

MURILO SANTOS

**COMBINANDO CARACTERÍSTICAS
COMPLEMENTARES EM MODELOS
ESCONDIDOS DE MARKOV: UMA OTIMIZAÇÃO
PARA O RECONHECIMENTO DE CARACTERES
MANUSCRITOS**

CURITIBA

2009

MURILO SANTOS

**COMBINANDO CARACTERÍSTICAS
COMPLEMENTARES EM MODELOS
ESCONDIDOS DE MARKOV: UMA OTIMIZAÇÃO
PARA O RECONHECIMENTO DE CARACTERES
MANUSCRITOS**

Dissertação apresentada ao Programa de Pós-Graduação em Informática da Pontifícia Universidade Católica do Paraná como requisito parcial para obtenção do título de Mestre em Informática.

Área de Concentração: Ciência da Computação

Orientador: Prof. Dr. Alceu de Souza Britto Jr.

Co-orientador: Prof. Dr. Luiz Eduardo S. de Oliveira

CURITIBA

2009

Agradecimentos

Agradeço aos meus colegas de mestrado que me ajudaram nos estudos e foram parceiros em muitos desafiadores trabalhos escolares. Especial agradecimento a Aldelir Fernando Luiz e Ruth Angelina Martins.

À minha família pelo incondicional apoio e incentivo aos estudos, e que me deu toda a base de valores para concluir mais esta etapa de minha vida. Agradeço ao meu pai Marcolino João dos Santos Filho, a minha mãe Marli Terezinha dos Santos, meu irmão Marlon Santos e minha namora Vivian Oliveira de Mira.

Gostaria também de agradecer a toda a direção e professores do PPGIA pelo ótimo trabalho que está sendo desenvolvido e que pude desfrutá-lo por alguns anos. E principalmente ao meu amigo e orientador Dr. Alceu de Souza Britto Jr. que soube com seu vasto conhecimento e grande paciência me conduzir de tal forma a poder chegar aqui com a certeza que a missão foi cumprida.

Sumário

Agradecimentos	i
Sumário.....	i
Lista de Figuras	iii
Lista de Tabelas	v
Lista de Abreviaturas.....	vi
Resumo	vii
Abstract.....	viii
Introdução	1
1.1 Motivação	1
1.2 Definição do Problema	2
1.3 Objetivos	2
1.4 Método Proposto	3
1.5 Estrutura do Trabalho	4
Estado da Arte	5
2.1 Otimização de MEM	5
2.2 Reconhecimento de Manuscritos.....	10
2.3 Outras Contribuições	12
2.4 Considerações Finais	17
Fundamentação Teórica.....	18
3.1 Modelos Escondidos de Markov	18
3.2 Algoritmo de Baum-Welch	20
3.3 Algoritmo de Viterbi	22
3.4 Considerações Finais	23

Método Proposto	24
4.1 Visão Geral do Método	24
4.1.1 Métodos de Extração de Características	25
4.1.2 Modelos Escondidos de Markov	29
4.2 Estratégias de Otimização	30
4.2.1 Múltiplas matrizes B.....	30
4.2.2 Múltiplos Modelos	33
4.2.3 Aumento da Base de Treinamento Utilizando DUF-sd	37
4.2.4 Redução dos Ruídos na Base de Treinamento.....	39
4.2.4.1 Redução dos Ruídos na Base de Treinamento sem DUF-sd.....	39
4.2.4.2 Redução dos Ruídos na Base de Treinamento com DUF-sd	40
4.3 Considerações Finais	41
Experimentos	42
5.1 Base de Dados	42
5.2 Experimentos Considerando Múltiplas Matrizes B	43
5.3 Experimentos Considerando Múltiplos MEMs	43
5.4 Experimentos Aumentando a Base de Treinamento utilizando DUF-sd	44
5.5 Experimentos Considerando Redutor de Ruídos	45
Conclusão	47
Referências Bibliográficas.....	50

Lista de Figuras

Figura 2.1	Imagens de caracteres transformados: (a) imagem de tamanho normalizado, (b) imagem de transformação polar, (c) imagem rotacionada 90° e (d) Imagem composta. Figura adaptada de [Nopsuwanchai, 2003].....	7
Figura 2.2	Diferença entre (a) colunas verticais de pixels onde foi aplicado ACP, e (b) sub blocos de colunas verticais onde bloco-base ACP foi aplicado. Figura adaptada de [Nopsuwanchai, 2003].....	8
Figura 2.3	Exemplo de extração de características da letra “e” (a)horizontal, (b)vertical, (c)esquerda diagonal, (d) direita diagonal [Arica, 2002]. Figura adaptada de [Arica, 2002]	14
Figura 4.1	Esquema implícito de zoneamento obtido pela combinação de MEMs de coluna e linha. Figura adaptada de [Britto, 2004].....	25
Figura 4.2	Média de direção circular $\bar{\alpha}$ e variância s_{α} para distribuição $F(\alpha_i)$. Figura adaptada de [Britto, 2004].....	26
Figura 4.3	Transições em uma coluna do caractere “e”, e as observações direcionais para estimar a direção média das transições 3 e 5.....	27
Figura 4.4	Configurações de Concavidade. Figura adaptada de [Britto, 2004].	29
Figura 4.5	Visão geral da estratégia de Múltiplos Modelos	36
Figura 4.6	Sequência de observações geradas pelo processo DUF-sd a partir da sequência original.	37
Figura 4.7	Sequência de observação original e duas novas sequências geradas a partir da original	38
Figura 4.8	Sequência de observações com as frequências de aparecimento na base de treinamento.....	39

Figura 4.9	Sequência de observação original e mais duas novas sequências criadas utilizando o RR.....	39
Figura 4.10	Uma das sequências de observações da base de treinamento do caractere <i>a</i> minúsculo contendo 42 símbolos.	40

Lista de Tabelas

Tabela 2.1	Taxas de reconhecimentos de palavras em (%). [Arica, 2002].....	14
Tabela 2.2	Resultados da combinação dos três classificadores. A taxa de reconhecimento do classificador base foi de 80.48% para C ₁ , 71.57 para C ₂ e 78.65 para C ₃ . [Günter, 2004].....	15
Tabela 4.1	Matriz B que representa uma probabilidade meteorológica.....	31
Tabela 4.2	Umidade relativa do ar para cada observação da matriz B	31
Tabela 5.1	Taxas de reconhecimento final em (%).	44
Tabela 5.2	Comparação com Tabalhos Relacionados.....	44
Tabela 5.3	Taxas de reconhecimentos para classes maiúsculas e minúsculas considerando diferentes limiares DUF-sd	46

Lista de Abreviaturas

ACP	Análise do Componente Principal
CF	Características de Fundo
CS	Classificação e Segmentação
CT	Características de traço
DUF	Deixe Um de Fora
DUF-sd	Deixe Um de Fora - sem duplicação
DUFT-r	Deixe Um de Fora Treinamento
DUFT-s	Deixe Um de Fora Teste
ECB	Estágio de Classificação Base
ECT	Extração de Características de Traço
ECTF	Extração de Características de Traço e de Fundo
HMM	Hidden Markov Model
LOB	Lancaster-Oslo/Bergen
MEM	Modelos Escondidos de Markov
MIM	Máxima Informação Mútua
MLP	Multilayer Perceptron
MSV	Máquina vetor de Suporte
PV	Projeções Vertical
RR	Redução de Ruídos
SM	Semelhança Máxima

Resumo

Este trabalho avalia diferentes estratégias para otimização de um sistema de reconhecimento de caracteres manuscritos baseado em Modelos Escondidos de Markov (MEM). Os algoritmos de Baum-Welch e Viterbi são utilizados para treinamento e classificação dos modelos, respectivamente. Estratégias como a utilização de múltiplas matrizes B e múltiplos modelos foram desenvolvidas e avaliadas. Além disso, foi avaliado o aumento da base de treinamento utilizando a técnica DUF-sd (Deixe Um de Fora – sem duplicações), na qual novas sequências de observações são geradas a partir das sequências originais para aumentar a base de treinamento e torná-la mais consistente. Por último, uma estratégia para a redução de ruídos foi utilizada para minimizar o impacto negativo de símbolos com baixa probabilidade na base de treinamento. Os resultados experimentais baseados em 52 classes de caracteres alfabéticos contendo mais de 23.000 amostras, mostrou que as estratégias propostas neste trabalho para otimizar o método de reconhecimento baseado em MEM são promissoras.

Palavras-Chave: (Otimização de MEM, Reconhecimento de Caracteres Manuscritos, Modelos Escondidos de Markov).

Abstract

This work evaluates different strategies for optimization of a handwriting character recognition system based on HMMs (Hidden Markov Models). The Baum Welch's and Viterbi's algorithms are used for model training and testing. Strategies such as the use of multiples B matrices and multiples HMMs were developed and evaluated. Moreover, it was evaluated the increasing of the training database by using the LOO (Leave One Out) strategy, in which new observation sequences are generated from the original ones in order to make the database more consistent. Finally, a scheme for noise reduction was used to decrease the negative impact of low-probability symbols in the training database. The experimental results on the NIST SD19, considering 52 character classes and 23,000 samples, have shown that the strategies presented in this work to optimize the recognition method based on HMMS are promising.

Keywords: (HMM optimization, handwritten character recognition, Hidden Markov Model)

Capítulo 1

Introdução

As pesquisas na linha de reconhecimento de manuscritos têm se tornado cada vez mais importantes para a sociedade em virtude de contribuírem para o aprimoramento de *softwares* que automatizam processos de digitação de documentos, os quais demandam muito tempo e dinheiro.

Muitos sistemas já foram propostos para reconhecimentos de dígitos, caracteres e textos manuscritos baseados em MEMs, porém poucos esforços têm sido direcionados para a otimização destes sistemas.

1.1 Motivação

O uso da modelagem Markoviana merece destaque, pois esta tem mostrado ser uma alternativa interessante uma vez que modelos para caracteres isolados podem ser combinados dinamicamente durante o reconhecimento de cadeias numéricas ou palavras cursivas, como mostrado com sucesso em [Britto, 2004] e [Cavalin, 2006]. Além disso, é possível adicionar aos modelos conhecimento referente à interação entre os caracteres os quais variam na forma de acordo com o contexto, ou seja, a sua posição na palavra. Contudo, o processo de modelagem baseado em modelos de Markov é complexo e diferentes fatores devem ser levados em consideração, tais como: a topologia dos modelos – número de estados e observações por estado; a maneira como se representar e estruturar as características utilizadas, dentre outras. O método proposto em [Britto, 2004] é um exemplo típico deste tipo de abordagem, e tem sido utilizado com êxito no reconhecimento de caracteres isolados,

cadeias numéricas e palavras cursivas. Contudo, ainda restam pontos a serem investigados na busca da otimização dos modelos Markovianos utilizados.

1.2 Definição do Problema

Conforme já destacado, a modelagem de um sistema de reconhecimento de manuscritos baseado em modelos discretos de Markov exige o estudo e avaliação de diferentes aspectos relacionados à topologia dos modelos, representação e combinação de características, bem como definição de uma base de treinamento representativa do problema em questão. Neste trabalho tomamos como base o sistema proposto em [Britto, 2001], o qual utiliza um conjunto de modelos de Markov para o reconhecimento de caracteres manuscritos. Neste sistema, características complementares extraídas do traço e do fundo das imagens de caracteres são combinadas em um vetor de 47 características, sendo 34 relacionadas ao traço e mais 13 relacionadas ao fundo. Estas características são extraídas separadamente para as colunas e linhas da imagem, assim, o método funciona com dois MEMs para cada classe de caractere. Tais modelos discretos foram originalmente otimizados em termos de topologia, número de estados e símbolos por estado, contudo, há ainda aspectos não investigados tais como: a) qual a melhor forma de combinar características nos modelos; e b) como reduzir o impacto de ruídos na base de treinamento. Desta forma, este será o foco deste trabalho.

1.3 Objetivos

Esta dissertação tem como objetivo avaliar diferentes estratégias para a otimização de sistemas de reconhecimento de manuscritos que utilizam a modelagem Markoviana. Para tal, considera-se a investigação de diferentes abordagens para a combinação das características complementares utilizadas no método como a utilização de vários MEMs e várias matrizes B, bem como a avaliação de uma estratégia para a redução do impacto de ruídos supostamente presentes nas sequências de observações discretas utilizadas no treinamento dos mesmos. Investiga-se a geração de novas sequências de observações com base em filtragem para detecção de observações cuja ocorrência está abaixo de um limiar experimentalmente estabelecido.

Durante as pesquisas, o método proposto em [Britto, 2004] para reconhecimento de caracteres manuscritos em 52 classes (letras maiúsculas e minúsculas) foi utilizado como referencia.

1.4 Método Proposto

O método proposto busca avaliar diferentes estratégias para otimização de um sistema baseado em MEMs aplicado ao reconhecimento de caracteres manuscritos.

A primeira estratégia proposta consiste em combinar as características complementares (traço e fundo) em diferentes matrizes B (matriz que fornece a probabilidade das observações ou símbolos por estado do MEM) para cada MEM, sendo que cada matriz B poderá computar um tipo diferente de característica, tais como: características de traço e características de fundo. Para que os MEMs possam interagir com várias matrizes B será aplicada a proposta apresentada em [Elms, 1996] no qual as probabilidades fornecidas pelas matrizes B, consideradas independentes, são multiplicadas. No algoritmo de Baum-Welch utilizado para treinamento dos MEMs e no algoritmo de Viterbi utilizado para testes dos MEMs, as matrizes B são substituídas pela equação resultante da multiplicação das matrizes B na qual cada matriz B representa um tipo de característica a ser modelada.

A segunda estratégia proposta consiste em criar vários MEMs, um para cada tipo de característica dentre elas: características de traço, características de fundo, e separadamente para as características de linhas e de colunas. O sistema é alterado para a extração de vários tipos de características onde para cada tipo de característica será gerado um *codebook* distinto. Os algoritmos utilizados para treinamento e teste dos modelos são alterados para interagir com os vários tipos de características extraídas, porém interagindo com cada uma delas separadamente.

Além de buscar diferentes formas de combinar as características utilizadas no reconhecimento, investiga-se o impacto do aumento da base de treinamento através da manipulação das sequências de observação. A idéia básica consiste em criar novas sequências de observações com base nas sequências originais. Novas sequências são criadas deixando uma observação da sequência original de fora, técnica denominada de DUF-sd (Deixe Um de Fora – sem duplicações). Finalmente, investiga-se a redução de ruídos na base de treinamento. Esta implementação foi realizada de forma única e também utilizando a técnica DUF-sd como auxílio. O funcionamento consiste em excluir observações que têm baixa representatividade na base de treinamento de acordo com um limiar estabelecido. Estas observações por terem uma baixa representatividade são consideradas possíveis ruído presentes na base de treinamento. Esta técnica utilizada juntamente com a técnica DUF-sd ao invés de excluir

observações cria novas sequências, porém as observações que são possíveis candidatas a ruído são deixadas de fora durante a criação de novas sequências.

1.5 Estrutura do Trabalho

O Capítulo 2 traz uma revisão bibliográfica de trabalhos relacionados à otimização de MEMs para reconhecimento de manuscritos e de alguns trabalhos relacionados ao reconhecimento de caracteres manuscritos. No Capítulo 3 é apresentada a fundamentação teórica, com uma apresentação dos MEMs e os algoritmos de Baum-Welch e de Viterbi utilizados neste trabalho. Os métodos propostos são apresentados no Capítulo 4. No Capítulo 5 são apresentados os experimentos referentes aos métodos propostos, enquanto no Capítulo 6 apresenta-se a conclusão contendo contribuições e propostas de trabalhos futuros.

Capítulo 2

Estado da Arte

Este capítulo apresenta alguns trabalhos que tratam da otimização de MEMs para reconhecimento de manuscritos e outros trabalhos relacionados ao próprio reconhecimento de manuscritos. As maiores contribuições nesta linha são: Otimização de MEMs, extração de características, conjuntos de classificadores, reconhecimento de dígitos, caracteres e palavras manuscritas.

2.1 Otimização de MEM

Em [Günter, 2003] foram avaliados três métodos para otimização de um sistema de reconhecimento de palavras manuscritas baseados em MEM, aplicados nas seguintes definições: número de estados do modelo; número de interações de treinamento; e número de misturas Gaussianas para cada estado. A base de dados utilizada para os testes foi a *Full English Sentence Database* utilizando 3850 palavras que foram divididas entre treinamento e validação.

- Para o número de estados, duas técnicas foram avaliadas.

Número constante: O número de estados para cada MEM é definido pela constante s . O melhor valor de s é determinado usando o conjunto de validação. A performance deste sistema atingiu uma taxa de reconhecimento de 66,23% para o qual o número de estados em cada MEM foi definido como 14.

Número flexível: A constante f representa o número de estados para cada MEM, definida como a média do tamanho do vetor de sequência de observações. Em geral o tamanho médio

é igual a média da largura do caractere. Este sistema atingiu a taxa de reconhecimento de 70,92%.

- Para o número de interações para treinamento e o número de Gaussianas foram avaliadas três estratégias:

Estratégia 1: Um algoritmo é executado e a cada interação é incrementada uma gaussiana, um número fixo de interações C de treinamento é executado. Quando for atingido o número desejado de gaussianas, o classificador é testado no conjunto de validação imediatamente após cada ciclo de treinamento para achar o melhor número de interações de treinamento para o sistema final.

Estratégia 2: A segunda estratégia é idêntica a primeira quando $C=0$. Isto significa que o número de gaussianas é incrementado até o número desejado sem nenhuma interação de treinamento após o incremento.

Estratégia 3: Nesta estratégia a procura pelo melhor número de interações de treinamento é realizada após cada incremento de gaussiana, o que significa que o sistema é otimizado imediatamente após cada incremento de gaussiana.

A terceira estratégia produziu um resultado melhor que a segunda, porém a estratégia que alcançou o melhor resultado foi a primeira que atingiu uma taxa de reconhecimento de 82.13%.

Em [Nopsuwanchai, 2003] foram apresentadas três técnicas para otimização de um sistema de reconhecimento de caracteres manuscritos isolados Tailandeses baseado em MEM, sendo elas: MIM (Máxima Informação Mútua), imagens compostas, e bloco base ACP (Análise do Componente Principal); as quais são descritas por:

- O critério MIM é a probabilidade posterior da correta transcrição. Sendo:

$$F_{MMI}(\lambda) = \sum_{r=1}^R \log \frac{p_{\lambda}(O_r | S_r)^K P(S_r)^K}{\sum_s p_{\lambda}(O_r | s)^K P(s)^K} \quad (2.2)$$

Em que λ é o parâmetro do MEM, O_r são os dados observados do arquivo de treinamento r' e S_r é a correta transcrição do arquivo r' (neste caso apenas um único caractere).

A soma \sum_s no denominador é calculada sobre todas as possibilidades de sentenças (caracteres neste caso) ou, na prática, somente as mais parecidas. A escala K , o qual será

tipicamente menor que um (ex.: na faixa de $\frac{1}{10}$ a $\frac{1}{20}$), é uma escala no *log* de semelhança que permite mais sentenças competirem com a sentença correta e melhorar generalizações no conjunto de testes. Experimentos em grandes vocabulários de reconhecimento de voz [Woodland, 2002] mostraram que escalas de probabilidades são essenciais para um bom conjunto de testes de desempenho. Treinamento discriminativo é mais sensível ao montante de dados de treinamento que em treinamento de SM (Semelhança Máxima), como é claro nos resultados apresentados em [Woodland, 2002] e neste trabalho. Utilizando 180 amostras de caracteres para treinamento a taxa de reconhecimento de SM foi de 88,54% contra 91,38% de MIM.

- Na técnica de imagem composta a imagem é concatenada com uma rotação de 90°, e é gerada uma versão de si própria aplicando a transformação polar. A transformação polar, similar à transformação *log-polar* [Tistarelli, 1993] largamente usada em pesquisas de visão computacional, consiste no mapeamento de pontos da imagem $f(x, y)$ a pontos na imagem polar $g(r, \theta)$. Esta definição foi adaptada à ‘origem’ $O = (o_x, o_y)$ dada pela centróide $(o_x = \bar{x}, o_y = \bar{y})$ da imagem. A imagem original e as duas imagens transformadas provêm similares resultados de reconhecimento individualmente, mas quando concatenadas em uma imagem composta como na (figura 2.1) elas provem substancial melhoramento.

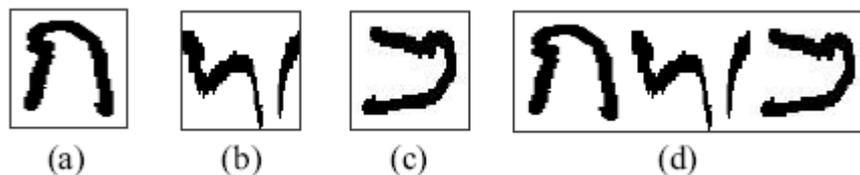


Figura 2.1: Imagens de caracteres transformados: (a) imagem de tamanho normalizado, (b) imagem de transformação polar, (c) imagem rotacionada 90° e (d) Imagem composta. Figura adaptada de [Nopsuwanchai, 2003].

Em um dos experimentos a taxa de reconhecimento para imagens inteiras foi de 85,47% e para imagens compostas de 89,36%.

- A simples aplicação do ACP é projetar o primeiro componente principal d do vetor de pixel H -dimensional em um vetor d -dimensional, o qual é referido a uma técnica ACP padrão. No bloco-base ACP é iniciado um bloco de imagem, ao qual irá tipicamente ser da altura $w \times$

H de blocos de pixels com a pequena largura w (ex.: [1,4]). Este é dividido verticalmente em sobreposições de sub-blocos de altura h (ex.: 16), com sucessivos sub-blocos movendo-se por um vertical *offset* O_v (ex.: 8) até eles cobrirem toda a altura vertical (figura 2.2b). O ACP é então aplicado a cada $w \times h$ sub-bloco para reduzir sua dimensão a uma pequena dimensão d' . O vetor de tamanho d' dos sub-blocos são concatenados para formar um vetor de características d . Na implementação, cada sub-bloco tem sua própria matriz para extrair o componente principal, calculado da covariância do sub-bloco no treinamento.

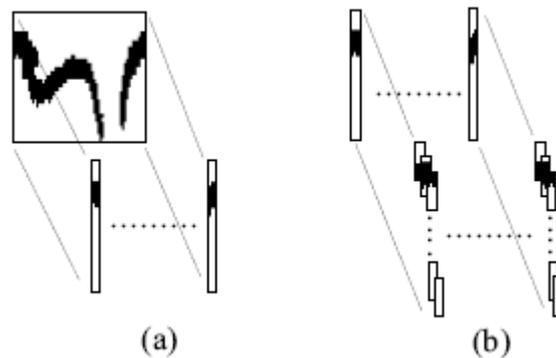


Figura 2.2: Diferença entre (a) colunas de pixels onde foi aplicado ACP, e (b) sub-blocos de colunas onde bloco-base ACP foi aplicado. Figura adaptada de [Nopsuwanchai, 2003].

Com uma largura de bloco (w) 4 e um vetor de característica com dimensão (d) 42 o resultado do ACP padrão foi de 87,77% para imagens compostas, e para o bloco-base ACP com $h = 16$ e $O_v = 8$ o resultado foi de 88,23%.

A base de dados para os experimentos não é pública sendo composta por 120 amostras de cada caractere para cada um de 20 escritores, dividida igualmente entre base de treinamento e testes.

Em [Ko, 2008] é abordado um método recente para otimização de um sistema para reconhecimento de dígitos manuscritos. O método utiliza as técnicas DUFT-r (Treinamento Deixe Um de Fora) e DUFT-s (Teste Deixe Um de Fora). No método quando existem inesperados ruídos em uma sequência de observações, a estrutura matemática fica impossibilitada de achar o relacionamento entre cada observação corrente e a próxima observação, pois a sequência fica irreconhecível para modelos de treinamento. Este fenômeno é chamado de ruído de sequência de observação, o DUFT-MEM tem como objetivo diminuir estes ruídos aumentando assim a taxa de reconhecimento. O DUFT-MEM é composto por três

componentes: otimização do codebook, DUFT-r e DUFT-s. Estas técnicas são descritas a seguir.

- a) O primeiro passo do DUFT-MEM é a otimização do codebook. Geralmente um codebook MEM é gerado a partir de um procedimento de quantização de vetores, e cada palavra código pode ser realmente considerada como uma centróide de um cluster no espaço de características. O índice XB é projetado para mensurar o desempenho da clusterização *fuzzy* [Bezdek, 1984], mas isto pode também ser adequado à clusterização *crisp* [Pakhira, 2004]. O índice XB é matematicamente justificado em [Xie, 1991]. Quanto menor o valor do índice XB , melhor a clusterização deve ser. Podemos usar o índice XB para rastrear os dados e achar o número de clusters com o menor índice XB . Este número é então escolhido para ser o tamanho do codebook, e as palavras códigos são geradas usando uma simples clusterização K-Means. Após a criação dos codebooks, é aplicado o processo DUF para o treinamento MEM.
- b) A técnica DUF para o treinamento MEM é um simples processo que gera sequências de observações a partir de uma única sequência. Supondo que para uma sequência S_i , temos as observações $O_{1_i}, O_{2_i}, \dots, O_{M_i}$, em que M_i é o número de observações da sequência S_i . A partir da sequência de observações original, obtemos então novas sequências simplesmente tirando uma observação por vez. Dado um conjunto de dados $S = \{S_i, 1 \leq i \leq N\}$, onde N é o número de sequências, e cada sequência tem M_i observações, cada sequência original pode assim gerar novas sequências M_i , e para todas as sequências N , temos o número total de sequências como:

$$\sum_{1 \leq i}^N M_i \quad (2.3)$$

Para o treinamento MEM, todas as novas sequências e as sequências originais são usadas, assim se obtém um total de sequências para treinamento de $\left(\sum_{1 \leq i}^N M_i \right) + N$.

O processo DUFT-r nos dá mais amostras para treinamento. Isto pode reduzir o efeito de ruídos nas sequências de observações. Quando os ruídos são reduzidos nas sequências, o MEM tem mais amostras corretas para treinamento.

- c) O teste DUF tem um processo similar ao treinamento DUF. A única diferença é que o processo DUF (Deixe Um de Fora) é aplicado nas sequências de teste ao invés das

sequências de treinamento. Supondo que exista uma sequência de teste $S'i$, temos então as observações $O_{1_i}, O_{2_i}, \dots, O_{M_i}$, onde M_i é o número de observações para a sequência $S'i$. Novamente, geramos novas sequências de observações a partir da sequência original simplesmente tirando uma observação a cada tempo. Após as novas sequências serem geradas, todas as novas sequências M_i e a sequência original de teste são submetidas ao teste. Suas semelhanças são combinadas e a classe é rotulada com a mais alta semelhança selecionada.

Para os experimentos foram utilizadas 10 classes de numerais da base *NIST SD19*. Para treinamento 150000 amostras da série hsf_0 até hsf_3 foram utilizadas e para testes 60089 amostras da série hsf_7. A taxa de reconhecimento atingiu 99,72%.

2.2 Reconhecimento de Manuscritos

Em [Britto, 2001] é apresentado um método para reconhecimento de cadeias de dígitos manuscritos composto por dois estágios baseado em MEM. O primeiro estágio é composto de 3 módulos: pré-processamento, Extração de Características de Traço (ECT) e Classificação/Segmentação (CS). Primeiramente o pré-processamento faz a correção da inclinação, suaviza o contorno e calcula o *bounding box* (espaço onde o objeto se encontra) da string, o módulo ECT varre a palavra da esquerda para a direita enquanto um vetor baseado em informações do traço é calculado para cada coluna da palavra. Este vetor é mapeado como um símbolo discreto disponível em um codebook previamente construído. No módulo CS modelos de letras treinados em caracteres isolados considerando as informações contextuais, são combinadas com a sequência provinda do módulo ECT.

O segundo estágio consiste na verificação das hipóteses geradas no primeiro estágio, e é composto de dois módulos: Extração de Características de Traço e de Fundo (ECTF) e Verificação, que consiste em um classificador MEM treinado com caracteres isolados sem levar em conta nenhuma informação contextual da palavra. Um novo conjunto de características combinando informações de traço e fundo para melhorar o desempenho do reconhecimento dos caracteres é gerado. A probabilidade do conjunto de caracteres calculada no módulo de verificação é adicionada à probabilidade de hipóteses obtidas em segmentação e

reconhecimento do estágio ECB. O resultado probabilístico final é usado para reclassificar a pontuação do conjunto de caracteres das hipóteses.

Os experimentos foram realizados utilizando a base de dados NIST SD19, A taxa de reconhecimento de dígitos isolados foi de 98,02%. Foram selecionados 50.000 exemplos para treinamento, 10.000 para validação e 10.000 para teste.

Em [Koerich, 2002] a proposta para classificação dos caracteres baseada em redes neurais, do tipo *Multilayer Perceptron* (MLP). Entre várias redes neurais, a MLP é uma das mais usadas, especialmente para o problema de classificação de padrões. Para construir um classificador MLP é necessário determinar o número de camadas e o número de neurônios em cada camada. Na teoria, para o problema de classificação de uma m -classe no qual a união de m classes de formas distintas em todo o espaço de entrada, são necessárias um total de m saídas para representar todas as possibilidades de decisão de classificação. Diferentes estratégias podem ser utilizadas para o reconhecimento de caracteres manuscritos isolados. As mais utilizadas são:

- 52-classes: letras maiúsculas e minúsculas representando um único caractere são consideradas classes diferentes (ex.: “A” e “a” são duas classes distintas);
- 26-classes: letras maiúsculas e minúsculas representando um único caractere são misturadas em uma única classe chamada de *metaclass* (ex.: “A” e “a” formam a *metaclass* “Aa”).

De acordo com os experimentos realizados com a base de dados NIST se observa que é preferível misturar letras maiúsculas e minúsculas em uma única classe (*metaclass*). Os experimentos mostraram que a combinação de letras maiúsculas e minúsculas chegando ao total de 52 classes obteve um percentual de reconhecimento na fase de teste de 85,51% enquanto a mistura das classes de letras maiúsculas e minúsculas totalizando 26 classes obteve um percentual de reconhecimento na fase de teste de 88,10%. Para os testes foi utilizada a base NIST SD19 considerando um conjunto de 178.792 amostras de letras maiúsculas e 208.066 amostras de letras minúsculas.

Em [Bortolozzi, 2005], é apresentado um resumo sobre os recentes avanços no reconhecimento de caracteres e dígitos manuscritos, que são considerados uma subclasse de caracteres. Pesquisas nesta linha tem basicamente considerado: a) métodos de extração de características, b) métodos de classificação, e c) sistemas baseados em diferentes estratégias,

tais como a combinação de múltiplos classificadores, o uso de múltiplas *templates* e o uso de módulos de verificação.

Em geral, os métodos de extração de características para reconhecimento de dígitos descritos na literatura são baseados em dois tipos de características: estatística e estrutural. As características estatísticas são derivadas das estatísticas de distribuição de pontos, tais como zoneamento, momentos, projeções de histograma ou direções de histograma [Kimura, 1992] [Gader, 1996] [Cheung, 1998]. Características estruturais são baseadas em topologias e propriedades geométricas do caractere, como traço e suas direções, pontos de fim, ou interseções do segmento e *loops* [Hirano, 1997] [Cai, 1998].

Nos últimos anos, contribuições significantes para aumentar as taxas de reconhecimentos foram alcançadas com diferentes combinações de classificadores [Suen, 1999] [Liu, 2004], e o uso de MSV [Teow, 2002] [Milgram, 2004].

Os pesquisadores de novas estratégias para reconhecimento de dígitos manuscritos têm levado as pesquisas a desenvolver modos alternativos para tratar a variedade restrita de numerais manuscritos. Neste contexto, três exemplos são interessantes: 1) o conjunto de 23 protótipos e o modelo deformável base de reconhecimento em *Bayesian framework* proposta em [Cheung, 1998]; 2) a regra básica de classificador estrutural usada como um módulo de verificação para a *Multilayer Feedforward Locally Connected Network* apresentada em [Zhou, 1997]; e, 3) a combinação de imagens flexíveis com 10.000 *templates* proposta em [Belongie, 2002].

2.3 Outras Contribuições

Em [Freitas, 2007] é apresentada uma metodologia que define *metaclasses* para o problema de reconhecimento de caracteres manuscritos. A proposta é baseada em discordância entre caracteres, e utiliza a distância Euclidiana computada entre matrizes de confusão. É aplicado o conceito de discordância para medir a diferença entre classificadores. A idéia é usar informações da matriz de confusão para cada classificador individual e computar a distância entre estas matrizes, que representa a discordância entre os classificadores. Para definir as *metaclasses* consideram-se diversos classificadores, a idéia é achar grupos de classes que melhor podem ser representadas por um conjunto de classificadores. Comparando duas classes de caracteres podemos definir que elas podem ser representadas por um conjunto de classificadores que apresentem uma discordância média

entre eles. As classes de caracteres que são representadas pelos mesmos classificadores fazem parte de uma mesma *metaclass*. Em um primeiro estágio do sistema de reconhecimento o caractere é inserido no sistema e é definido como sendo um representante de uma das *metaclasses* pelo conjunto de classificadores que obtiver a maior pontuação. Em um segundo estágio o caractere é testado pelos classificadores da referida *metaclass* sendo assim escolhido como pertencente a uma das classes de caracteres que esta *metaclass* representa.

Os experimentos foram realizados utilizando a base IRONOFF, a qual é composta por 26 classes de caracteres letras maiúsculas das séries B (B27 ... B52) com 10.510 amostras. Estas foram divididas em treinamento, validação e testes utilizando respectivamente: 60%, 20% e 20% do total aplicando o conceito de *metaclass* a melhora no desempenho atingiu 5%, chegando a uma taxa de reconhecimento de 90,4%

Em [Arica, 2002] é proposto um método de extração de características por esqueletização no qual cada caractere é representado por um conjunto de códigos de tamanho fixo. A representação é baseada em varrer o caractere em várias direções e achar a média (centro dos pixels) do traço de pixels em cada linha varrida. As linhas são divididas em regiões, que são codificadas pela potência 2 (veja Figura 2.3). Cada média é rotulada pelo código da região à qual aquela pertence. A média do traço de pixels indica um ponto de esqueletização do caractere para cada particular direção. Cada linha varrida é representada pela soma dos códigos das médias na linha varrida. A sequência de observação é obtida pela concatenação do conjunto de dígitos de códigos das linhas em todas as direções. Figuras 2.3a, 2.3b, 2.3c e 2.3d, mostram a varredura de linhas horizontal, vertical, diagonal esquerda e direita, e o resultado da sequência de observação em cada direção respectivamente.

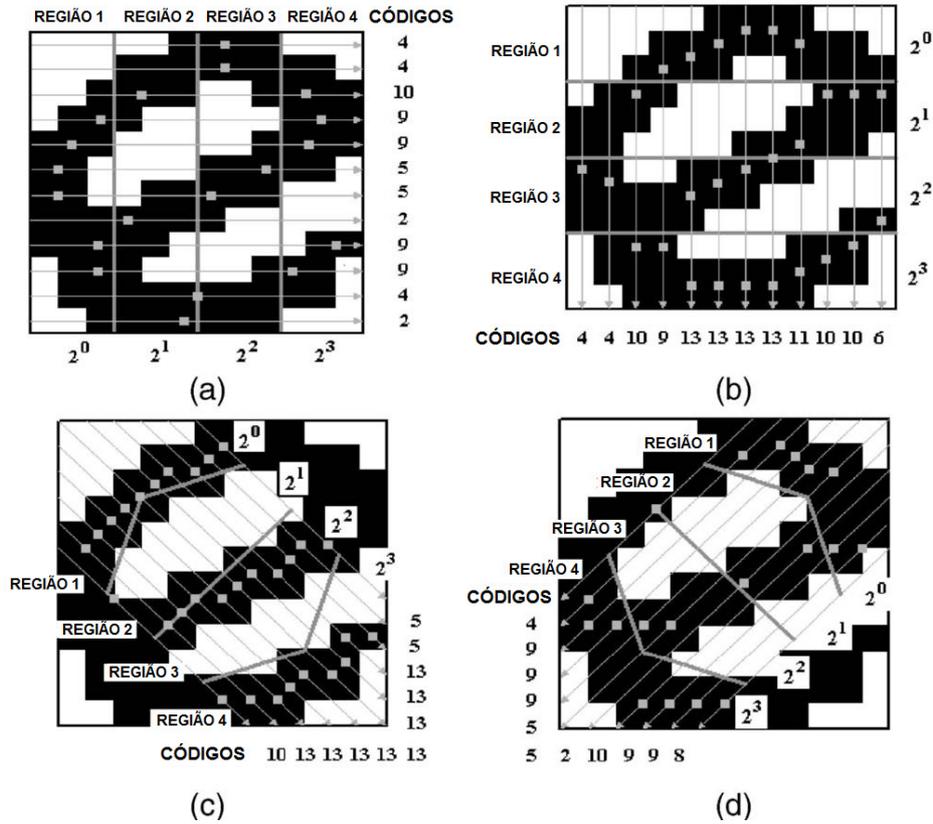


Figura 2.3: Exemplo de extração de características da letra “e” (a)horizontal, (b)vertical, (c)esquerda diagonal, (d) direita diagonal. Figura adaptada de [Arica, 2002].

O número de linhas varridas (n) nas direções diagonal é igual ao número de linhas varridas nas direções horizontais quando n é ímpar; do contrário ele é $n - 1$ para manter a simetria em relação ao eixo diagonal. Pontos indicam as médias do traço em cada direção varridas.

O método de extração de características conduz a uma sequência de observações de tamanho fixo, o qual é igual ao total dos números de linhas escaneadas em todas as direções.

Os experimentos foram realizados com a base Lancaster-Oslo/Bergen (LOB), os resultados podem ser conferidos na tabela 2.1.

Tabela 2.1: Taxas de reconhecimentos de palavras em (%), [Arica, 2002].

	Dados de Teste	Dicionário			
		50	1000	30000	40000
Base LOB	2000	92,3	90,8	89,1	88,8

Em [Günter, 2004] a proposta é um multiclassificador onde é analisada a combinação de diferentes classificadores para reconhecimento de palavras manuscritas. São utilizados três diferentes classificadores baseados em MEMs, C1, C2 e C3. No classificador C1 a largura por onde a janela percorre representa um pixel e nove características geométricas são extraídas de cada posição da janela. As características geométricas incluem as frações dos pixels pretos na janela, o centro da gravidade, e o momento de segunda ordem. No segundo classificador C2 é usada uma janela deslizante para extração de características, que é particionada em um grid de 4X4, a média do valor de cinza do pixel de cada célula é usada como vetor de característica. O terceiro classificador C3 é uma versão discretizada do classificador C1 onde todos os vetores de características são agrupados em um cluster e cada vetor de característica é rotulado com o número do cluster ao qual ele pertence. Após isto apenas o rótulo do vetor de característica é considerado.

Dois sistemas de combinação de classificadores são utilizados, os dois baseados em votação majoritária, no qual a melhor escolha de cada classificador é considerada. A classe de palavra que é mais frequente é a melhor escolha dos classificadores e é considerada a saída da combinação dos classificadores. A Tabela 2.2 mostra o resultados da combinação dos três classificadores. Os dois sistemas de combinação de votos diferem de como tratam empates

- Voting maximum score (*v_score*): Empates são decididos pela regra máxima no qual a classe de palavra que tiver maior pontuação entre todas as classes e entre todos os classificadores é escolhida.
- Voting priority (*v_priority*): Neste caso empates são decididos optando pelo classificador pré-definido que geralmente é o classificador que tem o melhor desempenho.

A base IAM foi utilizada para testes, em que 18.920 palavras foram utilizadas para treinamento, e 3.264 palavras foram utilizadas para testes. A tabela a seguir mostra os resultados obtidos para os multiclassificadores *v_score* e *v_priority*, sendo que o *v_priority* utiliza como classificador base cada um dos classificadores propostos.

Tabela 2.2: Resultados da combinação dos três classificadores. A taxa de reconhecimento do classificador base foi de 80,48% para C1, 71,57 para C2 e 78,65 para C3. [Günter, 2004].

<i>v_score</i>	<i>v_priority (C1)</i>	<i>v_priority (C2)</i>	<i>v_priority (C3)</i>
83,36%	83,46%	81,83%	82,75%

Em [Gatos, 2006] foram empregados dois tipos de extração de características de uma forma híbrida. A primeira divide a imagem em zonas verticais e horizontais, em cada zona é calculada a densidade dos pixels dos caracteres da palavra. A segunda é baseada em calcular a área que é formada através da projeção de perfil *upper/lower* da palavra. A imagem da palavra é dividida em duas seções separadas por uma linha horizontal que passa através do centro da massa da imagem. Perfis de palavras *upper/lower* são computados considerando cada coluna, a distância entre a linha horizontal e o mais próximo pixel de caractere do limite *upper/lower* da palavra. O vetor híbrido de características é fornecido pela equação 2.1. O tamanho correspondente do vetor de característica é igual a $Z_H Z_V + 2P_V$.

$$f(i) = \begin{cases} f^z(i) = \sum_{x=x_s(i)}^{x_e(i)} \sum_{y=y_s(i)}^{y_e(i)} im(x,y), & i = 0 \dots Z_H Z_V - 1 \\ f^{p_{up_ar}}(i) = \sum_{x=x_s(i-Z_H Z_V)}^{x_e(i-Z_H Z_V)} y_{up}(x), & i = Z_H Z_V \dots Z_H Z_V + P_V - 1 \\ f^{p_{lo_ar}}(i) = \sum_{x=x_s(i-Z_H Z_V + P_V)}^{x_e(i-Z_H Z_V + P_V)} y_{lo}(x), & i = Z_H Z_V + P_V \dots Z_H Z_V + 2P_V - 1 \end{cases} \quad (2.1)$$

Z_H e Z_V são os números totais de zonas nas direções horizontais e verticais e P_V é o número total de blocos formado em cada zona produzida (*upper/lower*).

Para os experimentos foram utilizadas 26.970 palavras da base IAM v3.0 de manuscritos, divididas em 23.171 palavras para treinamento e 3.799 palavras para testes. A taxa de reconhecimento obtida foi de 81,05%.

Em [Britto, 2004] o método de extração consiste em varrer o caractere da esquerda para a direita (características de colunas) e de baixo para cima (características de linhas). Informações da imagem e do fundo são combinadas em um vetor de 47 características: 34 de traço e 13 de fundo.

O vetor de características da imagem de traço é composto por características calculadas levando em conta os pixels de colunas e linhas. As características são baseadas em transições de pixels do fundo para a imagem e vice-versa. Para cada transição, a média da direção e a variância correspondente são obtidas pela média estatística.

O vetor de características do fundo da imagem é baseado em informações de concatividade. Estas características são usadas para destacar a topologia e as propriedades geométricas das classes dos caracteres. Cada característica de concatividade representa um número de pixels brancos que pertencem a uma específica configuração de concatividade. Cada direção é explorada até se encontrar um pixel preto ou o limite da janela. O pixel branco (fundo da imagem) é rotulado se for encontrado em duas direções consecutivas de pixel preto. Assim se obtêm 9 possíveis configurações de concatividade e consideram-se mais 4 para detectar precisamente a presença de *loops*.

Os caracteres isolados disponíveis na base NIST SD19 foram utilizados nos experimentos. As séries hsf_0, hsf_1, hsf_2 e hsf_3 em um total de 74.880 amostras foram utilizadas para o treinamento. As séries hsf_7 e hsf_4 com 23.670 e 23.941 amostras foram utilizadas para validação e teste respectivamente. Os resultados de reconhecimento foram de 90% para letras maiúsculas, 84% para letras minúsculas e 87% para letras maiúsculas e minúsculas juntas.

2.4 Considerações Finais

Neste capítulo foram apresentados alguns estudos que contribuíram para o desenvolvimento deste trabalho. As propostas de otimização aqui estudadas sugerem diversas estratégias como: alterações no número de estados do modelo, no número de interações de treinamento, no número de misturas Gaussianas para cada estado, nas escalas de probabilidade, inserção de imagens compostas e de bloco base ACP, otimização do codebook e aumento da base de treinamento e teste. Algumas propostas apresentadas nestes estudos são de suma importância como a proposta de extração de características desenvolvidas em [Britto, 2001] e o trabalho de [Ko, 2008] que trouxe base para desenvolvimento de uma das estratégias aqui apresentadas. A proposta em [Britto, 2004] que utiliza de um sistema para reconhecimento de caracteres manuscritos será a base do método proposto neste trabalho.

Capítulo 3

Fundamentação Teórica

Neste capítulo é apresentada uma breve fundamentação teórica para o método proposto. O foco do estudo neste capítulo é a modelagem estatística usando MEMs e, particularmente, dois algoritmos de MEMs que são os algoritmos Baum-Welch e o algoritmo de Viterbi. O algoritmo de Baum-Welch é utilizado para treinamento dos modelos e o algoritmo de Viterbi para classificação. Na seção 3.1 é apresentado o método estatístico dos MEMs, na seção 3.2 o algoritmo de Baum-Welch, na seção 3.3 o algoritmo de Viterbi e na seção 3.4 as considerações finais.

3.1 Modelos Escondidos de Markov

Em [Rabiner, 1989] MEM é considerado um processo estatístico duplamente embutido que contém um processo não observável (escondido), mas pode ser observado através de um outro conjunto estatístico de processos que produzem uma sequência de observações.

Um MEM é caracterizado pelos seguintes itens:

N , o número de estados no modelo.

M , o número dos distintos símbolos de observação por estado, por exemplo, o tamanho do alfabeto discreto. Os símbolos individuais são denotados como $V = \{v_1, v_2, \dots, v_M\}$.

O conjunto de probabilidades de observação do símbolo no estado $A = \{a_{ij}\}$ em que:

$$a_{ij} = P[q_{t+1} = S_j | q_t = S_i], \quad 1 \leq i, j \leq N \quad (3.1)$$

O conjunto de probabilidade de observação do símbolo no estado j , $B = \{b_j(k)\}$, em que:

$$b_j(k) = P[V_k \text{ em } t | q_t = S_j], \quad \begin{array}{l} 1 \leq j \leq N \\ 1 \leq k \leq M \end{array} \quad (3.2)$$

O conjunto de probabilidade do estado inicial $\pi = \{\pi_i\}$ em que:

$$\pi_i = P[q_1 = S_i], \quad 1 \leq i \leq N \quad (3.3)$$

Dado os apropriados valores de N , M , A , B , e π , o MEM pode ser usado como um gerador para fornecer uma sequência de observações:

$$O = O_1, O_2 \dots O_T \quad (3.4)$$

Cada observação O_t é um dos símbolos de V , e T é o número de observações na sequência, como descrita a seguir:

1. Escolha um estado inicial $q_1 = S_i$ de acordo com a probabilidade do estado inicial π .
2. Defina $t = 1$.
3. Escolha $O_t = V_k$ de acordo com a probabilidade do símbolo no estado S_i , ex.: $b_i(k)$.
4. Transite para um novo estado $q_{t+1} = S_j$ de acordo com a probabilidade do estado de transição para o estado S_i , a_{ij} .
5. Defina $t = t + 1$; retorne ao passo 3 se $t < T$; senão termine o processo.

O processo acima pode ser usado tanto como um gerador de observações ou um modelo de como uma dada sequência de observações foi gerada por um apropriado MEM.

3.2 Algoritmo de Baum-Welch

O algoritmo de Baum-Welch é um processo iterativo utilizado para maximizar a probabilidade das sequências de observações. Neste estudo este algoritmo será utilizado para treinamento dos modelos. Em [Rabiner, 1989] podemos encontrar a descrição deste procedimento, onde primeiro é definido $\xi_t(i, j)$, a probabilidade de estar no estado S_i no tempo t , e no estado S_j no tempo $t+1$, dado o modelo e a sequência de observações, ou seja:

$$\xi_t(i, j) = P(q_t = S_i, q_{t+1} = S_j | O, \lambda) \quad (3.5)$$

Através das definições das variáveis de *forward* e *backward*, pode-se escrever $\xi_t(i, j)$ na forma:

$$\xi_t(i, j) = \frac{a_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}{\sum_{i=1}^N \sum_{j=1}^N a_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)} \quad (3.6)$$

Nesta equação o termo numerador é $P(q_t = S_i, q_{t+1} = S_j, O | \lambda)$ e a divisão por $P(O | \lambda)$ dada a desejada medida da probabilidade.

Primeiramente é definido $\gamma_t(i)$ como a probabilidade de estar no estado S_i no tempo t , dada a sequência de observações e o modelo; pode-se relacionar $\gamma_t(i)$ com $\xi_t(i, j)$ pela somatória de j , onde:

$$\gamma_t(i) = \sum_{j=1}^N \xi_t(i, j) \quad (3.7)$$

Tendo as definições, pode-se denotar o número esperado de transições do estado S_i para o estado S_j como:

$$\sum_{t=1}^{T-1} \gamma_t(i) = \text{número esperado de transições de } S_i \quad (3.8)$$

$$\sum_{t=1}^{T-1} \xi_t(i, j) = \text{número esperado de transições de } S_i \text{ para } S_j \quad (3.9)$$

Usando as equações acima podemos fornecer um método de reestimação de parâmetros de um MEM, como um conjunto de equações de reestimações a seguir para π , A , e B .

A frequência esperada do estado S_i no tempo ($t = 1$)

$$\bar{\pi} = \gamma_1(i) \quad (3.10)$$

Números de transições esperadas do estado S_i para o estado S_j dividido pelo número de transições esperadas do estado S_i

$$\bar{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i, j)}{\sum_{t=1}^{T-1} \gamma_t(i)} \quad (3.11)$$

Esperado número de vezes do estado j e símbolo de observação V_k dividido pelo esperado número de vezes no estado j

$$\bar{b}_j(k) = \frac{\sum_{t=1}^T \gamma_t(j)}{\sum_{t=1}^T \gamma_t(j)} \quad (3.12)$$

3.3 Algoritmo de Viterbi

Viterbi é um dos mais usados algoritmos para encontrar o melhor caminho para uma sequência de estados. Em [Rabiner, 1989], é definido o algoritmo de Viterbi para encontrar a melhor sequência de estados, $Q = \{q_1 q_2 \dots q_T\}$, para uma dada sequência de observações $O = \{O_1 O_2 \dots O_T\}$. Para tanto é necessário definir a quantidade.

$$\delta_t(i) = \max_{q_1, q_2, \dots, q_{t-1}} P[q_1 q_2 \dots q_t = i, O_1 O_2 \dots O_t \mid \lambda] \quad (3.13)$$

Por exemplo: $\delta_t(i)$ é a melhor classificação (mais alta probabilidade) através de um caminho, no tempo t , no qual o calculo para a primeira observação t é o final do estado em S_i . Pela introdução temos

$$\delta_{t+1}(j) = [\max_i \delta_t(i) a_{ij}] \cdot b_j(O_{t+1}) \quad (3.14)$$

Para reaver a sequência de estados, precisamos manter o rastro do argumento no qual maximizado por cada t e j . Isto será feito através da matriz $\psi_t(j)$. O completo processo para achar a melhor sequência de estados é discutido a seguir:

Inicialização:

$$\delta_1(i) = \pi_i b_i(O_1), \quad 1 \leq i \leq N \quad (3.15a)$$

$$\psi_1(i) = 0 \quad (3.15b)$$

Recursão:

$$\delta_t(j) = \max_{1 \leq i \leq N} [\delta_{t-1}(i) a_{ij}] b_j(O_t), \quad 2 \leq t \leq T \quad (3.16a)$$

$$1 \leq j \leq N$$

$$\psi_t(j) = \arg \max_{1 \leq i \leq N} [\delta_{t-1}(i) a_{ij}], \quad 2 \leq t \leq T \quad (3.16b)$$

$$1 \leq j \leq N$$

Terminação:

$$P^* = \max_{1 \leq i \leq N} [\delta_T(i)] \quad (3.17a)$$

$$q_T^* = \arg \max_{1 \leq i \leq N} [\delta_T(i)] \quad (3.17b)$$

Refazendo o caminho (sequência de estados):

$$q_t^* = \psi_{t+1}(q_{t+1}^*), \quad t = T-1, T-2, \dots, 1 \quad (3.18)$$

3.4 Considerações Finais

Neste capítulo foi apresentada a fundamentação teórica que tem como principal foco os MEMs que são gerados pelos algoritmos de Baum-Welch e Viterbi para treinamento e classificação dos modelos, respectivamente. O principal objetivo deste capítulo foi obter um conhecimento abrangente dos MEMs e dos dois algoritmos utilizados para sua geração neste trabalho já que estes serão as principais etapas para o desenvolvimento do método proposto.

Capítulo 4

Método Proposto

Neste capítulo é apresentado o método proposto para o reconhecimento de caracteres manuscritos. Este método tem como objetivo o reconhecimento de caracteres utilizando métodos de otimização em um sistema baseado em MEM.

A Seção 4.1 traz uma visão geral do método apresentado em [Britto, 2004], para um melhor entendimento da otimização que é proposta na seção 4.2. As considerações finais são apresentadas na Seção 4.3.

4.1 Visão Geral do Método

A principal premissa do método de reconhecimento utilizado está na combinação de características complementares extraídas do fundo e traço da imagem de caracteres manuscritos. Um esquema de zoneamento baseado em colunas e linhas, fornece um modo de dividir o caractere em regiões, absorvendo a variação de tamanho inerente a este tipo de padrão (ver Figura 4.1). A imagem de um caractere é varrida da esquerda para direita e de cima para baixo para a extração das características de colunas e de linhas, respectivamente. No método base estas características são combinadas em um mesmo vetor de valores contínuos e dois conjuntos de modelos de Markov são utilizados, a saber: modelos de colunas e modelos de linhas. Cada conjunto contém um modelo para cada classe de caractere.

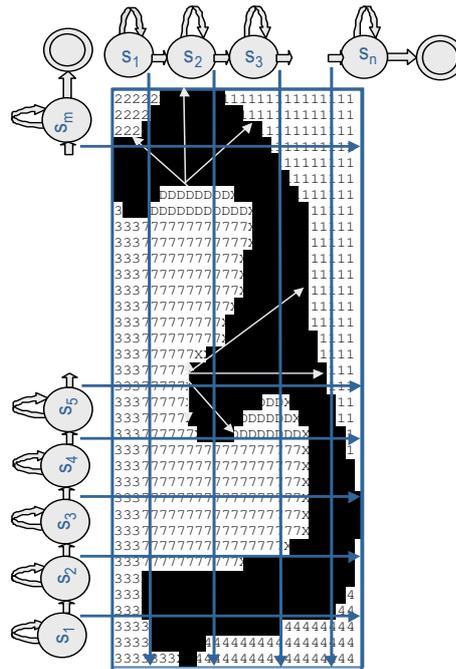


Figura 4.1: Esquema implícito de zoneamento obtido pela combinação de MEMs de coluna e linha. Figura adaptada de [Britto, 2004].

4.1.1 Métodos de Extração de Características

O método de extração de características consiste em varreduras da imagem do caractere da esquerda para a direita (características de colunas) e de cima para baixo (características de linhas). Para cada linha e coluna da imagem são calculadas características do traço e do fundo do caractere, sendo 34 características do traço (baseadas em transições e superfície) e 13 de fundo (baseadas em concavidades).

Características do Traço (CT): as características do traço consistem em informações locais e globais calculadas levando em conta os *pixels* do traço (ou em inglês, *foreground pixels*) da imagem do caractere, obtidas para cada coluna e linha. As características locais são baseadas em transições dos *pixels* de fundo para os *pixels* do traço e vice-versa. Para cada transição, a direção média e a variância correspondente são obtidas através de estimativa de média circular. Estas estimativas são mais sutis para observações direcionais, uma vez que são baseadas em escala circular. Como exemplo, dada a observação direcional $\alpha_1=1^0$ e $\alpha_2=359^0$, estas fornecem a direção média ($\bar{\alpha}$) de 0^0 ao invés de 180^0 calculado pela estimativa

convencional. Considere $\alpha_1, \dots, \alpha_i, \dots, \alpha_n$ como o conjunto de observações direcionais com distribuição $F(\alpha_i)$ e tamanho N .

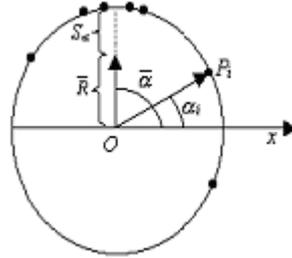


Figura 4.2: Média de direção circular $\bar{\alpha}$ e variância S_α para distribuição $F(\alpha_i)$. Figura adaptada de [Britto, 2004].

A Figura 4.2 mostra que α_i representa o ângulo entre o vetor unitário \overline{OP}_i e o eixo horizontal, enquanto P_i é o ponto de interseção entre \overline{OP}_i e a circunferência de raio unitário. As coordenadas cartesianas de P_i são definidas como:

$$(\cos(\alpha_i), \sin(\alpha_i)) \quad (4.1)$$

A direção média circular $\bar{\alpha}$ das N observações direcionais na circunferência unitária, correspondem à direção do vetor resultante \bar{R} , obtido pela soma dos vetores unitários $(\overline{OP}_1, \dots, \overline{OP}_i, \dots, \overline{OP}_N)$. O centro de gravidade (\bar{C}, \bar{S}) das N coordenadas $(\cos(\alpha_i), \sin(\alpha_i))$ definido por:

$$\bar{C} = \frac{1}{N} \sum_{i=1}^N \cos(\alpha_i) \quad \text{e} \quad \bar{S} = \frac{1}{N} \sum_{i=1}^N \sin(\alpha_i) \quad (4.2)$$

As coordenadas \bar{C} e \bar{S} são usadas para estimar o tamanho médio de \bar{R} , como:

$$\bar{R} = \sqrt{(\bar{C}^2 + \bar{S}^2)} \quad (4.3)$$

A direção circular média pode ser obtida resolvendo uma das equações a seguir:

$$\cos(\bar{\alpha}) = \frac{\bar{C}}{\bar{R}}, \quad \text{ou} \quad \sin(\bar{\alpha}) = \frac{\bar{S}}{\bar{R}} \quad (4.4)$$

Finalmente, a variância circular de $\bar{\alpha}$ é calculada como:

$$S_{\alpha} = 1 - \bar{R} \quad 0 \leq S_{\alpha} \leq 1 \quad (4.5)$$

Para estimar $\bar{\alpha}$ e S_{α} para cada transição de um caractere temos que considerar $\{0^{\circ}, 45^{\circ}, 90^{\circ}, 135^{\circ}, 180^{\circ}, 225^{\circ}, 270^{\circ}, 315^{\circ}\}$ como um conjunto de observações direcionais, enquanto $F(\alpha_i)$ é computado através da contagem do número de *pixels* pretos sucessivos na direção α_i de uma transição até aparecer um *pixel* branco. Na Figura 4.3 é possível observar as transições em uma das colunas do caractere “e”, as quais são enumeradas de 1 a 6. Além disso, destacam-se as possíveis observações direcionais das transições 3 e 5.

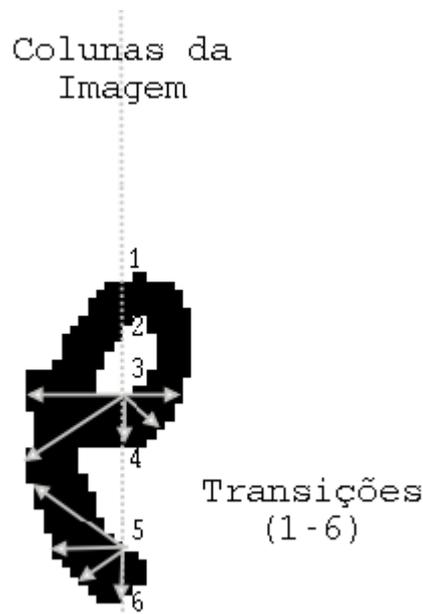


Figura 4.3: Transições em uma coluna do caractere “e”, e as observações direcionais para estimar a direção média das transições 3 e 5.

Adicionalmente à informação direcional, calculam-se duas outras características locais: a) posição relativa de cada transição, levando em conta o topo do caractere e b) se a transição

pertence ao contorno interno ou externo, a qual mostra a presença de contornos fechados na imagem do caractere. Para cada coluna consideram-se 8 possíveis transições. Neste ponto o vetor de característica é composto por 32 valores. As características globais são baseadas em projeções verticais (PV) de pixels pretos para cada coluna, e a derivada de PV entre colunas adjacentes. Finalmente, as características CT perfazem um total de 34 valores normalizados entre 0 e 1.

Características de Fundo (CF): consistem em informações referentes às concavidades do caractere. Estas características são usadas para destacar as propriedades de topologia e geometria das classes de caracteres. Cada característica de concavidade representa o número de *pixels* brancos que pertencem a uma configuração específica de concavidade. O rótulo para cada *pixel* é escolhido baseado nos códigos de quatro direções. Cada direção é explorada até encontrar um *pixel* preto ou o limite imposto pela menor caixa que contém o caractere (*bounding box*). Um *pixel* branco é rotulado se ao menos duas direções consecutivas atingirem *pixels* pretos. Assim, obtém-se 9 possíveis configurações. No entanto, consideramos mais quatro configurações, para detectar com mais precisão a presença de contornos fechados. Desta forma, o número total de características CF é 13. O vetor de concavidade é normalizado entre 0 e 1, dividindo-se cada entrada pelo total de códigos de concavidade computado para cada coluna ou linha da imagem do caractere. A Figura 4.4 mostra as 9 configurações de concavidade e também 4 configurações (A,B,C,D) para falsos contornos fechados.

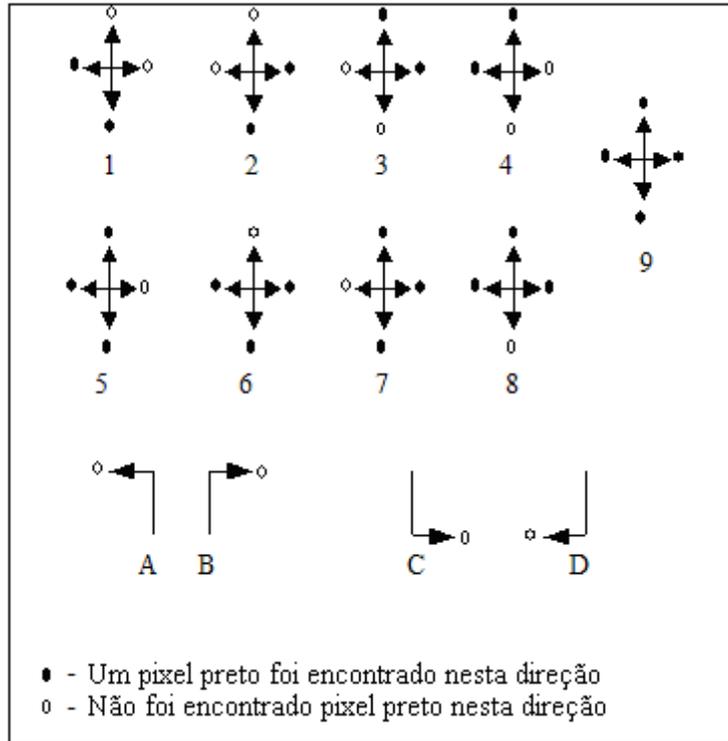


Figura 4.4: Configurações de Concavidade. Figura adaptada de [Britto, 2004].

4.1.2 Modelos Escondidos de Markov

Cada classe de caractere é representada por dois MEMs discretos: um baseado em colunas $(\lambda_c^a, \lambda_c^b, \dots, \lambda_c^z)$ e outro baseado em linhas $(\lambda_l^a, \lambda_l^b, \dots, \lambda_l^z)$ da imagem do caractere. Deste modelo de colunas e linhas, provêm um modo de combinar características complementares de traço e de fundo, em um sistema de zoneamento como mostrado na Figura 4.1. A entrada de cada modelo são sequências de observações discretas, geradas a partir da discretização dos vetores de características extraídos dos caracteres. Para tal, cada vetor é mapeado para um de 256 possíveis símbolos discretos, disponíveis em um *codebook* previamente construído, usando o algoritmo de quantização vetorial K-means [Britto, 2004]. Assim, a saída do método de extração de características consiste em sequências de observações discretas para cada caractere: ao menos uma sequência de observações extraídas das colunas e outra das linhas.

A topologia do modelo de caractere é definida levando em conta o reconhecimento do texto manuscrito. Isto significa o uso de um modelo esquerda-direita com o número de estados definido como descrito em [Rabiner, 1989].

4.2 Estratégias de Otimização

Quatro novas estratégias foram investigadas. Duas destas estratégias dizem respeito à combinação das características CT e CF, além da combinação de características em um mesmo vetor já implementada no método original. No método base informações de traço e fundo são combinadas em um vetor de 47 características, sendo 34 de traço e mais 13 de fundo. Estas características são extraídas separadamente para as colunas e linhas, assim o método funciona com dois MEMs para cada classe de caractere. As estratégias de otimizações avaliadas para combinações de CT e CF consistem em diferentes abordagens para a combinação das características complementares nos modelos. A primeira estratégia para combinar CT e CF é usar diferentes matrizes de observação por estado (ou diferentes matrizes B). Para cada modelo de Markov de linha e de coluna dos caracteres, são implementadas duas matrizes B, as quais se processa características de traço e fundo separadamente, tanto para linhas como para colunas. Por tanto, cada uma das duas matrizes B de cada MEM processa um tipo de característica diferente, traço e fundo, separadamente para linhas e colunas. A segunda estratégia é a combinação, considerando-se múltiplos modelos de Markov, ou seja, quatro modelos (2 modelos baseados em CT e CF extraídos das colunas e 2 modelos baseados em CT e CF extraídos das linhas do caractere) ou seja, teremos quatro MEMs, um para cada tipo de característica. Uma terceira e nova estratégia se refere ao aumento da base de treinamento introduzindo a abordagem chamada de DUF-sd (Deixe Um de Fora – sem duplicações), que aumenta a base de treinamento gerando novas sequências de observações a partir das sequências originais. A quarta e última estratégia se refere ao estudo da redução do impacto de ruídos, normalmente presentes nas sequências de observações discretas utilizadas no treinamento dos modelos de Markov, conforme apresentado em [Santos, 2008]. Investiga-se a geração de novas sequências de observações com base em filtragem, para detecção de observações, cuja ocorrência está abaixo de um limiar experimentalmente estabelecido. Os resultados experimentais são apresentados na Seção 5.

4.2.1 Múltiplas matrizes B

Esta estratégia utiliza múltiplas matrizes B (matriz que fornece a probabilidade das observações ou símbolos por estado do MEM) para cada MEM. Para entendermos melhor o funcionamento de uma matriz B, a Tabela 4.1 mostra o uso de uma matriz B em um modelo para previsão meteorológica. Cada uma das observações, que está na faixa entre zero e três,

representa uma faixa de umidade relativa do ar conforme Tabela 4.2. Cada fração numérica nas colunas, representa a probabilidade de uma umidade relativa do ar ocorrer diante de um dado estado meteorológico. Considerando a umidade do ar fornecida pelas observações e os possíveis estado meteorológicos, podemos então estimar a probabilidade de uma certa umidade relativa do ar ocorrer em um determinado estado meteorológico.

Estados/Obs	0	1	2	3
Ensolarado	0.4	0.3	0.2	0.1
Chuvoso	0.1	0.2	0.3	0.4
Nublado	0.1	0.2	0.5	0.2

Tabela 4.1: Matriz B que representa uma probabilidade meteorológica.

Obs	Umidade
0	umidade = < 25%
1	25% < umidade =< 50%
2	50% < umidade =< 75%
3	75% < umidade =< 100%

Tabela 4.2: Umidade relativa do ar para cada observação da matriz B.

Neste trabalho as matrizes B são utilizadas para fornecer as probabilidades de símbolos por estado. Nesta estratégia específica cada matriz B contém apenas informações sobre um tipo de características, a saber: a) traço coluna; b) traço linha; c) fundo coluna; e d) fundo linha. O intuito da estratégia é separar as características de traço e de fundo para que cada matriz B apresente um conjunto de probabilidades mais adequado. Logo, neste caso, cada modelo considerando duas matrizes de observações independentes a cada passo no tempo discreto tem os parâmetros $\lambda = (A, B^0, B^1, \pi)$, em que B^0 é a matriz de probabilidade de observações por estado $b^0_j(k)$, que nos dá a probabilidade de observar o símbolo (ou observação) k no estado j , e B^1 , de maneira similar, é a matriz de probabilidade de observações por estado $b^1_j(k)$. No treinamento e no reconhecimento $b_j(k)$ é substituído por $b_j(k^0, k^1)$, o qual é computado como:

$$b_j(k^0, k^1) = b_j^0(k^0) \times b_j^1(k^1) \quad (4.6)$$

Desta forma cada um dos dois MEMs, um que contém características de linha e o outro de colunas, passarão a ter suas subcategorias (traço e fundo) processadas em matrizes B separadas, gerando assim um total de quatro matrizes B.

Alterações no algoritmo de Baum-Welch utilizado para treinamento dos modelos, e no algoritmo de Viterbi utilizado para classificação dos modelos, foram realizadas a fim de se adequarem aos parâmetros da equação (4.6). Seguem a seguir as alterações necessárias para cada algoritmo:

- Baum-Welch:

1) Alteração da equação (3.6), da probabilidade de estar no estado S_i no tempo t , e estado S_j no tempo $t+1$.

$$\xi_t(i, j) = \frac{a_t(i) a_{ij} b_j(O_{t+1}^0, O_{t+1}^1) \beta_{t+1}(j)}{\sum_{i=1}^N \sum_{j=1}^N a_t(i) a_{ij} b_j(O_{t+1}^0, O_{t+1}^1) \beta_{t+1}(j)} \quad (4.7)$$

2) Alteração da equação (3.12) em que é o esperado número de vezes do estado j e símbolo de observação V_k dividido pelo esperado número de vezes no estado j .

$$\bar{b}_j(k^0, k^1) = \frac{\sum_{t=1}^T \gamma_t(j)}{\sum_{t=1}^T \gamma_t(j)} \quad (4.8)$$

- Viterbi

1) Alteração da equação (3.15a) de inicialização.

$$\delta_1(i) = \pi_i b_i(O_1^0, O_1^1), \quad 1 \leq i \leq N \quad (4.9)$$

2) Alteração da equação (3.16a) de recursão.

$$\delta_t(j) = \max_{1 \leq i \leq N} [\delta_{t-1}(i) a_{ij}] b_j(O_t^0, O_t^1), \quad 2 \leq t \leq T \quad (4.10)$$

$$1 \leq j \leq N$$

As alterações nos algoritmos foram bem sucedidas, bem como a estratégia como pode ser verificado na seção 5.2. Isto comprova que as matrizes de probabilidade se adaptam melhor ao modelo processando um único tipo de característica.

4.2.2 Múltiplos Modelos

Neste trabalho os MEMs são utilizados para treinamento e classificação dos caracteres que são representados por sequências de observações de suas características. O sistema base o qual foi utilizado para otimização neste estudo, utiliza originalmente dois MEMs, sendo um baseado em características de colunas que tem suas subdivisões em características de traço e fundo, e o outro MEM baseado em características de linhas também como suas subdivisões em características de traço e fundo. A proposta neste contexto é a criação de múltiplos MEMs que consiste em combinar quatro MEMs, cada MEM treinado com base em um tipo de característica, sendo: a) traço coluna; b) traço linha; c) fundo coluna; d) fundo linha.

Foram realizadas alteração nos módulos do sistema para combinar os quatro modelos; a seguir podemos observar as descrições dos módulos com suas funcionalidades originais e as alterações realizadas.

- 1) Extração de características: este módulo é responsável pela extração de características de traço e de fundo de cada caractere, sendo extraídas 34 características de traço e 13 de fundo, somando 47 características que são extraídas separadamente para as características de colunas e linhas. Estas características são representadas por sequências de observações numéricas e são armazenadas em arquivos para serem utilizadas nos módulos seguintes. Neste módulo, alterações não foram realizadas, já que apesar das características de colunas e linhas estarem sendo extraídas e armazenadas em arquivos diferentes, mais suas subdivisões traço e fundo estarem sendo armazenadas em um mesmo arquivo, a separação destas subdivisões, é possível ser realizada nos módulos seguintes.

- 2) Pré Vetor de Quantização: é o módulo que concatena todas as sequências de observações extraídas no módulo anterior em dois grandes arquivos, um para linhas e outro para colunas que serão usados para a construção dos *codebooks*. Este módulo foi alterado para concatenar quatro grandes arquivos, sendo cada um deles baseados em um tipo de observação sendo: traço coluna, traço linha, fundo coluna e fundo linha. Cada um dos quatro arquivos são gerados, extraindo individualmente seus tipos de características dos arquivos gerados no módulo anterior de extração de características.
- 3) Vetor de Quantização: no sistema base original com os dois arquivos gerados no módulo anterior era aplicado o algoritmo de K-Means em cada um deles onde eram gerados dois *codebooks*, um para linhas e outro para colunas. Cada *codebook* contendo 256 *codewords*. No sistema otimizado utilizamos os quatro arquivos gerados no módulo anterior para aplicar o algoritmo de K-Means em cada um deles e gerar quatro *codebooks*, cada um deles baseados em um tipo de característica, traço coluna, traço linha, fundo coluna e fundo linha e cada *codebook* também contendo 256 *codewords*.
- 4) Sequências de Observações: este módulo utiliza os *codebooks* gerados no módulo anterior para gerar sequências de observações discretizadas a partir das sequências de observações gerada no módulo de extração de característica. Originalmente estas sequências de observações discretizadas eram criadas utilizando dois *codebook*, um de sequências de observações de linhas e outro de colunas, gerando assim dois tipos de arquivos de sequência de observações discretizadas, ou seja, um para as características de colunas e outro para as características de linhas, cada um com suas subcategorias de traço e fundo. No sistema otimizada são utilizados quatro *codebooks* cada um com um tipo de característica, traço coluna, traço linha, fundo coluna e fundo linha, assim gerando quatro tipos de sequências de observações discretizadas.
- 5) Definição de número de estados com Wang: neste módulo é aplicado o algoritmo de Wang que utiliza as sequências de observações geradas no módulo anterior para descobrir qual o melhor número de estados a ser utilizado por cada classe de caractere na fase de treinamento em cada MEM. Originalmente este módulo avaliava as sequências de observações, geradas no módulo anterior para linhas e colunas e

gerava duas listas de números de estados para cada classe de caractere, um para as linhas e outro para as colunas. No sistema otimizado este módulo, gera quatro listas de números de estados para cada classe de caractere, baseados nos quatro tipos de sequências de observações geradas no módulo anterior.

- 6) **Treinamento:** este módulo utiliza o algoritmo de Baum-Welch para treinar os MEMs em cada classe de caractere. Originalmente eram utilizados dois MEMs para cada classe de caractere, um para as características de colunas e outro para as características de linhas. No sistema otimizado são utilizados quatro MEMs para cada classe de caractere. Cada um dos quatro MEMs para cada classe de caractere, processa um tipo de característica, traço coluna, traço linha, fundo coluna e fundo linha.
- 7) **Classificação:** Na última fase que é a de classificação, cada modelo é testado individualmente pelo algoritmo de Viterbi onde se obtém as probabilidades de cada classe de caractere e para cada uma das classes temos quatro probabilidades, sendo cada uma destas resultantes de cada um dos quatro MEMs. Para se definir a classe com a maior probabilidade, todas as quatro probabilidades, cada uma geradas por um MEM de cada classe de caracteres são somados e a classe que obtiver a maior probabilidade, é escolhida, como mostra a equação (4.11).

$$\lambda^{a\dots z} = \lambda_{tc}^{a\dots z} + \lambda_{tl}^{a\dots z} + \lambda_{fc}^{a\dots z} + \lambda_{fl}^{a\dots z} \quad (4.11)$$

Na Figura 4.5 podemos observar as sequências de fases utilizadas no método de múltiplos MEMs.

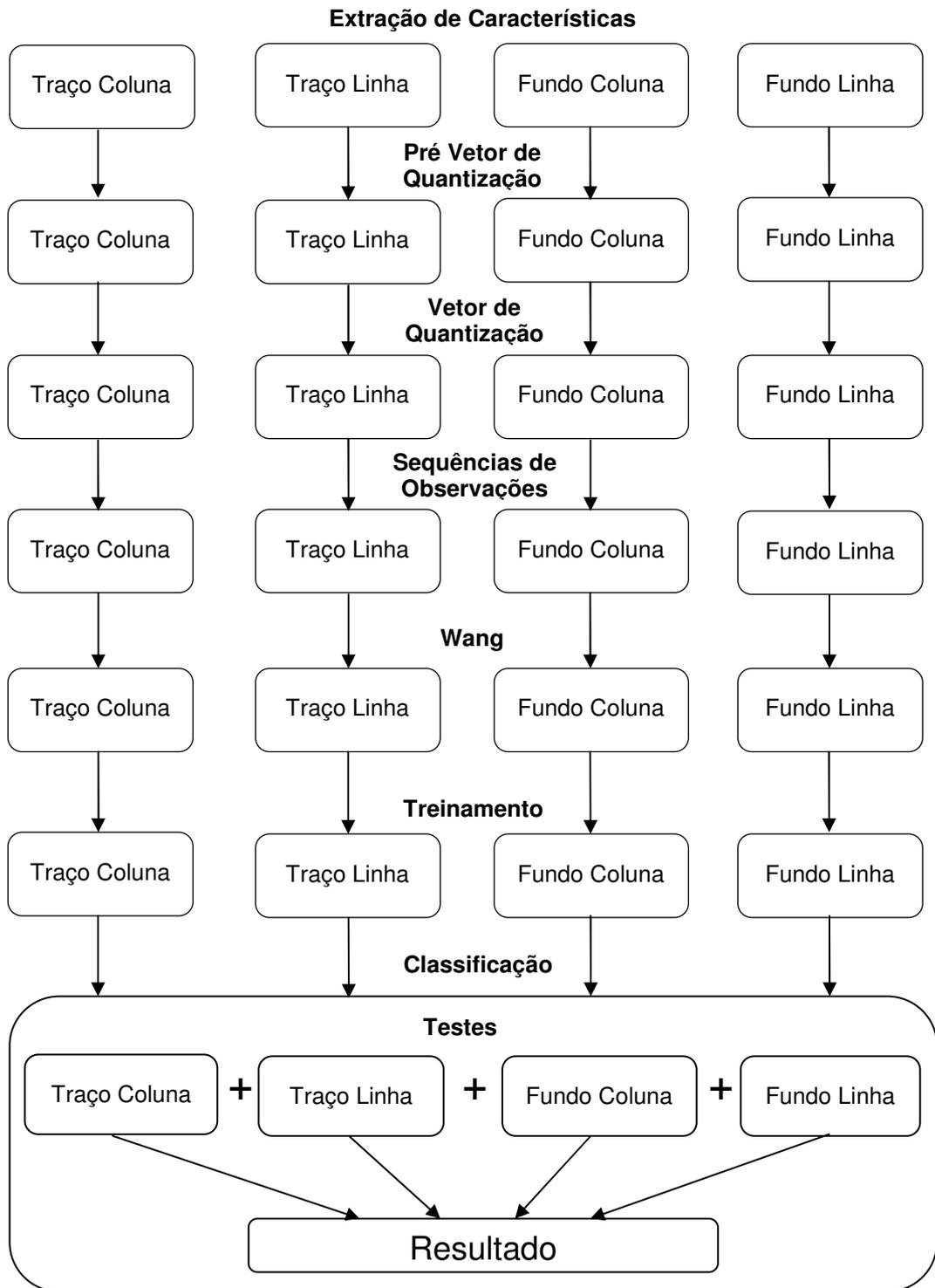


Figura 4.5: Visão geral da estratégia de Múltiplos Modelos.

Como na estratégia do uso de múltiplas matrizes B a intenção é que cada modelo criado se adapte melhor ao tipo de características nele inserido.

4.2.3 Aumento da Base de Treinamento Utilizando DUF-sd

O aumento na base de sequências de observações discretas utilizadas no treinamento dos modelos de caracteres pode ser obtido através da geração de novas sequências a partir das existentes. Aumentando a base de treinamento, ela se torna mais consistente, sendo que para sistemas baseados em MEM, isto é de suma importância para uma melhor eficiência. Para o aumento da base de treinamento apresentamos a estratégia aqui denominada de “Deixe um de fora – sem duplicação” (DUF-sd). A técnica DUF-sd é similar a técnica “Deixe um de fora” (DUF) apresentada em [Ko, 2008], porém a técnica DUF-sd apresenta um método para rejeição da criação de sequências duplicadas, como veremos nesta seção.

O sistema consiste em replicar a sequência já existente, porém a sequência replicada não conterá apenas um observação da sequência original. Na figura 4.6 podemos notar uma sequência original e logo abaixo uma sequência gerada, nota-se que apenas a primeira observação foi retirada da sequência original onde isto gerou uma nova sequência, o processo continua replicando a sequência original porém retirando uma outra observação da sequência original até que várias sequências sejam geradas cada uma sem uma observação da sequência original. O número de sequência a ser replicada é igual ao número de símbolos na sequência, pois cada sequência replicada não conterá um símbolo diferente da sequência original. Desta forma a base de treinamento conterá a sequência original mais as sequências geradas.

Para cada sequência de observações M , novas sequências serão geradas, onde M será igual ao número de observações disponíveis na sequência processada. Em uma sequência de observação com cinco símbolos, por exemplo, vai gerar mais cinco novas diferentes sequências. Então, na base teremos a sequência original mais as cinco novas sequências geradas, ficando um total de seis sequências (ver Figura 4.6).

Sequência Original:	(a,b,c,d,e)
Sequências Geradas:	(b,c,d,e)
	(a,c,d,e)
	(a,b,d,e)
	(a,b,c,e)
	(a,b,c,d)

Figura. 4.6: Sequência de observações geradas pelo processo DUF-sd a partir da sequência original.

Se usássemos a técnica DUF como implementada em [Ko, 2008], muitas sequências de observações repetidas serão criadas, pelo fato de existirem dentro das sequências de observações, muitas observações iguais uma após a outra e neste caso removendo uma observação, criará uma nova sequência e após isto removendo a observação seguinte que é igual a anterior removida, irá criar uma sequência duplicada, como mostrado nas duas primeiras sequências geradas da Figura 4.7.

Sequência Original:	(a,a,b)
Sequências Geradas:	(a,b)
	(a,b)
	(a,a)

Figura 4.7: Sequência de observação original e três novas sequências geradas a partir da original.

Para resolver este problema, uma regra que rejeita a criação de sequências duplicadas foi implementada; desta maneira sequências de observações duplicadas e inúteis não serão criadas tornando o processo mais rápido e eficiente. Por este motivo chamamos a técnica de DUF-sd (Deixe Um de Fora – sem duplicações)

Aumentando a base de treinamento utilizando a técnica DUF-sd, tivemos um pequeno aumento na taxa de reconhecimento como mostra a Tabela 5.1; notamos que o método traz resultados positivos e pode ser utilizado para novos estudos como foi feito neste trabalho e é mostrado na próxima seção.

4.2.4 Redução dos Ruídos na Base de Treinamento

Na base de treinamento existem muitas observações que podem ser consideradas ruídos, ou seja, observações que não ajudam e sim atrapalham no processo de treinamento dos modelos. As observações com baixa frequência de aparecimento na base de treinamento são possíveis candidatas a serem ruídos, pois observações de uma classe de caractere tende a se repetirem com certa frequência. Estes possíveis ruídos atrapalham no processo de treinamento assim prejudicando a taxa de reconhecimento. Desta forma criando um “reductor de ruído” (RR) para a base de treinamento, teremos teoricamente um aumento na taxa de reconhecimento. Porém não criamos um, mas sim dois tipos de redutores de ruídos que veremos a seguir.

4.2.4.1 Redução dos Ruídos na Base de Treinamento sem DUF-sd

Este reductor de ruído permite reduzir o impacto de observações com baixa frequência de aparecimento na base de treinamento. A idéia então é deixar de fora as observações cuja frequência de aparecimento na base de treinamento é menor que um determinado limiar, experimentalmente definido.

Na Figura 4.8 podemos observar uma sequência de observações com suas respectivas frequências de aparecimento na base de treinamento. Na observação de número 1 se nota uma frequência de aparecimento de 0,3%. Isto indica que em toda base de treinamento da classe onde esta sequência pertence à frequência de aparecimento da observação 1, é de 0,3%.

Se determinarmos um limiar de 1%, significa que as observações 1 e 4 da figura 4.8, serão eliminadas da base de treinamento. Ou seja, determinamos que observação que tem sua frequência de aparecimento na base de treinamento na classe ao qual está inserida abaixo de 1%, são possivelmente ruídos e por isto as eliminamos.

$$O_{0,3\%}^1, O_{1,2\%}^2, O_{2,8\%}^3, O_{0,8\%}^4, O_{1,4\%}^5$$

Figura 4.8: Sequência de observações com as frequências de aparecimento na base de treinamento

4.2.4.2 Redução dos Ruídos na Base de Treinamento com DUF-sd

Um outro método avaliado foi a combinação do RR com a técnica DUF-sd da seguinte forma. Na técnica DUF-sd cada uma das observações na sequência vai ser em algum momento deixada de fora para criar uma nova sequência como já mostramos na seção anterior, porém utilizando o RR integrado ao DUF, permitimos que somente observações que são possíveis candidatas a serem ruídos, sejam eliminadas na sequência original para criar uma nova sequência. Por exemplo, se estipularmos um limiar de 1% somente as observações 1 e 4 da Figura 4.8 serão deixadas de fora para criar novas sequências, ou seja, teremos a sequência original mais duas novas sequências, uma duplicando a sequência original porém sem a observação 1 e outra duplicando a sequência original porém sem a observação 4 como mostra a Figura 4.9.

$$\begin{array}{c}
 O_{0,3\%}^1, O_{1,2\%}^2, O_{2,8\%}^3, O_{0,8\%}^4, O_{1,4\%}^5 \\
 O_{1,2\%}^2, O_{2,8\%}^3, O_{0,8\%}^4, O_{1,4\%}^5 \\
 O_{0,3\%}^1, O_{1,2\%}^2, O_{2,8\%}^3, O_{1,4\%}^5
 \end{array}$$

Figura 4.9: Sequência de observação original e mais duas novas sequências criadas utilizando o RR

Desta forma, criamos novas sequências sem as observações que são prováveis ruídos, mas mantemos estas observações na sequência original e em outras sequências que são geradas a partir da sequência original, para que os modelos sejam treinados com várias possibilidades. As figuras aqui apresentadas mostram poucas observações por sequência, porém em uma base real existem inúmeras sequências de observações e cada sequência contém uma média de 30 a 60 observações como mostra a Figura 4.10. Assim cada vez que o método é aplicado inúmeras novas sequências são criadas. O método DUF-sd sem o RR apresentado na seção 4.2.3, irá criar um número de sequências de observações ainda maior, pois gera novas sequências a partir de cada observação. Porém o método DUF-sd implementado com o RR reduz drasticamente o número de novas sequências geradas tornando o processo mais rápido.

1 121 207 24 24 25 46 46 46 48 48 48 48 211 127 55 81 81 81 81 12
49 57 81 81 182 82 83 83 83 83 23 23 23 72 72 118 87 167 167 8 43

Figura 4.10: Uma das sequências de observações da base de treinamento do caractere *a* minúsculo contendo 42 símbolos.

4.3 Considerações Finais

Neste capítulo foi apresentada uma visão geral do método, abordando o método de extração de características, modelos escondidos de Markov e o principal foco deste trabalho que é a otimização do sistema. Os métodos de otimizações se dão pela avaliação de diferentes estratégias de combinação de características extraídas dos caracteres, como a utilização de múltiplas matrizes B e múltiplos MEMs. Também foram abordados para otimização do sistema, o aumento da base de treinamento, utilizando DUF-sd e a redução de ruídos com e sem a estratégia DUF-sd.

Capítulo 5

Experimentos

Este capítulo apresenta a base de dados utilizada bem como todos os experimentos relacionados a aplicação do método proposto no Capítulo 4. Estes experimentos trazem os resultados obtidos nas estratégias propostas, bem como a comparação com trabalhos realizados.

Na seção 5.1 é apresentada a base de dados utilizada, na seção 5.2 os experimentos considerando múltiplas matrizes B, na seção 5.3 múltiplos MEMs, na seção 5.4 o aumento da base de treinamento utilizando DUF-sd e na seção 5.5 o redutor de ruídos.

5.1 Base de Dados

A base de dados utilizada para os experimentos foi a NIST SD19 de caracteres. Foram utilizadas 74,880 amostras das séries *hsf_0*, *hsf_1*, *hsf_2*, e *hsf_3* para treinamento, 23,670 amostras da série *hsf_7* para validação, e 23.941 amostras da série *hsf_4* para testes. Duas linhas de experimentos foram seguidas: um experimento foi baseado na combinação de CT e CF em múltiplas matrizes B, e o outro na combinação de CT e CF em múltiplos modelos, a combinação de CT e CF em um mesmo vetor é apresentada em [Britto, 2004]. As estratégias RR e DUF não são utilizadas nos experimentos baseados em diferentes matrizes B, pelo fato que é necessário ter a mesma quantidade de observações nas sequências a fim de combiná-las. Contudo, é possível observar a contribuição das estratégias RR e DUF quando consideram-se múltiplos MEMs.

5.2 Experimentos Considerando Múltiplas Matrizes B

Para os experimentos de combinar múltiplas matrizes B às sequências de observações extraídas das imagens, e discretizadas em um codebook de 256 entradas, foram inseridas no módulo de treinamento em duas fases. Na primeira fase, o módulo de treinamento recebe apenas as sequências de observações de colunas que contém observações de traço e fundo. O módulo é carregado com estas observações e o algoritmo de Baum-Welch recebe e trata as observações de traço e fundo separadamente, ou seja, no algoritmo existe uma matriz B para as características de traço e outra para as características de fundo. A segunda fase segue o mesmo princípio da primeira com a diferença que na segunda o módulo recebe somente as características de linhas. O módulo de treinamento gera uma base de dados que é carregado no módulo de testes, o algoritmo de Viterbi neste módulo recebe as probabilidades de duas matrizes B e separadamente para linhas e colunas. O caractere que é escolhido como o mais provável para representar uma sequência de observação, é aquele que apresenta a maior probabilidade somando as duas matrizes B, que contém as características de traço e fundo das linhas, e após somando com as probabilidades das duas matrizes B que contém as características de traço e fundo das colunas.

Os experimentos combinando múltiplas matrizes B mostraram uma melhoria na taxa de reconhecimento comparado ao método original, conforme observa-se na Tabela 5.1. Este aumento na taxa de reconhecimento se deve ao fato de cada matriz B tratar individualmente as características de traço e fundo.

5.3 Experimentos Considerando Múltiplos MEMs

Nos experimentos em múltiplos MEMs desde a fase de extração de características até a fase de testes, cada tipo de características (traço coluna, traço linha, fundo coluna e fundo linha) são processadas individualmente. Cada um destes tipos de características gera seu próprio codebook e nos treinamentos e testes cada um destes tipos de características possui seu próprio MEM. Somente no final da fase de testes, é que as probabilidades geradas por estas características individualmente, são somadas e o caractere que obtiver a maior probabilidade, é escolhido para representar as sequências de observações.

Combinando múltiplos modelos obtivemos melhorias nas taxas de reconhecimento, semelhantes à obtida na estratégia de combinar múltiplas matrizes B como mostra a Tabela 5.1. Contudo, usando a estratégia de combinar distintos MEMs temos o diferencial de poder

considerar quantidades de vetores de observações diferentes em cada MEM e também diferentes quantidades de observações por sequência. Estas particularidades dão a esta estratégia, a liberdade de avaliar as estratégias RR e DUF manipulando-se assim a sequência de observações da base de treinamento, ou seja, para aplicar as novas estratégias, utilizamos como sistema base, o sistema que já utiliza a estratégia de múltiplos MEMs, onde já obteve sucesso e é flexível para tal feito – conforme aplicado nos próximos experimentos.

Tabela 5.1: Taxas de reconhecimento em (%)

Letras	Vetor Único	Múltiplas Matrizes B	Múltiplos Modelos	Múltiplos Modelos + DUF	Múltiplos Modelos + Redutor de Ruído	
					Sem DUF Limiar 0.05%	Com DUF
Minúsculas	84,0	85,03	84,94	84,96	83,34	85,67 DUF 0.12
Maiúsculas	90,0	91,44	91,14	91,29	89,46	91,62 DUF 0.16
52 Classes	87,0	88,27	87,76	87,87	85,76	88,22 DUF min. 0.11, mai. 0.18

Tabela 5.2: Comparação com Trabalhos Relacionados, [Bortolozzi, 2005].

Método	Treinamento	Validação	Teste	Taxa Rec. (%)
(Oh, 1998) Minúsculas (26 classes)	26,000	-	11,941	90.0
(Dong, 2001) Minúsculas (26 classes)	23,937	-	10,688	92.3
(Koerich, 2002) Maiúsculas (26 classes)	37,440	12,092	11,941	92.3
Minúsculas (26 classes)	37,440	11,578	12,000	84.6
Ambas (52 classes)	74,880	23,670	23,941	85.5
(Britto, 2004) Maiúsculas (26 classes)	37,440	12,092	11,941	90.0
Minúsculas (26 classes)	37,440	11,578	12,000	84.0
Ambas (52 classes)	74,880	23,670	23,941	87.0

5.4 Experimentos Aumentando a Base de Treinamento utilizando DUF-sd

A motivação para o estudo da otimização da base de treinamento é o fato da base de treinamento ser uma peça chave para o sucesso de um sistema de reconhecimento de manuscritos.

Os experimentos utilizando a técnica DUF-sd para o aumento da base de treinamento foram realizados, utilizando o sistema onde já foi implementada a estratégia de múltiplos modelos, já que esta estratégia já apresentou sucesso e sua flexibilidade permitiu a inserção da técnica DUF-sd, assim combinando as duas técnicas. A técnica DUF-sd obteve um pequeno resultado positivo conforme mostra a Tabela 5.1. Porém podemos notar que a técnica é promissora, pois neste mesmo trabalho na seção 4.2.4 esta técnica foi novamente utilizada, porém combinada com a técnica de redução de ruídos onde os resultados mostraram uma boa eficiência.

Este estudo teve como intuito não só aumentar a taxa de reconhecimento destes sistemas, mas também, difundir a técnica aqui empregada, que se mostrou promissora e inspira futuros estudos.

5.5 Experimentos Considerando Redutor de Ruídos

A seguir veremos o resultado dos experimentos da técnica de redução do impacto de ruídos na base de observações discretas, das duas formas que foram implementados, sendo:

- Sem DUF-sd: foram realizados testes utilizando um sistema base já implementado a estratégia de múltiplos modelos e com a base de letras minúsculas, ao qual o reconhecimento expende uma dificuldade maior. Em dois testes os limiares utilizados foram de 0.01% e 0.05% e os resultados atingiram as seguintes taxas de reconhecimento respectivamente 83.97% e 83.34%, ou seja menor que a taxa de reconhecimento original, apresentada para múltiplos modelos como mostra a Tabela 5.1. Este método nesta forma de implementação não obteve sucesso, pois uma observação que tem uma baixa representatividade na base de treinamento de uma determinada classe pode ser um provável ruído, mas também pode ser uma observação fundamental para que determinada sequência seja entendida pelo modelo.

- Com DUF-sd: nesta técnica o sistema base utilizado é também o sistema que já tem implementado a estratégia de múltiplos modelos, a redução de ruídos é realizada através da geração de novas sequências a partir das existentes. A técnica DUF-sd já aqui apresentada, foi utilizada para a escolha da observação a ser deixada de fora; foram avaliados diferentes valores de limiares como apresentado na Tabela 5.3. O valor que apresentou melhores resultados para letras minúsculas foi de 0.12 e para letras maiúsculas de 0.16, ou seja, foram deixadas de fora para geração de novas sequências, as observações cujas frequências de

aparecimento na base de treinamento eram menores, que 0.12% para letras minúsculas e de 0.16% para letras maiúsculas.

Tabela 5.3: Taxas de reconhecimentos para classes maiúsculas e minúsculas considerando diferentes limiares DUF-sd

	Limiar	Taxa de reconhecimento em (%)
Minúsculas	0,07	85,36
"	0,08	85,45
"	0,09	85,48
"	0,10	85,53
"	0,11	85,63
"	0,12	85,67
"	0,13	85,43
"	0,14	85,48
"	0,15	85,53
"	0,16	85,47
"	0,17	85,35
<hr/>		
Maiúsculas	0,11	91,43
"	0,12	91,42
"	0,13	91,42
"	0,14	91,59
"	0,15	91,56
"	0,16	91,62
"	0,17	91,48
"	0,18	91,48
"	0,19	91,52
"	0,20	91,56
"	0,21	91,44

Para os testes em 52 classes, ou seja, maiúsculas e minúsculas foram inicialmente utilizados os melhores limiares apresentados para cada classe de caractere, sendo 0.12 para minúsculas e 0.16 para maiúsculas, chegando assim a uma taxa de reconhecimento de 88.10%. Porém, em novos experimentos realizados, notou-se que uma pequena alteração nestes limiares, trouxeram melhorias na taxa de reconhecimento. Os limiares que apresentaram melhor resultado para 52 classes foram de 0.11 para classes minúsculas e de 0.18 para classes maiúsculas, chegando assim a uma taxa de reconhecimento de 88,22%.

Capítulo 6

Conclusão

Este trabalho apresentou diferentes estratégias para otimizar um sistema baseado em MEM para reconhecimento de caracteres manuscritos. Quatro abordagens foram avaliadas, onde todas elas apresentaram resultados positivos, aumentando assim as taxas de reconhecimento do sistema base, como mostrado nos experimentos.

A primeira estratégia abordada foi a combinação de características em múltiplas matrizes B. O intuito desta estratégia foi separar as características de traço coluna, traço linha, fundo coluna e fundo linha em diferentes matrizes B. Desta forma a taxa de reconhecimento teve um aumento considerável, em virtude dos modelos de probabilidade das matrizes B se adequarem melhor a cada um dos tipos das características específica, assim um tipo de característica não interferiu no outro.

A estratégia de combinar múltiplos MEMs também apresentou melhoras nas taxas de reconhecimento do sistema. Esta técnica separou e processou as características dos caracteres em quatro grandes divisões sendo, traço coluna, traço linha, fundo coluna e fundo linha. O intuito desta separação foi instigar os modelos a se adequarem melhor a cada tipo de característica. Após separadas e processadas estas quatro divisões só foram reunidas novamente na fase de classificações, em que suas probabilidades foram somadas para definir o resultado.

Tanto a estratégia de combinar múltiplos MEMs como a estratégia de combinar múltiplas matrizes B foi focada em tratar tipos de características individualmente. Os resultados destas duas estratégias foram semelhantes. A avaliação e comparação destas foi importante para definir qual das duas apresenta melhor resultado, assim também auxiliando

em trabalhos futuros. Uma particularidade da estratégia de combinar múltiplos MEMs foi crucial para a avaliação da estratégia de redução de impactos de ruídos na base de treinamento. Enquanto na estratégia de combinar múltiplas matrizes B não podemos alterar o número de observações, e nem o número de sequências de observações das características de colunas comparado as características de linhas e vice versa, a estratégia de combinar múltiplos MEMs nos dá esta possibilidade.

O aumento da base de treinamento utilizando a estratégia DUF trouxe um pequeno aumento na taxa de reconhecimento do sistema. Porém esta técnica se mostrou promissora já que como foi apresentado esta, foi de suma importância para o sucesso da técnica de redução de ruídos, onde as técnicas RR e DUF foram unidas para gerar uma única estratégias bem sucedida.

A estratégia de redução do impacto de ruídos na base de treinamento, foi implementada, utilizando como sistema base, o sistema no qual já foi implementado a estratégia bem sucedida de utilização de múltiplos MEMs. O objetivo desta estratégia foi diminuir o número de observações dos caracteres, que são possíveis ruídos e interferem no processo de reconhecimento do sistema. Inicialmente criamos um sistema ao qual de acordo com um limiar pré-estabelecido eliminava possíveis ruídos, porém a estratégia implementada desta forma não obteve o sucesso esperado. Então em uma segunda implementação, utilizamos a técnica RR em conjunto com a técnica DUF, aumentando assim a base de treinamento, de forma tal, a inserir novos conjuntos de sequências de observações sem a presença de observações que são possíveis ruídos, porém mantemos estas observações na sequência original e em outras sequências que são geradas a partir da sequência original, para que os modelos sejam treinados com várias possibilidades. Os resultados obtidos com a redução de ruídos na base de treinamento em conjunto com a técnica DUF, se mostrou promissora, já que atingiu seu objetivo que foi aumentar a taxa de reconhecimento do sistema.

Na literatura, poucos trabalhos consideram 52 classes. A Tabela 5.2 apresenta alguns dos melhores resultados de trabalhos relacionados ao nosso. É possível observar que para 52 classes, os modelos otimizados são realmente promissores. Isso nos incentiva a investigar novos conjuntos de características, bem como uma avaliação mais detalhada do impacto de observações com baixa frequência em cada classe de caractere.

Contribuições: do lado social o aumento da taxa de reconhecimento de caracteres manuscritos aumenta as possibilidades de aplicações práticas como: checagens bancárias,

reconhecimento de endereços postais, leitura de formulários, entre outros. Do lado científico a avaliação de diferentes estratégias como:

- Múltiplas Matrizes B
- Múltiplos MEMs
- Aumento da base de treinamento utilizando DUF
- Redutor de Ruídos

Trazem auxílio a futuras pesquisas nesta área já que estas estratégias podem ser aplicadas em diferentes sistemas para reconhecimentos de manuscritos baseados em MEMs e até mesmo em sistema para reconhecimentos de voz também baseados em MEMs.

Futuros trabalhos podem ser desenvolvidos aplicando técnicas aqui avaliadas como o uso de múltiplos modelos. Esta técnica aumentou o número de amostras a serem classificadas, experimentos utilizando diferentes classificadores como NN (Neural Network) ou a combinação com o classificador existente são estratégias que podem ser investigadas. Em [Carvalho, 2002] os classificadores NN e MEM foram avaliados sozinhos e combinados em um sistema de reconhecimento de palavras manuscritas. A combinação de NN e MEM se mostrou promissora no sistema avaliado, se esta combinação também pode ser promissora no sistema de múltiplos modelos para reconhecimento de caracteres manuscritos é uma questão a ser levantada. As estratégias DUF e RR foram desenvolvidas na base de treinamento, estas estratégias com algumas modificações e implementações também podem ser aplicadas na base de testes. Este tipo de avaliações podem ser promissores já que as técnicas já se mostraram bem sucedidas para aplicação na base de treinamento.

Referências Bibliográficas

- [Arica, 2002] Arica, N. & Yarman-Vural, F. T., *Optical Character Recognition for Cursive Handwriting*. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 24, no. 6, 2002.
- [Belongie, 2002] Belongie, S.; Malik, J.; Puzicha, J. *Shape matching and object recognition using shape contexts*. IEEE Trans. Pattern Analysis and Machine Intelligence, 24 (2), 2002, p. 509-522.
- [Bezdek, 1984] Bezdek, J., Ehrlich, R. e Full, W. FCM: The fuzzy c-means clustering algorithm. Computers & Geosciences, vol 10, n. 2-3, pp. 191-203, 1984.
- [Bortolozzi, 2005] Bortolozzi, Flavio; Britto Jr, A. S.; Oliveira, Luiz Eduardo Soares; Morita, Marisa. *Recent Advances in Handwriting Recognition*. In: Umapada Pal; Swapan K. Parui; Bidyut B. Chaudhuri. (Org.). Document Analysis. Kalkuta: Chennai, v. 1, 2005, p. 1-30.
- [Britto, 2001] Britto Jr., A. S. *A Two-Stage HMM-Based Method For Recognizing Handwritten Numeral Strings*. Curitiba, 2001. 142 p., Tese de Doutorado, Departamento de Informática, Pontifícia Universidade Católica do Paraná, 2001.
- [Britto, 2004] Britto Jr., A. S. et al. *Foreground and Background Information in an HMM-based Method for Recognition of Isolated Characters and Numeral Strings*. Proc. of the 9th International Workshop on Frontiers in Handwritten Recognition, 2004, v. 1. p. 371-376.

- [Cai, 1998] Cai, J. & Liu, Z. *Integration of Structural and Statistical Information for Unconstrained Handwritten Numeral Recognition*. Proceedings of the Fourteenth International Conference on Pattern Recognition, Vol. I, 1998, p. 378-380.
- [Carvalho, 2002] Carvalho, J. M. ; Oliveira Jr., Jose Josemar de ; Freitas, Cinthia Obladen Alameida ; Sabourin, Robert . *Evaluating NN and HMM Classifiers for Handwritten Word Recognition*. In: XV Brazilian Symposium on Computer Graphics and Image Processing, 2002, Fortaleza CE. Proceedings of the SIBGRAPI'02. Los Alamitos, CA : IEEE CS Press, 2002. v. 1
- [Cavalin, 2006] Cavalin, P.; Britto JR, A. S. ; Oliveira, L. E. S. ; Sabourin, R.; Bortolozzi, F. *An Implicit Segmentation based Method for Recognition of Handwritten Strings of Characters*. In: Proc. of the The 21st Annual ACM Symposium on Applied Computing. Vol. 1, 2006, p. 836-840.
- [Cheung, 1998] Cheung, K. & Yeung, D. *A Bayesian Framework for Deformable Pattern Recognition with Application to Handwritten Character Recognition*. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 20, No. 12, 1998, p.1382-1388.
- [Dong, 2001] Dong, J. X.; Krzyzak, A.; Suen, C. Y. *Local learning framework for recognition of lowercase handwritten characters*. In Proc. Int. Workshop on Machine Learning and Data Mining in Pattern Recognition, Leipzig, Germany, 2001, p. 226-238.
- [Elms, 1996] Elms, A.J. *The representation and recognition of text using hidden Markov model*. Surrey, 1996, Tese de Doutorado, Department of Electronic and Electrical Engineering, University of Surrey, 1996.
- [Freitas, 2007] Freitas, C. O. A.; Soares, Luis Eduardo Oliveira; Aires, Simone; Bortolozzi, Flávio . *Zoning and metaclasses for character recognition*. In: The 22nd Annual ACM

Symposium on Applied Computing, 2007, Seoul. Proceedings of the 2007 ACM symposium on Applied Computing, 2007. v. 1. p. 632-636.

- [Gatos, 2006] Gatos, Pratikakis B. & Perantonis, S. J. *Hybrid Off-Line Cursive Handwriting Word Recognition*, Pattern Recognition, 2006. ICPR 2006. 18th International Conference on Volume 2, Issue , 2006, p. 998 – 1002.
- [Gader, 1996] Gader, P. D. & Khabou, M. A. *Automatic Feature Generation for Handwritten Digit Recognition*. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 18, No. 12, 1996, p. 1256-1261.
- [Günter, 2003] Günter, S. & Bunke, H. *Optimizing the Number of States, Training Iterations and Gaussians in an HMM-based Handwritten Word Recognizer*. Proceedings of the Seventh International Conference on Document Analysis and Recognition., Vol.1. 2003, p.472 – 476.
- [Günter, 2004] Günter, S. & Bunke, H. *Combination of three classifiers with different architectures for handwritten word recognition*. 9th International Workshop on Frontiers in Handwriting Recognition, Kokubunji, Tokyo, Japan, 2004, p. 63-68,.
- [Hirano, 1997] Hirano, T.; Okada, Y.; Yoda, F. *Structural Character Recognition Using Simulated Annealing*. Proceedings of the Fourth International Conference on Document Analysis and Recognition, Vol. 2, 1997, p.507-510.
- [Kimura, 1992] Kimura, F. & Shridhar, M. *Segmentation-recognition algorithm for handwritten numeral strings*. Machine Vision Applications, No. 5, 1992, p. 199-210.
- [Ko, 2008] KO, Albert ; SABOURIN, Robert ; BRITTO JR, A. S. . *Leave-One-Out-Training and Leave-One-Out-Testing Hidden Markov Models for a Handwritten Numeral Recognizer: the Implication of Single Classifier and Multiple Classifications*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2008.

- [Koerich, 2002] Koerich, A. L. *Large Vocabulary Off-Line Handwritten Word Recognition*. Montreal-Canada, 2002. 314 p., Tese de Doutorado, École de Technologie Supérieure, 2002.
- [Liu, 2004] Liu, C-L. & Narukawa, K. *Normalization Ensemble for Handwritten Character Recognition*. IWFHR-9, 2004, p. 69-74
- [Milgram, 2004] Milgram, J.; Cheriet, M.; Sabourin, R. *Speeding up the decision making of Support Vector Classifier*. IWFHR-9, 2004, p. 57-62.
- [Nopsuwanchai, 2003] R. Nopsuwanchai, & D. Povey. *Discriminative training for HMM-based offline handwritten character recognition*. Proceedings of Seventh International Conference on Document Analysis and Recognition, Vol. 1, 2003, p. 114-118.
- [Oh, 1998] Oh, I. S.; Suen, C. Y. *Distance features for neural network-based recognition of handwritten characters*. International Journal on Document Analysis and Recognition, 1998, p. 73-88.
- [Pakhira, 2004] M. K. Pakhira, S. Bandyopadhyay and U. Maulik, "Validity index for crisp and fuzzy clusters," Pattern Recognition, vol. 37, No. 3, pp. 487-501, 2004
- [Rabiner, 1989] Rabiner L. R. *A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition*. Proc. of the IEEE, Vol. 77, No. 2, 1989, p.257-286.
- [Santos, 2008] Santos, Murilo; Oliveira, Luiz E. S.; Sabourin, Robert; Koerich, Alessandro L.; Britto Jr., Alceu. S. *Combinando Características Complementares em Modelos Escondidos de Markov: uma otimização para o reconhecimento de caracteres manuscritos*. In: XXXIV Conferencia Latino Americana de informática (CLEI 2008), Santa Fé. Anais da Conferência Latinoamericana de Informática (CLEI 2008), 2008. v. 1. p. 260-269.

- [Suen, 1999] Suen, C.Y.; Liu, K.; Strathy, N.W. *Sorting and recognizing cheques and financial documents*. In: S. W. Lee, Y. Nakano (Eds.), *Document Analysis Systems: Theory and Practice*, Springer, Berlin, 1999, p. 173-187.
- [Teow, 2002] Teow, L.-N. & Loe, K.-F. *Robust vision-based features and classification schemes for off-line handwritten digit recognition*. *Pattern Recognition* 35 (11), 2002, p. 2355-2364.
- [Tistarelli, 1993] M. Tistarelli & G. Sandini. *On the advantage of polar and log-polar mapping for direct estimation of time-to-impact from optical flow*. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 15, No 4, 1993. p. 401–410.
- [Woodland, 2002] P. Woodland & D. Povey. *Large scale discriminative training of hidden markov models for speech recognition*. *Computer Speech and Language*, Vol. 16, No. 1, 2002, p. 25–47.
- [Xie, 1991] X. L. Xie and G. Beni, *A validity measure for fuzzy clustering*, *IEEE Transactions of Pattern Analysis and Machine Intelligence*, 1991, p. 841-847.
- [Zhou, 1997] Zhou, J.; Qiang, G.; Suen, C. Y. *A High Performance Hand-printed Numeral Recognition System with verification Module*. *Proceedings of the Fourth International Conference on Document Analysis and Recognition*, Vol. 2, 1997, p. 293-297.