilustração de segmentação de imagem utilizada nos experimentos por meio da combinação de segmentações baseadas em subtração de fundo e em atributos de cores.

De cada imagem segmentada, foram extraídas inicialmente 9 características. Três delas são as coordenadas X e Y do centro de massa da larva e o primeiro momento de Hu, que foram obtidas por meio dos momentos estatísticos. As características restantes correspondem aos valores do histograma dos ângulos da larva, que são calculados através do cálculo da k-curvatura e serão agrupados em seis classes de acordo com seus valores. Portanto, para cada sequência de imagens, teremos um vetor de características de mesmo tamanho, em que cada elemento dele corresponde a uma coleção com 9 valores. Provavelmente devido à dimensão do vetor de características, não foram obtidos resultados consideráveis em experimentos iniciais com essas características.

Para contornar esse problema, foram extraídos 4 descritores de forma: *aspect ratio*, *form factor*, *roundness* e *compactness*. Essas quatro características foram utilizadas para classificar a postura da larva em cada quadro como esticada ou curvada/fechada. Dessa forma, cada quadro da sequência produz apenas uma característica discreta que assume dois valores possíveis. Para realizar essa classificação, foi criado um conjunto de treinamento composto por exemplos das duas classes obtidos de imagens selecionadas manualmente. Esse conjunto de treinamento foi utilizado por um algoritmo de aprendizagem (kNN - *k-Nearest Neighbor*) para realizar a classificação de todas as imagens restantes. Diferentemente da situação anterior, foram obtidos resultados preliminares animadores com a nova característica produzida.

5.2 Metodologia para treinamento e classificação

Dadas as informações obtidas das imagens, é necessária a manipulação adequada dessas informações. Por isso, é necessário o mapeamento desses dados de modo que eles possam ser reconhecidos pelos classificadores posteriormente. Nos experimentos, foram utilizados classificadores baseados em HMM's e baseados em algoritmos de aprendizagem de máquina. Como a natureza da informação produzida no passo anterior é discreta, consequentemente, foram utilizados apenas HMM's discretos, todos com probabilidades escaladas. Segue abaixo a descrição das metodologias utilizadas para o treinamento dos classificadores e obtenção das classificações.

Como citado em capítulo anterior, no classificador baseado em HMM's, cada classe a ser reconhecida possuía um respectivo modelo. Tradicionalmente, para classificar uma determinada sequência de observações, esse tipo de classificador utiliza a comparação das probabilidades de geração da sequência de observações em cada um dos modelos. Ou seja, é encontrada a probabilidade da sequência pertencer a cada uma das classes definidas. Portanto, a sequência é classificada como pertencente à classe cujo modelo apresentar a maior probabilidade de gerá-la. Nos experimentos deste trabalho, além da abordagem tradicional, foi utilizada uma abordagem combinada para a construção do classificador.

Na abordagem combinada, além da probabilidade apresentada por cada modelo, é utilizada a resposta produzida por um classificador baseado em aprendizagem de máquina em algumas circunstâncias. Caso algum outro modelo apresente uma probabilidade muito próxima à maior probabilidade encontrada para uma determinada sequência, é utilizado um classificador auxiliar para realizar a classificação final. As informações de entrada desse classificador são as probabilidades de cada modelo e sua saída, obviamente, é a classe à qual a entrada pertence. Para os experimentos dessa abordagem, foi utilizada a configuração de HMM's que apresentou os melhores resultados nos demais experimentos.

Para avaliar os resultados do classificador baseado em HMM's, foram exploradas separadamente duas características do HMM que podem interferir consideravelmente nos resultados, que são as probabilidades iniciais e o critério de parada no treinamento. Foram analisados três conjuntos de probabilidades iniciais diferentes (aleatório, pré-computado manualmente e pré-computado automaticamente) e três critérios de parada diferentes (número de iterações, baseado em limiar e baseado no aumento de probabilidades). Nos experimentos com probabilidades pré-computadas automaticamente, houve ainda variação no número de estados (de 2 a 4), diferentemente dos outros, em que foram utilizados apenas 2 estados.

O conjunto de probabilidades iniciais aleatório considera as mesmas probabilidades de ocorrência inicial para os estados do modelo e as mesmas probabilidades de transição entre eles. Além disso, define aleatoriamente as probabilidades de emissão de símbolos de cada um desses estados. De acordo com [38], para HMM's discretos, probabilidades iniciais aleatórias (ou uniformes, no caso de π e A) são adequadas para a reestimação de probabilidades em vários casos. No entanto, em alguns casos, boas probabilidades iniciais de emissão de símbolos (B) podem garantir o sucesso no treinamento dos modelos.

Dado esse fato, o conjunto de probabilidades iniciais pré-computado automaticamente utiliza probabilidades uniformes para π e A e utiliza auxílio do algoritmo de clusterização *K-Means*[1] para obter as probabilidades de B dos modelos. Para isso, o número de médias do algoritmo corresponde ao número de estados do modelo. Ou seja, cada média é associada a um estado.

[1] No Apêndice C, encontra-se uma breve revisão sobre o algoritmo K-Means para auxílio na compreensão.

Em seguida, utilizam-se sequências de observações cujas classes são conhecidas para formar o conjunto de treinamento. Com base no resultado produzido pelo *K-Means*, são alcançadas as probabilidades de emissão de símbolos para cada estado do respectivo modelo ao qual a sequência de treinamento pertence.

Cada símbolo da sequência de observações é considerado como um valor a ser agrupado pelo algoritmo. Logo, ao final de sua execução, serão produzidos K grupos de símbolos, sendo K o número de estados definido para o modelo. Como cada grupo está associado a um estado, as probabilidades de emissão de símbolos de um estado podem ser calculadas por contagem simples dos elementos de seu grupo. Com o intuito de evitar probabilidades nulas, antes da contagem, foi acrescentada uma ocorrência para cada um dos símbolos reconhecidos pelo modelo em todos os estados.

O algoritmo *K-Means* realiza sucessivos agrupamentos dos valores envolvidos. Dessa forma, a cada agrupamento (ou iteração), cada um desses valores pertencerá a uma das K médias do algoritmo. É fato que, de uma iteração para outra, um valor que anteriormente pertencia a uma determinada média pode passar a pertencer a outra média. Portanto, o número de iterações pode interferir no resultado final produzido. Como não existe um número genérico de iterações, deve ser estipulado um critério de parada para esse agrupamento. Neste trabalho, o critério utilizado corresponde à ausência de mudanças no agrupamento de uma iteração para outra. Ou seja, o algoritmo para o processamento quando todos os valores permanecem na mesma média em duas iterações sucessivas.

Por fim, o conjunto de probabilidades iniciais pré-computado manualmente é baseado em análise humana prévia para estimação das probabilidades. Um HMM é dito oculto pois a sequência de estados percorrida nas gerações de símbolos é desconhecida. No entanto, dada a existência de sequências de estados de um determinado modelo – ou seja, as sequências de estados percorridas no modelo para geração de algumas sequências de símbolos –, é possível alcançar as probabilidades de ocorrência inicial dos estados e transição entre eles também por contagem simples. Dada ainda a existência das sequências de observações que foram geradas pelas sequências de estados citadas, pode-se também, da mesma forma, calcular as probabilidades de emissão de símbolos em cada estado.

Com base nisso, o objetivo da análise humana era identificar em cada sequência de observações os respectivos estados emissores dos símbolos dessa sequência para se obter as probabilidades iniciais dos modelos. Para isso, foi atribuído um significado humanamente perceptível a cada um dos estados dos modelos. Como cada símbolo das sequências de observações corresponde originalmente a uma imagem de uma larva, foi definido que um estado do modelo estaria relacionado a larvas esticadas e o outro a larvas curvadas. A Figura 5.2(a) ilustra um exemplo de larva esticada e as Figuras 5.2(b) e 5.2(c) ilustram exemplos de larvas curvadas.

Nesse sentido, cada símbolo das sequências de observações analisadas por um especialista produzia um respectivo estado. Logo, os estados produzidos para uma determinada sequência de observações correspondem à respectiva sequência de estados geradora da sequência de observações analisada. Como dito, fazendo uso dessas sequências, pode-se calcular todas as probabilidades iniciais dos modelos.

Após a obtenção das probabilidades iniciais dos modelos nos experimentos acima citados, os parâmetros desses modelos foram reestimados utilizando o algoritmo *Baum-Welch*. O critério de parada dessa reestimação era o número de iterações do algoritmo, ou seja, o número de atualizações de todas as componentes do modelo. Em cada reestimação, eram realizadas 9

(a) Esticada (b) Curvada (c) Curvada

Figura 5.2: Imagens de larvas em diferentes posturas.

iterações sobre o modelo com base em sequências de observações de treinamento. Para avaliar se o número de iterações definido era suficiente para a aplicação, foram realizados outros três experimentos voltados ao critério de parada do algoritmo, descritos abaixo.

O primeiro critério observado determina que, para continuar a reestimação, a soma das probabilidades das sequências utilizadas no treinamento considerando os novos parâmetros têm que ser maiores que a soma das probabilidades dessas mesmas sequências considerando os parâmetros antigos [56]. O segundo critério para a reestimação quando a diferença entre a probabilidade das sequências de observações no modelo novo e a probabilidade no modelo anterior é menor que um limiar previamente definido (nos experimentos, esses limiares variaram de 10^{-1} a 10^{-6}). O terceiro critério, de modo similar ao critério utilizado nos experimentos anteriores, considera o número de iterações do algoritmo (considerando 50, 100, 500 e 1000 iterações).

Adicionalmente, foram analisadas as taxas de acerto para esta aplicação utilizando uma combinação entre classificadores. Como mencionado em capítulo anterior, em alguns casos, podem ser obtidas melhores taxas de acerto com classificadores baseados em HMM's quando combinados a técnicas de aprendizagem de máquina. Uma das combinações possíveis é utilizar essas técnicas quando o resultado do classificador baseado em HMM's pode ser duvidoso. Essa dúvida pode ocorrer, por exemplo, quando dois modelos apresentam probabilidades muito próximas para uma mesma entrada. De fato, essa foi a combinação utilizada neste trabalho, como descrito abaixo.

Nessa combinação, foi definido um limiar que varia entre 0 e 1. Para uma dada entrada, são calculadas as probabilidades de geração da respectiva sequência de observações por cada um dos modelos. A maior probabilidade obtida é utilizada como referência. Em seguida, é calculada a distância entre essa probabilidade e as probabilidades apresentadas pelos outros modelos. Caso a proporção de qualquer uma dessas distâncias com relação à maior probabilidade obtida for maior que o limiar anteriormente definido, a entrada é classificada como pertencente à classe do modelo de maior probabilidade. Caso contrário, todas as probabilidades dos modelos são enviadas a um classificador auxiliar. Com base nesses valores, esse segundo classificador executa a decisão final e classifica a entrada.

Nos experimentos realizados, os classificadores baseados em HMM's foram combinados com os algoritmos de árvores de decisão (J48), *k-Nearest Neighbor* (IBK), máquinas de vetor de suporte (SMO) e redes neurais (MLP). Esses algoritmos foram utilizados também de forma independente, com o intuito de comparar os resultados obtidos com os classificadores baseados em HMM's. No entanto, como esses classificadores utilizam um número fixo de características de entrada para inferir uma resposta, foram necessários pré-processamentos dos dados de entrada, descritos abaixo.

Foram realizados três pré-processamentos nos dados de entrada, que correspondem às posturas da larva em cada uma das imagens. No primeiro, foram contadas as ocorrências de cada postura nas sequências de posturas. Portanto, para esta aplicação, cada sequência de posturas produzia 2 características, que correspondem à quantidade de ocorrências da postura esticada e à quantidade de ocorrências da postura curvada. No segundo, são computadas as quantidades de ocorrências sucessivas de posturas. Nesse caso, cada sequência de posturas produzia 4 características, duas correspondendo aos números de ocorrências de duas mesmas posturas sucessivamente e duas aos números de ocorrências de duas posturas diferentes uma após a outra. No terceiro, utiliza-se apenas uma característica, que corresponde ao número de mudanças de duas posturas.

Para ficar mais claro como são realizados esses pré-processamentos, considere que uma sequência de imagens produziu a sequência de posturas ($E-E-E-C-C-C-C-C-C-E-E$), em que E representa a postura esticada e C curvada. Para essa sequência de posturas, o primeiro pré-processamento produziria os valores $E = 5$ e $C = 6$, que corresponde a contagem de cada postura. O segundo produziria os valores $EE = 3$ (número de ocorrências sucessivas da postura esticada), $EC = 1$ (número de ocorrências da postura esticada seguida da postura curvada), $CE = 1$ (posturas curvadas seguidas de esticada) e $CC = 5$ (posturas curvadas sucessivas). O terceiro produziria o valor 2, pois houve apenas duas mudanças (de E para C e de C para E).

5.3 Configuração dos experimentos

Os experimentos deste trabalho podem ser classificados em 4 grupos diferentes. Com o intuito de facilitar a compreensão, os resultados de todos esses experimentos, juntamente com a análise desses resultados, também serão apresentados no capítulo seguinte de acordo com esses grupos.

No grupo 1, foram analisados os desempenhos de classificadores baseados em HMM's considerando os conjuntos de probabilidades iniciais desses modelos. Após a configuração inicial das probabilidades dos modelos, todos os classificadores foram treinados da mesma forma, com o algoritmo *Baum-Welch*. A primeira abordagem considerou probabilidades iniciais aleatórias para todos os HMM's do classificador; a segunda utilizou probabilidades pré-computadas manualmente; e, na última, as probabilidades foram pré-computadas automaticamente.

Os experimentos do grupo 2 também consideraram classificadores baseados em HMM's, porém, analisando apenas o critério de parada durante o treinamento. No segundo critério de parada utilizado, os valores do limiar utilizados eram na forma 10^{-x}, em que $x \in \mathbb{Z}$, variando entre 1 a 6. Já no terceiro critério, foram atribuídos os valores 50, 100, 500 e 1000 para o número de iterações. Como o primeiro critério analisado é fixo, não foram realizadas variações.

O grupo 3 contém os experimentos com classificadores que combinam a utilização de HMM's com algoritmos de aprendizagem de máquina, que, nesses experimentos, correspondem ao IBk, J48, MLP e SMO. Assim, de acordo com o que foi apresentado, caso as probabilidades apresentadas pelos HMM's não sejam suficientes para discriminar a classe à qual a entrada pertence, esses algoritmos são utilizados para executarem a decisão final. Nesses experimentos, foi variado o limiar percentual em uma unidade inteira, de 0 a 100%.

Por fim, no grupo 4, estão os experimentos que utilizaram classificadores baseados apenas em algoritmos de aprendizagem de máquina. Nesses experimentos, foram utilizados os mesmos quatro algoritmos utilizados nos experimentos do grupo anterior. Como citado, para utilização desses classificadores foi necessário o pré-processamento das características para produção de

um número definido de características.

Capítulo 6

Resultados e análise

Os desempenhos dos classificadores serão apresentados e analisados com auxílio de gráficos ROC e taxas de acerto. Seguem abaixo as taxas de acerto dos classificadores baseados em HMM's para os experimentos com larvas separados por probabilidades iniciais. Nos principais experimentos, serão mostradas as matrizes de confusão para cada tipo de classificador por comportamento e a melhor curva de ROC por comportamento com base na sua AUC. Em seguida, serão expostos os resultados dos experimentos com algoritmos de aprendizagem.

6.1 Experimentos do grupo 1

No experimento em que as probabilidades iniciais dos HMM's do classificador eram aleatórias, foi obtida a taxa de acerto de 59%. A Tabela 6.1 e a Figura 6.1 mostram, respectivamente, a matriz de confusão e as curvas ROC desse experimento. Na figura, são mostradas duas curvas, cada uma representando uma classe e ambas com área de 0.6250.

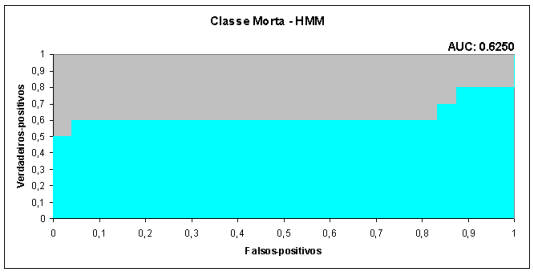
	Aleatório		Manual		Autom. (2)		Autom. (3)		Autom. (4)	
	Viva	Morta	Viva	Morta	Viva	Morta	Viva	Morta	Viva	Morta
Viva	14	10	22	2	22	2	23	1	22	2
Morta	4	6	4	6	2	8	2	8	2	8

Tabela 6.1: Matrizes de confusão dos experimentos com probabilidades iniciais variadas para os HMM's. O primeiro par de colunas apresenta a matriz de confusão considerando probabilidades aleatórias; o par seguinte considera probabilidades pré-computadas manualmente; e os outros 3 últimos pares consideram, nesta ordem, probabilidades pré-computadas automaticamente com 2, 3 e 4 estados.

Quando as probabilidades iniciais dos modelos foram alteradas para probabilidades pré-computadas manualmente, a taxa de acerto aumentou para 82%, como pode ser observado pela matriz de confusão, também apresentada na Tabela 6.1. Com essa configuração, as curvas ROC do classificador, ilustradas na Figura 6.2, foram alteradas. Com essa alteração, a AUC das curvas passaram a ser 0.8292.

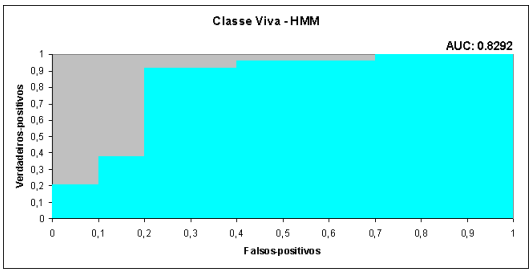
Por fim, quando utilizadas probabilidades iniciais pré-computadas automaticamente para os modelos, foi obtida a taxa de acerto de 88% com 2 e 4 estados e de 91% com 3 estados. As curvas

(a) Curva ROC para a classe Viva.

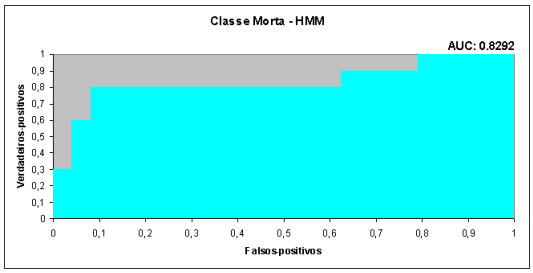


(b) Curva ROC para a classe Morta.

Figura 6.1: Curvas ROC para experimento com HMM's com probabilidades iniciais aleatórias.



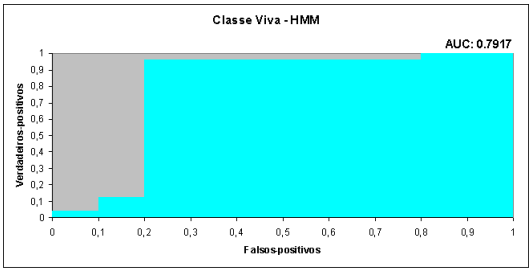
(a) Curva ROC para a classe Viva.



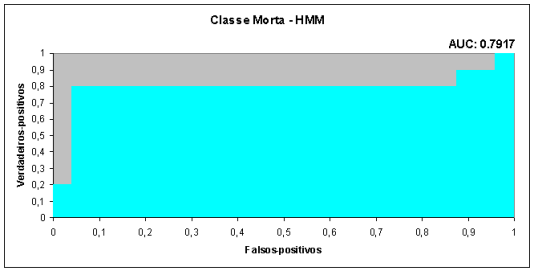
(b) Curva ROC para a classe Morta.

Figura 6.2: Curvas ROC para experimento com HMM's com probabilidades iniciais pré-computadas manualmente.

ROC com essas configurações são mostradas nas Figuras 6.3 – que corresponde ao classificador com 2 estados – e 6.4 – que corresponde aos classificadores com 3 e 4 estados. A área sob a curva do classificador com 2 estados é de 0.7917 e com 3 e 4 estados é de 0.7875.



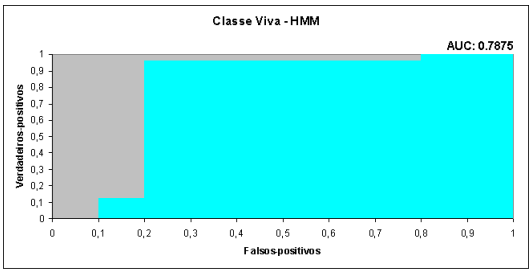
(a) Curva ROC para a classe Viva.



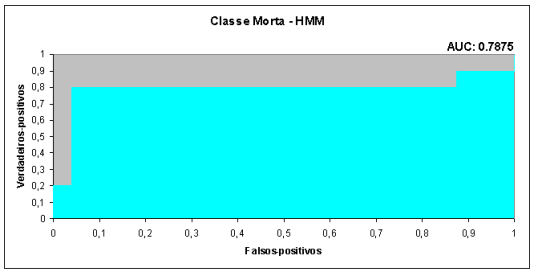
(b) Curva ROC para a classe Morta.

Figura 6.3: Curvas ROC para experimento com HMM's com probabilidades iniciais pré-computadas automaticamente com 2 estados.

Foi possível observar que, em conformidade ao afirmado por Rabiner em [38], os HMM's cujas probabilidades iniciais são predefinidas (obtidas de forma manual ou automática) apresentaram melhores resultados em comparação aos HMM's com probabilidades iniciais aleatórias. Ainda nesse quesito, nota-se ainda que os HMM's com probabilidades iniciais pré-computadas automaticamente alcançaram resultados aproximados àqueles com probabilidades inicias pré-computadas manualmente. Utilizando como critério apenas a taxa de acerto, os resultados foram superiores aos computados manualmente. Essa informação mostra que, em alguns casos, o cálculo manual



(a) Curva ROC para a classe Viva.



(b) Curva ROC para a classe Morta.

Figura 6.4: Curvas ROC para experimento com HMM's com probabilidades iniciais pré-computadas automaticamente com 3 e 4 estados.

das probabilidades iniciais (que pode ser exaustivo e impraticável) torna-se desnecessário, dada a existência de mecanismos automáticos para obtenção dessas probabilidades.

Dentre todas as configurações utilizadas e considerando o critério de parada inicialmente definido[1], a melhor taxa de acerto obtida com a utilização de classificadores baseados em HMM's foi alcançada com probabilidades iniciais pré-computadas automaticamente com modelos com 3 estados. No entanto, considerando as curvas ROC dos classificadores, esse classificador foi inferior ao classificador com HMM's cujas probabilidades iniciais foram pré-computadas manualmente, que apresentou a melhor AUC dentre todos os classificadores.

6.2 Experimentos do grupo 2

No experimento para análise do critério de parada do treinamento dos modelos, como apresentado abaixo, não foram verificadas mudanças consideráveis. Como descrito anteriormente, foram utilizados três novos critérios de parada contra o critério padrão (que considerou 9 iterações). Nos parágrafos abaixo, são apresentados os resultados obtidos considerando cada um dos modelos iniciais.

Com probabilidades iniciais aleatórias, poucas alterações foram encontradas. No primeiro critério, houve um aumento na taxa de acerto de 12%. No entanto, não trata-se de um aumento considerável, pois todas as instâncias foram classificadas como pertencentes à classe viva. Como essa classe possui mais instâncias no conjunto de treinamento, a probabilidade final se elevou. No segundo critério, não houve alteracão na taxa de acerto. O mesmo ocorreu com o terceiro critério quando o número de iteracões foi menor que 500. A partir disso, todas as instâncias também foram classificadas como pertencentes à classe viva, como ocorreu com o primeiro critério.

Considerando probabilidades iniciais pré-computadas manualmente, foi notada uma pequena alteração nas taxas de acerto em dois critérios de parada. O primeiro não apresentou alterações. No segundo, considerando um limiar de 10^{-1}, houve um aumento de 3% na taxa de acerto. Com limiares iguais a 10^{-2} e 10^{-3}, o aumento foi de 12%. Quando os limiares assumiram valores menores a esses, o aumento foi de 6%. No terceiro critério, houve um aumento de 6% em todas as variações de iterações. A Tabela 6.2 apresenta as matrizes de confusão para esses dois critérios.

[1]Nove iterações para parada do treinamento.

	Segundo						Terceiro	
	10^{-1}		10^{-2} 10^{-3}		10^{-4} 10^{-5} 10^{-6}		50 a 1000	
	Viva	Morta	Viva	Morta	Viva	Morta	Viva	Morta
Viva	21	3	24	0	24	0	24	0
Morta	2	8	2	8	4	6	4	6

Tabela 6.2: Matrizes de confusão para o segundo e terceiro critérios de parada. No segundo critério, são considerados diferentes limiares. No terceiro, todas as variações de iterações (de 50 a 1000) apresentaram os mesmos resultados.

Por fim, a taxa de classificação do classificador cujos modelos tinham probabilidades iniciais pré-computadas automaticamente (com 2 estados) reduziu em todos os critérios. Com o segundo critério, a redução foi de 17% e, com o primeiro e terceiro critérios, de 3%, como apresenta as matrizes de confusão na Tabela 6.3.

	Primeiro		Segundo		Terceiro	
	Viva	Morta	Viva	Morta	Viva	Morta
Viva	23	1	24	0	23	1
Morta	4	6	10	0	4	6

Tabela 6.3: Matrizes de confusão para modelos com probabilidades iniciais pré-computadas automaticamente e diferentes critérios de parada.

Constatou-se que houve uma leve interferência nos resultados de classificação inicialmente obtidos. O aumento máximo registrado foi de 12% com o segundo critério de parada e probabilidades iniciais pré-computadas manualmente. Ainda com essas probabilidades iniciais, foram obtidos aumentos de até 6%. Considerando a taxa de acerto, foram registrados outros aumentos não superiores ao máximo acima citado. No entanto, esses aumentos não foram considerados significantes pois os classificadores foram invariantes, classificando todas as entradas como pertencentes à mesma classe, que pode ser justificado pela ocorrência de *overfitting* durante o treinamento. Todos os resultados diferentes desses apresentaram redução nas taxas de acerto. Isso leva a crer que o número de iterações inicialmente utilizado é razoavelmente adequado para o conjunto de dados.

Como a maioria dos resultados dos experimentos para avaliação do critério de parada não apresentaram diferenças significativas com relação aos resultados iniciais, não foram confeccionadas as curvas ROC desses experimentos para análise. O mesmo ocorre com os resultados dos experimentos do grupo 3.

6.3 Experimentos do grupo 3

Os últimos resultados obtidos com classificadores baseados em HMM's são provenientes do experimento de combinação desses classificadores com algoritmos de aprendizagem de máquina. Embora a aplicação dessa abordagem seja favorável em alguns casos, como observado em outros

trabalhos, nesta aplicação, ela não foi positiva. Em todos os experimentos, não houve modificações nas taxas de acerto quando o limiar utilizado era inferior a 82%. Quando superior, todas as taxas encontradas foram inferiores àquelas obtidas com o classificador isolado. Nesse experimento, como mencionado, foi utilizada a configuração de modelos que apresentou os melhores resultados. Sendo assim, foram utilizados modelos com probabilidades iniciais pré-computadas automaticamente com 3 estados.

6.4 Experimentos do grupo 4

Para comparação com os experimentos anteriores, serão apresentados abaixo os resultados dos experimentos realizados com algoritmos de aprendizagem de máquina sobre o mesmo conjunto de dados. Nesta ordem, as Tabelas 6.4, 6.5 e 6.6 apresentam as matrizes de confusão obtidas com a utilização dos algoritmos IBk, J48, SMO e MLP com os pré-processamentos que produzem 2, 4 e 1 características[2]. Em seguida, são apresentadas nas Figuras 6.5, 6.7 e 6.9 as melhores curvas ROC para cada um dos 3 pré-processamentos com base no valor de suas AUC's. Em contrapartida, as Figuras 6.6, 6.8 e 6.10 apresentam as piores curvas. As demais curvas podem ser visualizadas no Apêndice B.

	IBk (k=1)		J48		SMO		MLP	
	Viva	Morta	Viva	Morta	Viva	Morta	Viva	Morta
Viva	23	1	23	1	24	0	23	1
Morta	0	10	0	10	7	3	3	7

Tabela 6.4: Matrizes de confusão dos experimentos com algoritmos de aprendizagem de máquina (2 características).

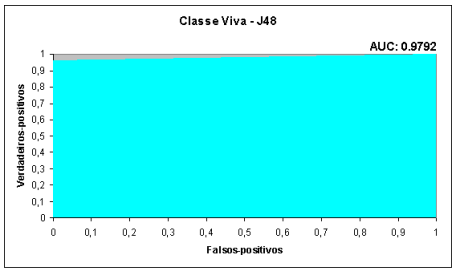
	IBk (k=2)		J48		SMO		MLP	
	Viva	Morta	Viva	Morta	Viva	Morta	Viva	Morta
Viva	23	1	24	0	24	0	23	1
Morta	2	8	2	8	6	4	3	7

Tabela 6.5: Matrizes de confusão dos experimentos com algoritmos de aprendizagem de máquina (4 características).

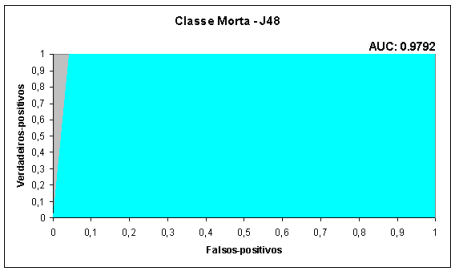
	IBk (k=2)		J48		SMO		MLP	
	Viva	Morta	Viva	Morta	Viva	Morta	Viva	Morta
Viva	24	0	23	1	24	0	24	0
Morta	2	8	2	8	10	0	7	3

Tabela 6.6: Matrizes de confusão dos experimentos com algoritmos de aprendizagem de máquina (1 característica).

[2]Com exceção do número de vizinhos mais próximos do algoritmo IBk, foram utilizados os valores padronizados definidos pela ferramenta Weka para os parâmetros dos algoritmos.

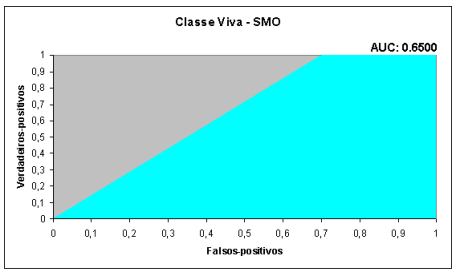


(a) Obtida com o algoritmo J48 para classe Viva.

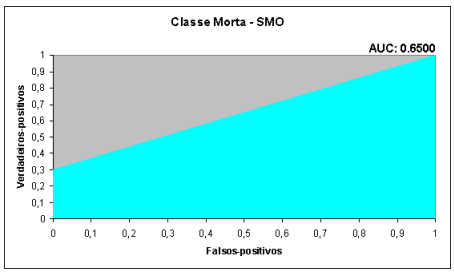


(b) Obtida com o algoritmo J48 para classe Morta.

Figura 6.5: Melhores curvas ROC entre algoritmos de aprendizagem de máquina (utilizando 2 características).

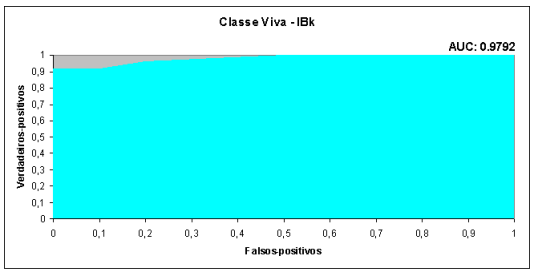


(a) Obtida com o algoritmo SMO para classe Viva.

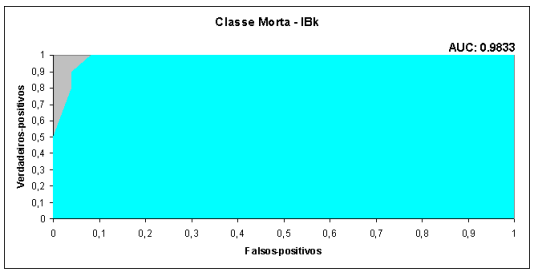


(b) Obtida com o algoritmo SMO para classe Morta.

Figura 6.6: Piores curvas ROC entre algoritmos de aprendizagem de máquina (utilizando 2 características).



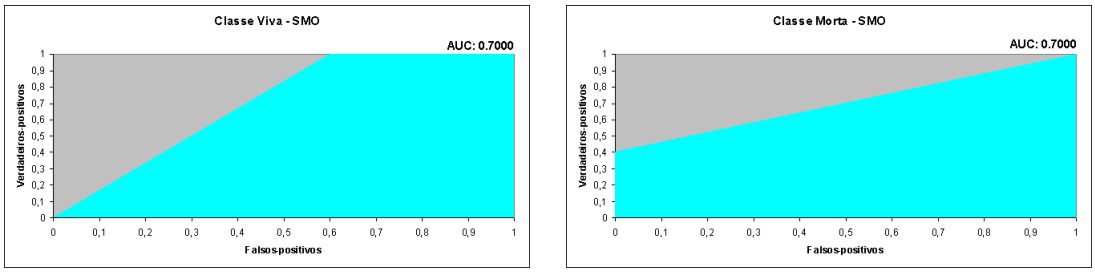
(a) Obtida com o algoritmo IBk para classe Viva.



(b) Obtida com o algoritmo IBk para classe Morta.

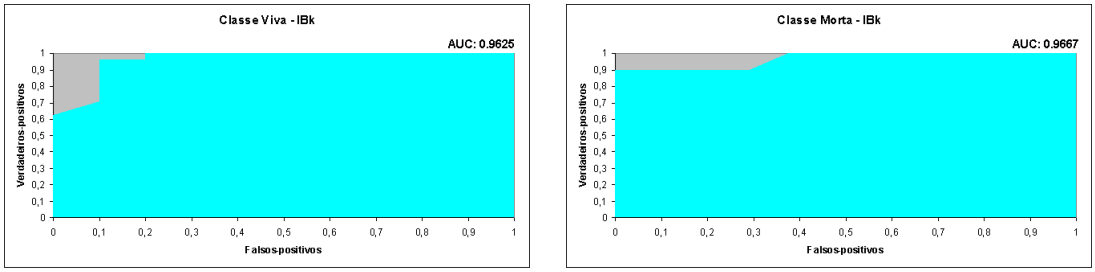
Figura 6.7: Melhores curvas ROC entre algoritmos de aprendizagem de máquina (utilizando 4 características).

Analisando isoladamente os classificadores com aprendizagem de máquina, os melhores resultados foram obtidos com o algoritmo IBk, cuja curva da classe viva apresentou área de 0.9792 e da classe morta de 0.9833, e os piores resultados foram obtidos com o algoritmo SMO, com AUC de 0.5 para ambas as classes. Utilizando esses valores como referenciais, o desempenho dos classificadores baseados em HMM's foi mediano, dado que suas AUC's ficaram entre 0.8292 e 0.6250 para as duas classes. No entanto, desconsiderando que os resultados do classificador com MLP foi muito próximo aos resultados do classificador com HMM, esse último foi superior apenas ao classificador com SMO.



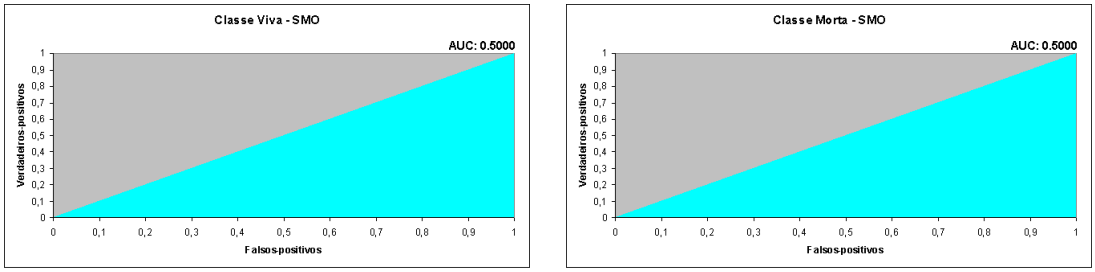
(a) Obtida com o algoritmo SMO para classe Viva. (b) Obtida com o algoritmo SMO para classe Morta.

Figura 6.8: Piores curvas ROC entre algoritmos de aprendizagem de máquina (utilizando 4 características).



(a) Obtida com o algoritmo IBk para classe Viva. (b) Obtida com o algoritmo IBk para classe Morta.

Figura 6.9: Melhores curvas ROC entre algoritmos de aprendizagem de máquina (utilizando 1 característica).



(a) Obtida com o algoritmo SMO para classe Viva. (b) Obtida com o algoritmo SMO para classe Morta.

Figura 6.10: Piores curvas ROC entre algoritmos de aprendizagem de máquina (utilizando 1 característica).

Como mencionado acima, embora a estrutura dos HMM's seja adequada para entradas de tamanhos variáveis e com informações temporais, os resultados dos classificadores baseados nesses modelos para a aplicação aqui atacada não foram favoráveis em comparação com outras alternativas. Isso, provavelmente, se deve ao fato de existirem poucas variações (ou nenhuma) nas características da classe morta no conjunto de dados utilizado, o que não ocorre nas características da outra classe. Desse modo, com auxílio de pré-processamento dessas características, os outros classificadores podem ter se tornado mais eficazes na classificação dos comportamentos desta aplicação.

Capítulo 7

Considerações finais

Neste trabalho, foram realizados experimentos com larvas do mosquito transmissor da dengue, o *Aedes aegypti*, com o objetivo de contribuir com um sistema de visão computacional criado para auxiliar pesquisadores das áreas biológicas da Universidade Católica Dom Bosco (UCDB) em testes laboratoriais que analisam a eficácia de larvicidas extraídos de plantas no combate às larvas desse inseto. Esse sistema – construído e mantido por um grupo de profissionais da área de visão computacional, o INOVISAO – faz parte de um projeto de pesquisa, denominado LARVIC, criado por pesquisadores de diversas áreas da UCDB.

O papel desse sistema é auxiliar na contagem de larvas vivas e mortas contidas em recipientes com concentrações de larvicidas. Atualmente, essa contagem é totalmente realizada por especialistas, que observam os recipientes e anotam em determinados instantes as quantidades de larvas vivas e mortas. Com a inserção de um sistema eficaz de visão computacional que automaticamente executa esse trabalho, é possível obter melhores resultados, em comparação com a metodologia atual, devido à inexistência de erros humanos prováveis decorrentes de fadiga, exaustão, subjetividade, falta de conhecimento, dentre outros.

Contudo, para obtenção de um sistema computacional eficiente, são necessários o estudo, aplicação e testes de técnicas adequadas para solucionar o problema-alvo. Nesse sentido, o objetivo desse trabalho foi analisar a aplicação de modelos de Markov ocultos na etapa de classificação dos comportamentos das larvas. Em outras palavras, visou-se analisar se, com base nas informações obtidas das imagens em etapas anteriores, esses modelos estatísticos são adequados para diferenciar corretamente larvas vivas de larvas mortas.

Para isso, foram realizados diferentes experimentos envolvendo tais modelos, com o intuito de realizar uma investigação abrangente nesta aplicação. Nos experimentos, primeiramente, foi analisado o conjunto de probabilidades inicial dos modelos. Segundo a literatura, essas probabilidades podem interferir significativamente no processo de treinamento e, consequentemente, na classificação. Foram analisados três conjuntos de probabilidades, compostos por 1) probabilidades uniformes e aleatórias, 2) probabilidades pré-computadas manualmente e 3) probabilidades pré-computadas automaticamente. Com os resultados desses experimentos, foi possível constatar que os classificadores com modelos cujas probabilidades iniciais foram pré-computadas obtiveram melhores resultados que aqueles com modelos aleatórios. Além disso, foi observado que os resultados obtidos com modelos pré-computados automaticamente foram muito próximos aos obtidos com modelos pré-computados manualmente, o que indica que, em alguns casos, a tarefa onerosa de obtenção de probabilidades iniciais pode ser desnecessária.

Em seguida, foram analisados diferentes critérios de parada para o algoritmo de treinamento, o *Baum-Welch*. Os resultados desses experimentos puderam ser utilizados para verificar se o critério de parada utilizado nos experimentos anteriores era adequado. De fato, poucas modificações consideráveis foram observadas nesses experimentos com relação aos experimentos anteriores. Com isso, pode-se inferir que, considerando o custo computacional e a proporção da diferença em desempenho de classificação, o critério anteriormente utilizado (de nove iterações) foi apropriado para a aplicação.

Por fim, considerando classificadores que utilizam HMM's, o último experimento averiguou a combinação de tais classificadores com algoritmos de aprendizagem de máquina. De acordo com outros trabalhos, em alguns casos, a utilização desses algoritmos como auxiliares de classificadores baseados em HMM's para determinadas circunstâncias pode elevar o poder de classificação desses classificadores. Neste trabalho, os classificadores foram combinados com quatro algoritmos – árvores de decisão, *k-Nearest Neighbor*, máquinas de vetor de suporte e redes neurais. O parâmetro utilizado para ativação dos algoritmos foi baseado na existência de probabilidades máximas próximas entre modelos associados a diferentes classes. Nessas circunstâncias, os algoritmos realizaram as classificações finais.

Em todos os experimentos que envolveram classificadores combinados, não houve aumento nas taxas de acerto quando o limiar utilizado foi inferior a 82%. Nos outros casos, todas as taxas encontradas foram inferiores àquelas obtidas com o classificador trabalhando isoladamente. Para esses experimentos, foram utilizados classificadores com modelos com probabilidades iniciais pré-computadas automaticamente com 3 estados, pois foi a configuração que apresentou os melhores resultados. Esses resultados podem ser provenientes da inexistência de variações relevantes nas imagens utilizadas, especialmente, nas imagens de larvas mortas.

De acordo com os resultados obtidos com classificadores baseados nos mesmos quatro algoritmos de aprendizagem utilizados nas combinações, pôde-se observar que o desempenho dos classificadores baseados em HMM's aplicados à obtenção de taxas de mortalidade de larvas foi considerável, porém, inferior ao desempenho de outros classificadores.

Provavelmente, esse resultado está relacionado ao padrão dos comportamentos existentes na base de dados, que, possivelmente, pode ser mais bem classificado utilizando números finitos de características e, em alguns casos, desconsiderando padrões temporais. Outra possível causa é a existência de ruídos nas entradas, que, por conta do pré-processamento, podem ter sido amenizados, favorecendo alguns classificadores. Seguem abaixo algumas sugestões de trabalhos futuros que podem ser desenvolvidos para contribuir cientificamente com esta aplicação.

A análise de um conjunto maior de diferentes amostras de larvas vivas e mortas, como sequências de imagens de larvas em processo de falência, pode encontrar padrões que permitam a aplicação favorável de HMM's na etapa de classificação desse problema devido ao fator temporal. A utilização de outros algoritmos de ajuste de parâmetros dos modelos, como realizado em [22], pode permitir a identificação de comportamentos não considerados pelo algoritmo *Baum-Welch*. Por fim, é cabível também analisar o desempenho dos classificadores baseados em algoritmos de aprendizagem de máquina (como J48, IBk e MLP, que apresentaram bons resultados) em amostras com padrões diferenciados. Adicionalmente, podem ser analisados os desempenhos de outros classificadores (ou combinações de classificadores) que possam ser adequados ao problema aqui estudado.

Apêndices A

Conceitos suplementares

A.1 Algoritmos Forward e Backward

A Figura A.1 demonstra graficamente os valores da variável α na análise de uma sequência de observações de tamanho 5 em um HMM com 3 estados. O valor de $\alpha_i(s_x)$ indica a probabilidade do estado s_x ter gerado a *i-ésima* observação da sequência, desconsiderando as observações seguintes. Por exemplo, na Figura A.1, $\alpha_3(1)$ representa a probabilidade do estado 1 ter gerado a terceira observação da sequência, desconsiderando o quarto e o quinto símbolo da sequência de observações.

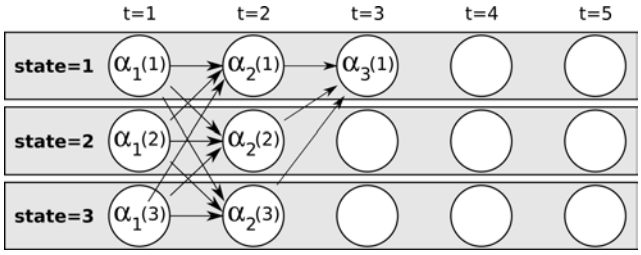


Figura A.1: Treliça representando os valores de α nas possíveis sequências de estados geradoras de uma sequência de observações de 5 símbolos em um HMM com 3 estados.

Adicionalmente, temos a Figura A.2, demonstrando graficamente os valores da variável β utilizando a mesma sequência de observações e o mesmo HMM. O valor da variável $\beta_i(s_x)$ representa a probabilidade de geração dos símbolos $o_{i+1}, ..., o_T$ da sequência de observações, dada a geração do *i-ésimo* símbolo da sequência de observações no estado s_x. Na figura, $\beta_3(1)$ representa a probabilidade de geração do quarto e do quinto símbolo da sequência de observações, dado que o terceiro símbolo foi gerado no estado 1.

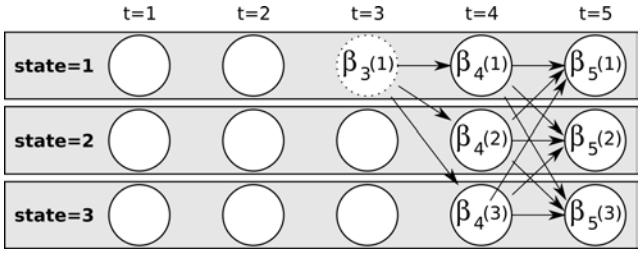


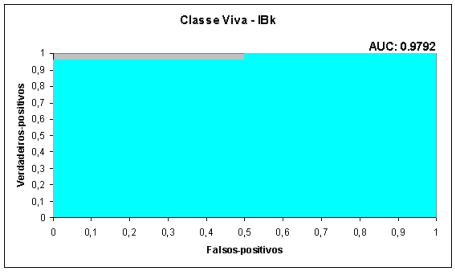
Figura A.2: Treliça representando os valores de β nas possíveis sequências de estados. Note que o círculo que contém o valor $\beta_3(1)$ está tracejado, pois a probabilidade de geração do terceiro símbolo não está agregada no valor da variável.

Apêndices B

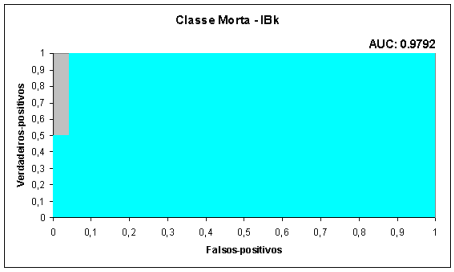
Curvas ROC adicionais

Seguem abaixo as curvas obtidas por meio da utilização de algoritmos de aprendizagem de máquina em que foram realizados 3 pré-processamentos diferentes. Os algoritmos utilizados foram IBk, J48, SMO e MLP.

B.1 Pré-processamento com 2 características

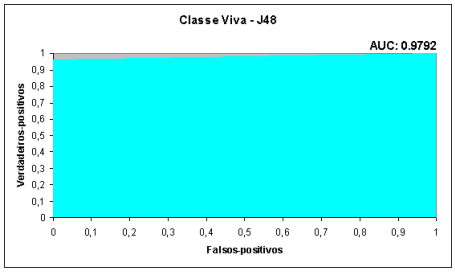


(a) Curva ROC para a classe Viva.

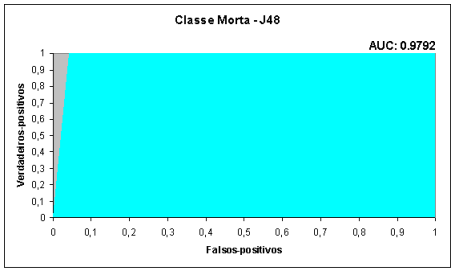


(b) Curva ROC para a classe Morta.

Figura B.1: Curvas ROC para algoritmo IBk (2 características).

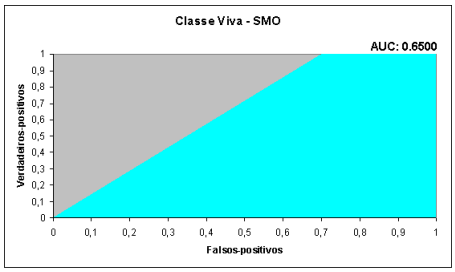


(a) Curva ROC para a classe Viva.

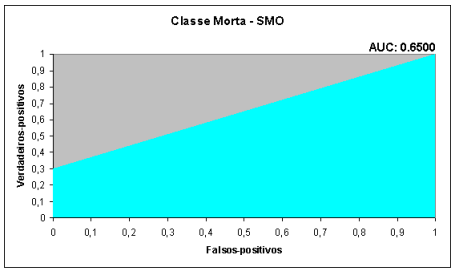


(b) Curva ROC para a classe Morta.

Figura B.2: Curvas ROC para algoritmo J48 (2 características).

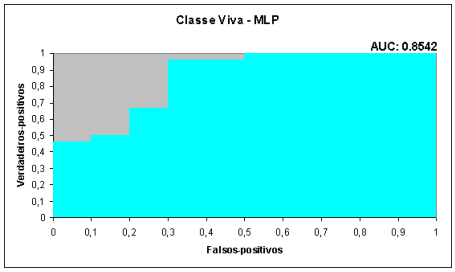


(a) Curva ROC para a classe Viva.

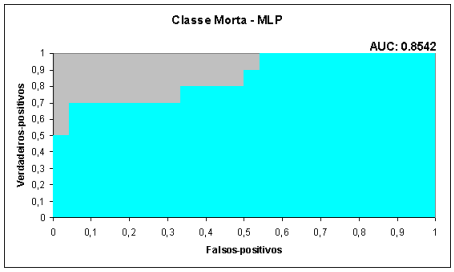


(b) Curva ROC para a classe Morta.

Figura B.3: Curvas ROC para algoritmo SMO (2 características).



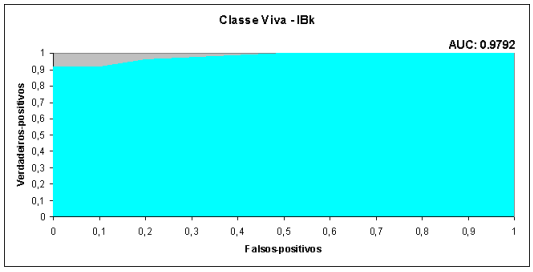
(a) Curva ROC para a classe Viva.



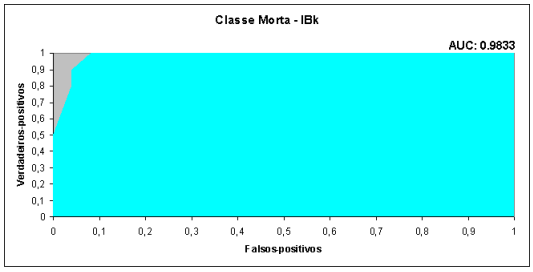
(b) Curva ROC para a classe Morta.

Figura B.4: Curvas ROC para algoritmo MLP (2 características).

B.2 Pré-processamento com 4 características



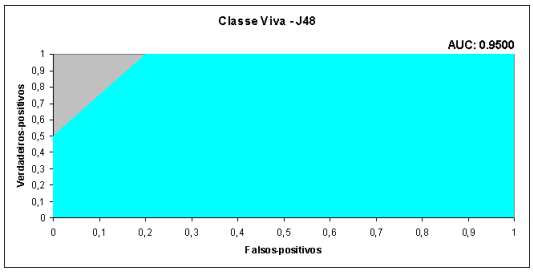
(a) Curva ROC para a classe Viva.



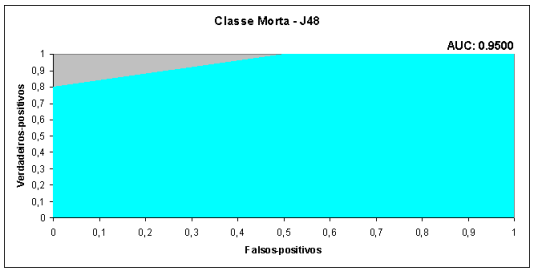
(b) Curva ROC para a classe Morta.

Figura B.5: Curvas ROC para algoritmo IBk (4 características).

B.3 Pré-processamento com 1 característica

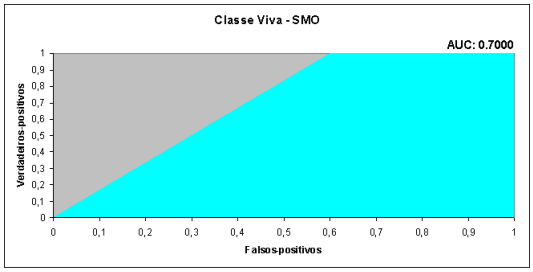


(a) Curva ROC para a classe Viva.

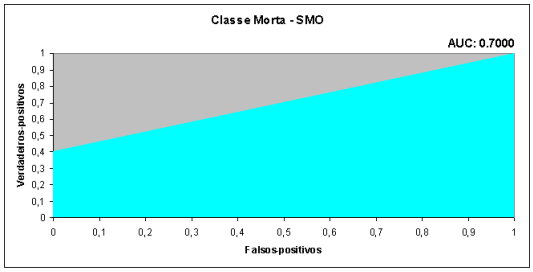


(b) Curva ROC para a classe Morta.

Figura B.6: Curvas ROC para algoritmo J48 (4 características).

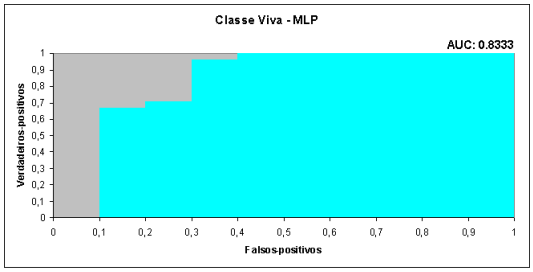


(a) Curva ROC para a classe Viva.

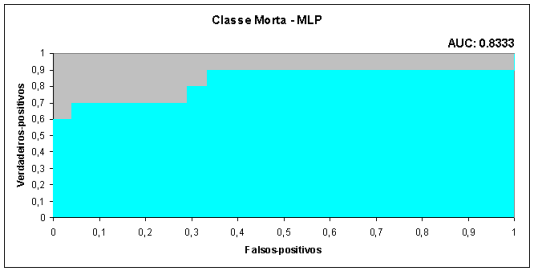


(b) Curva ROC para a classe Morta.

Figura B.7: Curvas ROC para algoritmo SMO (4 características).

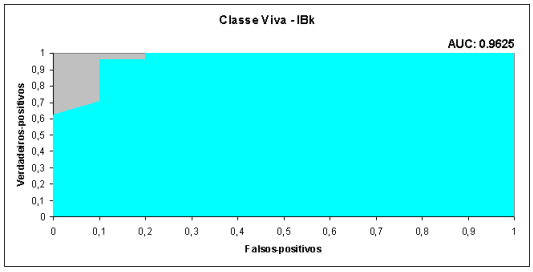


(a) Curva ROC para a classe Viva.

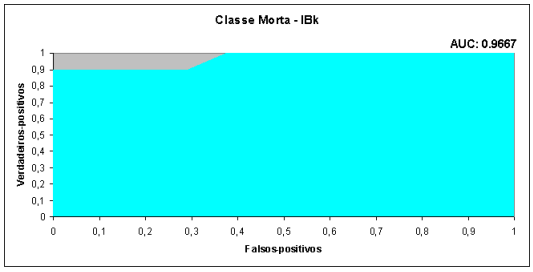


(b) Curva ROC para a classe Morta.

Figura B.8: Curvas ROC para algoritmo MLP (4 características).

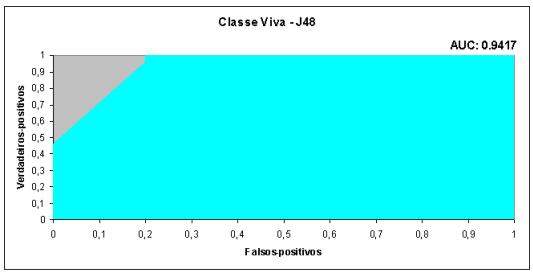


(a) Curva ROC para a classe Viva.

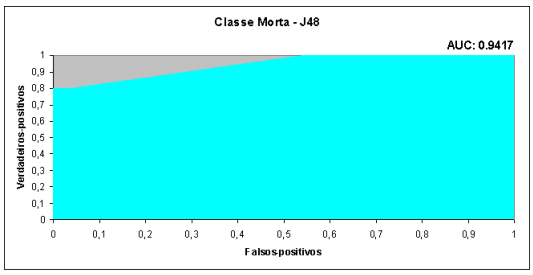


(b) Curva ROC para a classe Morta.

Figura B.9: Curvas ROC para algoritmo IBk (1 característica).

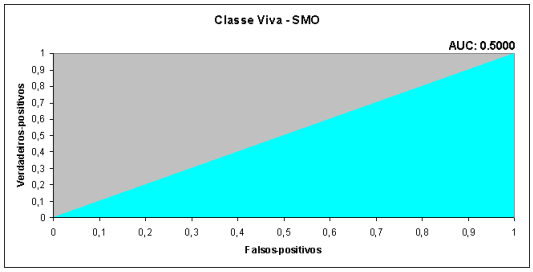


(a) Curva ROC para a classe Viva.

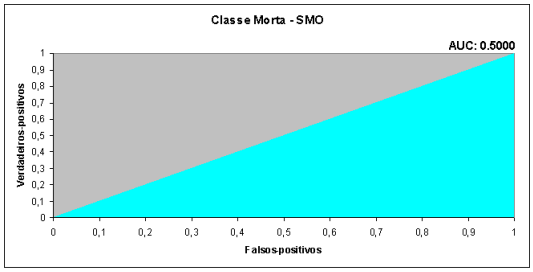


(b) Curva ROC para a classe Morta.

Figura B.10: Curvas ROC para algoritmo J48 (1 característica).

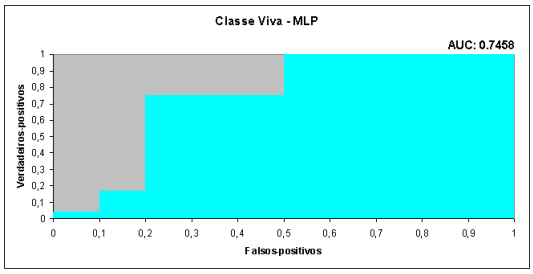


(a) Curva ROC para a classe Viva.

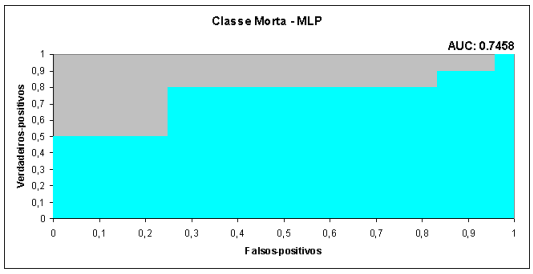


(b) Curva ROC para a classe Morta.

Figura B.11: Curvas ROC para algoritmo SMO (1 característica).



(a) Curva ROC para a classe Viva.



(b) Curva ROC para a classe Morta.

Figura B.12: Curvas ROC para algoritmo MLP (1 característica).

Apêndices C

Algoritmo K-Means

O algoritmo K-Means agrupa um conjunto com N pontos em K grupos de forma não-supervisionada [24, 43]. Um exemplo prático de aplicação desse algoritmo seria o agrupamento de produtos comercializados em dois grupos – rentáveis e não rentáveis – de acordo com suas características, como lucro percentual, unidades vendidas, entre outras. Ao fim do agrupamento, alguns produtos estarão no grupo de produtos rentáveis e os outros no outro grupo. Essa informação poderia ser utilizada, por exemplo, em comércios durante a aquisição de produtos para venda.

No algoritmo, devem ser definidos o número de grupos e a localização de cada um desses grupos. Adicionalmente, deve ser estipulada a métrica a ser utilizada para realização dos agrupamentos, como as distâncias Euclidiana e de Mahalanobis, por exemplo. Além disso, deve ser ainda estabelecido o critério de parada do algoritmo. Portanto, o número de iterações do algoritmo é variável de acordo com essas definições. A cada iteração, o algoritmo rotula cada um dos pontos como pertencente a um dos grupos definidos, utilizando a métrica definida. Em seguida, atualiza os valores de localização de cada um dos grupos, de acordo com os pontos associados a cada um deles. Esse processo se repete até que o critério de parada seja satisfeito.

Para melhor compreensão, considere o exemplo dado no início deste capítulo. A Tabela C.1 apresenta os produtos a serem agrupados com suas respectivas características. Nesse caso, o número de grupos é igual a dois (rentáveis e não rentáveis). Algumas alternativas podem ser utilizadas para a definição dos valores iniciais da localização dos grupos. Uma delas é a utilização de valores aleatórios. Outra é a utilização de alguns pontos do conjunto, que também podem ser escolhidos aleatoriamente. Para esse exemplo, utilizaremos a segunda alternativa e definimos a localização de um grupo como sendo $(50; 1700)$ e do outro como $(80; 1500)$.

Cód.	Lucro %	Vendas mensais
1	50	1700
2	80	1500
3	20	2000
4	110	2
5	40	10

Tabela C.1: Exemplo de produtos a serem agrupados.

Produtos			Distâncias		Grupo
Cód.	Lucro %	Vendas mensais	G_1	G_2	
1	50	1700	0.0	202.24	G_1
2	80	1500	202.24	0.0	G_2
3	20	2000	301.50	503.59	G_1
4	110	2	1699.06	1498.30	G_2
5	40	10	1690.03	1490.54	G_2

Tabela C.2: Primeira iteração do algoritmo para o exemplo dado.

Ainda no exemplo, podemos definir que a métrica utilizada será a distância euclidiana[1]. Por fim, a parada do algoritmo ocorrerá quando nenhum ponto apresentar alteração de grupo de uma iteração para outra. Para um ponto qualquer X, essa alteração ocorre se, na iteração t, esse ponto for rotulado como pertencente a uma determinada classe e, na iteração $t + 1$, for rotulado como pertencente a outra classe. Dadas essas definições, o algoritmo está apto para ser executado.

Para execução de uma iteração do algoritmo, devemos calcular a distância entre cada ponto e todos os grupos. Por fim, o ponto é rotulado como pertencente ao grupo que apresentar a menor distância. A Tabela C.2 mostra a distância entre cada ponto e os dois grupos, representados por G_1 e G_2. Em seguida, rotulamos cada ponto como pertencente ao grupo mais próximo a esse ponto. Por exemplo, de acordo com a métrica definida, o terceiro ponto (que representa o produto de código 3) está mais próximo do grupo G_1 (301.50) que do grupo G_2 (503.59). Logo, esse ponto foi rotulado como pertencente ao grupo G_1, como apresentado na tabela. O mesmo procedimento é executado para todos os demais pontos.

Em seguida, após a rotulação de todos os pontos, a localização dos grupos é atualizada de acordo com os pontos pertencentes a eles após o passo anterior. Essa atualização é realizada por meio da média dos pontos selecionados. Por exemplo, de acordo com as rotulações obtidas, os pontos referentes aos produtos de códigos 1 e 3 pertencem ao grupo G_1 e os outros pontos ao grupo G_2. Logo, com base na média dos dois pontos, a localização do grupo G_1 passa a ser $(150.75; 352.92)$ e do grupo G_2, pela média dos outros três pontos, passa a ser $(1197.11; 996.28)$. Realizada a atualização, a próxima iteração pode ser realizada. Para isso, o mesmo ocorre: calculam-se as distâncias entre pontos e grupos e, posteriormente, rotulam-se os pontos. A Tabela C.3 apresenta as distâncias de cada ponto (considerando agora as novas localizações) para cada um dos pontos e a respectiva rotulação atribuída a eles.

Note que, na primeira iteração, o ponto referente ao produto de código 2 foi associado ao grupo G_2 e, na segunda, ao grupo G_1, o que caracteriza uma alteração de grupo para esse ponto. Portanto, de acordo com o critério definido, o algoritmo continua sua execução e parte para a terceira iteração. A Tabela C.4 apresenta as distâncias e o rótulo de cada ponto para essa iteração. Como todos os rótulos atribuídos na iteração anterior foram mantidos, o critério de parada foi satisfeito e o algoritmo é encerrado. Nesse exemplo, o algoritmo agrupou os produtos 1, 2 e 3 em um grupo (que pode ser dos rentáveis) e os demais no outro grupo.

[1]A distância euclidiana entre dois pontos de dimensão M é definida por $\sqrt{\sum_{i=1}^{M}(x_i - y_i)^2}$, em que x_i e y_i correspondem ao valor da i-ésima dimensão do primeiro e do segundo ponto.

Produtos			Distâncias		Grupo
Cód.	Lucro %	Vendas mensais	G_1	G_2	
1	50	1700	150.75	1196.29	G_1
2	80	1500	352.88	996.01	G_1
3	20	2000	150.75	1497.07	G_1
4	110	2	1849.52	503.11	G_2
5	40	10	1840.01	495.36	G_2

Tabela C.3: Segunda iteração do algoritmo para o exemplo dado.

Produtos			Distâncias		Grupo
Cód.	Lucro %	Vendas mensais	G_1	G_2	
1	50	1700	33.33	1694.18	G_1
2	80	1500	235.25	1494.01	G_1
3	20	2000	268.35	1994.76	G_1
4	110	2	1732.37	35.23	G_2
5	40	10	1723.36	35.23	G_2

Tabela C.4: Terceira iteração do algoritmo para o exemplo dado.

Referências Bibliográficas

[1] H. Bahi and M. Sellami. Combination of vector quantization and hidden markov models for arabic speech recognition. In *AICCSA '01: Proceedings of the ACS/IEEE International Conference on Computer Systems and Applications*, page 96, Washington, DC, USA, 2001. IEEE Computer Society.

[2] M. A. S. F. Bezerra. Segmentação de pele em imagens utilizando redes neurais. In *I Workshop de Visão Computacional*, Universidade Metodista de Piracicaba - UNIMEP, Piracicaba, SP, 2005.

[3] A. C. S. Braga. *Curvas ROC: Aspectos funcionais e aplicações*. PhD thesis, Universidade do Minho, Braga, Portugal, 2000.

[4] C. J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2:121–167, 1998.

[5] R. J. C. Cenens and J. V. Impe. Evaluation of different shape parameters to distinguish between flocs and filaments in activated sludge images. *Water science and technology : a journal of the International Association on Water Pollution Research*, 45:85–91, 2002.

[6] A. M. Cunha and L. Velho. Hidden markov models. Technical report, Instituto Nacional de Matemática Pura e Aplicada, 2002.

[7] R. Cutler and L. Davis. View-based detection and analysis of periodic motion. In *ICPR '98: Proceedings of the 14th International Conference on Pattern Recognition*, volume 1, page 495. IEEE Computer Society, 1998.

[8] F. Dellaert. The expectation maximization algorithm. Technical Report GIT-GVU-02-20, Georgia Institute of Technology, College of Computing, February 2002.

[9] J. B. Dias and K. P. Souza. Reconhecimento de gestos utilizando modelos de markov ocultos. Monografia (Graduação), 2006. Universidade Católica Dom Bosco, Campo Grande, MS.

[10] S. Eickeler, A. Kosmala, and G. Rigoll. Hidden Markov Model Based Continuous Online Gesture Recognition. In *Int. Conference on Pattern Recognition (ICPR)*, pages 1206–1208, Brisbane, 1998.

[11] A. El-Yacoubi, R. Sabourin, M. Gilloux, and C. Y. Suen. *Off-line handwritten word recognition using hidden Markov models*, pages 191–230. CRC Press, Inc., Boca Raton, FL, USA, 1999.

[12] T. Fawcett. An introduction to roc analysis. *Pattern Recognition Letters*, 27(8):861–874, 2006.

[13] A. Feldman and T. Balch. Automatic identification of bee movement. Technical report, Georgia Institute of Technology, Atlanta, Georgia 30332, USA, 2003.

[14] R. S. Fibiger. Estudo comparativo de técnicas de visão computacional para detecção de pele humana. Monografia (Graduação), 2004. Universidade Católica Dom Bosco, Campo Grande, MS.

[15] J. S. Fonseca and G. A. Martins. *Curso de estatística*. Editora Atlas, 1982.

[16] M. A. D. Frick. Caracterização de minério de ferro por visão computacional. Dissertação de Mestrado, 2008. Universidade Federal de Santa Maria, Santa Maria, RS.

[17] W. N. Gonçalves, V. A. S. Silva, B. B. Machado, J. A. Silva, K. P. Souza, and H. Pistori. Técnicas de segmentação baseadas em subtração de fundo e modelos de cores: Um estudo comparativo. In *XXVIII CILAMCE - Iberian Latin American Congress on Computational Methods in Engineering*, 2007.

[18] J. Heikkilä and O. Silvén. A real-time system for monitoring of cyclists and pedestrians. In *VS '99: Proceedings of the Second IEEE Workshop on Visual Surveillance*, page 74. IEEE Computer Society, 1999.

[19] B. Q. Huang, C. J. Du, Y. B. Zhang, and M.-T. Kechadi. A hybrid hmm-svm method for online handwriting symbol recognition. In *ISDA '06: Proceedings of the Sixth International Conference on Intelligent Systems Design and Applications*, pages 887–891, Washington, DC, USA, 2006. IEEE Computer Society.

[20] O. Javed, K. Shafique, and M. Shah. A hierarchical approach to robust background subtraction using color and gradient information. In *MOTION '02: Proceedings of the Workshop on Motion and Video Computing*, page 22. IEEE Computer Society, 2002.

[21] M. J. Jones and J. M. Rehg. Statistical color models with application to skin detection. *International Journal of Computer Vision*, 46(1), 2002.

[22] N. Liu, R. I. A. Davis, B. C. Lovell, and P. J. Kootsookos. Effect of initial hmm choices in multiple sequence training for gesture recognition. In *ITCC '04: Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'04)*, volume 2, page 608, Washington, DC, USA, 2004. IEEE Computer Society.

[23] N. Liu and B. C. Lovell. Gesture classification using hidden markov models and viterbi path counting. In *Proceedings of the Seventh International Conference on Digital Image Computing: Techniques and Applications*, pages 273–282, 2003.

[24] D. J. C. MacKay. *Information Theory, Inference & Learning Algorithms*, chapter 20. Cambridge University Press, 2003.

[25] M. Matetic, S. Ribaric, and I. Ipsic. Qualitative modelling and analysis of animal behaviour. *Applied Intelligence*, 21(1):25–44, 2004.

[26] A. M. McIvor. Background subtraction techniques. In *Image and Vision Computing New Zealand*, 2000.

[27] J. A. Montero and L. E. Sucar. Feature selection for visual gesture recognition using hidden markov models. In *Proceedings of the Fifth Mexican International Conference in Computer Science*, volume 0, pages 196–203, Los Alamitos, CA, USA, 2004. IEEE Computer Society.

[28] M. Morita and L. S. Oliveira. Introdução aos modelos escondidos de markov. Technical report, PPGIA-PUCPR, Curitiba-Brazil, November 1998.

[29] M. Nixon and A. Aguado. *Feature Extraction & Image Processing*. Newnes, January 2002.

[30] J. J. Oliveira Júnior, J. M. Carvalho, C. O. A. Freitas, and R. Sabourin. Evaluating nn and hmm classifiers for handwritten word recognition. In *SIBGRAPI '02: Proceedings of the 15th Brazilian Symposium on Computer Graphics and Image Processing*, pages 210–217, Washington, DC, USA, 2002. IEEE Computer Society.

[31] H. Pistori, E. J. de Arruda, A. R. Roel, K. R. de Andrade Porto, V. V. V. A. Odakura, and J. A. Colombo. Larvic - monitoramento da criação e bioensaios com larvas de aedes aegyptti l. e artemia salina l. utilizando visão computacional. Technical report, Universidade Católica Dom Bosco e Tec-Sinapse Tecnologia da Informação Ltda., 2007.

[32] H. Pistori, P. S. Martins, M. C. Pereira, and J. J. Neto. Sigus - plataforma de apoio ao desenvolvimento de sistemas para inclusão digital de pessoas com necessidades especiais. In *IV Congresso Iberdiscap: Tecnologias de Apoio a Portadores de Deficiência*, Vitória, ES, February 2006.

[33] H. Pistori and K. P. Souza. Tecnologia adaptativa aplicada à biotecnologia: Estudos de caso e oportunidades. In *WTA2010 - IV Workshop de Tecnologia Adaptativa*. Escola Politécnica da USP, São Paulo, SP, 2010.

[34] K. R. A. Porto, A. R. Roel, M. M. Silva, R. M. Coelho, E. J. D. Schleder, and A. H. Jeller. Atividade larvicida do óleo de anacardium humile saint hill sobre aedes aegypti. *Revista da Sociedade Brasileira de Medicina Tropical*, 41:1–4, 2008.

[35] R. C. Prati, G. E. A. P. A. Batista, and M. C. Monard. Curvas roc para a avaliação de classificadores. *Revista IEEE América Latina*, 2008.

[36] D. Prescher. A tutorial on the expectation-maximization algorithm including maximum-likelihood estimation and em training of probabilistic context-free grammars. *ArXiv Computer Science e-prints*, December 2004.

[37] J. H. S. Queiroz, H. Pistori, K. R. Porto, and A. A. Roel. Rastreamento de múltiplas larvas utilizando técnicas de visão computacional: Resultados preliminares. In *Workshop de Iniciação Científica - XXII SIBGRAPI*, Rio de Janeiro, RJ, 2009.

[38] L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. In *Proceedings of the IEEE*, volume 77, pages 42–53. IEEE Computer Graphics and Applications, 1989.

[39] L. R. Rabiner and B.-H. Juang. *Fundamentals of Speech Recognition*, chapter 10. Prentice Hall PTR, 1993.

[40] S. Ramachandrula and S. Thippur. Scalar quantization of features in discrete hidden markov models. In *International Conference on Information, Communications and Signal Processing*, 1997.

[41] A. P. L. Rossi, K. R. A. Porto, L. C. de Oliveira, A. R. Roel, and E. J. Arruda. Efeito tóxico de íons metálicos, edta e sbti (inibidor de tripsina de soja, tipo kunitz) contra aedes aegypti (culicidae) e artemia salina (artemidae). *QUÍMICAHOJE - Revista da Federação Nacional dos Profissionais da Química*, pages 9–13, 2007.

[42] J. C. Russ. *The image processing handbook (3rd ed.).* CRC Press, Inc., Boca Raton, FL, USA, 1999.

[43] S. Russel and P. Norvig. *Inteligência Artificial.* Elsevier, 2004.

[44] S. Sayayama and S. Takahashi. On the use of scalar quantization for fast hmm computation. In *International Conference on Acoustics, Speech, and Signal Processing*, 1995.

[45] J. A. Silva and W. N. Gonçalves. Modelos ocultos de markov aplicados na identificação de comportamento de animais. Monografia (Graduação), 2007. Universidade Católica Dom Bosco, Campo Grande, MS.

[46] J. A. Silva, W. N. Gonçalves, B. B. Machado, H. Pistori, and A. S. Souza. Comparação de descritores de formas no reconhecimento de objetos. In *III Workshop de Visão Computacional*, UNESP - São José do Rio Preto - SP, 2007.

[47] K. Siriboon, A. Jirayusakul, and B. Kruatrachue. Hmm topology selection for on-line thai handwritten recognition. In *CW '02: Proceedings of the First International Symposium on Cyber Worlds (CW'02)*, page 0142, Washington, DC, USA, 2002. IEEE Computer Society.

[48] W. Skarbek and A. Koschan. Colour image segmentation - a survey. Technical report, Technical University of Berlin, 1994.

[49] K. P. Souza and H. Pistori. Implementacão de um extrator de características baseado em momentos da imagem. *SIBGRAPI '05: Proceedings of the 18th Brazilian Symposium on Computer Graphics and Image Processing*, 2005.

[50] S. Theodoridis and K. Koutroumbas. *Pattern recognition.* Academic press, 1999.

[51] M. A. Q. Truyenque. Uma aplicação de visão computacional que utiliza gestos da mão para interagir com o computador. Dissertação de Mestrado, 2005. Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, RJ.

[52] Q. Wang and S. Ju. A mixed classifier based on combination of hmm and knn. In *ICNC '08: Proceedings of the 2008 Fourth International Conference on Natural Computation*, pages 38–42, Washington, DC, USA, 2008. IEEE Computer Society.

[53] Q. Wu, F. Merchant, and K. Castleman. *Microscope Image Processing.* Academic Press, 2008.

[54] F. Ye. Automatic audio classification using hmm combined with svm. *Performance Computing Technology*, 2005.

[55] J. Ye, H. Yao, and F. Jiang. Based on hmm and svm multilayer architecture classifier for chinese sign language recognition with large vocabulary. In *ICIG '04: Proceedings of the Third International Conference on Image and Graphics*, pages 377–380. IEEE Computer Society, 2004.

[56] S. J. Young, G. Evermann, M. J. F. Gales, T. Hain, D. Kershaw, G. Moore, J. Odell, D. Ollason, D. Povey, V. Valtchev, and P. C. Woodland. *The HTK Book, version 3.4.* Cambridge University Engineering Department, Cambridge, UK, 2006.

[57] L.-G. Zhang, X. Chen, C. Wang, Y. Chen, and W. Gao. Recognition of sign language subwords based on boosted hidden markov models. In *ICMI '05: Proceedings of the 7th international conference on Multimodal interfaces*, pages 282–287, New York, NY, USA, 2005. ACM Press.

[58] M. Zhao, C. Chen, and Z. Wu. Robust background subtraction in hsv color space. In *Multimedia Systems and Applications*, pages 325–332. SPIE, 2002.

[59] M. Zimmermann and H. Bunke. Hidden markov model length optimization for handwriting recognition systems. In *IWFHR '02: Proceedings of the Eighth International Workshop on Frontiers in Handwriting Recognition (IWFHR'02)*, page 369, Washington, DC, USA, 2002. IEEE Computer Society.

[60] W. Zucchini, D. Raubenheimer, and I. L. MacDonald. Modelling animal motivation by means of a hidden markov model. In *55th Session of the International Statistical Institute*, 2005.