Eduardo Vellasques

Classificação de Pontos de Segmentação de Dígitos Manuscritos

Curitiba - PR

2006

Eduardo Vellasques

Classificação de Pontos de Segmentação de Dígitos Manuscritos

Dissertação de mestrado apresentada como pré-requisito de conclusão do curso de Mestrado em Informática, da Pontifícia Universidade Católica do Paraná, área de concentração: Visão, Imagem e Robótica.

Orientador:

Prof^o. Dr^o. **Luiz Eduardo Soares de Oliveira**

Co-orientadores:

Prof^o. Dr^o. **Alceu de Souza Britto Jr.**

Prof^o. Dr^o. **Robert Sabourin**

Pontifícia Universidade Católica do Paraná

Curitiba - PR

2006

Agradecimentos

A Deus toda honra, toda a glória e todo o louvor.

À Gislaine, minha noiva, pela compreensão, apoio e inspiração.

Aos meus pais, Antonio Carlos e Suely, por acreditarem que este sonho era possível e por terem me ensinado o que de mais importante eu já aprendi.

Aos meus orientadores, Dr. Luiz Eduardo Soares de Oliveira, Dr. Alceu de Souza Britto Jr. e Dr. Robert Sabourin, pela amizade, atenção e dedicação.

Aos professores do PPGIA, especialmente ao Dr. Alessandro Koerich e ao Dr. Jacques Facon pelo ensino e suporte.

À minha família pelo encorajamento.

Aos amigos de laboratório e de trabalho Paulo, Fausto, Israel, Márcio, Diogo, Neimar, Diego, Cheila, Arlete, Tânia e Fabiano pelo companheirismo e incentivo.

Aos meus avós Dirce e Antônio (*in memoriam*) pelo apoio nos tempos de graduação.

Aos professores que participaram de minha banca, Dr. Flávio Bortolozzi e Dr. Rafael Dueire Lins pelas valorosas contribuições.

Resumo

Este trabalho apresenta um método de classificação de pontos de segmentação de dígitos manuscritos. O método proposto funciona como um filtro, a ser aplicado em sistemas baseados na estratégia segmentação-reconhecimento. Esse tipo de estratégia de segmentação geralmente implica em um grande número de hipóteses de segmentação, que são posteriormente avaliadas por um classificador de dígitos isolados. Filtrar hipóteses de segmentação desnecessárias, além de contribuir para a redução do custo computacional (uma vez que o custo computacional do filtro é menor do que o do classificador de dígitos isolados) também contribui para um aumento das taxas de classificação. Através da redução do número de hipóteses de segmentação, muitas chamadas a um classificador de dígitos isolados (de 132 características) são migradas para um classificador de pontos de segmentação (filtro) de apenas 42 características. Além disso, a estratégia adotada permite que isso seja possível com um número bem menor de chamadas ao filtro do que geralmente é necessário quando se utiliza somente o classificador de dígitos isolados.

Características de alto poder de discriminância (MCA) foram utilizadas. Esse tipo de caracteristica baseia-se em análise das concavidades antes e depois da segmentação. Dessa forma, é possível detectar casos de sobre-segmentação. Detectar sobre-segmentação foi a maneira encontrada para encontrar hipóteses desnecessárias, já que hipóteses desnecessárias geram casos de sobre-segmentação. Além disso o conjunto de características escolhido é invariante à quantidade e tipo de pontos de segmentação, tamanho da cadeia, etc. Dessa forma o método proposto pode ser aplicado a qualquer sistema baseado na estratégia segmentação-reconhecimento.

O uso de um classificador SVM (para detectar casos de sobre-segmentação) e de um sistema de custo baseado em análise de curvas ROC (para otimizar o desempenho do filtro proposto) tornam o método proposto adaptativo e livre de heurísticas.

Os resultados experimentais mostram que de fato o método proposto torna possível a redução do custo computacional aliada a melhora nas taxas de classificação.

Abstract

This work presents a method to classify segmentation cuts for handwritten digits. The proposed method works as a filter, which is applied on segmentation-based recognition systems. In this strategy, the number of segmentation hypothesis created is usually high. These segmentation hypothesis are individually evaluated by a general-purpose classifier.

Filtering unnecessary cuts, not only reduces the computational effort (as the computational cost of the proposed filter is smaller than the computational cost of the general-purpose classifier) but also increases the recognition rate of the general-purpose classifier.

Through the reduction in the number of segmentation hypothesis, many calls to the general-purpose classifier, which works with 132 features are migrated to a segmentation cut classifier (filter) which works with only 42 features. Moreover, in the proposed strategy, this is done with a significantly smaller number of calls to the filter (when compared to the number of calls to the general-purpose classifier).

High discriminant features (MCA) were used. This type of feature relies on concavity analysis before and after the segmentation is done. This makes possible the detection of over-segmentation within a given connected component. Detecting over-segmentation was the way chosen to detect unnecessary cuts, as an they usually generate over-segmentation. Besides, the chosen feature set is invariant to the number and type of segmentation cuts, string length. This makes possible the use of the proposed method in every system which relies on segmentation-based recognition.

The use of an SVM classifier (to detect over-segmentation) and of ROC analysis (to optimize the performance of the proposed filter) makes the proposed method adaptive and heuristics-free.

The experimental results show us that the proposed method makes possible to reduce the computational cost and to increase the recognition performance, at the same time.

Sumário

Agradecimentos **ii**

Resumo **iii**

Abstract **iv**

Sumário **v**

Lista de Tabelas **viii**

Lista de Figuras **ix**

Abreviações **xii**

1 Introdução **1**

1.1 Definição do Problema 1

1.2 Objetivo 7

1.3 Justificativa 8

1.4 Proposta 9

1.5 Contribuição 11

1.6 Organização 12

2 Estado da Arte **14**

2.1 Pré-processamento 14

2.2 Segmentação 15

2.2.1 Chen e Wang [5] 16

2.2.2 Sadri *et al* [19] 18

2.2.3 Lei *et al* [14] 19

2.2.4 Fujisawa *et al* [8] 21

2.2.5 Fenrich e Krishnamorthy [7] 23
2.3 Filtragem 25
2.4 Resumo 27

3 Fundamentação Teórica **28**
3.1 Máquinas de Vetores de Suporte 28
3.1.1 Estimando probabilidades 31
3.2 *Receiving Operator Characteristics* 32
3.2.1 Desempenho 32
3.2.2 Algoritmo 36
3.2.3 Análise sensível a custo 38
3.3 Resumo 40

4 Método Proposto **41**
4.1 Arquitetura 41
4.2 Conjunto de características 43
4.3 Estratégia de verificação [16] 45
4.4 Bases de dados 49
4.5 Treinamento 51
4.6 Esquema de custo baseado em ROC 52
4.6.1 Esquemas de custo 53

5 Experimentos **55**
5.1 Processo de avaliação 55
5.1.1 Rotulação 55
5.2 Aspectos metodológicos 56
5.3 Desempenho para pontos de segmentação necessários 59

5.4 Desempenho para pontos de segmentação desnecessários - com conhecimento do tamanho da cadeia de caracteres 60
5.4.1 Resultados 61
5.5 Desempenho para pontos de segmentação desnecessários - sem conhecimento do tamanho da cadeia de caracteres 62
5.5.1 Análise de custo 63
5.5.2 Impacto no desempenho do classificador de dígitos isolados . . . 64
5.6 Impacto no custo computacional 68
5.6.1 Número de hipóteses 68
5.6.2 Número de chamadas ao classificador 69
5.6.3 *Total number of feature-value* 70
5.7 Análise de erro 71
5.8 Resumo 74

6 Conclusão **75**

Lista de Tabelas

1 Tipos de conexão entre dígitos numéricos [5]. 5
2 Algumas métricas utilizadas em problemas de duas classes [6]. 34
3 Esquemas de custo utilizados nos experimentos 63
4 Número de hipóteses 69
5 Número de chamadas ao classificador 70
6 *Total number of feature-values* global dos experimentos realizados 72

Lista de Figuras

1 Problemas mais comuns em manuscritos [11]. 3

2 Espaço das estratégias de segmentação [3]. 6

3 Hipóteses de segmentação geradas para uma amostra de dígitos conectados [17]. 7

4 Esquema proposto. 10

5 Diagrama de blocos do sistema proposto por Oliveira *et al* [16]. 11

6 Correção de inclinação que gerou erro em imagem conectada. 15

7 Remoção de ligadura. (a) Imagem contendo ligadura. (b) Detecção da ligadura. (c) Imagem após remoção da ligadura. 16

8 Estratégia para detecção de ruído [12]. 27

9 Hiperplano separando as classes -1 e 1. Os vetores de suporte estão com um círculo ao redor [2]. 29

10 Situações possíveis de classificação em um classificador binário. 33

11 ROC contendo cinco classificadores discretos [6] 35

12 Curva ROC típica [6] 35

13 Mecanismo de custo iso-performance [21] 39

14 Mecanismo de rejeição baseado em análise de curva ROC [21] 40

15 Distribuição dos pontos de segmentação para as classes de dígitos isolados 1, 4, 7 e 8 [16]. 42

16 Exemplo de sobre-segmentação do dígito “2”. (a) Imagem original. (b) Imagem sobre-segmentada. 42

17 Exemplo de ICL para um dígito [16]. 44

18 Exemplo de MCA [16]. 44

19 Interação entre verificadores e classificador de dígitos isolados. Os classificadores v_o e v_u são dois verificadores. O classificador e_{13} é um classificador de dígitos isolados [16]. 47
20 13 características de Freeman. (a) Vetor de características. (b) Concavidades. (c) Direções auxiliares. (d) 4-direções de Freeman. [16] 48
21 Esquema de zoneamento para extração de características de subsegmentação [16]. 49
22 Exemplo do dígito 2 "sobre-segmentado" onde o maior segmento pode ser facilmente confundido com um segmento obtido por um ponto de segmentação necessário. 51
23 Impacto da escolha de um ponto operacional nas taxas de TP, FP, TN e FN. 54
24 Análise ROC para rotulações utilizando outros limiares no classificador de dígitos isolados. (a) 75%. (b) 80%. (c) 85%. (d) 90%. (e) 95%. 58
25 Pontos ótimos classificados como desnecessários. 59
26 Pontos desnecessários classificados como necessários. 61
27 Impacto do método proposto em termos de taxa de erro *versus* taxa de rejeição. 62
28 Pontos operacionais para esquemas de custo propostos. 64
29 Taxas de reconhecimento de cadeias de caracteres sem o uso de filtro, com o uso de filtro (sem mecanismo de custo), com o uso do *low-level verifier* e com o uso do filtro+*low-level verifier*. 65
30 Amostra beneficiada pelo aumento no custo de falso negativo. 66
31 Taxas de reconhecimento de cadeias de caracteres utilizando o filtro proposto aliado aos esquemas de custo da Tabela 3. 67
32 Taxas de reconhecimento de cadeias de caracteres utilizando o filtro proposto + *low level* aliado aos esquemas de custo da Tabela 3. 67
33 Melhores resultados. 68

34 Erro de classificação para sem filtro e filtro (C9) e acerto para *low level* e filtro 0-rejeição. (a) Hipótese sub-ótima. (b) Hipótese ótima. (c) Hipótese desnecessária. 73
35 Erro na detecção de parte sobre-segmentada. (a) Hipótese ótima. (b) Hipótese desnecessária. 73
36 Hipótese ótima classificada como desnecessária, causando erro de classificação. (a) Hipótese ótima. (b) Hipótese desnecessária. 74

Abreviações

AUC	Area Under Curve
CFN	Cost of False Negative
CI	Contextual Information
CTN	Cost of True Negative
FN	False Negative
FP	False Positive
ICL	Initial Concavity Level
IP	Intersection Point
LL	Low Level
MCA	Multi-level Concavity Analisys
MCC	Multi-Class Classifier
MLP	Multi-Layer Perceptron
NIST	National Institute of Standards and Technology
OCC	One-Class Classifier
ROC	Receiving Operator Characteristics
SVM	Support Vector Machine
TI	Tecnologia da Informação
TN	True Negative
TP	True Positive
TVF	Total number of feature-value

1 Introdução

Reconhecimento de escrita manuscrita é uma área de pesquisa bastante ativa. Apesar da disseminação do uso de TI nos últimos 20 anos, o uso de documentos manuscritos é ainda imprescindível em algumas áreas. Além de ser um campo de estudo de alguns dos principais problemas na área de reconhecimento de padrões. Os avanços em termos de capacidade de processamento e de armazenamento tornaram viável a criação de soluções de reconhecimento automático de escrita manuscrita para as mais diversas finalidades - reconhecimento de cheques, reconhecimento de envelopes postais, indexação de documentos históricos, entre outras - de maneira cada vez mais condizente com a realidade da escrita humana (tolerante a variações de estilo, ruído, etc). Existem várias propostas em relação à estrutura de um sistema desse tipo na literatura, mas de uma forma geral, tais sistemas possuem quatro etapas principais: aquisição, pré-processamento, segmentação, representação e reconhecimento. Dessas etapas, segmentação é uma das mais complexas já que é necessário separar dígitos e outros componentes conectados entre si para que possam ser reconhecidos posteriormente. Porém, geralmente, não se conhece *a priori* a quantidade de dígitos que devem ser separados.

1.1 Definição do Problema

Segmentação é a etapa na qual componentes são separados para depois serem reconhecidos. Esta etapa depende bastante da etapa de pré-processamento que por sua vez influencia bastante a etapa de reconhecimento, uma vez que primitivas extraídas incorretamente podem não ser reconhecidas posteriormente ou pior, podem ser confundidas.
Em sistemas de reconhecimento de manuscritos, essa etapa costuma ser subdividida em três outras etapas: identificação de blocos de texto, identificação de componentes conectados e segmentação de dígitos conectados.

Na identificação dos blocos de texto, o fundo da imagem é separado do

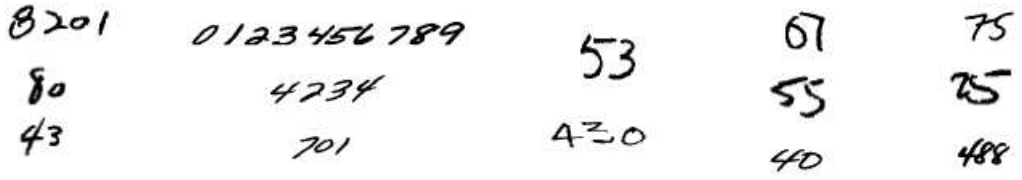
texto. Esse fundo pode conter informação ruidosa como textura, marca d'água, etc. Em seguida, esse texto é agrupado em blocos de acordo com a necessidade do problema a ser resolvido como parágrafos, linhas, etc. Feito isso, para cada um desses blocos é feita a separação dos componentes conectados. Um componente conectado é um conjunto de *pixels* $P[m, n] = \{p(x, y)^*\}$ no qual a partir de qualquer ponto $p(x, y)$ é possível chegar a qualquer outro ponto $q(x, y)$ percorrendo somente pontos de $P[m, n]$, isto é, se não existir pelo menos um caminho entre $q(x, y)$ e $p(x, y)$ composto somente de pontos $P[m, n]$, não podemos dizer que este ponto $q(x, y)$ pertence ao componente conectado $P[m, n]$.

Um classificador associa uma imagem I a uma classe $c_i \in C$ na qual C é o conjunto total de classes que um classificador pode reconhecer. Um classificador verifica qual a classe que mais se assemelha com aquela entrada (imagem). Essa associação pode ser feita de maneira discreta (simplesmente atribui-se uma classe àquela imagem) ou de maneira contínua, através de uma medida de similaridade, como distância até um hiperplano (SVM) ou ainda probabilidade condicional à classe, $P(x \mid \omega_i)$ que mede a probabilidade de um conjunto de medições de uma imagem (x) ter sido extraído de uma imagem da classe ω_i. Em um sistema de reconhecimento de cadeias numéricas, as classes costumam ser os próprios dígitos assim como outros caracteres úteis no reconhecimento como separadores, sinais, operadores numéricos entre outros.

Em ambientes controlados como escrita em tempo real (dispositivos móveis), formulários pré-formatados, a incidência de dígitos conectados costuma ser pequena, isto é, geralmente um componente conectado contém somente informação (imagem) referente a uma classe.

No entanto, em ambientes menos controlados, um componente conectado pode conter informações de mais de uma classe como dígitos conectados com outros dígitos ou com outros tipos de elementos como separadores ou até mesmo apenas parte de um elemento como um pedaço de um caracter quebrado. Na Figura 1 podemos ver alguns exemplos de problemas que acabam dificultando o reconhecimento de dígitos manuscritos e muitas

vezes quebrando essa relação de um-para-um entre primitivas e componentes conectados. Na Figura 1a podemos ver o problema da variação de tamanho. Esse tipo de problema



(a) (b) (c) (d) (e)

Figura 1: Problemas mais comuns em manuscritos [11].

não afeta muito a detecção de componentes conectados, mas afeta bastante a segmentação desse tipo de componente uma vez que a relação entre algumas propriedades estruturais de cada um dos dígitos isolados poderiam eventualmente fornecer boas informações a respeito de um ponto de segmentação como podemos ver em Chen e Wang [5]. No entanto este tipo de característica é bastante sensível a variações de escala. A Figura 1b por sua vez, vemos alguns exemplos de inclinação. Este tipo de problema também não afeta a detecção de componentes conectados mas tem um grande impacto em sistemas de segmentação implícita [11]. A Figura 1c demonstra casos de dígitos quebrados. Este tipo de problema afeta bastante a detecção de componentes conectados uma vez que partes de um mesmo dígito acabam ficando distribuídas entre vários componentes distintos. Uma maneira de mitigar problemas deste tipo é através do uso de métodos de pré-processamento após a detecção de componentes conectados como podemos ver em Oliveira [16]. A não correção deste problema irá afetar todas as demais etapas (segmentação de dígitos isolados, reconhecimento, etc). Finalmente, a Figura 1d apresenta amostras de sobreposição. Este tipo de problema também afeta pouco a detecção de componentes conectados mas assim como a inclinação, também tem um impacto grande em sistemas

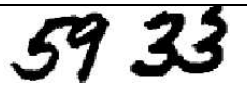
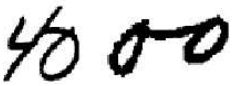
baseados em segmentação implícita. Por fim, podemos ver o próprio problema de dígitos conectados. O impacto na detecção de componentes conectados é que um componente conectado acaba contendo mais do que um dígito tornando necessário o uso de um método de segmentação específico para este tipo de problema.

Dessas três sub-etapas, identificação de blocos de texto, identificação de componentes conectados e segmentação de dígitos conectados, a última é uma das mais complexas. Daqui por diante iremos utilizar apenas a palavra segmentação ao invés de segmentação de dígitos conectados, uma vez que esse é o foco deste trabalho. Quando for necessário se referir a qualquer outro tipo de segmentação, isto será feito de maneira explícita.

Segmentação é um problema do tipo "ovo e galinha": é essencial que primitivas sejam separadas para posterior reconhecimento mas ao mesmo tempo essa separação exige algum reconhecimento. Devido a complexidade dessa tarefa, mesmo com todos os avanços em termos de processadores, esta etapa ainda é considerada um dos principais gargalos num sistema de reconhecimento de manuscritos [17]. Por esse motivo, segmentação é uma área de pesquisa bastante ativa, com diversas abordagens e propostas. Basicamente, existem dois tipos de conexões entre dígitos: simples (ocorre apenas em um ponto) e múltipla (ocorre em diversos pontos). Cada uma dessas conexões possui formas diversas como podemos ver na Tabela 1. Segundo Wang *et al* [24], 89% dessas conexões são conexões simples e 11% são conexões múltiplas. Segundo esse mesmo estudo, 85% das conexões ocorrem em cadeias contendo apenas dois caracteres.

Casey e Lecolinet [3] propõem uma taxonomia para os métodos de segmentação existentes. Segundo essa taxonomia, existem três estratégias principais de segmentação:

Tabela 1: Tipos de conexão entre dígitos numéricos [5].

Categoria	Tipo de Conexão	Exemplo
Conexão simples (89%)		
		
		
		
Conexão múltipla (11%)		

1. Dissecação: Métodos baseados nesta estratégia procuram informações estruturais que possam indicar a provável localização de uma conexão, como por exemplo, vales na projeção vertical dos *pixels* pretos da imagem. Esse tipo de estratégia acarreta em um número menor de hipóteses de segmentação (em relação à segmentação-reconhecimento). Mas por outro lado, faz uso intenso de heurísticas, o que acaba prejudicando a robustez do método, isto é, o método fica muitas vezes limitado a um conjunto de premissas determinadas para um dado problema.

2. Segmentação-reconhecimento: Esta estratégia consiste em utilizar um classificador para validar o segmento obtido. Os métodos que se concentram nesse tipo de estratégia costumam gerar inúmeras hipóteses de segmentação (sobre-segmentação), que são posteriormente submetidas a um classificador. Este classificador tentará associar cada um dos prováveis segmentos de cada uma dessas hipóteses a uma das classes do domínio do problema. A hipótese de segmentação que obtiver a probabilidade mais alta é escolhida.

3. Holística: Esta estratégia consiste em não segmentar componentes que contém mais do que um elemento. Ao invés de segmentação, opta-se pelo uso de premissas mais complexas como palavras inteiras. Essa estratégia é bastante utilizada em problemas envolvendo escrita cursiva. Uma das limitações desse tipo de estratégia é o número de premissas, que costuma ser alto.

Essas estratégias formam um espaço ortogonal de três dimensões (Figura 2) e os métodos de segmentação existentes encontram-se dispersos nesse espaço. Isto é, existe um espaço contínuo de estratégias de segmentação e não um conjunto discreto das mesmas.

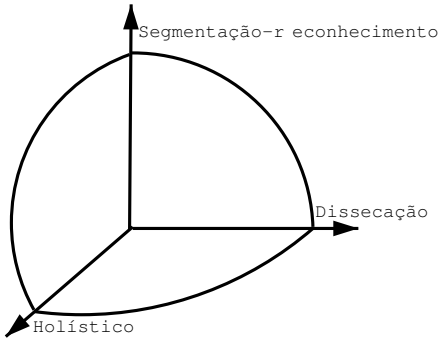


Figura 2: Espaço das estratégias de segmentação [3].

Existe ainda uma outra taxonomia, que divide as estratégias existentes em dois grupos - segmentação explícita e implícita. De forma geral, nos algoritmos baseados em segmentação explícita, um ponto de segmentação em potencial é definido por uma série de regras (vales, junções, etc), isto é, um ponto é intrinsecamente ligado a características da imagem, como contorno, fundo, entre outras. Já nos algoritmos baseados em segmentação implícita, a escolha de um ponto de segmentação é "arbitrária". A segmentação geralmente é feita por uma linha vertical, abrangendo todo o comprimento da imagem e nenhuma característica da imagem em si como contorno, fundo ou qualquer outra, é utilizada para definir o ponto onde será feita a segmentação.

Apesar dos métodos de segmentação baseados em segmentação-reconhecimento fazerem menor uso de heurísticas, quando comparados com métodos baseados em dissecação, eles geram muitas hipóteses de segmentação desnecessárias. Essas hipóteses além de causarem um aumento no custo computacional também levam a erros de classificação. A Figura 3 mostra um exemplo de dígitos conectados segmentados por um algoritmo de segmentação baseado em segmentação-reconhecimento [16]. Além do aumento no custo computacional, essas hipóteses levaram a um erro de classificação no classificador de dígitos isolados proposto por Oliveira *et al* [16], após avaliadas todas as hipóteses. Nesse caso uma amostra de "56" foi classificada como "510".

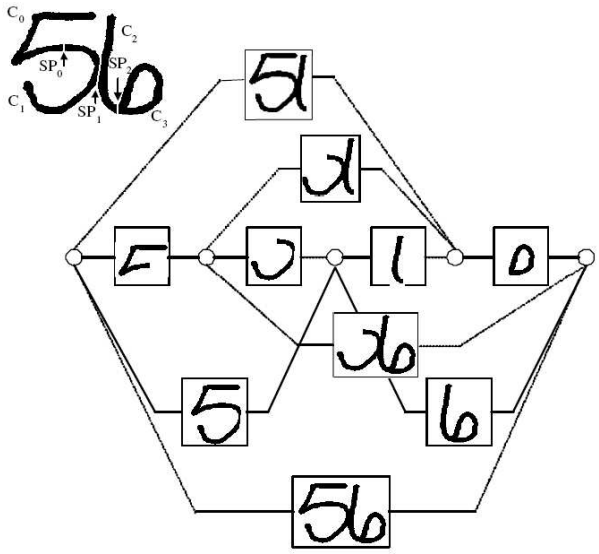


Figura 3: Hipóteses de segmentação geradas para uma amostra de dígitos conectados [17].

1.2 Objetivo

Neste trabalho propomos um método de classificação de pontos de segmentação para cadeias de dígitos manuscritos. O objetivo principal deste método é reduzir a quantidade de hipóteses de segmentação gerada pelos métodos de segmentação baseados

na estratégia segmentação-reconhecimento [3].

Através do uso desse método, espera-se obter uma redução sensível do custo computacional de sistemas baseados em segmentação-reconhecimento.

Um outro objetivo do método proposto é reduzir o uso de heurísticas nos métodos de segmentação (evitando o uso de estratégias baseadas em dissecação), mas ao mesmo tempo restringindo o número de hipóteses a serem submetidas ao classificador de dígitos isolados. Espera-se que o método proposto classifique pontos de segmentação de maneira adaptativa e através de características com alta discriminância, que permitam a sua aplicação em qualquer outro problema.

Por fim, dada a natureza do problema, classificar pontos de segmentação como necessários ou desnecessários antes de qualquer reconhecimento, espera-se que o método proposto possua tolerância a ruído.

1.3 Justificativa

A segmentação tem um papel importantíssimo em sistemas de reconhecimento de escrita cursiva. Um erro de segmentação geralmente acarreta em um erro de reconhecimento. E mesmo com os progressos em termos de processadores, essa etapa costuma ser um gargalo em muitos sistemas [17]. Principalmente no que tange reconhecer hipóteses de segmentação desnecessárias geradas por métodos baseados em segmentação-reconhecimento.

Por isso, um método que consiga reduzir o número de hipóteses desnecessárias, sem fazer uso de heurísticas que acabam acarretando em perda de flexibilidade, como ocorre nos métodos baseados em dissecação, é importante. Essa redução do número de hipóteses desnecessárias pode trazer dois tipos de benefícios - redução da complexidade do grafo de segmentação, levando a uma redução do custo computacional e melhora nas taxas de acerto pelo classificador de dígitos isolados, uma vez que hipóteses desnecessárias geral-

mente acabam se tornando ruído na etapa de reconhecimento. Em relação a redução do custo computacional, considerando que a quantidade de hipóteses de segmentação para uma amostra de dígitos conectados é 2^n onde n é o número de pontos de segmentação, eliminar um ponto de segmentação significa reduzir pela metade o número de hipóteses de segmentação. Considerando que uma hipótese que contenha somente um ponto de segmentação gera dois segmentos, que são individualmente submetidos a um classificador de dígitos isolados, podemos concluir que a redução na quantidade de chamadas ao classificador é ainda mais significativa do que a redução do número de hipóteses de segmentação.

O impacto do uso de heurísticas na flexibilidade de um método de segmentação pode ser visto em Oliveira *et al* [17], no qual o método de segmentação proposto por Pal *et al* [18] sofre uma redução drástica em seu desempenho quando testado em uma base sintética de dígitos numéricos conectados. Isto ocorre porque as premissas propostas para o problema original (reconhecimento de campo numérico em cheques bancários franceses) não são válidas para o novo problema (base sintética).

1.4 Proposta

A partir das duas restrições dos métodos de segmentação existentes: número de hipóteses de segmentação e uso de heurísticas, e dada a necessidade de aliar redução do número de hipóteses de segmentação à redução do uso de heurísticas, propomos a estratégia segmentação-reconhecimento e tentamos minimizar a sua desvantagem (excesso de hipóteses de segmentação). Isso é feito através de um filtro, colocado entre a etapa de segmentação e a etapa de reconhecimento (Figura 4).

Esse filtro é aplicado ao sistema proposto por Oliveira *et al* [16], no entanto dada a sua natureza, o mesmo pode ser aplicado a qualquer sistema baseado na estratégia segmentação-reconhecimento. A Figura 5 mostra de maneira geral como esse sistema modular está organizado. Podemos perceber que o filtro proposto é aplicado logo após o

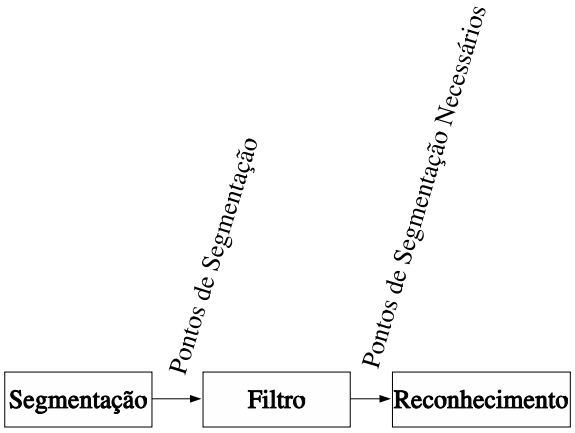


Figura 4: Esquema proposto.

processo de segmentação.

O método proposto encontra pontos de segmentação desnecessários e os elimina antes deles serem submetidos a etapa de reconhecimento. Esse método é capaz de encontrar pontos de segmentação desnecessários em cadeias de dígitos numéricos conectados, independente do tamanho ou do tipo de conexão entre cada par de dígitos. Por esse motivo foi necessário um estudo aprofundado da área de segmentação para que fosse possível encontrar características que aliassem alto poder de discriminância a uma alta adaptabilidade (independentes do tipo de conexão ou do número de dígitos).

O método proposto utiliza um classificador para reconhecer os pontos desnecessários de forma adaptativa. Dada a natureza do problema (classificar um ponto de segmentação como necessário ou desnecessário), o classificador escolhido foi um SVM, uma vez que o mesmo foi concebido para atuar em problemas de duas classes.

Por isso é necessário também um estudo criterioso do tipo de classificador, para que se avaliem as vantagens e as desvantagens do mesmo. Por se tratar de um problema de duas classes, é utilizada análise da curva ROC (*Receiving Operator Characteristics*) do classificador a fim de compreender de maneira ampla o comportamento do mesmo. Uma curva

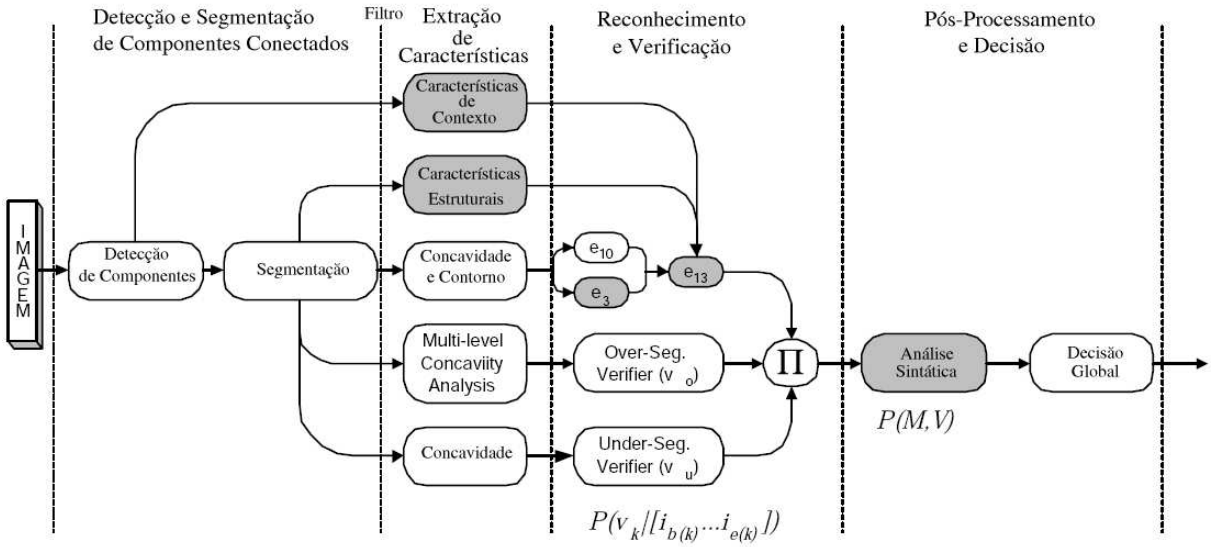


Figura 5: Diagrama de blocos do sistema proposto por Oliveira *et al* [16].

ROC mostra para um dado classificador de duas classes (não-discreto), a relação entre as taxas de falso positivo (FP) e verdadeiro positivo (TP) de todos os seus pontos operacionais. Um ponto operacional representa o comportamento de um classificador (em termos de TP e FP) para um dado limiar imposto ao *score* desse classificador.

1.5 Contribuição

A principal contribuição deste trabalho à área de reconhecimento de manuscritos é a criação de um método para classificar pontos de segmentação de dígitos manuscritos conectados. Uma vez encontrado um ponto desnecessário, o mesmo é rejeitado antes de ser avaliado por um classificador de dígitos isolados. Dessa maneira, torna-se possível uma redução sensível da complexidade do grafo de segmentação em sistemas de reconhecimento de dígitos manuscritos. E essa redução do número de hipóteses é conseguida de forma adaptativa, livre de heurísticas, o que torna possível aplicar o método proposto em

cadeias numéricas de qualquer tamanho e para os mais diversos tipos de conexões possíveis. Essa redução livre de heurísticas pode ser considerada como mais uma contribuição deste trabalho.

Sob um outro aspecto, foi possível estudar de uma forma prática e objetiva as Máquinas de Vetores de Suporte. Espera-se que esse estudo prático forneça para a comunidade científica um referencial sobre as vantagens e desvantagens desse tipo de classificador.

Por fim, faz-se uso de análise das características operacionais (ROC) dos classificadores utilizados. Esse ferramental é bastante utilizado pela comunidade científica (em áreas distintas como diagnósticos médicos, análise de sinais e até mesmo reconhecimento de padrões) mas pouco utilizado na análise de desempenho de sistemas de reconhecimento de manuscritos.

Os resultados preliminares desta pesquisa foram aceitos para publicação no 10^{th} *International Workshop on Frontiers in Handwriting Recognition* [23].

1.6 Organização

Esta dissertação está organizada em seis capítulos. Este capítulo apresenta uma breve apresentação do trabalho proposto. No Capítulo 2 apresentamos alguns métodos de segmentação encontrados na literatura, bem como diferentes estratégias utilizadas para melhorar o desempenho dos sistemas de reconhecimento de dígitos. Através deste estudo será possível compreender melhor os desafios envolvendo segmentação de dígitos conectados. No Capítulo 3 mostraremos a fundamentação teórica em relação a reconhecimento de padrões e principalmente em relação a SVM e ROC. No Capítulo 4 apresentaremos o método proposto bem como algumas alternativas ao mesmo. Também mostraremos as vantagens e desvantagens de cada uma das opções. No Capítulo 5 encontram-se os aspectos metodológicos, os resultados experimentais do método proposto e uma análise

dos erros ocorridos. O Capítulo 6 conclui o trabalho e apresenta algumas expectativas em relação a trabalhos futuros.

2 Estado da Arte

Neste capítulo apresentamos uma breve introdução sobre pré-processamento, apenas para fins de contextualização e uma apresentação mais detalhada sobre segmentação, onde serão analisadas as técnicas mais relevantes da atualidade.
Também apresentamos alguns fundamentos de filtragem, propostos por Landgrebe *et al* [12].

2.1 Pré-processamento

A etapa de pré-processamento é fundamental para o desempenho das etapas posteriores. Não há nenhum tipo de extração de informação simbólica da imagem. Uma imagem bruta é fornecida como entrada para esta etapa e uma imagem pré-processada é obtida como saída. Um dos objetivos desta etapa é eliminar ou pelo menos reduzir falhas ocorridas na aquisição ou na produção do documento como ruído, inclinação de linha de base e de caracter, iluminação inadequada, etc. Um outro objetivo é tornar a imagem adequada a um determinado método de segmentação, como por exemplo conversão entre sistemas de cores, binarização e realce de bordas.
Um dos maiores problemas em relação a pré-processamento é o seu impacto nas etapas posteriores. Muitas vezes algoritmos de pré-processamento acabam gerando erros nas etapas posteriores como segmentação e reconhecimento, por esse motivo, ultimamente a máxima "menos é mais" tem sido constantemente adotada. Não que isso implique em uma corrida para a abolição do uso de pré-processamento, já que existe uma dependência muito grande das outras etapas em relação a esta. Ao invés disso, o foco tem sido usar essas técnicas de maneira racional.
Um exemplo do impacto de um método de pré-processamento nas etapas posteriores pode ser visto na Figura 6. Neste caso, um algoritmo de correção de inclinação foi aplicado a uma amostra de uma cadeia de dois dígitos conectados (“69”). A correção aplicada acabou

transformando a imagem original numa cadeia que foi posteriormente reconhecida como sendo uma cadeia de três dígitos (“181”), com uma probabilidade alta, por um sistema baseado em segmentação implícita [11].

(a) (b)

Figura 6: Correção de inclinação que gerou erro em imagem conectada.

2.2 Segmentação

A literatura nos mostra algumas tentativas no sentido de reduzir o número de hipóteses de segmentação. Em Sadri *et al* [19], são avaliadas a sobreposição e a altura dos segmentos resultantes através do uso de limiares pré-definidos, eliminando as hipóteses que não satisfaçam tais limiares. Entretanto, o uso de limiares limita o método de classificação àquele problema. Chen e Wang [5] utilizam oito características estruturais de pontos de segmentação e dos próprios componentes para modelar pontos ótimos através do uso de uma mistura de Gaussianas. O uso de características estruturais não é desejado, uma vez que esse tipo de característica tem uma baixa discriminância.

Nesta sub-seção apresentamos uma série de algoritmos de segmentação, os quais nos dão uma boa idéia da quantidade de algoritmos disponíveis na literatura.

2.2.1 Chen e Wang [5]

Método

Este método baseia-se em características de contorno e de fundo. Um algoritmo de afinamento (Hilditch) é utilizado para encontrar esqueletos de fundo e contorno. Características de bifurcações, extremidades e arestas desses esqueletos são utilizadas para detectar prováveis pontos de segmentação. Um algoritmo procura relacionar pontos de fundo com pontos de contorno formando linhas de corte. Após criar as linhas de corte, um outro algoritmo procura remover ligaduras (Figura 7). Por fim, usa-se uma mistura de Gaussianas de tamanho M=20 para determinar qual é a melhor hipótese de segmentação.

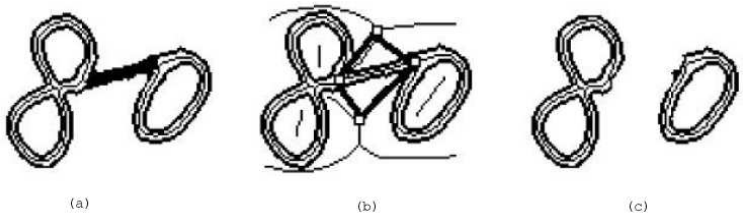


Figura 7: Remoção de ligadura. (a) Imagem contendo ligadura. (b) Detecção da ligadura. (c) Imagem após remoção da ligadura.

As seguintes características são utilizadas para treinar a mistura de Gaussianas e posteriormente reconhecer as hipóteses:

1. Relação entre as larguras dos dois segmentos.
2. Relação entre as alturas dos dois segmentos.
3. Relação entre as áreas (quantidade de píxeis pretos) dos dois segmentos.
4. Relação entre largura e altura de cada segmento (duas características, uma para cada segmento).

5. Relação entre distância horizontal da sobreposição entre os dois segmentos e a menor largura (largura do segmento mais estreito).

6. Distância normalizada entre o centro da linha de corte e o centro da imagem (par de segmentos).

7. Relação entre a área do caminho de segmentação e a altura da imagem.

A hipótese de segmentação com probabilidade mais alta é escolhida.

Resultados

Segundo os autores, a taxa de acerto obtida através do uso deste método é de 96% (do total de hipóteses não-rejeitadas) e a de erro de 4%. O total de hipóteses rejeitadas foi de 7,8%. Esses resultados foram obtidos em uma base de 4500 amostras (4178 amostras da base NIST SD19 e 322 de uma base criada pelos autores). Foram utilizadas 823 amostras para treinar as misturas de Gaussianas e 3677 amostras para testar o algoritmo. A base de teste (3677 amostras) possui 1,5% de casos de ligadura e 4,9% de casos de conexões múltiplas.

Considerações

A etapa de extração de características não faz uso de heurísticas, o que é algo desejável. No entanto, a etapa de construção das linhas de corte o faz. Cabe uma outra observação importante em relação ao uso de um método de afinamento. Apesar de esqueletos fornecerem informações importantes sobre a existência de um ponto de segmentação, os métodos de afinamento, amplamente utilizados para a extração de esqueletos, possuem um custo computacional elevado. Uma análise criteriosa da relação custo/benefício do uso deste tipo de método é imprescindível. De qualquer forma, algumas das características estruturais propostas podem eventualmente ser úteis no nosso método. Apesar do fato de características estruturais gerarem bastante confusão, combiná-las com características não-estruturais

de maneira complementar, com um tipo de característica compensando as deficiências do outro tipo, pode ser uma estratégia atraente. Uma outra idéia bastante interessante é a estratégia de identificar e remover ligaduras, uma vez que esse tipo de componente além de desnecessário, quando não removido acaba aumentando a variabilidade intra-classe, afetando de forma negativa o desempenho da etapa de reconhecimento.

2.2.2 Sadri *et al* [19]

Método

Este método baseia-se em características de contorno e de fundo e faz uso de operações de pré-processamento (suavização, correção de inclinação). Um método de afinamento é utilizado para se obter o esqueleto da imagem. Pontos de junção (*IP*, do inglês *Intersection Point*) desse esqueleto e informações do perfil da imagem são utilizados como características.

Quatro listas de pontos-chaves são obtidas - contorno superior/inferior e fundo superior/inferior. Heurísticas são utilizadas para associar esses pontos-chaves e formar as linhas de corte.

Um outro algoritmo, também baseado em heurísticas, avalia os caminhos de segmentação gerados. São verificadas restrições (limiares) referentes a sobreposição entre as imagens (*overlap*) e altura das mesmas.

Resultados

Segundo os autores, este método foi testado num conjunto de 835 imagens da base NIST SD19, todas contendo dígitos conectados. A avaliação baseou-se em análise visual e segundo o autor, foi possível obter sucesso em 95,3% dos casos, isto é, a melhor linha de corte (o método proposto gera mais do que uma) encontrada pelo método proposto é igual a linha de corte ótima. Além disso, em 88,6% dos casos o método proposto gera uma única linha de corte que é igual à ótima.

Considerações

O método proposto parece ser promissor. Considerando que o mecanismo utilizado para construir e avaliar os caminhos de segmentação depende bastante de heurísticas, podemos concluir que as características utilizadas para detectar os pontos-chaves são de fato robustas. No entanto, essas características aplicam-se a esqueletos e como o custo computacional dos métodos de afinamento (utilizados para extrair esses esqueletos) é alto, faz-se necessária uma análise criteriosa da relação custo/benefício do uso deste método em um dado problema ou sistema de reconhecimento de manuscritos.

2.2.3 Lei *et al* [14]

Método

Os autores propõem um método de segmentação de dígitos conectados baseado na estratégia segmentação-reconhecimento. Neste método, características do contorno e de projeção dos píxeis pretos são utilizadas para a identificação dos prováveis pontos de segmentação. São aplicados alguns métodos de pré-processamento (normalização de altura, operação morfológica de fechamento). A espessura média do contorno é utilizada para eliminar ruído (componentes conectados cuja espessura do contorno médio seja menor do que a média de toda a cadeia).

Primeiramente, é feita uma tentativa de classificar um componente conectado, utilizando um classificador de dígitos isolados (*nested-subset classifier*). Se o reconhecimento falha, um algoritmo de segmentação de dígitos conectados é então aplicado ao componente conectado. São utilizadas características do contorno externo e da projeção dos píxeis pretos para identificar prováveis pontos de segmentação.

A análise do contorno divide-se em duas fases - análise do contorno superior e análise do contorno inferior. Essa análise consiste basicamente em detectar bordas no

contorno externo. Heurísticas são utilizadas para escolher os prováveis pontos de segmentação entre todas as arestas identificadas.

Heurísticas também são utilizadas para identificar pontos nas projeções de píxeis pretos - todas as linhas que tiverem uma quantidade de píxeis pretos menor do que o dobro da espessura de contorno média, estimada anteriormente e que possuam uma linha vizinha com uma quantidade de píxeis pretos maior ou igual que o dobro da espessura de contorno estimada são consideradas como possíveis pontos de segmentação.

Heurísticas são utilizadas novamente para construir as linhas de corte, conectando os pontos-chaves identificados anteriormente.

Há um estágio de verificação. Neste estágio é utilizada a abordagem segmentação-reconhecimento. A imagem é dividida em vários fragmentos (de acordo com as linhas de corte identificadas anteriormente). Em seguida é utilizado um algoritmo de busca de *clique* em grafos para organizar esses fragmentos. Uma função de otimização tenta achar o caminho que maximize a probabilidade de reconhecimento do *clique*. A hipótese com a probabilidade mais alta é escolhida como sendo a hipótese ótima.

Resultados

Os autores testaram este método numa base contendo 3359 amostras de cadeias numéricas de dois dígitos (conectados). Essas amostras foram extraídas da base NIST SD19 (séries hsf_0, hsf_2, hsf_3, hsf_4, hsf_6 e hsf_7). Foram utilizadas 500 amostras para estimar os parâmetros do modelo (treinamento). Segundo os autores, a taxa de acerto do método proposto foi de 97,7% com 0% de taxa de rejeição. Os autores também avaliaram esse método em uma base contendo 525 amostras de cadeias numéricas de três dígitos (conectados). Segundo os autores, nesse teste foi possível segmentar com sucesso 93,3% dos casos.

Considerações

Os resultados obtidos são excelentes. Além disso, o método proposto não faz uso de um algoritmo de afinamento. No entanto heurísticas são usadas em algumas situações - para determinar as arestas, para encontrar os pontos-chaves de contorno, para encontrar os pontos-chaves de projeção dos píxeis pretos, para conectar os pontos encontrados, formando as linhas de corte. Mas dado o desempenho do método proposto, talvez reduzir o uso de heurísticas e aplicar um método de classificação inteligente de pontos (como o proposto no nosso trabalho) pode ser uma estratégia promissora.

2.2.4 Fujisawa *et al* [8]

Método

Os autores propõem um método de segmentação baseado na estratégia segmentação-reconhecimento. Este método utiliza características de contorno. Esses contornos são classificados segundo uma "medida de anormalidade" (*measure of abnormality*). Segundo os autores, essa medida de anormalidade mostra o quanto um componente conectado ultrapassa a área estimada de um dígito isolado (*estimated field area*). É feita uma classificação do componente conectado como dígito isolado ou dígito conectado com base em dois limiares, calculados a partir da largura média dos dígitos isolados. O uso de dois limiares permite que se atribua um nível de conectividade ao mesmo.

O algoritmo de construção das linhas de corte é dividido em dois. Um para construir linhas de cortes para casos onde dois *loops* estão conectados e um outro para todos os demais casos. Este último algoritmo é executado por primeiro. Inicialmente, o contorno é dividido em dois - superior e inferior. Em seguida, é aplicada uma operação chamada *vertical marginal operation* que basicamente consiste em varrer cada um dos dois contornos (superior e inferior) individualmente, da esquerda para a direita, procurando as colunas mais externas (duas) no caso do contorno inferior e mais internas (duas) no caso do contorno superior onde existam mais do que dois pontos pretos na mesma. Em seguida,

é feita uma busca pelo ponto candidato em cada um dos contornos. Essa busca se dá no espaço entre os dois pontos encontrados anteriormente pela *vertical marginal operation*, através do uso de heurísticas. Após isso, o ponto encontrado é deslocado de maneira que fique próximo do ponto onde o contorno do componente conectado se expande bruscamente (local onde há uma grande variação na espessura do contorno, no sentido vertical).

A segunda etapa consiste em segmentar *loops* conectados. Este algoritmo primeiramente separa os *loops* em dois grupos - *loops* do componente esquerdo e *loops* do componente direito. Se houver sobreposição (a sobreposição é calculada com base num limiar), o algoritmo para e indica que houve um erro. Caso contrário, é feita uma busca por pontos de mínima e de máxima em cada um dos *loops*. Em seguida, é feita uma busca por pontos correspondentes a esses pontos de mínima e de máxima, mas no contorno externo do componente. Depois, é feita uma busca pelo ponto mais baixo do contorno superior e pelo ponto mais alto do contorno inferior, no espaço entre os dois pontos localizados anteriormente. Então é traçada uma reta conectando esses pontos, que será a linha de corte.

Além desses dois algoritmos, os autores apresentam um outro algoritmo, útil para conectar partes de dígitos quebrados. Esse algoritmo, basicamente verifica se um componente é um pedaço de um dígito quebrado ou não. Então, uma estratégia de segmentação-reconhecimento é aplicada para tentar reunir componentes quebrados. São feitas várias tentativas de união entre pedaços de componentes quebrados e outros componentes. A hipótese que obtiver a probabilidade mais alta é escolhida.

Resultados

Os autores criaram uma base sintética, consistindo de 46 combinações de dígitos conectados que segundo eles aparecem com mais frequência. Para cada uma dessas combinações foram geradas 20 amostras. Segundo os autores, 95% dessas amostras foram corretamente

segmentadas e consequentemente, reconhecidas corretamente. As demais foram rejeitadas.

Considerações

Os autores mencionam algumas das limitações do método proposto. Uma delas refere-se a erros para estimar a espessura vertical do contorno. Uma outra limitação é o fato do método proposto não segmentar dígitos com ocorrências de conexões múltiplas (funciona somente para conexões simples). Uma vantagem do método proposto é o fato do mesmo não fazer uso de um algoritmo de afinamento.

2.2.5 Fenrich e Krishnamorthy [7]

Método

A proposta dos autores é a de criar um método para aplicações de tempo real, útil para segmentar código postal em envelopes postais americanos. Inicialmente é feita a detecção de componentes conectados. Para isto, este método faz uma divisão recursiva, *top-down*, até que as linhas em cada um dos subconjuntos sejam contíguas. Segundo os autores, esta abordagem possui um custo computacional baixo.

Em seguida, o algoritmo procura por separadores (traço). Esta estapa está diretamente ligada ao domínio do problema e tem como propósitos auxiliar a identificação da quantidade de dígitos (cinco ou nove) e eliminar separadores, que prejudicariam o reconhecimento dos dígitos isolados. A existência de um traço indica que a cadeia possui nove dígitos. Em seguida, é aplicado um método de agrupamento, que tem como objetivo agrupar pedaços de dígitos quebrados aos seus respectivos dígitos. O método de agrupamento utilizado é baseado num conceito chamado proximidade hierárquica. Este método consiste basicamente em definir uma série de restrições de proximidade (descobertas empiricamente) que são depois utilizadas para agrupar exemplos que não satisfaçam

a uma delas. Primeiramente os componentes são classificados como dígitos isolados ou dígitos conectados. Essa classificação é feita através do produto da largura normalizada do componente pela área mesmo. Em seguida é feita a segmentação dos dígitos conectados propriamente dita. Esse método é sub-dividido em quatro outros métodos, que são aplicados ao componente conectado de maneira sequencial de acordo com a frequência e confiabilidade de cada método. O processo todo é iterativo, define-se uma vizinhança inicial. Então o método é aplicado àquela vizinhança inicial. Caso nenhum dos quatro métodos obtenha sucesso, aumenta-se o tamanho da vizinhança e o processo todo é repetido. Primeiramente é aplicado um método baseado na análise da projeção vertical dos *pixels* pretos. Basicamente, vales nessas projeções tornam-se candidatos a linhas de corte. Se esse vale é maior do que a espessura média do traço e corta a imagem em mais do que dois ou três pontos, este método é abortado e o próximo é executado. O próximo método, chamado *upper-lower*, procura por vales no contorno superior e picos no contorno inferior. Se o vale e o pico encontrados estão a um distância entre si menor do que um determinado limiar, uma linha de corte é gerada entre eles.

Caso contrário é feita uma outra tentativa de segmentação através de um terceiro método, chamado *upper-contour* que é similar ao *upper-lower*, mas aplica-se somente ao contorno superior. Este método procura por um vale no contorno superior e tenta gerar uma linha de corte desse vale até o limite inferior da imagem. Para isso, o método vai procurando, a partir do vale encontrado, de cima para baixo, píxeis pretos que possuam tanto à sua esquerda quanto à sua direita somente píxeis pretos ou somente píxeis brancos. Esses píxeis são adicionados a lista de píxeis da linha de corte.

Caso este método falhe, um quarto método, chamado *lower-contour*, é invocado. Este método funciona da mesma maneira que o anterior (*upper-contour*) mas inicia-se no pico do contorno inferior e com direção ao topo da imagem.

Por fim é aplicado um método de remoção de partes supérfluas.

Resultados

Os autores utilizaram uma base contendo 450 amostras de códigos postais retirados de envelopes postais americanos para fins de treinamento (ajustar limiares, modificar o método para situações não previstas inicialmente). Em seguida, foram utilizadas 928 amostras (também de códigos postais retirados de envelopes postais americanos) para testar o método proposto. Segundo os autores, o método proposto segmentou corretamente as amostras em 93,6% dos casos.

Considerações

Dada a natureza do problema, é possível saber com antecedência o tamanho da cadeia de caracteres. Um aspecto interessante do método proposto é o foco em redução de custo computacional. O algoritmo para agrupamento de dígitos quebrados aparentemente depende bastante de heurísticas. O método em geral faz bastante uso de heurísticas. Podemos observar isso na própria etapa de treinamento, onde os autores dizem que o propósito do treinamento é encontrar limiares que maximizem a taxa de acerto do método proposto.

2.3 Filtragem

Apesar dos avanços em termos de classificadores, informação ruidosa ainda é uma fonte de problemas. Boa parte dos problemas relacionados a informação ruidosa se devem ao fato de muitas vezes, a amostragem de ruído ser insuficiente ou até mesmo inexistente. Isso leva muitas vezes a erros de classificação, já que dada a sua imprevisibilidade, ruídos acabam sendo confundidos com classes do domínio do problema. Um outro problema é que mesmo quando um classificador não confunde o ruído com uma classe existente, tempo de processamento acaba sendo perdido.

Em relação a reconhecimento de dígitos manuscritos, hipóteses de segmentação desnecessárias podem ser consideradas informação ruidosa. Os resultados experimen-

tais do método proposto irão mostrar que de fato essas hipóteses além de causar aumento no custo computacional, também acaba levando a erros de classificação pelo classificador de dígitos isolados.

Há na literatura algumas propostas destinadas principalmente a reduzir o impacto negativo de informações ruidosas nas taxas de reconhecimento. Em Oliveira *et al* [16], é utilizada uma estratégia de verificação. Nessa estratégia, são adicionados mais classificadores que irão reforçar ou rebaixar a opinião do classificador principal.
Uma outra estratégia é a de classificação em cascata. Nunes *et al* [15] propõem um mecanismo de seleção de caracteristicas. Em seguida, os classificadores obtidos através desse método são organizados em cascata, isto é, os classificadores são organizados em série. Cada classificador possui um nível diferente de rejeição. Caso a decisão de um classificador seja rejeitar a amostra, a mesma é submetida a um outro classificador, e assim por diante. O conceito de classificador em cascata pode ser visto com mais detalhes em [10] e [9].

Landgrebe *et al* [12] propõem uma outra estratégia para detectar ruído (*outliers*) em situações onde há ruído variável ou baixa amostragem de ruído. Essa estratégia consiste em usar um *One-Class Classifier* (D_{OCC}) [20] para identificar o que faz parte do domínio do problema. O que é classificado corretamente é então submetido a um outro classificador (D_{MCC}) que irá classificar a amostra de maneira mais específica (Figura 8). Os autores apresentam essa estratégia como uma opção ao uso de rejeição (*reject-option*), já que segundo os autores há uma limitação no uso de rejeição uma vez que um classificador com um bom desempenho de classificação não irá necessariamente ter um bom desempenho de rejeição. Através dessa estratégia torna-se possível adequar cada um dos classificadores às particularidades de cada uma das tarefas (rejeição, que passa a ser encarada como detecção de ruído e classificação).

A principal vantagem dessa estratégia é que o filtro é posicionado antes do classificador principal (ao contrário do verificador que irá reforçar ou rebaixar a opinião

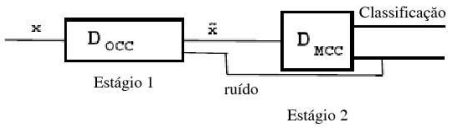


Figura 8: Estratégia para detecção de ruído [12].

do classificador principal). Com isso, em caso de ruído, são evitadas chamadas ao classificador principal (o que não ocorre quando se usa um verificador).

2.4 Resumo

Este capítulo apresentou uma revisão de alguns métodos de segmentação existentes na literatura, bem como de algumas técnicas relacionadas ao método proposto (pré-processamento, filtragem). Foi possível observar exemplos práticos de alguns problemas comuns à segmentação de dígitos manuscritos, a serem superados pelo método proposto. No capítulo seguinte, será abordado um outro aspecto importante a ser compreendido: a fundamentação teórica do método proposto (classificadores, ferramentas de avaliação).

3 Fundamentação Teórica

Neste capítulo apresentamos alguns dos conceitos teóricos referentes a reconhecimento de padrões úteis para a melhor compreensão do método proposto. Inicialmente apresentamos a fundamentação teórica das Máquinas de Vetores de Suporte, proposta por Vapnik [22]. Por fim apresentaremos o conceito de ROC (*Receiving Operator Characteristics*).

3.1 Máquinas de Vetores de Suporte

Máquinas de Vetores de Suporte representam uma nova abordagem em classificação. O princípio básico é, dada uma base de treinamento $\{x_i, y_i\}, i = 1, ..., l, y_i \in \{-1, 1\}, x_i \in \Re^d$, (onde x_i representa um vetor e y_i representa um rótulo para um dado vetor), identificar elementos (vetores) que representem aquela base. Em um problema de classificação binária isto significa identificar os vetores que definam o hiperplano que separa as duas classes. O algoritmo busca encontrar o hiperplano mais "espesso" possível. A Figura 9 mostra esses vetores para o caso linear com $\{x, y\} \in \Re^2$. Podemos perceber nesta figura o hiperplano separando amostras de duas classes diferentes, sendo as margens desse desse hiperplano definidas por alguns vetores. Esses são os vetores de suporte.

Esse hiperplano que separa as duas classes é definido pela equação $x \cdot w + b = 0$. Essa equação pode ser entendida como a equação de uma reta na forma $z = a \cdot x + b$. Todo vetor x que faça parte desse hiperplano deve satisfazer essa equação, onde w e b correspondem aos parâmetros a (inclinação) e b (deslocamento) da reta. Segundo este princípio, a base de treinamento é dividida da seguinte forma:

$$x_i \cdot w + b \geq +1 \; para \; y_i = +1 \tag{1}$$

$$x_i \cdot w + b \leq -1 \; para \; y_i = -1 \tag{2}$$

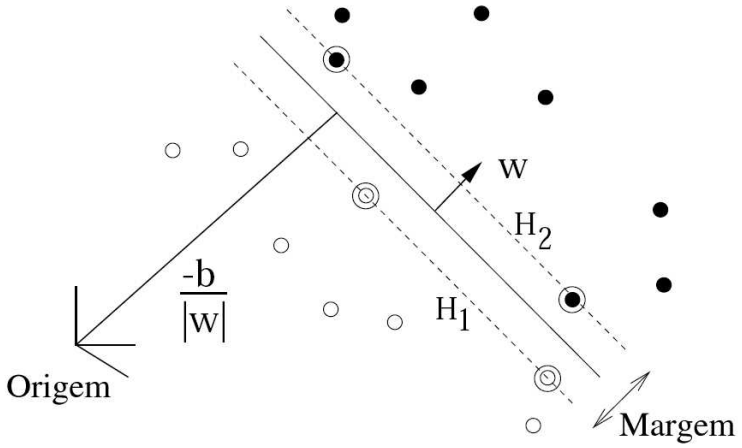


Figura 9: Hiperplano separando as classes -1 e 1. Os vetores de suporte estão com um círculo ao redor [2].

que podem ser combinadas na inequação:

$$y_i(x_i \cdot w + b) - 1 \geq 0, \ \forall i \tag{3}$$

Nas aplicações práticas, é necessário adicionar uma certa tolerância, isto é, o hiperplano precisa aceitar alguns "desvios". Para isso é adicionado um erro ξ_i.

$$x_i \cdot w + b \geq +1 - \xi_i, \ para \ y_i = +1 \tag{4}$$

$$x_i \cdot w + b \leq -1 + \xi_i, \ para \ y_i = -1 \tag{5}$$

Esse hiperplano pode ser encontrado minimizando $\| \ w \ \|^2$ sujeito à restrição da Equação 3. Este é um problema de otimização. Para isso é utilizada a otimização de Lagrange. São adicionados coeficientes de Lagrange positivos $\alpha_i, i = 1, ..., l$, para cada uma das restrições da Equação 3. Esses coeficientes são multiplicados pela restrição (3) e o resultado dessa mutiplicação é subtraído da função objetivo ($\| \ w \ \|^2$), resultando no

Lagrangiano:

$$L_P \equiv \frac{1}{2} \parallel w \parallel^2 - \sum_i \alpha_i y_i (x_i \cdot w + b) + \sum_i \alpha_i \tag{6}$$

que deve ser minimizado em relação a w e b. Além disso, a derivada parcial de L_P em relação a α_i deve tender a zero, sujeito a $\alpha_i \geq 0$ (Restrição 1). Esse problema também pode ser visto como maximizar L_P sujeito à restrição de que o gradiente de L_P em relação a w e b tenda a zero, também sujeito a $\alpha_i \geq 0$ (Restrição 2). Esse é um caso de formulação dupla para um problema, isto é, o mínimo de L_P para a Restrição 1, ocorre nos mesmos valores de w, b e α que o máximo de L_P para a Restrição 2. Quando os gradientes de L_P em relação a w e b tendem a zero, obtemos:

$$w = \sum_i \alpha_i y_i x_i \tag{7}$$

$$\sum_i \alpha_i y_i = 0 \tag{8}$$

podemos substituir essas equações na Equação 6, obtendo:

$$L_D \equiv \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i \cdot x_j \tag{9}$$

que deve ser maximizada, sujeita a:

$$0 \leq \alpha_i \leq C_i \tag{10}$$

$$\sum_i \alpha_i y_i = 0 \tag{11}$$

na qual C_i é a tolerância a erros do hiperplano. A solução (vetores que definem esse hiperplano) é:

$$\sum_{i=1}^{Ns} \alpha_i y_i x_i \tag{12}$$

na qual Ns é a quantidade total de vetores de suporte encontrados. Isto funciona para dados linearmente-separáveis. Para dados não linearmente-separáveis é necessário fazer um mapeamento da dimensão $\Re^d$ para uma dimensão mais alta (H):

$$\Phi : \Re^d \mapsto H \tag{13}$$

Uma função $K(x_i, x_j) = \Phi(x_i)\Phi(x_j)$ pode ser utilizada para fazer esse mapeamento. Para verificarmos em qual lado do hiperplano um vetor x está, basta utilizarmos a equação abaixo:

$$\sum_{i=1}^{Ns} \alpha_i y_i K(s_i, x) + b \tag{14}$$

na qual $K(s_i, x)$ é qualquer função que satisfaça a condição de Mercer, s_i é um vetor de suporte e x é um vetor de teste (alguma amostra que estamos tentando classificar). Essa função $K(s_i, x)$ é conhecida como função de *Kernel*.

Os $K(x, y)$ mais conhecidos são:

$$K(x, y) = (x \cdot y + 1)^p \tag{15}$$

conhecido como *Kernel* polinomial, no qual p é o grau do polinômio.

$$K(x, y) = e^{-||x-y||^2/2\sigma^2} \tag{16}$$

conhecido como *Kernel* Gaussiano, no qual σ é o desvio padrão.

$$K(x, y) = tanh(\kappa x \cdot y - \delta) \tag{17}$$

conhecido como *Kernel* Sigmóide, no qual κ pode ser entendido como um fator (*scale*) e δ o deslocamento desejado.

3.1.1 Estimando probabilidades

Um dos problemas em relação a SVM é que os mesmos não utilizam um modelo probabilístico. No entanto, existem vários métodos na literatura, destinados a converter a saída descalibrada de um classificador SVM (que na verdade representa a distância de

um vetor até o hiperplano que separa as duas classes) em uma probabilidade. No método proposto foi utilizada a biblioteca LIBSVM [4]. O LIBSVM utiliza o método proposto por Platt [1] para converter essa saída em uma probabilidade. Neste método, dados empíricos são utilizados para estimar uma função sigmóide. Essa sigmóide é então utilizada para mapear as saídas descalibradas do SVM em probabilidades, retornando um *score* entre 0 e 1 para cada resultado do classificador.

3.2 *Receiving Operator Characteristics*

O uso de curva ROC para análise de taxas de acertos e taxas de falsos positivos tem origem na teoria de sinais. Foi difundida entre a comunidade médica, na área de sistemas de diagnósticos e acabou atraindo a atenção da comunidade de aprendizagem de máquina, como forma de avaliar algoritmos e classificadores, principalmente para problemas de classificação envolvendo duas classes.

3.2.1 Desempenho

Em um problema de duas classes, pode-se dizer que uma das classes é a que temos interesse, e por isso a chamamos de positiva e a outra é indesejável e por isso a chamamos de negativa. Durante a classificação, o classificador pode gerar duas respostas possíveis (positiva e negativa) para cada uma das classes (também positiva e negativa), gerando quatro situações distintas de classificação:

1. Verdadeiro Positivo (TP), quando uma amostra positiva é corretamente classificada como positiva.

2. Falso Positivo (FP), quando uma amostra negativa é erroneamente classificada como positiva.

3. Verdadeiro Negativo (TN), quando uma amostra negativa é corretamente classificada como negativa.

4. Falso Negativo (FN), quando uma amostra positiva é erroneamente classificada como negativa.

A Figura 10 mostra essas quatro situações possíveis.

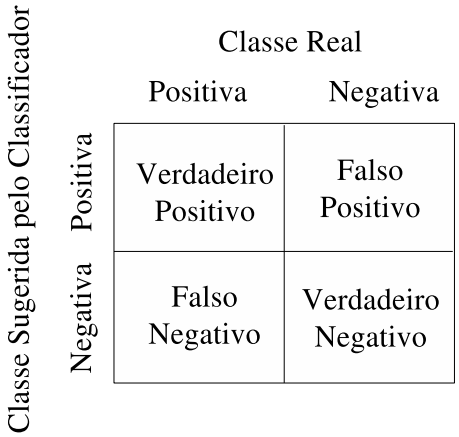


Figura 10: Situações possíveis em um classificador binário.

É possível extrair algumas métricas dos totais de cada uma das situações acima. Essas métricas podem ser vistas na Tabela 2, na qual TP, FP, TN, P e N são os totais de verdadeiros positivos, falsos positivos, verdadeiros negativos, positivos e negativos, respectivamente.

A característica operacional (ROC) de um classificador é definida como a sua capacidade de classificar corretamente em relação aos seus erros. No caso, é avaliada a taxa de verdadeiros positivos *versus* a taxa de falsos positivos. Uma gráfico de ROC típico possui duas dimensões. O eixo X representa as taxas de falsos positivos e o eixo Y representa as taxas de verdadeiros positivos.

Um classificador pode associar uma amostra a uma classe de duas maneiras - de maneira

Tabela 2: Algumas métricas utilizadas em problemas de duas classes [6].

Nome	Métrica
Taxa de verdadeiros positivos (*Recall*)	$\frac{TP}{P}$
Taxa de falsos positivos	$\frac{FP}{P}$
Precision	$\frac{TP}{TP+FP}$
Accuracy	$\frac{TP+TN}{P+N}$
F-measure	$\frac{2}{1/precision+1/recall}$

discreta (simplesmente diz o rótulo de uma amostra) ou através de um *score* que pode ser calibrado (por exemplo, probabilidade) ou não (medida de distância como no SVM). No caso de um classificador discreto, o seu comportamento é representado por um ponto no gráfico ROC como podemos ver na Figura 11. Neste caso, como o mesmo não emite um *score* durante a classificação, a função de decisão também é discreta. Ou seja, não é possível variar a função de decisão para verificar outros possíveis resultados de FP e TP. Por isso os desempenhos dos classificadores A, B, C, D e E nessa figura são representados por pontos.

Já para um classificador baseado em um *score*, a função de decisão irá resultar em um limiar. Todas as amostras com *score* acima do mesmo são classificadas como positivas e no caso contrário como negativas. Neste caso, existem inúmeros limiares possíveis e o gráfico ROC será uma curva, com as taxas para todos os limiares possíveis, de $-\infty$ até $+\infty$. Uma curva típica pode ser vista na Figura 12.

Quanto mais próxima a curva estiver do canto superior esquerdo do gráfico, melhor é o desempenho do classificador. No pior caso, a curva se encontrará sobre a diagonal, o que significa que o desempenho é o mesmo que atribuir rótulos aleatoriamente. Uma curva ótima irá tocar o canto superior esquerdo do gráfico. Sabendo disso, podemos

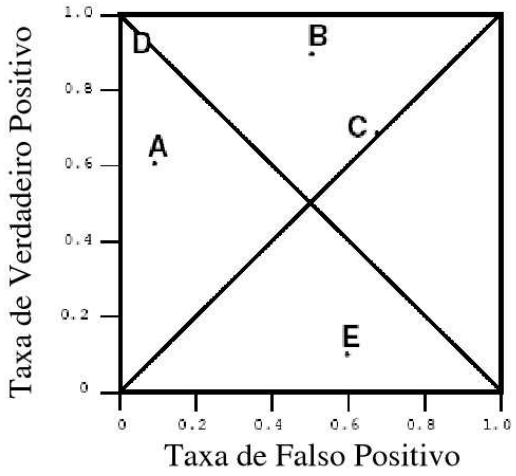


Figura 11: ROC contendo cinco classificadores discretos [6]

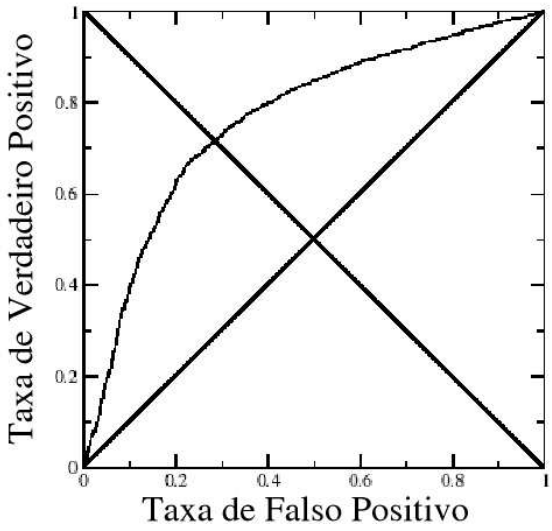


Figura 12: Curva ROC típica [6]

utiizar a área sob a curva (AUC, do inglês *Area Under Curve*) como uma métrica para comparar curvas ROC. Uma curva ótima terá um AUC igual a 1. Já para o desempenho aleatório (pior caso), como a curva toca a diagonal principal, a AUC da mesma será igual a 0,5.

Uma das vantagens da análise de curva ROC é que através dela é possível compreender de maneira ampla o comportamento de um classificador. Além disso, a curva ROC não é sensível a distribuição das probabilidades *a priori* das classes.

3.2.2 Algoritmo

A análise da curva ROC parte do princípio que um classificador gera um *score* para uma determinada instância e esse *score* irá determinar o grau de associação entre essa instância e uma classe [6]. A curva ROC mostra então os efeitos de se estabelecer limiares, classificando os elementos maiores do que um dado limiar como positivos e os menores como negativos.

Existe um algoritmo de geração de curva ROC baseado em força bruta, que basicamente considera cada um dos *scores* das amostras de teste como prováveis limiares e em seguida contabiliza a quantidade de FPs e TPs para cada um desses limiares.

No entanto, existe um fato que torna possível implementar um algoritmo com um menor custo computacional. Se ordenarmos todos esses *scores* e começarmos a fazer a contagem a partir do maior é óbvio que os elementos já contabilizados como positivos (independente de falso ou verdadeiro) serão contabilizados novamente nos próximos limiares. Dada essa constatação, é possível gerar uma curva ROC simplesmente ordenando as amostras pelos seus *scores* e partindo da amostra de maior *score* (até a de menor), avaliar se a mesma é um FP ou um TP e incrementar o seu respectivo contador (FP ou TP). Podemos ver esse algoritmo com mais detalhes no Algoritmo 1.

Algoritmo 1 Método computacionalmente eficiente para geração de curva ROC [6].

Entradas: L, conjunto de amostras de teste; $f(i)$, *score* da amostra i; P e N, totais de amostras positivas e negativas, respectivamente.
Saídas: R, lista de pontos da curva ROC.
Restrição: $P > 0$ e $N > 0$.

{Inicialização}
$L_{ordenado} \leftarrow L$ ordenado pelos *scores* f (decrescente)
$FP \leftarrow TP \leftarrow 0$
$R \leftarrow \langle\rangle$
$f_{ant} \leftarrow -\infty$
$i \leftarrow 1$
{Contabilização das amostras}
enquanto $i \leq \mid L_{ordenado} \mid$ **faça**
  se $f(i) \neq f_{prev}$ **então**
    adiciona $\left(\frac{FP}{N}, \frac{TP}{P}\right)$ a R
    $f_{prev} \leftarrow f(i)$
  fim se
  se $L_{ordenado[i]}$ é uma amostra positiva **então**
    $TP \leftarrow TP + 1$
  senão
    $FP \leftarrow FP + 1$
  fim se
  $i \leftarrow i + 1$
fim enquanto
adiciona $\left(\frac{FP}{N}, \frac{TP}{P}\right)$ a R

3.2.3 Análise sensível a custo

O objetivo de um sistema de reconhecimento de padrões é minimizar um risco (aqui chamado de custo). Muitas vezes, altas taxas de acerto acabam por não trazer benefícios práticos já que nem sempre o benefício trazido por um acerto é diretamente proporcional ao custo que um erro acarreta. Podemos usar como ilustração um sistema de reconhecimento de cheques. Alguém com experiência poderia dizer que o benefício obtido por substituir o processamento manual de cheques por um sistema computadorizado é de R$0,50 por folha de cheque. Por outro lado, um erro de reconhecimento poderia causar prejuízos diretamente proporcionais ao valor reconhecido. Por exemplo, em uma situação extrema, um cheque de R$1.000,00 reconhecido como R$100,00 poderia causar a um banco, um prejuízo de R$900,00, ou seja, 1.800 vezes maior do que o benefício de R$0,50 trazido pelo processamento automático do mesmo.

Esse equilíbrio entre benefício obtido por um acerto e custo gerado por um erro é fundamental e a análise da curva ROC de um classificador é um instrumento bastante utilizado para este fim [21].

Como vimos anteriormente, uma curva ROC ótima tende a ficar próxima do canto superior esquerdo do gráfico. Como cada ponto dessa curva corresponde a um dado limiar (de forma mais genérica a um classificador), podemos concluir que o melhor desempenho é aquele que se localiza no ponto ao mesmo tempo mais à esquerda e mais acima da curva. Uma forma de encontrar esse ponto é utilizar uma reta, de 45° de inclinação e procurar o ponto no qual essa reta toca a curva (Figura 13). Nesse esquema, estamos considerando que ocorrências de TP e FP possuem o mesmo custo (iso-performance).

Em algumas situações é desejável que os custos de FP e TP sejam diferentes. Ao fazermos isto, estamos mudando a inclinação dessa reta. Tortorella [21] propõe um mecanismo de rejeição baseado nesse princípio. Ao invés de simplesmente procurar o ponto ótimo da curva para um dado mecanismo de custo, Tortorella procura por um in-

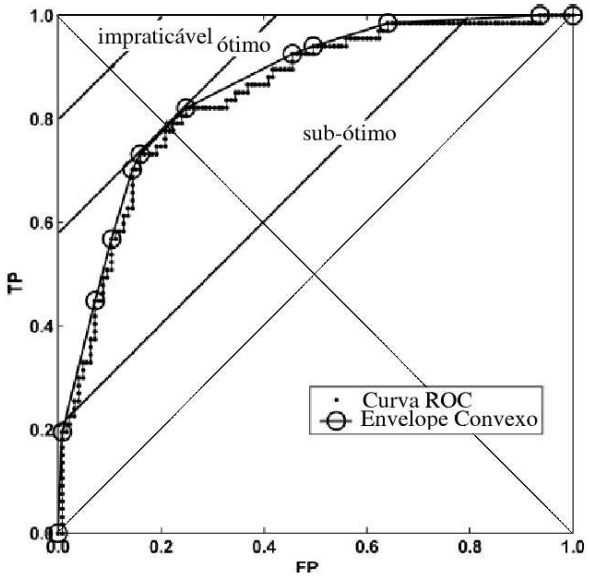


Figura 13: Mecanismo de custo iso-performance [21]

tervalo (dois limiares). Todas as amostras dentro desse intervalo são então rejeitadas (o objetivo é estabelecer uma distância segura entre as duas distribuições, de maneira a minimizar o erro causado pela sobreposição entre elas). Podemos ver o esquema proposto por Tortorella na Figura 14. Nessa figura, podemos observar que duas retas tocam a curva ROC. Cada uma dessas retas é definida por um sistema de custos, uma para os custos envolvendo amostras positivas e outra para custos envolvendo amostras negativas. Os pontos onde essas retas tocam a curva definem um intervalo de limiares que irá minimizar o custo estimado para esse mecanismo de custo proposto. Esses limiares funcionam como um mecanismo de rejeição, isto é, rejeitam-se as amostras cujo *score* esteja dentro do intervalo definido por esses dois limiares.

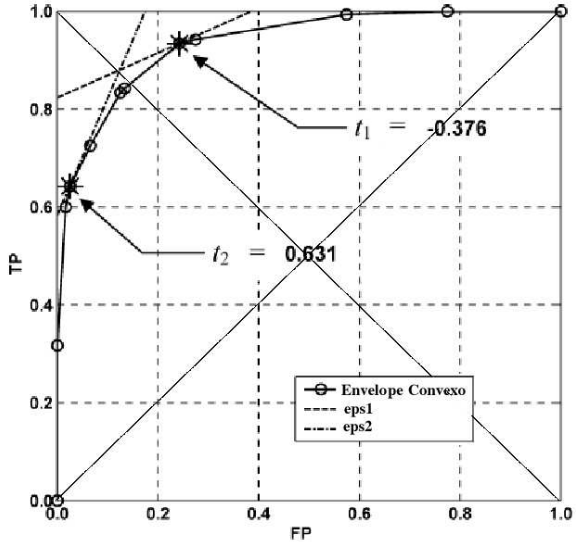


Figura 14: Mecanismo de rejeição baseado em análise de curva ROC [21]

3.3 Resumo

Este capítulo apresentou alguns fundamentos necessários à compreensão do método proposto neste trabalho. Foram apresentados dois possíveis classificadores para este método (os dois foram avaliados nos experimentos realizados). Também foi apresentado o conceito de ROC. A análise de curvas ROC terá um duplo propósito neste trabalho - avaliar o desempenho dos classificadores (permitindo uma análise ampla de um classificador, bem como comparações entre classificadores) e otimizar a decisão do classificador (filtro) com base em um sistema de custo. No capítulo seguinte, será feita uma apresentação do método proposto.

4 Método Proposto

Neste capítulo apresentaremos o método proposto para classificação de pontos de segmentação em dígitos manuscritos conectados. Este método tem como objetivo reduzir o número de hipóteses geradas por algoritmos de segmentação baseados na estratégia segmentação-reconhecimento. Este tipo de estratégia tem uma vantagem em relação à dissecação - faz menos uso de heurísticas - mas em compensação tem uma desvantagem, gera um grande número de hipóteses. Através do uso do método aqui proposto, usando como filtro em métodos baseados em segmentação-reconhecimento, esperamos obter redução de quantidade de hipóteses de segmentação, sem recorrer ao uso de heurísticas.

4.1 Arquitetura

A idéia deste método é classificar pontos de segmentação de cadeias de dígitos numéricos conectados, de qualquer tamanho, tanto para conexões simples quanto para conexões múltiplas. As Figura 4 e 5 mostram como o método proposto interage com as etapas de segmentação e de reconhecimento em um sistema de reconhecimento de dígitos manuscritos baseado em segmentação-reconhecimento [16]. Como não é possível saber se um ponto é necessário ou não através da sua localização, foi evitado o uso de características estruturais para modelar esses pontos. Foram realizados alguns experimentos com características estruturais como as utilizadas por Chen *et al* [5], mas os resultados obtidos mostraram que as mesmas possuem um poder de discriminância insuficiente.

Em Oliveira *et al* [16], demonstrou-se que características estruturais não possuem um desempenho satisfatório. A Figura 15 mostra o resultado de um experimento, no qual um algoritmo de segmentação foi aplicado a amostras normalizadas (altura e largura) dos dígitos 1, 4, 7 e 8 da base NIST SD19. Esses pontos foram plotados e como podemos notar, não existe um padrão que discrimine as classes de pontos.

Um ponto de segmentação pode aparecer em qualquer lugar para uma determinada classe de dígito numérico como podemos ver na Figura 15. Além disso, é possível notar por esta figura que não há relação entre a localização de um ponto e a classe do dígito. Realizamos alguns experimentos com características estruturais que comprovaram a deficiência das mesmas para modelar pontos de segmentação.

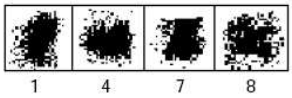
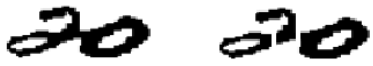


Figura 15: Distribuição dos pontos de segmentação para as classes de dígitos isolados 1, 4, 7 e 8 [16].

Sobre-segmentação pode ser definida como uma segmentação ocorrida em um dígito isolado e é causada por um ponto desnecessário (um ponto desnecessário é um ponto que segmenta a imagem indevidamente). Há uma relação direta entre ponto desnecessário e sobre-segmentação. Por isso, para detectar ponto desnecessário, tentaremos descobrir se o mesmo causou uma sobre-segmentação. Um exemplo de sobre-segmentação pode ser visto na Figura 16.



(a) (b)

Figura 16: Exemplo de sobre-segmentação do dígito “2”. (a) Imagem original. (b) Imagem sobre-segmentada.

4.2 Conjunto de características

Os algoritmos de segmentação existentes na literatura nos mostram várias estratégias e métodos para identificar pontos candidatos à segmentação. Apesar de alguns deles terem sido muito bem sucedidos em encontrar pontos de segmentação necessários, o número elevado de pontos candidatos gerados por alguns desses algoritmos é um bom indicativo sobre o quão difícil pode ser encontrar características que identifiquem de forma precisa se um ponto é ou não desnecessário.

Uma outra abordagem seria tentar identificar se um dos segmentos de uma determinada hipótese de segmentação é ou não uma ocorrência de sobre-segmentação. Oliveira *et al* [16] propõem um método para detectar sobre-segmentação. Este método chama-se *Multilevel Concavity Analysis* (MCA). Primeiramente, cada um dos *pixels* de fundo do componente conectado é analisado e recebe um rótulo (*label*), indicando a quantidade de *pixels* existentes em cada uma das 4-direções de Freeman. Com isso obtém-se o *Initial Concavity Level* (ICL) de cada um destes *pixels*. Em seguida, faz-se o mesmo processo, mas somente para um dos segmentos (para aquele que deseja-se verificar se é ou não uma ocorrência de sobre-segmentação). Através destas análises, obtém-se duas imagens contendo apenas os rótulos (imagens rotuladas) sendo uma imagem do componente conectado (I_{orig}) e outra a imagem do hipotético caso de sobre-segmentação (I_{seg}). Um exemplo de ICL pode ser visto na Figura 17.

Gerados I_{seg} e I_{orig}, é feita uma comparação píxel-a-píxel de seus ICLs. Como resultado é gerada uma nova imagem rotulada, chamada MCA. Cada píxel de I_{seg} e I_{orig} é comparado. Se ambos possuem rótulos diferentes, eles recebem um novo rótulo no MCA, indicando que houve uma mudança em suas concavidades. Caso contrário o mesmo rótulo é atribuído ao píxel na MCA também. Píxeis de fundo de I_{seg} também recebem um rótulo especial no MCA. Criado o MCA, é extraída a informação contextual (CI, do inglês *Contextual Information*) do hipotético caso de "sobre-segmentação". O CI de I_{seg} é o ICL

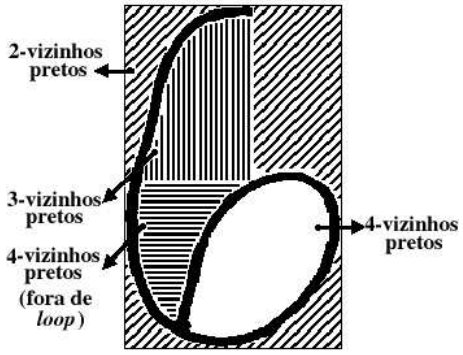


Figura 17: Exemplo de ICL para um dígito [16].

das áreas acima e abaixo de I_{seg}, incluindo todos os possíveis *pixels* de contorno. Um exemplo de um MCA pode ser visto na Figura 18, onde podemos observar que de fato as características do MCA ajudam a destacar a parte sobre-segmentada do caracter original.

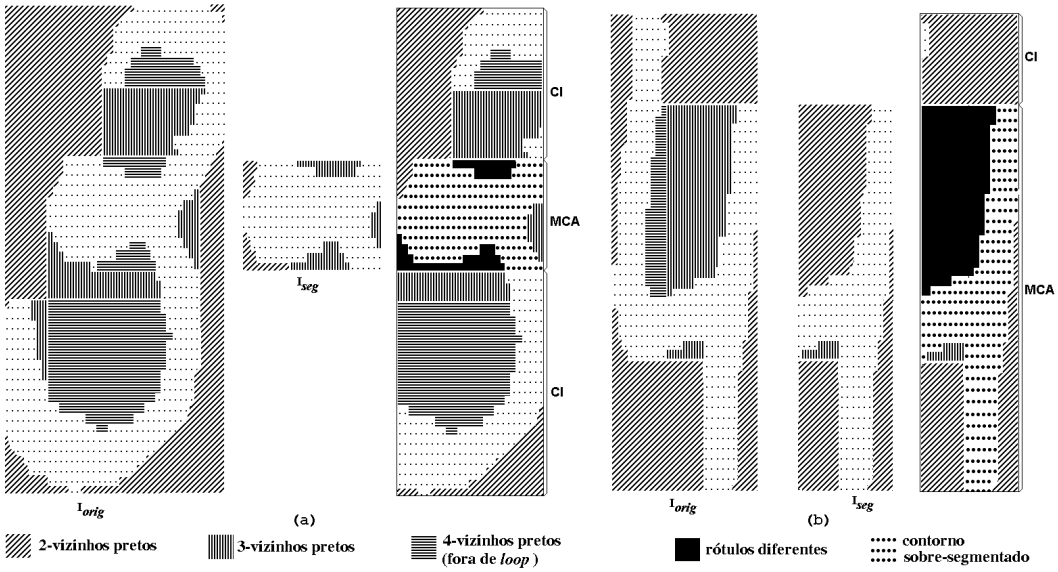


Figura 18: Exemplo de MCA [16].

Existem sete características de MCA que podem ser extraídas de uma imagem:

1. Número de *pixels* de fundo cercados por dois *pixels* pretos.

2. Número de *pixels* de fundo cercados por três *pixels* pretos.

3. Número de *pixels* de fundo cercados por quatro *pixels* pretos, mas que não estão dentro de um *loop*.

4. Número de *pixels* de fundo dentro de um *loop*.

5. Número de *pixels* de fundo que sofreram uma mudança em seu nível de concavidade (*label*).

6. Número de *pixels* de pretos dentro da região do MCA.

7. Número de *pixels* de pretos fora da região do MCA mas dentro da região extendida.

Foi utilizado um esquema de zoneamento (*zoning*) para extrair essas características. Após criada a imagem de rótulos MCA, a mesma é dividida em 2×3 regiões de igual tamanho. Em seguida as sete características de MCA são extraídas de cada região. Os vetores de características de todas as seis regiões são concatenados em um único vetor, contendo 42 características. Para cada segmento de cada imagem é extraído um vetor (a classificação é feita segmento por segmento).

4.3 Estratégia de verificação [16]

A estratégia de verificação demonstrada aqui foi proposta por Oliveira *et al* [16]. Essa estratégia servirá como base de comparação para o método proposto e tem como objetivo confirmar as hipóteses de classificação geradas pelo classificador de dígitos, verificar se as mesmas são válidas ou não. Ao invés de filtrar as amostras desnecessárias, como nós propomos, são adicionados outros dois classificadores, um de sub-segmentação e outro de sobre-segmentação, que irão reforçar ou punir os resultados do classificador de dígitos isolados. As probabilidades dos três classificadores (dígito isolado, sub-segmentação e sobre-segmentação) são combinadas em um modelo probabilístico [16]. Neste caso, o que

os classificadores tentam aferir é a probabilidade da amostra não ser um caso de sobre-segmentação ou sub-segmentação. Caso essa probabilidade (de não ser) seja baixa, o fato de ser utilizada probabilidade combinada irá fazer com que a probabilidade final daquele dígito seja penalizada. Podemos ver a estratégia de verificação na Figura 19. Podemos ver nessa figura que cada segmento é submetido a um classificador específico. No caso, e_{13} é o classificador de dígitos isolados, v_o é um classificador utilizado para detectar casos de sobre-segmentação (amostras segmentadas além do necessário) e v_u é um classificador utilizado para detectar casos de sub-segmentação (amostras que deveriam ter sido segmentadas e não foram). Também podemos ver nessa figura que para um dado segmento, os resultados dos três classificadores são multiplicados entre si. No caso dos classificadores v_o e v_u são utilizadas as probabilidades dos neurônios de saída de índice zero, isto é, a probabilidade de não serem sobre-segmentação e sub-segmentação respectivamente. Isto é, amostras de sub-segmentação ou sobre-segmentação irão impor uma "punição" aos resultados do classificador de dígitos isolados.

O classificador de sobre-segmentação (v_o) utiliza as características de MCA vistas anteriormente. O classificador de sub-segmentação (v_u) utiliza 13 características de concavidade de Freeman (Figura 20), juntamente com o número de *pixels* de contorno. É atribuído um rótulo de 1 a 13 a cada píxel de fundo de acordo com o tipo de concavidade do mesmo. É feita uma verificação adicional em *pixels* de fundo cercados por quatro *pixels* para descobrir se o mesmo está em um contorno fechado (*loop*) ou não. A Figura 20c mostra as possíveis ocorrências de *pixels* de fundo cercados por quatro *pixels* de contorno mas que não estão em um contorno fechado. Basicamente essas setas representam o caminho a ser percorrido até encontrar uma "saída", isto é, um "buraco" naquele que aparentava ser um "loop" fechado.

Essas características são extraídas utilizando um esquema de zoneamento (*zoning*). Esse esquema pode ser visto na Figura 21. A imagem é dividida em três regiões

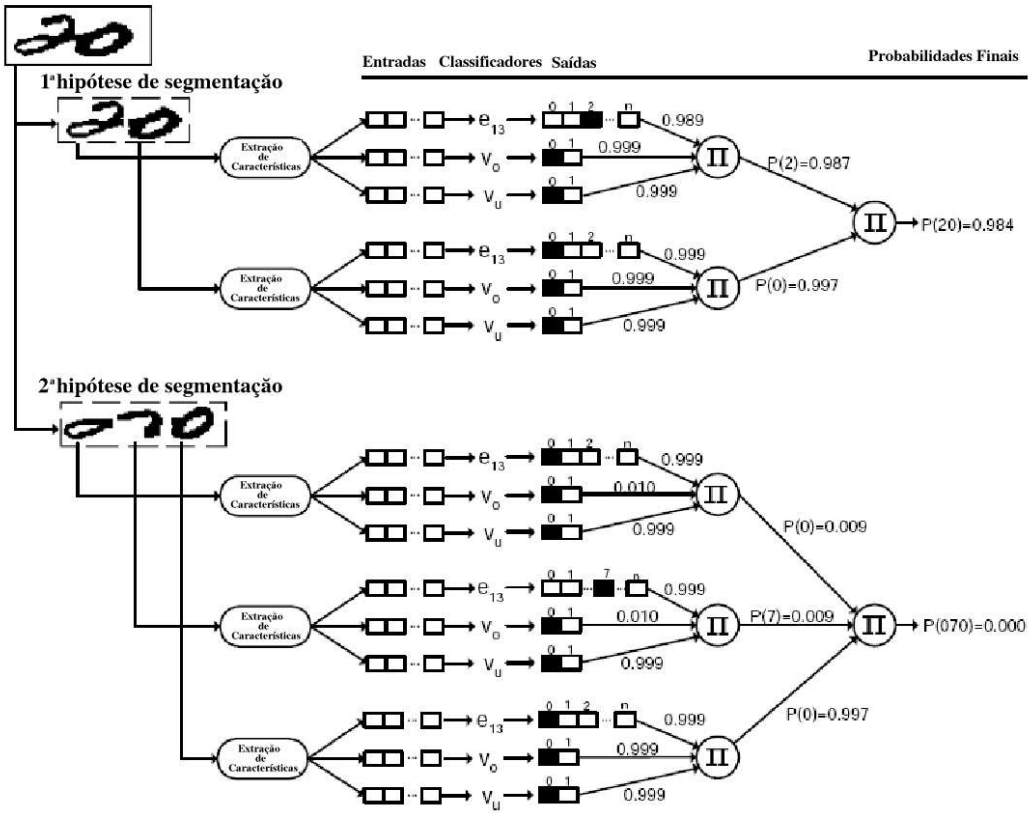


Figura 19: Interação entre verificadores e classificador de dígitos isolados. Os classificadores v_o e v_u são dois verificadores. O classificador e_{13} é um classificador de dígitos isolados [16].

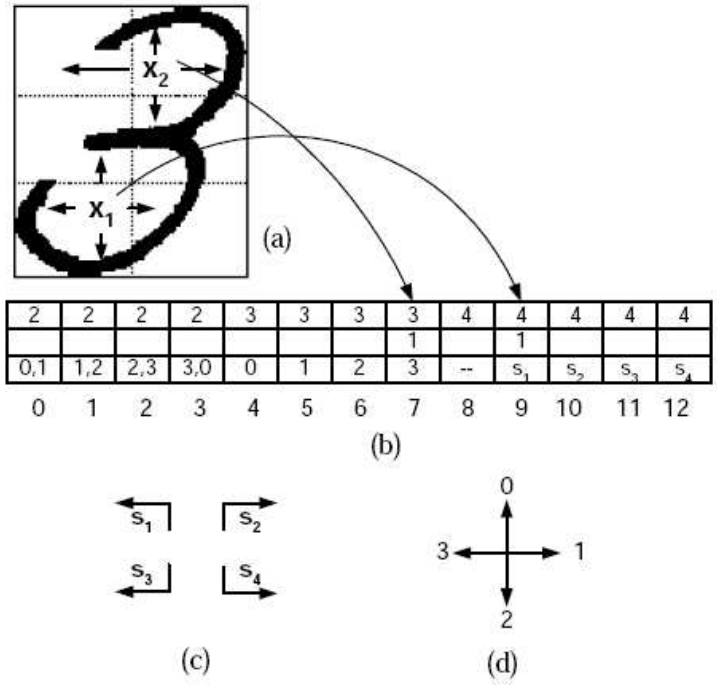
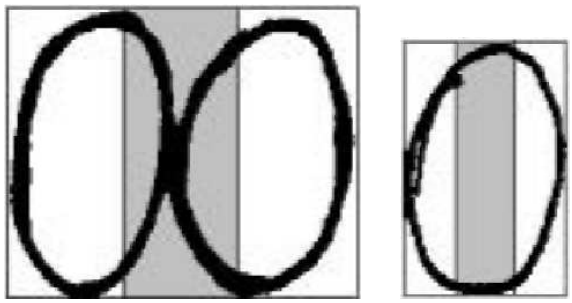


Figura 20: 13 características de Freeman. (a) Vetor de características. (b) Concavidades. (c) Direções auxiliares. (d) 4-direções de Freeman. [16]

verticais. Em cada região, são contabilizados o total de *pixels* de fundo para cada um dos 13 tipos de concavidade e o total de *pixels* de contorno, resultando num vetor de 42 características.



(a) (b)

Figura 21: Esquema de zoneamento para extração de características de sub-segmentação [16].

4.4 Bases de dados

Esta seção descreve as bases de dígitos conectados e isolados utilizadas tanto para treinar o filtro proposto bem como para avaliá-lo.

Duas bases diferentes foram utilizadas neste trabalho. A primeira é composta de 31.710 amostras de dígitos sobre-segmentados (gerada a partir de dígitos isolados da base NIST SD19, os quais foram submetidos a um algoritmo de segmentação [7]) e foi utilizada para treinar, validar (verificadores) e testar a classe de dígito sobre-segmentado (ponto de segmentação desnecessário). Essa base foi particionada da seguinte forma - 10.781 amostras para treinamento, 10.783 para validação e 10.146 para teste.

A segunda é composta de 15.000 amostras de dígitos conectados extraídos da base sintética proposta por Oliveira *et al* [17] e foi utilizada para treinar, validar (verificadores)

e testar a classe de ponto de segmentação necessário. Essa base foi particionada da seguinte forma - 5.000 amostras para treinamento, 5.000 para validação e 5.000 para teste.

A base sintética proposta por Oliveira *et al* [17] contém 273.452 amostras de pares de dígitos conectados (300 dpi, bi-tonal) e foi gerada a partir de 2.000 dígitos isolados extraídos da base NIST SD19 (subconjunto hsf_0), artificialmente conectados entre si. De acordo com Wang *et al* [24], 89% das conexões ocorrem em cadeias de dois dígitos de comprimento. Além disso, a maioria dos algoritmos de segmentação existentes limitam-se ao problema de segmentar cadeias de dois dígitos.

O algoritmo proposto para conectar esses dígitos entre si, baseia-se em duas regras simples - primeiramente, apenas dígitos criados pelo mesmo autor podem ser conectados (informação sobre o autor é disponibilizada pela base NIST SD19) e por segundo, os dígitos são primeiramente alinhados entre si (na altura da mediana) e depois simplesmente deslocados até se conectarem. Essas regras fazem com que as amostras criadas sejam bastante similares àquelas criadas por autores humanos, evitando distorções como conectar dígitos com variação de tamanho muito grande, diferentes níveis de inclinação ou contornos com variação muito grande de espessura. Além disso, somente dígitos corretamente classificados por um classificador de dígitos isolados [16] foram utilizados na construção desta base sintética, minimizando a interferência da etapa de reconhecimento na avaliação de métodos de segmentação.

Esta base contém 89% de amostras de conexão simples e 11% de amostras de conexões múltiplas. Um aspecto importante desta base é que pelo fato de ser sintética, existe informação de *ground-truth* (localização exata dos pontos de segmentação ótimos) disponível.

4.5 Treinamento

O foco está em encontrar ocorrências de sobre-segmentação. Para treinar a classe de dígito sobre-segmentado, foram utilizadas 10.781 amostras de dígitos isolados sobre-segmentados. O método de segmentação proposto por Fenrich [7] foi aplicado sobre amostras de dígitos isolados da base NIST SD19, gerando essas amostras sobre-segmentadas. Foram utilizados somente os menores segmentos obtidos em cada dígito para este fim. A idéia de utilizar somente os menores segmentos, deve-se ao fato de que na maioria dos casos, o maior segmento de um dígito isolado pode ser facilmente confundido com um segmento obtido por um ponto de segmentação necessário.

Foram realizados experimentos utilizando ambos os segmentos para treinar o modelo de dígito sobre-segmentado. Também foram realizados experimentos onde apenas dígitos cuja relação das áreas do menor e do maior segmento fosse menor do que um determinado limiar (foram avaliados limiares de 10%, 20%, 30% e 40%). O resultado obtido em todos esses experimentos foi bastante parecido.

A similaridade entre um segmento obtido por um ponto de segmentação necessário e o maior segmento de um dígito sobre-segmentado pode ser visto na Figura 22.

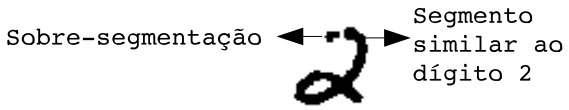


Figura 22: Exemplo do dígito 2 “sobre-segmentado” onde o maior segmento pode ser facilmente confundido com um segmento obtido por um ponto de segmentação necessário.

Para treinar o modelo de ponto de segmentação necessário, foi utilizada informação de *ground-truth* de 5.000 imagens da base sintética de dígitos conectados proposta por Oliveira *et al* [17]. Esse conjunto continha somente amostras de dois dígitos de com-

primento e com ocorrências de conexão simples. Como os pontos do *ground-truth* são ótimos, foi utilizada a informação de MCA dos dois dígitos, isto é, para cada uma das 5.000 amostras foram extraídos dois vetores de características, totalizando 10.000 vetores.

4.6 Esquema de custo baseado em ROC

Conforme apresentado por Tortorella [21], uma curva ROC pode ser uma ferramenta bem útil para encontrar os pontos operacionais de um determinado classificador para um dado esquema de custo. Tortorella utiliza dois tipos de custos na tentativa de definir dois pontos operacionais que por sua vez são utilizados para definir um intervalo de rejeição. Mas quando o que se tem em vista é filtrar pontos de segmentação, rejeição é uma ação que acaba não fazendo sentido, isto é, uma hipótese deve ser classificada como necessária e submetida a um classificador de dígitos isolados ou como desnecessária e eliminada. Por isso apenas um ponto operacional é necessário para atender a demanda do método proposto. Ele é obtido através de dois custos, custo de falso negativo (CFN) e custo de verdadeiro negativo (CTN, na verdade um benefício).
Esses dois custos irão nos dar o coeficiente de inclinação de uma reta:

$$m = -\frac{p(N) \cdot CTN}{p(P) \cdot CFN} \tag{18}$$

na qual $p(X)$ é a probabilidade *a priori* da classe X.

Para encontrar o ponto operacional para um dado esquema de custo (utilizando esses dois custos) basta procurar pelo ponto na curva ROC onde a linha com inclinação m toca a curva ROC.

Para gerar a curva ROC, foi utilizado um filtro treinado com uma base de 10.000 amostras da base sintética proposta por Oliveira *et al* [17]. Como base de teste para essa mesma finalidade foram utilizadas outras 10.000 amostras dessa mesma base sintética.

4.6.1 Esquemas de custo

A idéia principal do esquema de custo proposto está fundamentada no fato de que desperdiçar uma hipótese de segmentação necessária é muito pior em termos de impacto nas taxas de reconhecimento do que o benefício obtido por eliminar uma hipótese desnecessária, já que uma hipótese ótima descartada poderia ser eventualmente reconhecida com sucesso pelo classificador de dígitos isolados. Já em relação a uma hipótese desnecessária, muitas vezes quando não identificada acaba levando a erros de classificação (pelo classificador de dígitos isolados). No entanto, o grande problema é o impacto no custo computacional. Por este motivo, os esquemas de custo propostos buscam "punir" severamente este tipo de erro (falsos negativos).

De uma forma prática, essa estratégia minimizará a ocorrência de falsos negativos. A contrapartida é que a ocorrência de verdadeiros negativos também acaba sendo minimizada. A Figura 23 ilustra as implicações em se escolher um ponto operacional para um classificador binário. É possível perceber que qualquer mudança no ponto operacional x^* implica em novos valores de TP, FP, TN e FN.

Nos próximos estudos, pretendemos avaliar o método proposto em cadeias maiores que dois dígitos e com ocorrências de conexões múltiplas. Além disso, planejamos utilizar um mecanismo de seleção de características na tentativa de encontrar um conjunto de características que maximize o desempenho (em termos de melhora nas taxas de reconhecimento). Finalmente, pretendemos avaliar o desempenho do método proposto em outros algoritmos de segmentação baseados na estratégia segmentação-reconhecimento.

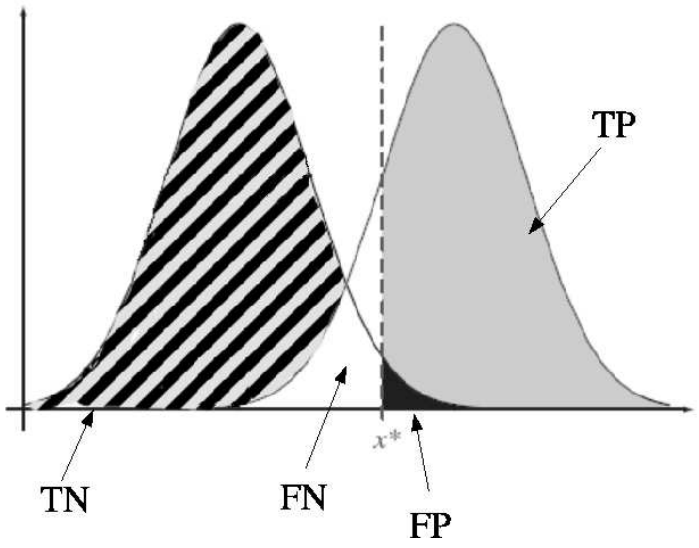


Figura 23: Impacto da escolha de um ponto operacional nas taxas de TP, FP, TN e FN.

5 Experimentos

5.1 Processo de avaliação

O processo de avaliação aqui descrito, tem como objetivo definir uma série de procedimentos a serem realizados para se compreender de forma clara o método proposto. Com boa compreensão do comportamento do método proposto, é possível fazer um ajuste fino (*fine-tuning*) do mesmo. Primeiramente definimos uma forma de rotular os pontos de segmentação gerados por um algoritmo de segmentação [7] entre pontos necessários e desnecessários. Feito isso, propomos um processo, composto de três etapas, a serem realizadas de maneira iterativa, de forma a melhorar o desempenho do método proposto. Análise da curva ROC será usada para este fim. Com isso, espera-se obter um claro entendimento da capacidade de generalização do método proposto.

É necessário entender o que queremos avaliar no método proposto. O primeiro passo a ser realizado, será avaliar o desempenho do filtro para as bases rotuladas, utilizando para isso análise das curvas ROC de cada base. Em seguida avaliaremos:

- O impacto do filtro proposto no classificador de dígitos isolados [16].
- O impacto da estratégia de verificação [16] no classificador de dígitos isolados.
- O impacto do filtro proposto na estratégia de verificação.

A medida de desempenho utilizada será taxa de erro de reconhecimento em função das taxas de rejeição.

5.1.1 Rotulação

Para avaliar as características operacionais do método proposto, foi necessário rotular os pontos gerados pelo método de segmentação [7] em pontos necessários e pontos

desnecessários. Para isso usamos o classificador de dígitos isolados proposto por Oliveira *et al* [16]. Foram utilizados dígitos isolados da base NIST SD19, previamente reconhecidos pelo classificador de dígitos isolados em questão, mitigando o viés do classificador no processo de rotulação.

Foram definidos limiares em relação à probabilidade retornada por esse classificador e a rotulação foi feita com base nesses limiares. Amostras cujos dígitos fossem reconhecidos corretamente por esse classificador com uma probabilidade maior que um determinado limiar foram rotuladas como necessárias. As demais foram classificadas como desnecessárias. Foram criadas cinco bases distindas, utilizando os seguintes limiares: 95%, 90%, 85%, 80% e 75%.

5.2 Aspectos metodológicos

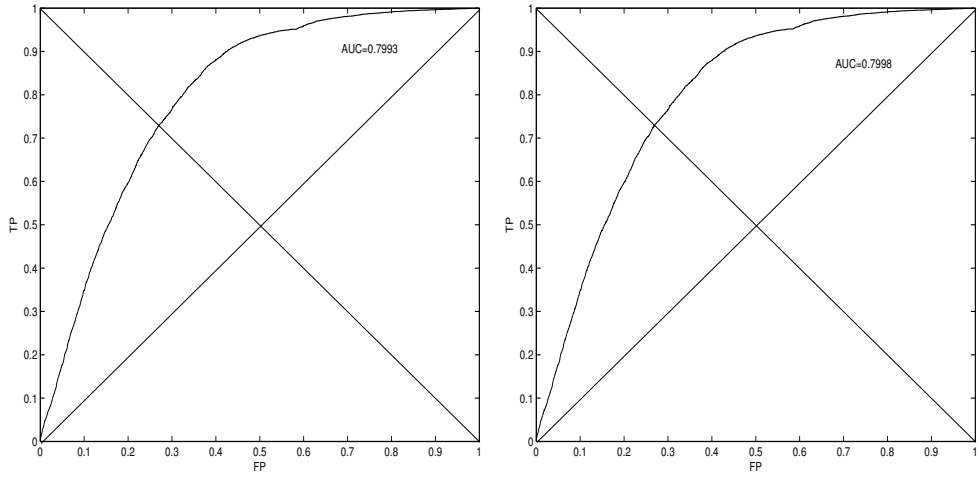
Os experimentos apresentados nesta dissertação foram realizados em um conjunto de 10.000 amostras de cadeias numéricas, todas elas contendo casos de dígitos conectados (conexão simples) e dois dígitos de comprimento. Essas amostras foram extraídas da base sintética proposta por Oliveira *et al* [17]. O algoritmo de segmentação proposto por Fenrich [7] foi utilizado para avaliar a eficiência do método proposto. Este algoritmo faz uso de características de contorno externo e de perfil na tentativa de achar pontos de segmentação em dígitos conectados. Este algoritmo foi escolhido por ser bastante popular na literatura. Além disso, vários outros algoritmos de segmentação baseiam-se no conjunto de características utilizado por este método. Este algoritmo gera em média 3,2 pontos de segmentação para cada cadeia de dois dígitos conectados (com um desvio padrão de ± 1).

Nenhuma hipótese de segmentação foi encontrada para 37 dessas 10.000 amostras. No total foram gerados 31.305 pontos de segmentação para 9.963 amostras. Foi estabelecido o seguinte critério para se descartar uma hipótese: o classificador de pontos de segmentação

deve reconhecer pelo menos um dos segmentos como um caso de “sobre-segmentação”. Antes de realizar os experimentos, foi necessário rotular cada uma dessas hipóteses como necessária ou desnecessária. Para isso foi utilizado o classificador de dígitos isolados proposto por Oliveira *et al* [16]. Este classificador é baseado em uma rede neural MLP e retorna uma probabilidade para cada imagem submetida a ele. Foi utilizado um mecanismo de rejeição. Basicamente, cada um dos segmentos da cadeia foi avaliado por este classificador e rejeitado se a probabilidade do classificador ficasse abaixo de um determinado limiar (95%). Caso contrário, verifica-se se houve um acerto na classificação. Todas as hipóteses em que houve erro ou rejeição foram rotuladas como desnecessárias. As demais (acertos) foram rotuladas como necessárias. Isto resultou em 19.380 amostras de pontos de segmentação desnecessários.

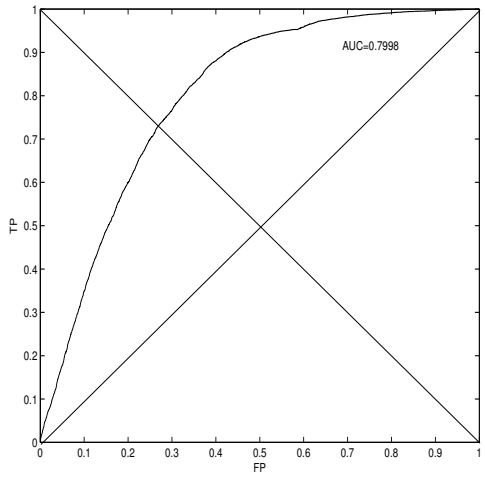
Para evitar qualquer viés, no processo de criação da base sintética foram utilizados apenas dígitos isolados reconhecidos corretamente por esse mesmo classificador de dígitos isolados [17]. Além disso, foram feitas outras rotulações, utilizando limiares de 75%, 80%, 85% e 90%. A análise das curvas ROC do filtro proposto para cada uma dessas rotulações pode ser vista na Figura 24 e mostra que a escolha desse limiar não afeta significativamente o desempenho do método proposto. Nessa figura podemos ver que a área sob a curva (AUC) é praticamente a mesma para todas as rotulações, respectivamente 0.7993 (para o limiar de 75%), 0.7998 (para o limiar de 80%), 0.7998 (para o limiar de 85%), 0.7999 (para o limiar de 90%) e 0.7964 (para o limiar de 95%).

Remover o máximo possível desses pontos foi o foco dos experimentos realizados. Um classificador SVM binário foi utilizado no método proposto. Os elementos dos vetores de características foram normalizados entre -1 e 1.

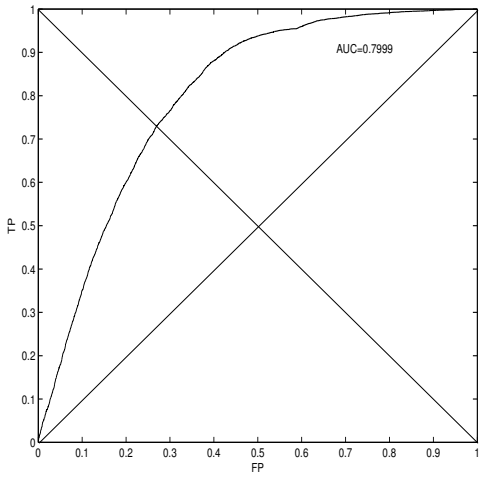


(a)

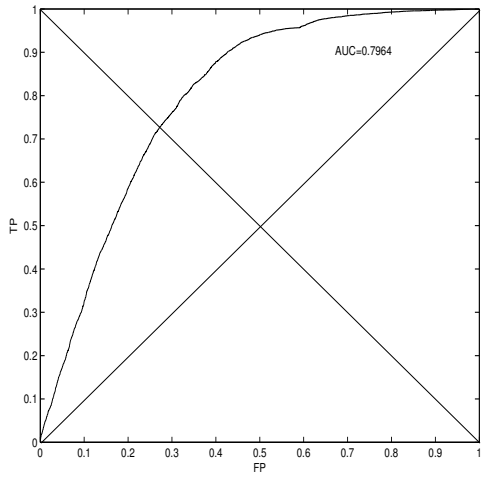
(b)



(c)



(d)



(e)

Figura 24: Análise ROC para rotulações utilizando outros limiares no classificador de dígitos isolados. (a) 75%. (b) 80%. (c) 85%. (d) 90%. (e) 95%.

5.3 Desempenho para pontos de segmentação necessários

No primeiro experimento, o método foi testado em pontos de segmentação necessários, obtidos do *ground-truth*. O objetivo deste experimento foi obter um parâmetro de referência do desempenho do classificador de sobre-segmentação. A taxa de acerto foi de 98,9%. Analisando os erros, foi possível observar que mais de 50% (0,7% do 1,1% de erro) dos erros ocorreram em amostras contendo os dígitos sete (“7”) ou um (“1”). A explicação para este fato pode ser a ausência de características de concavidade para este tipo de dígito. Por este motivo, eles são facilmente confundidos com um caso de sobre-segmentação já que os menores segmentos em casos de sobre-segmentação também possuem poucas características de concavidade. Mas em um caso real, o impacto deste tipo de erro será pequeno já que casos de conexões envolvendo estes tipos de dígitos (“1” ou “7”) são raros. O “1” por exemplo, é geralmente composto por um traço no sentido vertical, com pouca ou nenhuma protuberância, o que o torna pouco suscetível a se envolver em casos de conexões. Por um outro lado, características estruturais poderiam ajudar a reduzir este tipo de erro, mas elas acabam por adicionar ruído ao modelo, dada a sua baixa discriminância. Alguns exemplos de pontos ótimos classificados como desnecessários podem ser vistos na Figura 25.

Figura 25: Pontos ótimos classificados como desnecessários.

5.4 Desempenho para pontos de segmentação desnecessários - com conhecimento do tamanho da cadeia de caracteres

Apesar do método proposto funcionar sem conhecimento do tamanho da cadeia de caracteres, em algumas situações, como reconhecimento de código postal em envelopes postais, é possível saber *a priori* o tamanho da cadeia de caracteres. Por isso, saber como o método proposto se comporta quando há conhecimento prévio do tamanho da cadeia de caracteres é algo desejável. Neste experimento, foi avaliado o desempenho do método proposto para filtrar pontos de segmentação desnecessários. Existem dois aspectos importantes a serem avaliados - o número de pontos de segmentação filtrados e o impacto no resultado final do classificador de dígitos isolados. Primeiramente, foi realizado um experimento sem o uso do filtro em questão, para que fosse possível obter um parâmetro do impacto do mesmo nas taxas de reconhecimento das cadeias de dígitos. Como foi dito anteriormente, foi utilizado um classificador de dígitos isolados, baseado em MLP para classificar as cadeias de dígitos numéricos. Foi utilizado um limiar de 99% na probabilidade do filtro, isto é, para uma hipótese ser considerada desnecessária, pelo menos um dos segmentos da mesma deve ser classificado como um caso de sobre-segmentação com uma taxa de 99%. Apesar da escolha intuitiva, esse limiar condiz com o esquema de custo descrito na Tabela 3.

Em seguida, foi realizado um experimento com o filtro proposto. Foi utilizado o mesmo critério para identificar pontos desnecessários, isto é, o classificador SVM deveria reconhecer ao menos um dos segmentos como um caso de sobre-segmentação. No caso, 67,9% dos pontos desnecessários foram descartados. Além disso, foi possível obter uma melhora nas taxas de reconhecimento do classificador de dígitos isolados. Alguns dos casos de sobre-segmentação são similares a alguns dígitos isolados (em termos de características de MCA). Por isso, alguns pontos desnecessários foram classificados erroneamente como sendo necessários como podemos ver na Figura 26. Nas Figuras 26a, 26b e 26d, os

pontos de segmentação estão bem próximos dos respectivos pontos ótimos encontrados pelo algoritmo de segmentação. Na Figura 26c, o ponto de segmentação localiza-se à esquerda do *loop* inferior do dígito "6" (claramente um erro de segmentação). Mas este tipo de erro (falso positivo) não afeta de maneira significativa o desempenho do classificador de dígitos isolados (como por exemplo os falsos negativos), apesar de Oliveira *et al* [16] apresentar alguns casos de falsos positivos que levam a erros no classificador de dígitos isolados.

(a) (b) (c) (d)

Figura 26: Pontos desnecessários classificados como necessários.

5.4.1 Resultados

Foi avaliado o impacto do método proposto nas taxas de reconhecimento de cadeias numéricas de dígitos conectados. Os resultados desses experimentos, bem como a descrição dos mesmos podem ser encontrados em Vellasques *et al* [23]. De uma forma geral, podemos dizer que através do uso do filtro proposto, além da redução do número de hipóteses, foi possível obter uma melhora nas taxas de acerto do classificador de dígitos isolados.

A Figura 27 mostra o impacto na taxa de erro (em função da taxa de rejeição). Neste caso **Nenhum** e **SVM** corresponde aos resultados do classificador de dígitos isolados sem o uso de filtros e com o uso de um filtro baseado num classificador SVM,

respectivamente. Como podemos observar, houve melhoras em termos de desempenho do classificador de dígitos isolados. Esses resultados demonstram a eficiência do método proposto.

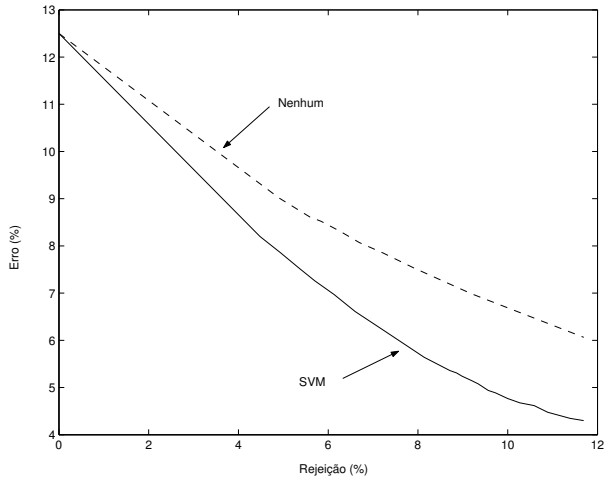


Figura 27: Impacto do método proposto em termos de taxa de erro *versus* taxa de rejeição.

5.5 Desempenho para pontos de segmentação desnecessários - sem conhecimento do tamanho da cadeia de caracteres

Os experimentos descritos até aqui consideram que o tamanho da cadeia de caracteres é conhecido *a priori*. Isso torna o problema relativamente mais simples. Mas espera-se que em condições reais não existam tais restrições em um sistema de reconhecimento de dígitos manuscritos (salvo em algumas exceções). No entanto a ausência dessa restrição acarreta em um aumento do custo computacional, já que em uma dada cadeia pode haver até 2^n hipóteses de segmentação, onde n é o número de pontos de segmentação.

O método proposto foi avaliado sob duas perspectivas - impacto nas taxas de reconhecimento do classificador de dígitos isolados e custo computacional da classificação como

um todo (filtragem+classificação de dígitos isolados).

5.5.1 Análise de custo

Para encontrar um esquema de custo que ao mesmo tempo minimizasse o custo computacional e os erros de classificação foram avaliados onze esquemas diferentes de custo, de C1 (*iso-cost*) a C11. O limiar SVM de cada um desses esquemas corresponde a deslocar o hiperplano de separação das classes (inicialmente na posição "0") para o local definido pelo limiar. Como as saídas do SVM não são balanceadas, a posição inicial do hiperplano (que é a utilizada nos experimentos sem esquema de custo) não é necessariamente a mesma do esquema de custo *iso-cost*.
Esses esquemas de custo podem ser vistos na Tabela 3.

Tabela 3: Esquemas de custo utilizados nos experimentos

Esquema	CTN	CFN	Limiar SVM	Limiar Probabilidade (Negativa)
C1	-10	10	1,422	1,21%
C2	-10	20	0,430	20,79%
C3	-10	30	-1,295	98,21%
C4	-10	40	-1,561	99,2%
C5	-10	50	-1,607	99,31%
C6	-10	60	-1,645	99,38%
C7	-10	70	-1,645	99,38%
C8	-10	80	-1,889	99,71%
C9	-10	90	-2,273	99,99%
C10	-10	100	-2,462	99,99%
C11	-10	110	-2,949	99,99%

Os pontos operacionais correspondentes a esses esquemas de custo podem ser vistos na Figura 28.

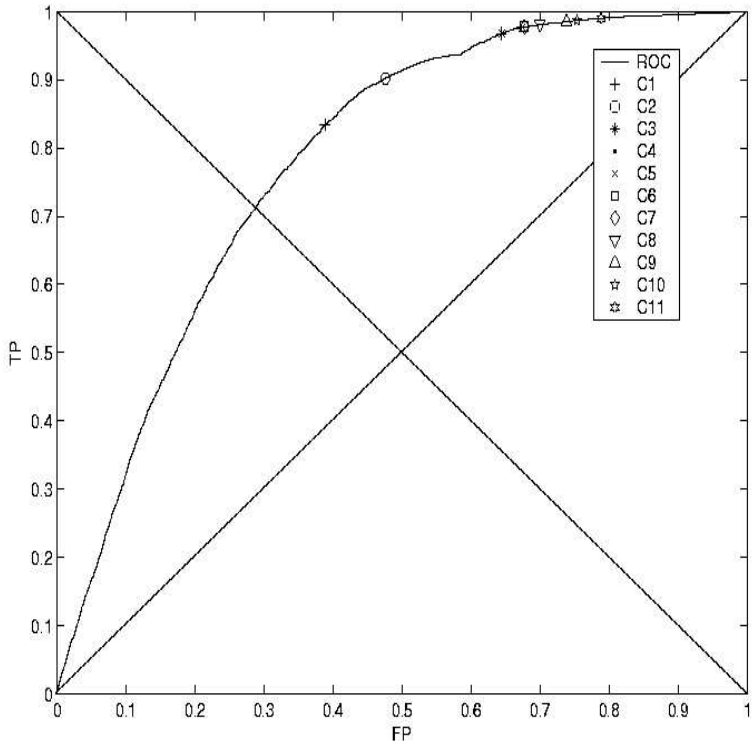


Figura 28: Pontos operacionais para esquemas de custo propostos.

5.5.2 Impacto no desempenho do classificador de dígitos isolados

Três tipos de experimentos foram realizados visando avaliar o impacto do método proposto no desempenho do classificador de dígitos isolados. Em todos eles, a métrica utilizada foi a mesma - Taxa de Erro $\times$ Taxa de Rejeição.
Primeiramente foi realizado o seguinte experimento:

- Reconhecimento de cadeias de dígitos manuscritos sem o uso do filtro proposto.

- Reconhecimento de cadeias de dígitos manuscritos usando o filtro proposto mas sem um mecanismo de custo.
- Reconhecimento de cadeias de dígitos manuscritos usando o *low-level verifier* [16].
- Reconhecimento de cadeias de dígitos manuscritos usando o filtro proposto (sem mecanismo de custo) em conjunto com o *low-level verifier*.

Os resultados deste experimento podem ser vistos na Figura 29.

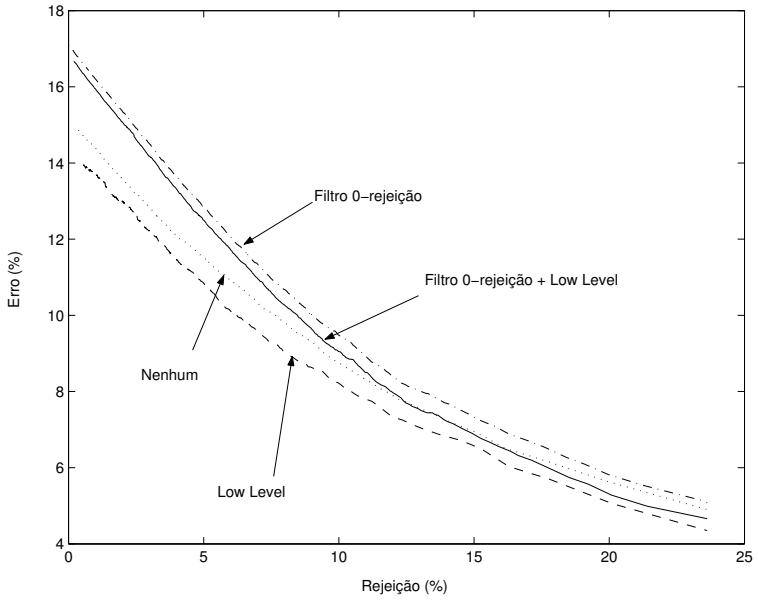


Figura 29: Taxas de reconhecimento de cadeias de caracteres sem o uso de filtro, com o uso de filtro (sem mecanismo de custo), com o uso do *low-level verifier* e com o uso do filtro+*low-level verifier*.

Podemos observar que o melhor resultado foi obtido através do uso do *low-level verifier*. Neste caso, o filtro proposto sem o uso de um mecanismo de custo acabou reduzindo o desempenho do classificador de dígitos isolados. Isto ocorreu porque muitas hipóteses necessárias foram classificadas como desnecessárias e descartadas pelo filtro.

Um exemplo desse tipo de problema pode ser visto na Figura 30, onde há apenas um ponto de segmentação e esse ponto é ótimo (que acarreta em duas hipóteses, uma com esse único ponto ativo e outra com o mesmo desativado). O filtro sem mecanismo de custo classificou esse ponto como desnecessário (um caso claro de falso negativo), o que fez com que a hipótese ótima fosse descartada. No entanto, ao atribuir um custo alto para ocorrência de falso negativo, o classificador tornou-se mais "exigente" em relação à detecção de pontos desnecessários e classificou o ponto como necessário.

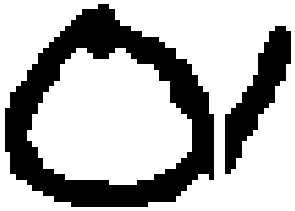


Figura 30: Amostra beneficiada pelo aumento no custo de falso negativo.

No segundo experimento, foi avaliado o impacto dos esquemas de custo propostos na Tabela 3 (Figura 31).

O melhor resultado foi obtido através da combinação CFN=90 e CTN=-10, que impõe um custo alto para a eliminação de um ponto necessário (ocorrência de falso negativo). No entanto os resultados são bem parecidos. Também é importante notar que o desempenho melhora gradativamente à medida em que aumenta-se o CFN (para valores de 10 a 90) quando então o desempenho começa a piorar.

No terceiro experimento foi adicionado um *low-level verifier* ao filtro proposto (Figura 32).

Neste caso, o melhor resultado foi obtido através da combinação CFN=100 e CTN=-10.

Na Figura 33 podemos ver uma comparação dos melhores resultados de cada um desses experimentos. Podemos observar que o melhor resultado foi obtido através do filtro

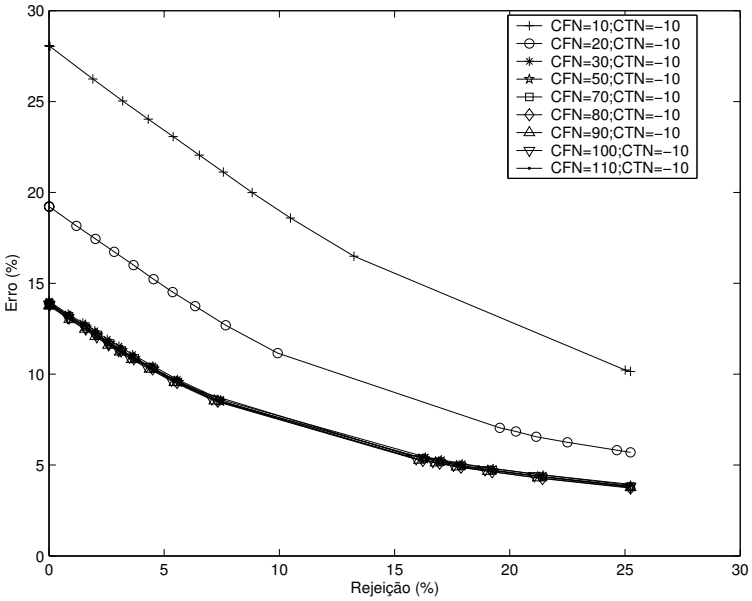


Figura 31: Taxas de reconhecimento de cadeias de caracteres utilizando o filtro proposto aliado aos esquemas de custo da Tabela 3.

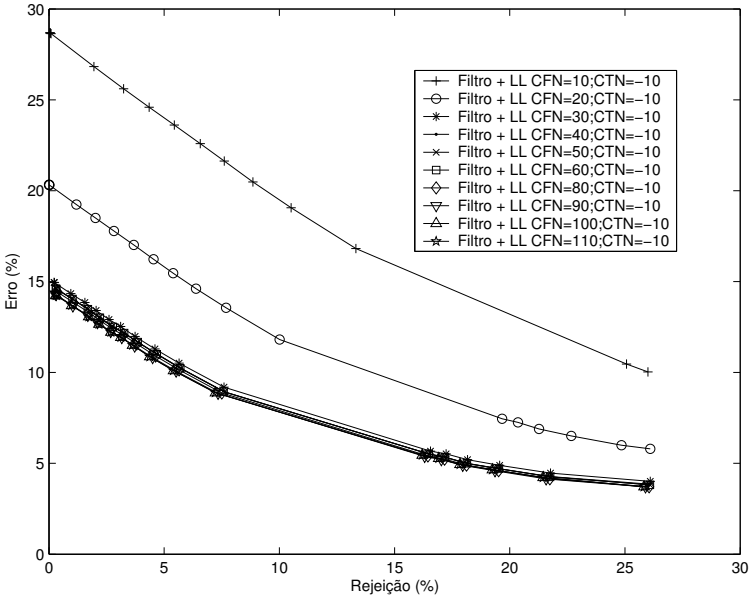


Figura 32: Taxas de reconhecimento de cadeias de caracteres utilizando o filtro proposto + *low level* aliado aos esquemas de custo da Tabela 3.

(esquema de custo CFN=90 e CTN=-10).

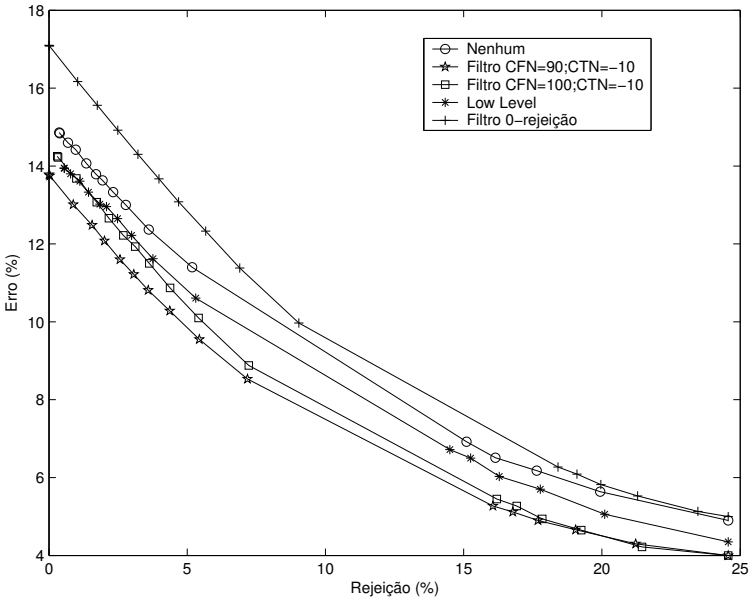


Figura 33: Melhores resultados.

5.6 Impacto no custo computacional

5.6.1 Número de hipóteses

Uma das métricas utilizadas para avaliar o impacto no custo computacional foi o número de hipóteses de segmentação. O algoritmo de segmentação avaliado [7] gerou 31.305 pontos de segmentação para uma base de 10.000 amostras. Como os experimentos foram realizados considerando não haver conhecimento do tamanho da cadeia, foi necessário avaliar todas as possíveis hipóteses de segmentação para cada amostra. Para cada amostra, o número de hipóteses é igual a 2^n onde n é o número de pontos. Por isso, os 31.305 pontos dessas 10.000 amostras se traduziram em 143.533 hipóteses de segmentação. Nas estratégias sem filtro e *low level* todas as hipóteses precisam ser avaliadas. Já em relação ao uso do filtro, quanto mais baixo o custo de falso negativo, menor a quantidade

de hipóteses a serem avaliadas pelo classificador de dígitos isolados. Isso ocorre porque à medida em que se aumenta o custo de falso negativo, o filtro se torna mais "exigente", isto é, passa a eliminar somente hipóteses para as quais o grau de certeza (de serem desnecessárias) seja alto. A Tabela 4 mostra isso. Nela podemos ver que o número de hipóteses para o esquema de custo C9 é maior do que para o C3. No entanto, os dois possuem um impacto semelhante no desempenho do classificador de dígitos isolados. Por isso, podemos dizer que C3 apresenta a melhor relação custo-benefício.

Tabela 4: Número de hipóteses

Esquema	Hipóteses
Sem Filtro	143.533
Filtro 0-rejeição	16.413
Low Level (LL)	143.533
Filtro C3	23.519
Filtro C9	37.898
Filtro C11	49.388

5.6.2 Número de chamadas ao classificador

Uma outra métrica utilizada para avaliar o impacto no custo computacional foi o número de chamadas ao classificador. Todos os experimentos foram realizados utilizando 0-rejeição no classificador de dígitos isolados. O número de chamadas ao filtro proposto, classificador de sub-segmentação, classificador de sobre-segmentação e classificador de dígitos isolados foi avaliado. Uma observação importante é que o classificador de dígitos isolados trabalha com 132 características, enquanto os demais classificadores trabalham com 42. O número de hipóteses submetidas ao classificador de dígitos isolados

também foi avaliado. Os resultados obtidos mostram que o desempenho (em termos de

Tabela 5: Número de chamadas ao classificador

Esquema	Filtro	Sub-Segmentação	Sobre-Segmentação	Dígitos Isolados
Sem Filtro	0	0	0	527.429
Filtro 0-rejeição	240.570	0	0	32.656
Low Level (LL)	0	527.429	527.429	527.429
Filtro C3	235.166	0	0	53.459
Filtro C9	268.710	0	0	100.122
Filtro C11	295.264	0	0	139.524

taxa de reconhecimento) quando utiliza-se somente o filtro é praticamente o mesmo para os esquemas de custo entre C3 e C11. No entanto o número total de chamadas ao classificador no esquema de custo C3 é 33,6% menor do que o número de chamadas no esquema de custo C11.

Já o número de chamadas ao classificador de dígitos isolados cai de 527.429 no caso sem filtro para 53.459 no filtro C3 (com um aumento nas taxas de reconhecimento).

Já em relação ao *low level verifier*, o número total de chamadas cai de 1.582.287 para 288.625 (filtro C3).

5.6.3 *Total number of feature-value*

Uma outra maneira de avaliar o impacto no custo computacional é através de uma métrica chamada *total number of feature-values* [13]. Essa métrica foi proposta

inicialmente para avaliar o custo computacional de sistemas compostos de classificadores em cascata e é calculada da seguinte maneira

$$TVF = \sum_{i=1}^{n} m_i x_i \tag{19}$$

na qual n é a quantidade de classificadores, m_i o número de características do classificador i e x_i o número de instâncias classificadas pelo classificador i.

Analisando por esta métrica, podemos concluir que há uma grande vantagem em eliminar hipóteses de segmentação desnecessárias utilizando o classificador de sobre-segmentação (antes que a mesma seja classificada pelo classificador de dígitos isolados). Uma única chamada ao classificador de sobre-segmentação possui um TVF de 42 enquanto uma chamada ao classificador de dígitos isolados possui um TVF de 132. Além disso, devido à estratégia utilizada (descartar a hipótese caso um dos segmentos seja classificado como sobre-segmentação), muitas vezes é possível descartar uma hipótese desnecessária sem a necessidade de avaliar todos os segmentos da mesma. Enquanto na estratégia *low-level verifier* todos os segmentos de cada hipótese tem que ser avaliados por três classificadores. A análise de TVF global dos experimentos realizados mostra o quão vantajoso é o método proposto em termos de custo computacional.

Um ponto importante desses resultados é o fato do TVF do filtro C3 ser 76% menor do que o TVF do caso sem filtro e ainda com um aumento nas taxas de reconhecimento. Além disso, os Filtros C3 e C9 possuem um desempenho bem parecido em termos de taxa de reconhecimento mas o TVF do filtro C3 é 31% menor do que o TVF do filtro C9.

5.7 Análise de erro

Apesar dos resultados obtidos mostrarem os benefícios do método proposto em termos de melhora nas taxas de reconhecimento de cadeias de dígitos manuscritos,

Tabela 6: *Total number of feature-values* global dos experimentos realizados

Esquema	TVF Global ($\times 10^7$)	Em relação ao TVF Global do caso Sem Filtro
Sem Filtro	6,96	0% menor
Filtro 0-rejeição	1,44	79% menor
Low Level (LL)	11,4	64% maior
Filtro 0-rejeição + LL	1,71	75% menor
Filtro C3	1,69	76% menor
Filtro C9	2,45	65% menor
Filtro C11	3,08	56% menor

ainda há espaço para melhora. Conhecer as principais limitações do método proposto é algo que pode encorajar estudos futuros e ajudar a esclarecer a respeito dos tipos de problemas não solucionados pelo método proposto.

Na Figura 34, uma amostra sobre-segmentada (Figura 34c) obtém uma probabilidade alta no classificador de dígitos isolados, resultando em um erro de classificação. Mas o filtro (C9) não consegue detectar a parte sobre-segmentada. O filtro sem esquema de custo por sua vez, identificou essa parte sobre-segmentada e eliminou a hipótese de segmentação. O *low level verifier* também identificou com sucesso a hipótese desnecessária. Nesse caso, combinar filtro e *low level* resolve o problema, já que a parte sobre-segmentada não identificada acaba sendo punida na estratégia *low level*.

Na Figura 35, ambos, filtro C9 e *low level verifier* falharam na detecção da parte sobre-segmentada (Figura 35b), causando um erro no classificador de dígitos isolados. O filtro 0-rejeição identificou corretamente a parte sobre-segmentada e a hipótese foi então eliminada. A hipótese ótima por sua vez foi corretamente classificada pelo classifi-

(a) (b) (c)

Figura 34: Erro de classificação para sem filtro e filtro (C9) e acerto para *low level* e filtro 0-rejeição. (a) Hipótese sub-ótima. (b) Hipótese ótima. (c) Hipótese desnecessária.

cador de dígitos isolados.

(a) (b)

Figura 35: Erro na detecção de parte sobre-segmentada. (a) Hipótese ótima. (b) Hipótese desnecessária.

Na Figura 36, a hipótese ótima (Figura 36a) foi erroneamente classificada como desnecessária pelo filtro C9. Essa mesma hipótese foi corretamente classificada nos casos sem filtro e *low level*. Uma hipótese desnecessária (Figura 36b) acabou não sendo detectada e obteve o melhor *score* no classificador de dígitos isolados,

Em alguns desses casos, o uso de um simples limiar poderia resolver o problema, no entanto, a estratégia do filtro é muito menos dependente de heurísticas do que utilizar um limiar.

(a) (b)

Figura 36: Hipótese ótima classificada como desnecessária, causando erro de classificação. (a) Hipótese ótima. (b) Hipótese desnecessária.

5.8 Resumo

Este capítulo apresentou os resultados experimentais do método proposto. Foi possível observar através desses resultados as principais vantagens do método proposto - redução do custo computacional e melhora no desempenho do classificador de dígitos isolados bem como alguns erros. No capítulo seguinte, será feita uma conclusão deste trabalho.

6 Conclusão

Nesta dissertação foi proposto um método de classificação de pontos de segmentação. O uso deste método pode ser justificado pelo número de pontos de segmentação gerados pelos algoritmos baseados na estratégia segmentação-reconhecimento. Os resultados obtidos são promissores e dão suporte à aplicabilidade deste método:

- Os experimentos realizados mostram que o método proposto reduz de maneira significativa o custo computacional. Essa redução se dá de duas maneiras, uma através da redução geral de número de chamadas de classificadores e outra através da migração de chamadas de um classificador complexo, que utiliza 132 características e modela 10 classes (classificador de dígitos isolados) para um que utiliza 42 características e modela somente duas classes. Essa redução do número de chamadas se deu através da redução do número de hipóteses de segmentação.

- Essa redução do número de hipóteses de segmentação foi obtida através de um método adaptativo, baseado em uma metodologia pré-definida e não através do uso de heurísticas como ocorre nos métodos de segmentação baseados em dissecação [3].

- Apesar dos experimentos terem sido realizados em cadeias numéricas de dois dígitos, com conexões simples, o uso de MCA torna possível a expansão deste método para cadeias de qualquer comprimento e com qualquer tipo de conexão, uma vez que o processo de identificar "sobre-segmentação" não depende nem do tamanho da cadeia nem da quantidade de pontos de segmentação. Uma vez identificado um caso de "sobre-segmentação" numa cadeia, basta descartá-la.

Apesar dos benefícios obtidos pelo método proposto, ainda há trabalho a ser realizado. Algumas atividades a serem realizadas futuramente incluem:

- Avaliar o método proposto em cadeias maiores que dois dígitos.

- Avaliar o método proposto em cadeias com ocorrências de conexões múltiplas.
- Utilizar um mecanismo de seleção de características.
- Utilizar um mecanismo de seleção de características.
- Avaliar o desempenho do método proposto em outros algoritmos de segmentação.

Referências

[1] *Probabilistic Outputs for Support Vector Machines and Comparison to Regularized Likelihood Methods*. MIT Press, 1999.

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

[3] R.G. Casey and E. Lecolinet. A survey of methods and strategies in character segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence,*, 18(7):690–706, 1996.

[4] C.C. Chang and C.J. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

[5] Y.K. Chen and J.F. Wang. Segmentation of single- or multiple touching handwritten numeral string using background and foreground analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1304–1317, 2000.

[6] T. Fawcett. Roc graphs: Notes and practical considerations for researchers, 2004.

[7] R. Fenrich and S. Krishnamoorthy. Segmenting diverse quality handwritten digit strings in near real-time. In *USPS*, pages 523–537, 1990.

[8] H. Fujisawa, Y. Nakano, and K. Kurino. Segmentation methods for character recognition: from segmentation to document structure analysis. In *Proc. of IEEE*, volume 80, pages 1079–1092, 1992.

[9] S. Gunter and H. Bunke. Creation of classifier ensembles for handwritten word recogntion using feature selection algorithms. In *Proc. of 8th IWFHR*, pages 183–188, 2002.

[10] Anil K. Jain, Robert P. W. Duin, and Jianchang Mao. Statistical pattern recognition: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1):4–37, 2000.

[11] A. Britto Jr. *A Two-Stage HMM-Based Method For Recognizing Handwritten Numeral Strings*. PhD thesis, Pontifícia Universidade Católica do Paraná, 2001.

[12] T.C.W. Landgrebe, D.M.J. Tax, P. Paclik, R.P.W. Duin, and C.M. Andrew. A combining strategy for ill-defined problems. In *Fifteenth Annual Symposium of the Pattern Recognition Association of South Africa*, pages 57–62, 2004.

[13] M. Last, H. Bunke, and A. Kandel. A feature-based serial approach to classifier combination. *Pattern Analysis and Aplications*, 5(4):385–398, 2002.

[14] Y. Lei, C. S. Liu, X.Q. Ding, and Q. Fu. A recognition based system for segmentation of touching handwritten numeral strings. In *Proceedings of IFHWR 9*, pages 294–299, 2004.

[15] C.M. Nunes, A.S. Britto Jr., C.A.A. Kaestner, and R. Sabourin. Feature subset selection using an optimized hill climbing algorithm for handwritten character recognition. In *SSPR/SPR*, pages 1018–1025, 2004.

[16] L.S. Oliveira. *Automatic Recognition of Handwritten Numerical Strings*. PhD thesis, École de Technologie Supérieure, 2003.

[17] L.S. Oliveira, A. Britto Jr., and R. Sabourin. A synthetic database to assess segmentation algorithms. In *8th International Conference on Document Analysis and Recognition (ICDAR 2005)*, pages 207–211, Seoul, South Korea, August 29-September 1st 2005. IEEE CS Press.

[18] U. Pal, A. Belaïd, and C. Choisy. Touching numeral segmentation using water reservoir concept. *Pattern Recognition Letters*, 24:261–272, 2003.

[19] J. Sadri, C.Y. Suen, and T.D. Bui. Automatic segmentation of unconstrained handwritten numeral strings. In *Proceedings of IWFHR 9*, pages 317–322, 2004.

[20] D.M.J. Tax. *One-class classification*. PhD thesis, Delft University of Technology, 2001.

[21] Francesco Tortorella. Reducing the classification cost of support vector classifiers through an roc-based reject rule. *Pattern Anal. Appl.*, 7(2):128–143, 2004.

[22] V. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag New York, Inc, 1995.

[23] E. Vellasques, L.S. Oliveira, R. Sabourin, A.S. Britto, and A.L. Koerich. Modeling segmentation cuts using support vector machines. In *Proceedings of IWFHR 10*, 2006.

[24] X. Wang, V. Govindaraju, and S. Srihari. Holistic recognition of touching digits. In *Proceedings of IFHWR 6*, pages 295–303, 1998.