Allan Reffson Granja Lima

Máquinas de Vetores Suporte na Classificação de Impressões Digitais

Fortaleza, Ceará

2002

Allan Reffson Granja Lima

Máquinas de Vetores Suporte na Classificação de Impressões Digitais

Dissertação de Mestrado

Orientador:
Fernando de Carvalho Gomes

Universidade Federal do Ceará
Departamento de Computação

Fortaleza, Ceará

2002

Tese de Mestrado sob o título *"Máquinas de Vetores Suporte na Classificação de Impressões Digitais"*, defendida por Allan Reffson Granja Lima e aprovada em 10 de dezembro de 2002, em Fortaleza, Estado do Ceará, pela banca examinadora constituída pelos doutores:

Prof. Dr. Fernando de Carvalho Gomes
Orientador

Prof. Dr. Stan Matwin
Universidade de Ottawa

Prof. Dr. Francisco Tavares Silva
Instituto Nacional de Pesquisas Espaciais

Resumo

Uma abordagem para classificar impressões digitais é apresentada nessa tese. Impressões digitais podem ser agrupadas em cinco categorias: *Plain Arch* , *Tented Arch* , *Left Loop* , *Right Loop* e *Whorl* . Um algoritmo para calcular o campo direcional de uma impressão digital é usado para extrair o vetor características de uma imagem de impressão digital e um classificador Máquina de Vetores Suporte (SVM) é executado tendo como entrada o vetor de características. O classificador SVM é testado sobre o banco de dados NIST-4 de 4.000 imagens de impressões digitais, que é referência mundial na área e foi coletado pelo *"Federal Bureau of Investigation" - FBI.*

Agradecimentos

- A Deus!
- À Niza, minha noiva e futura esposa, é claro!
- À meus pais!
- Ao meu orientador professor Fernando de Carvalho Gomes, pela segunda chance!
- FUNCAP, pelo suporte financeiro!
- A FIC - Faculdade Integrada do Ceará, mais precisamente aos Professores Cleto Prata, pela oportunidade de lecionar e, assim melhorar meu exercício de docência e, claro, o suporte financeiro.
- Aos amigos Edson Ricardo (Pagodeiro), Freud (Alex Sandro) pela companhia e ajuda no LCG.
- Ao criador da lista de inutilidades *Peleja*, Yuri Abitbol vulgo Vizir.
- Aos professores Creto, Ricardo Correa, Claudia Linhares e Manoel Campelo pela excelência no ensino.
- Ao Instituto Atlântico, especialmente ao Superintendente Eduardo Bernal e Gerente de Tecnologia Adriano, pela parceiria e credibilidade.
- A minha coordenadora de projeto Jaudênia pelos constantes puxões de orelha com relação a essa tese. Obrigado Jau, pela sua preocupação e ajuda com a realização desse trabalho.
- Ao professor doutor Nivando pelas incontáveis dicas em LaTeX, Linux e processamento digital de imagens.

Sumário

Lista de Figuras

Lista de Tabelas

1 Introdução

Com o avanço da humanidade e os benefícios da tecnologia, a população mundial cresceu e junto com ela a criminalidade. Cada vez mais, as instituições investem em sistemas de segurança, mesmo sabendo que tais sistemas possuem limitações. Isso fomentou novas tecnologias dedicadas à segurança, especialmente para o caso de identificação de pessoas. Entretanto, não são recentes as tentativas de construção de tais mecanismos. Formas automáticas e precisas de identificação ou reconhecimento de indivíduos tornaram-se, ao longo do tempo, ferramentas de auxílio fundamentais, tanto na área forense, como em organizações. Isso fez surgir uma nova área de pesquisa em ciência da computação: *Sistemas Automáticos de Identificação ou Verificação de Indivíduos.*

1.1 Biométrica

A *biométrica* tem se tornado um tópico cada vez mais importante, em face das crescentes necessidades de proteção e segurança no mundo hodierno. Ela é definida como a(s) característica(s) ou métrica(s) que identifica(m) de forma única um indivíduo. Qualquer característica fisiológica ou comportamental pode ser usada para construir uma métrica desde que satisfaça, total ou parcialmente, os seguintes requerimentos [1]:

1. **Universalidade:** todas pessoas devem ter a(s) característica(s);
2. **Unicidade:** duas pessoas não podem ter a(s) mesma(s) característica(s);
3. **Permanência:** a(s) característica(s) deve(m) ser invariante(s) com o tempo;
4. **Coletabilidade:** a(s) característica(s) pode(m) ser medida(s) quantitativamente.

Exemplos de sinais biométricos, não invasivos, são impressão digital, geometria da mão, íris, retina, face, voz e DNA.

Para se escolher uma biométrica, é necessário avaliar os requerimentos (citados anteriormente) da característica. Por exemplo: a íris é universal (quase todos os humanos as possuem), tem um grande índice de unicidade (no ocidente o índice de íris idênticas é praticamente nulo), mas tem baixa permanência (a íris muda com a idade, principalmente se o sujeito tem diabetes) e média coletabilidade (exige-se a ausência de óculos, lentes de contato e infecções, assim como imobilidade diante da câmera).

Impressões digitais são universais, não existem duas impressões absolutamente idênticas para o mesmo dedo (unicidade), têm alto índice de permanência (não muda jamais, porém podem ser prejudicadas por cicatrizes e queimaduras de terceiro grau) e altíssima coletabilidade [1]. Justamente devido à sua coletabilidade, as impressões digitais, numa tentativa de fraude, podem ser substituídas por moldes de borracha, ou mesmo por um dedo não vivo. Porém, *"Scanners"* de impressões digitais de última geração acoplam sensores capacitivos que medem a temperatura e a constante dielétrica da amostra. Uma vez que a combinação temperatura/constante dielétrica do dedo vivo é única, reduz-se a possibilidade de fraudes com outros materiais.

A identificação biométrica através de impressões digitais recebe maior atenção, devido à sua facilidade de uso, à universalidade de sua utilização e à disponibilidade de registros abrangentes, em alguns casos, como no Brasil, atingindo praticamente a totalidade da população economicamente ativa. Especial atenção deve ser dada à confiabilidade e à performance de sistemas de identificação que freqüentemente devem ser capazes de pesquisar, com eficiência e precisão, bases de dados de grandes dimensões, mantidas de forma distribuída [1, 6–8].

Alguns exemplos da aplicabilidade Sistemas de Identificação/Verificação de Impressões Digitais são:

- controle de acesso,
- segurança financeira (*"financial security"*),
- verificação da identidade de visitantes de prisioneiros (ou prisões),
- pagamento de benefícios,
- controle de embarque e desembarque em sistemas de transporte e turismo,
- acesso a redes e PCs,
- aplicações relacionadas com *"ATMs"* (bancos, cartões de crédito),

- acesso a clube exclusivos tais com golfe e esportes,
- cartões de identidade nacional (*"smartcard"*),
- acesso a carros, celulares, Internet, eleição,
- verificação de compra de armas de fogo e
- carteira de motorista.

1.2 Reconhecimento de Impressão Digital

Existem dois tipos de sistemas de reconhecimento de impressões digitais [7]:

- *Sistemas de Verificação* usam impressões digitais para verificar se um indivíduo é quem ele afirma ser. A impressão de entrada é comparada com uma impressão digital de referência que é obtida em um banco de dados, a partir de uma dada identidade. O processo de busca na base de dados dá-se de forma rápida, uma vez que a chave de acesso é entrada *a priori*. O sistema retorna sucesso se a comparação indicar semelhança nas duas impressões e retorna fracasso em caso contrário (Figura 1). O usuário fornece como entrada uma identificação (número de matrícula ou cartão magnético) e uma impressão digital que é coletada em um *"scanner"*. A imagem digitalizada é submetida a um algoritmo que irá extrair características da impressão digital. Enquanto isso, a identificação é usada para recuperar em um banco de dados o modelo de características da impressão digital correspondente. De posse dos dois modelos (o coletado e o recuperado), um algoritmo de comparação é aplicado. Se o número de semelhanças for superior ou igual a um limiar, então a pessoa é quem ela diz ser. Esse tipo de sistema tem sido usado para verificar o acesso de um grupo de pessoas a ambientes restritos ou sistemas.

- *Sistemas de Identificação* não assumem nenhuma identidade *a priori*, apenas a impressão digital do indivíduo a ser identificado. Em seguida, uma busca em uma volumosa base de impressões é efetuada, a fim de encontrar uma ou mais impressões semelhantes (processo de emparelhamento). Isto irá identificar o indivíduo ou apresentar uma reduzida relação de impressões. Nesse caso uma segunda chave, como a impressão da palma da mão, por exemplo, poderá servir para identificar o indivíduo.

  Como mostrado na Figura 2, uma imagem de impressão digital (completa ou parcial) é submetida ao sistema. A imagem digitalizada é submetida a um algoritmo

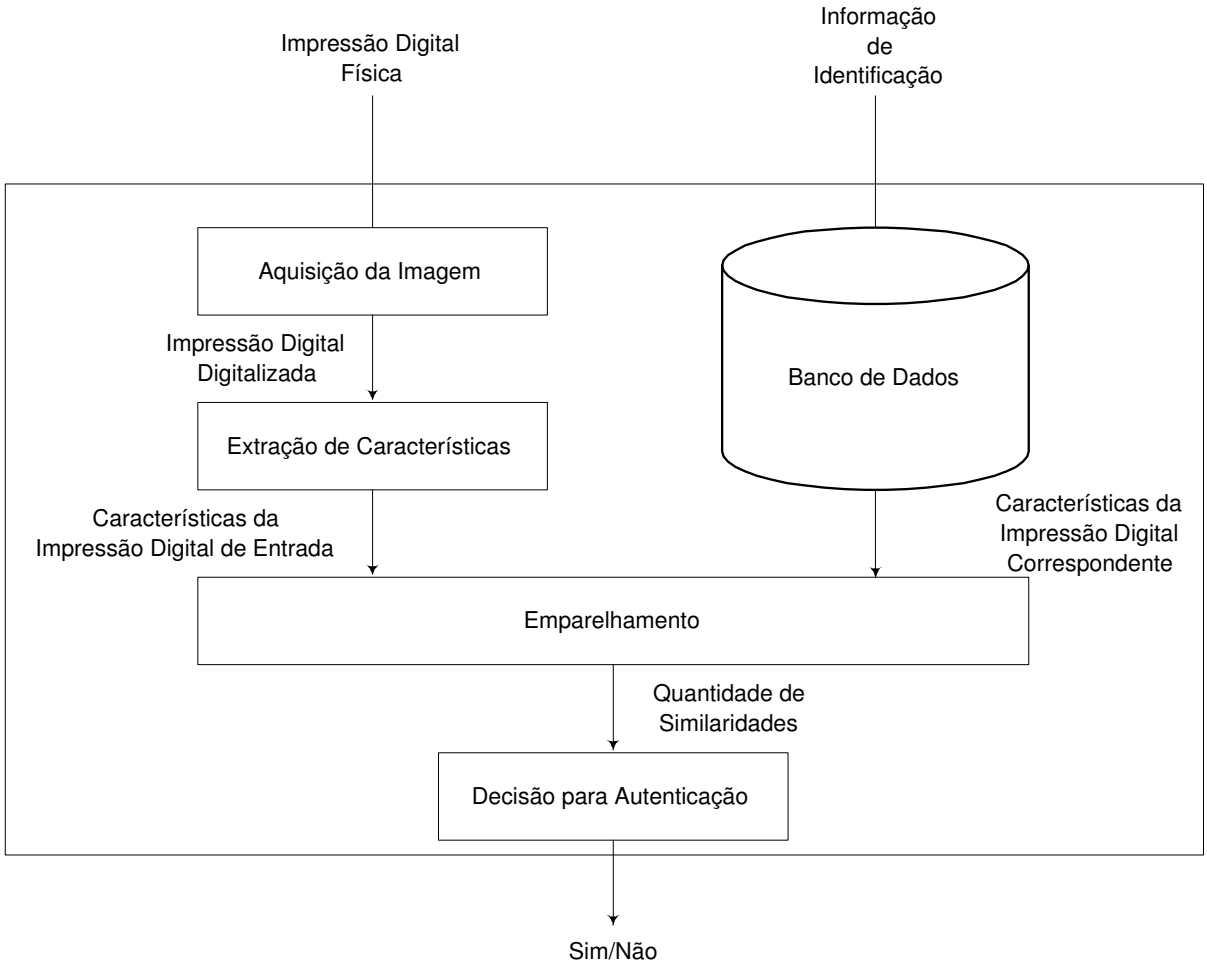


Figura 1: Diagrama de blocos de um sistema de verificação de impressão digital [1].

que irá extrair características da impressão digital. As características globais são repassadas ao algoritmo responsável por classificar a digital em alguma das categorias pré-estabelecidas. Se a categoria determinada for laço ou espiral, então um algoritmo de sub-classificação é aplicado às características, caso contrário a classe já está determinada. Após encontrar qual grupo de impressão digital pertence a imagem então uma consulta ao banco de dados é realizada para recuperar as impressões digitais pertencentes a esse grupo. A seguir um algoritmo de comparação é aplicado para encontrar as semelhanças entre a digital não identificada e cada uma das digitais recuperadas no banco de dados. Ao término da comparação, uma única ou um conjunto de impressões digitais é então retornado. No caso de uma lista de impressões digitais, um especialista no assunto ou outra medida pode ser usada para confirmar se a digital sem dono pertence ou não a lista. A principal aplicação reside na identificação de suspeitos, visando a obtenção dos registros criminais dos mesmos, caso existam, independentemente da identidade presumida. Outra aplicação

importante diz respeito à identificação de indivíduos presentes na cena do crime, a partir de fragmentos de impressões.

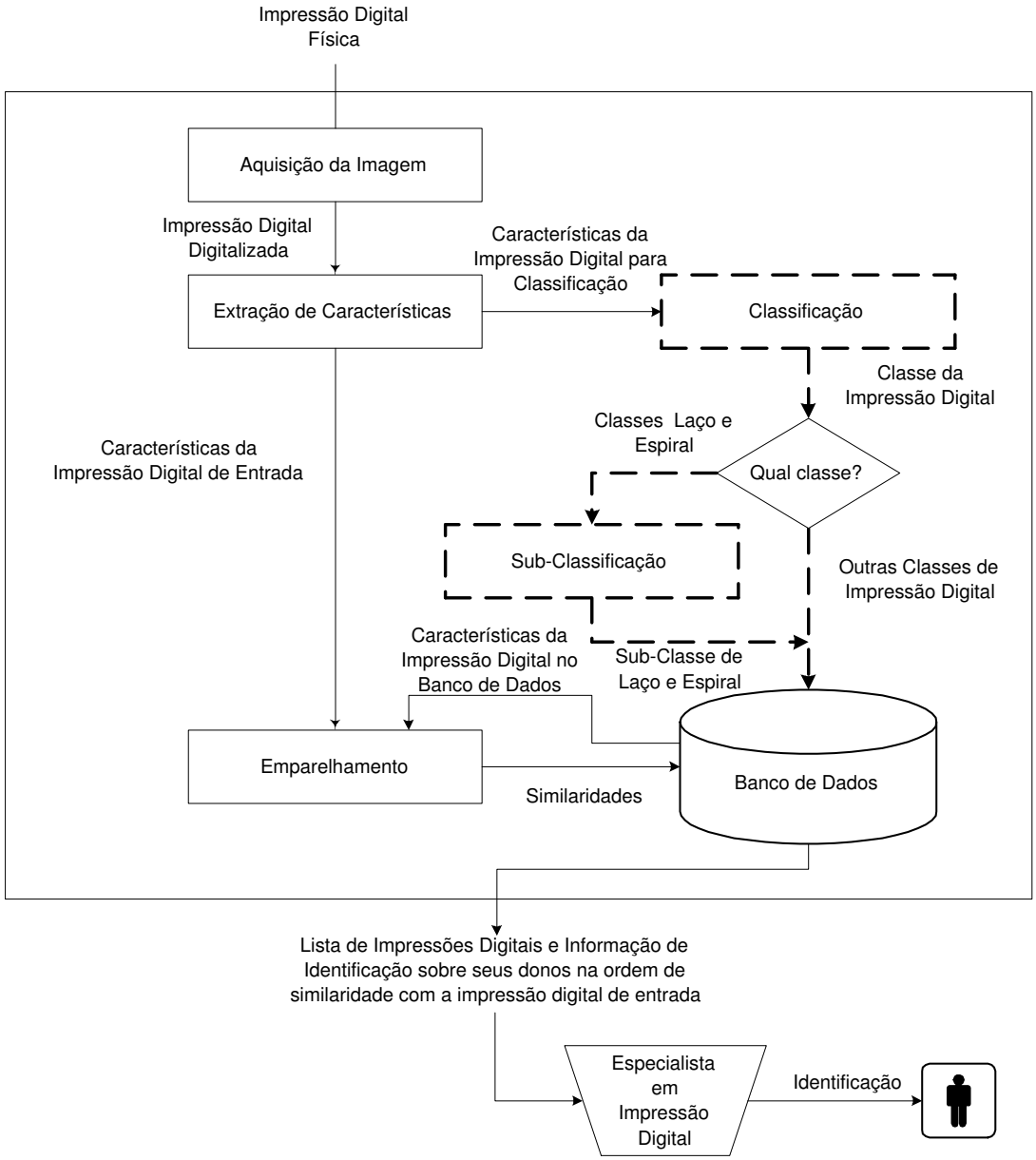


Figura 2: Diagrama de blocos de um sistema de identificação de impressão digital [1].

Tanto a verificação, como a identificação através de impressões digitais, exigem um tratamento preliminar da imagem relativa à impressão. Na seqüência, serão extraídas características (ou minúcias) a partir da imagem tratada. No caso de identificação, devido ao grande volume de dados presente na base, o processo de emparelhamento deve ser segmentado através de técnicas de classificação de impressões. Uma vez encontrada a classe, a busca segmentada é realizada através de algoritmo paralelo de busca de emparelhamentos.

1.3 Aquisição de Impressões Digitais

Os métodos mais comuns para adquirir digitais são impressão digital em tinta e *"scanner"*. Impressões Digitais em tinta mantêm-se como um dos mais populares e antigos métodos de aquisição de impressões digitais. Primeiro, uma impressão em tinta da impressão digital é adquirida após umedecimento dos dedos das mãos em tinta e, em seguida, impressa sobre papel. Segundo, os papéis com os padrões de impressões digitais são digitalizados. Esse método pode resultar em imagens de digitais bastante distorcidas e seria necessário um profissional treinado para tratar da imagem. Obviamente, esse método consome muito tempo e é inviável para sistemas de identificação e verificação *"on-line"* de impressão digital. Em contrapartida, *"scanners"* de impressões digitais são capazes de capturar diretamente impressões digitais em formato digital [7]. Esse método fornece imagens melhores e não necessitam de especialistas, mas imagens altamente distorcidas são possíveis devido a pele seca, oleosa, suja, úmida ou doente. A Figura 3 mostra duas impressões digitais de um mesmo dedo adquiridas sob diferentes situações (pele oleosa e pele seca). Em ambos os métodos, a imagem obtida é de alta resolução (aproximadamente 500 dpi) e tons de cinza (geralmente 256). Em todos os métodos disponíveis, as seguintes variações entre duas cópias da mesma impressão digital são possíveis [1]:

(a) pele oleosa *(b) pele seca*

Figura 3: Aquisição: duas imagens de uma mesma impressão digital.

1. translação e rotação devido ao posicionamento diferente da impressão digital no dispositivo de entrada;
2. escala espacial devido ao declive causado pela pressão na superfície;
3. diferença de contraste por causa do declive causado pela pressão e a densidade da tinta nos métodos baseados em tinta;

4. impressão digital incompleta, ou seja, imagem contendo apenas parte da impressão digital;

5. transformação de cisalhamento se o dedo exercer uma força de cisalhamento diferente na superfície;

6. perturbações locais, por exemplo, translação, rotação ou escala devido a uma pressão não uniforme ou força de cisalhamento;

7. quebras ou manchas causadas pelo contato não uniforme e densidade de tinta não uniforme em métodos baseados em tinta;

8. distorções não permanentes ou semi-permanentes tais como idade, doenças, cicatrizes, suor etc.

1.4 Características de Impressões Digitais

Para aplicarmos os algoritmos de aprendizagem sobre impressões digitais precisamos de atributos ou características que são comuns em impressões digitais. Quando a tarefa é identificação, as características mais comuns são *minúcias* (Seção 1.4.1) e em alguns casos *poros* (Seção 1.4.2). Já quando a tarefa é classificação, então as características usadas são os *ponto singulares core* e *delta* (Seção 1.4.3).

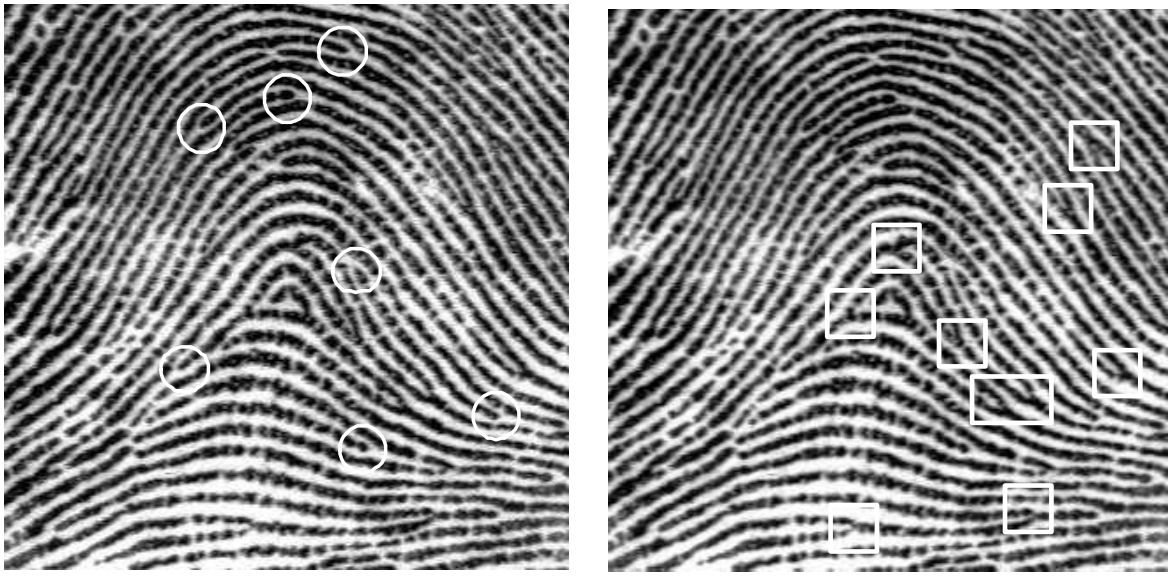
1.4.1 Minúcias

A forma mais comum de representação usada em identificação de impressões digitais são as características de Galton, estabelecidas pelos estudos de Sir F. Galton, em 1892 [1]. Uma linha[1] é definida como um simples segmento curvo e um vale é a região entre duas linhas adjacentes. As linhas e os vales se alternam em uma impressão digital, fluindo em uma direção local constante. Descontinuidades locais são chamadas *minúcias*. Galton define quatro tipos de minúcias. Mais tarde essas características foram refinadas e estendidas. Dezoito tipos diferentes de características de impressão digital estão enumeradas. Em sistemas automáticos, o conjunto de tipos de minúcias estão restritos a dois tipos: terminações de linha e bifurcações de linha (Figura 4); outros tipos de minúcias podem ser expressados em termos desses dois tipos.

[1]Do inglês, *"ridge"*

Uma impressão digital é representada pelas localizações, tipos e alguns atributos, tais como, orientação da minúcia. Essa representação reduz o problema de emparelhamento entre minúcias ao problema de emparelhamento em grafos. Verificação de impressão digital determina se duas impressões digitais são do mesmo dedo ou não. Por exemplo, um gabarito da impressão digital de um usuário é armazenando em um banco de dados com um número de identificação. O usuário fornece esse número de identificação ao sistema através de um teclado ou cartão e o sistema recupera o gabarito. Esse gabarito é então comparado com a impressão digital coletada *"on-line"* por um *"scanner"* para identificar a autenticidade do usuário.

Uma centena de anos de estudo em impressões digitais garante a singularidade da representação baseada em minúcia para uma população muito grande de humanos. Existem entre 50 a 150 minúcias sobre uma imagem completa de uma digital e, em sistemas automáticos, 10 minúcias idênticas são consideradas suficiente para estabelecer identidade. Os problemas dessa representação advêm da incerteza de algoritmos de extração de minúcias confiáveis e da dificuldade em se definir quantitativamente uma comparação confiável entre dois conjuntos de minúcias [1]. Duas outras dificuldades podem estar presentes na imagem: a presença de falsas minúcias e ausência de minúcias que estão presentes na digital.



(a) *(b)*

Figura 4: Minúcias: em (a) os círculos mostram as terminações e em (b) os retângulos mostram as bifurcações.

1.4.2 Poros

Uma característica menos comumente usada são os *poros*[2] sobre a superfície das linhas que também satisfazem a unicidade. [10]. Eles são usados como características auxiliares em alguns sistemas baseados em minúcias. A impressão digital é representada pela localização de poros e orientação local da linha sobre a qual eles residem e o problema novamente é reduzido a um problema de comparação de ponto [1].

1.4.3 Pontos Singulares

Codificar todo o padrão de linha e usar a imagem em tons de cinza são outras representações possíveis que satisfazem a condição de singularidade, mas a carga computacional é alta nessas representações e elas são também mais sensitivas as variações citadas anteriormente (seção 1.3). Especialmente em verificação *on-line*, a carga computacional é um parâmetro importante. Por essa razão, em alguns sistemas, as representações acima são derivadas de uma pequena, mas consistente parte da impressão digital. Tais simplificações resultam em perda do grau de unicidade, mas é razoável para propósitos de classificação e para sistemas de autenticação (verificação) que são usados por algumas centenas de pessoas [1].

Características que satisfazem a condição de singularidade podem ser usadas em comparação e classificação, mas há outras características que são mais adequadas para classificação, como veremos adiante na seção 1.5. A representação mais comum é o mapa de direção ou campo direcional (Figura 5) que é uma matriz de direções representando a orientação de linha e vale em cada localização sobre a imagem da impressão digital [1].

Os dois tipos especiais de características relacionadas ao mapa de direção são os pontos *core* e *delta* que são denominados de pontos de singularidade. O ponto *core* é definido como o ponto mais ao topo sobre a linha curva mais interna e um ponto *delta* é o centro de uma região triangular onde três diferentes direções se encontram, Figura 6. O número e a localização desses pontos são usados na definição das Classes de Henry. Alguns trabalhos mostram que usando apenas as localizações e os tipos desses pontos, um modelo na forma de um campo direcional que aproxima o mapa direcional pode ser construído. Outros trabalhos usaram funções *"B-splines"* para representar as curvas das linhas, que são úteis para compressão de imagens de impressão digital.

[2]Numa perspectiva histórica, as suspeitas de que poros poderiam ser usados para identificação foram estabelecidas no início de 1912 por E. Locard [9].

Figura 5: Campo Direcional para janelas de 16×16.

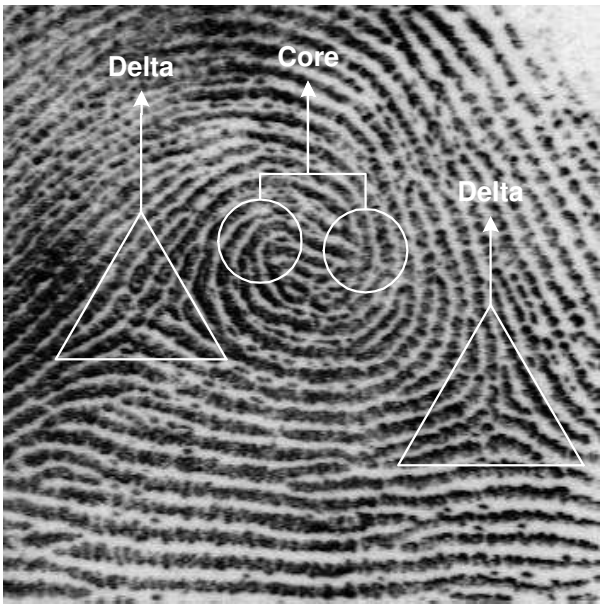


Figura 6: Pontos Core e Delta em uma impressão digital.

1.5 Classificação

O objetivo de classificação de impressões digitais é atribuir, a uma dada impressão digital, uma categoria específica de acordo com suas propriedades geométricas. Os principais propósitos de classificar impressões digitais são facilitar o gerenciamento de grandes bancos de dados de impressões digitais e acelerar o processo de identificação (emparelhamento) de impressões digitais [7].

O esquema mais antigo de classificação, que é usado na identificação manual de impressões digitais, é o esquema de Classificação de Henry (E. Henry, em 1900). Neste esquema, há cinco classes chamadas de *"Plain Arch"*, *"Tented Arch"*, *"Right Loop"*, *"Left Loop"* e *"Whorl"*[3], Figura 7. Essas classes são determinadas pelo fluxo da linha sobre a área core e o número e as localizações relativas dos pontos *core* e *delta*. Raramente, impressões digitais não podem ser atribuídas a nenhuma dessas classes e são assim associadas a uma classe chamada *acidental*. A classificação de Henry é eficiente para classificação manual, pois humanos podem facilmente identificar cada classe, mas nenhum ganho maior é obtido com o agrupamento do banco de dados em seis classes. Além disso, essas classes têm distribuições desiguais [7].

A classificação torna-se essencial para reduzir o conjunto de impressões digitais que o estágio de emparelhamento irá considerar. Desde que impressões digitais são desigualmente distribuídas nas classes (65% das impressões digitais são laços, 30% são espirais) e o número de classes principais é pequeno, o estágio de classificação sozinho não estreita suficientemente a busca no banco de dados de características. Isto motiva um estágio intermediário conhecido como *sub-classificação de impressão digital*. O *"Federal Bureau of Investigation" - FBI* possui um procedimento manual para sub-classificar "*loops*" (contagem de linha) e "*whorl*" (trilha da espiral) apresentado em [11].

Na tarefa de identificação de impressões digitais, apenas a classe correspondente à impressão digital de entrada é procurada e assim o tempo de busca é reduzido, quando as classes são igualmente distribuídas. Um sistema de classificação pode também ser usado como um sistema de reconhecimento para poucas pessoas onde cada classe corresponde a somente um indivíduo.

Sistemas de classificação automática de impressões digitais tentam principalmente implementar o esquema de classificação de Henry, adicionando as outras duas classes. Há quatro abordagens principais que têm sido tomadas para classificação automática de

[3]Foram mantidos os termos em inglês para evitar confusão com a literatura.

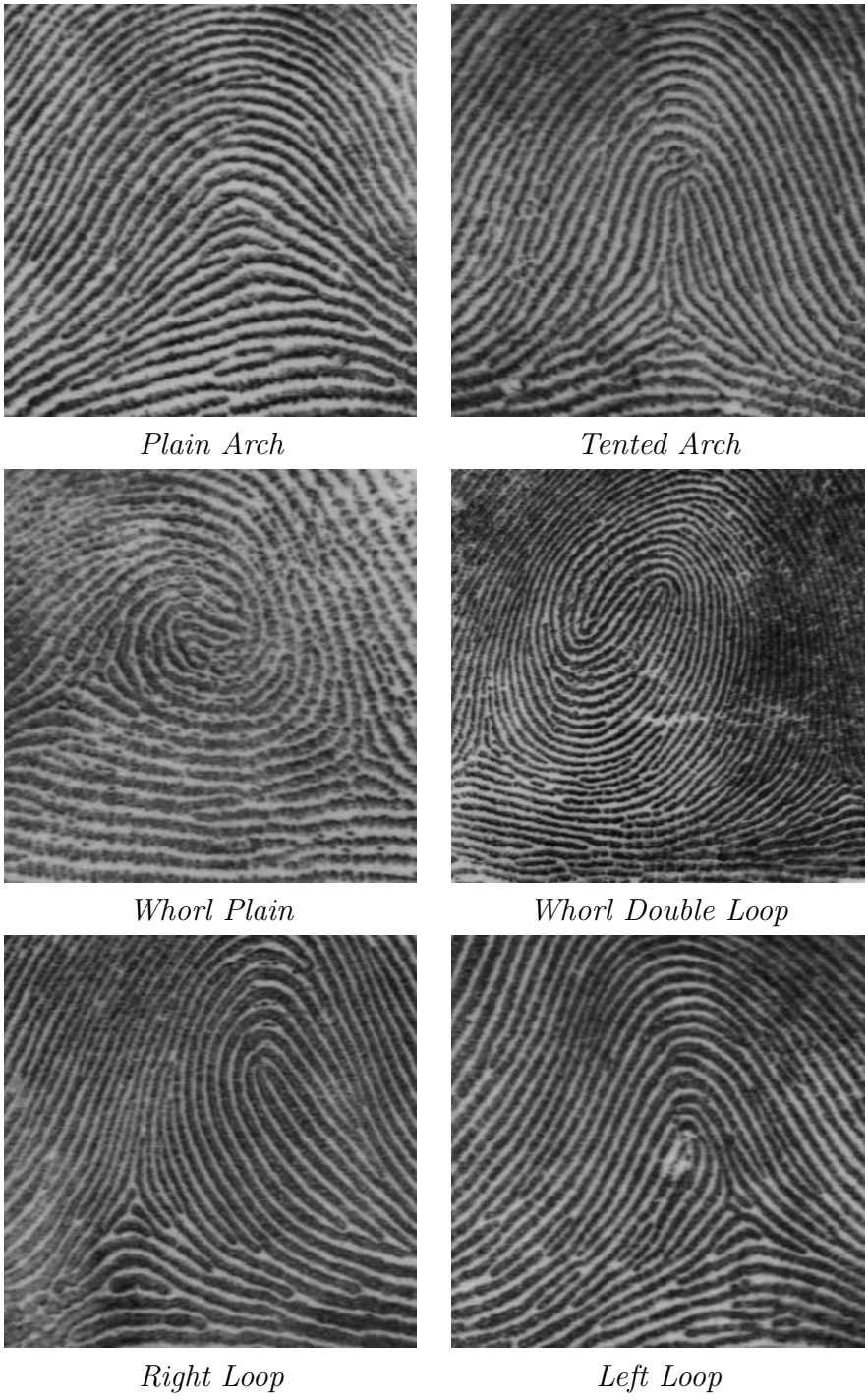


Figura 7: Cada impressão digital na figura pertence a uma subclasse diferente do esquema de classificação do FBI.

impressões digitais: *sintática*, *estrutural*, *redes neurais* e *estatísticas* [1, 12].

1.5.1 Abordagem sintática

Na abordagem sintática, os padrões de linhas e minúcias são aproximados à uma *"string"* de primitivas. Então classes pré-definidas (*e.g.* Classes de Henry) são modeladas como regras de produção ou um conjunto de gramáticas é inferido das amostras de treinamento. Quando um novo padrão chega a string de primitivas é formada e passada para um *"parser(s)"* cuja saída produz a classe do padrão de entrada [1, 12].

1.5.2 Abordagem estrutural

Na abordagem estrutural, as características baseadas em minúcias são extraídas e então representadas usando um grafo. A comparação estrutural é feita pela exploração da topologia das características. Outra abordagem estrutural é usar a topologia de singularidades. Os tipos e as localizações dos pontos core e delta e o fluxo da linha entre pares de pontos core e delta são usados para classificar impressões digitais nas classes pré-estabelecidas [1, 12].

1.5.3 Abordagem por rede neural

Na abordagem rede neural, um vetor de características é construído e classificado por um classificador rede neural. A transformada K-L de vetores direção normalizados são usados como características. Então as classes são formadas em uma maneira não supervisionada usando uma estrutura hierárquica de rede neural consistindo de uma versão modificada de mapas de características auto-organizados. A transformada K-L de vetores direção é usada como entrada para uma rede *"perceptron"* multi-camada que classifica o padrão de entrada em uma das classes. Características *"wedge-ring"*, que são extraídas da transformada de Fourier da imagem da impressão digital, são usadas como entrada para uma rede perceptron multi-camada. Características de textura juntas com algumas características direcionais são usadas como uma entrada para uma rede *"fuzzy"* perceptron multi-camada que é treinada para reconhecer as classes [1, 12].

1.5.4 Abordagem por estatística

Outra abordagem é usar classificadores estatísticos ao contrário de classificadores rede neural. Características *"wedge-ring"* obtidas da transformada hexagonal de Fourier de realce e afinamento de imagem de impressão digital são usadas como entrada para um classificador vizinho mais próximo [1, 12].

1.6 Eficiência de sistemas de reconhecimento de impressão digital

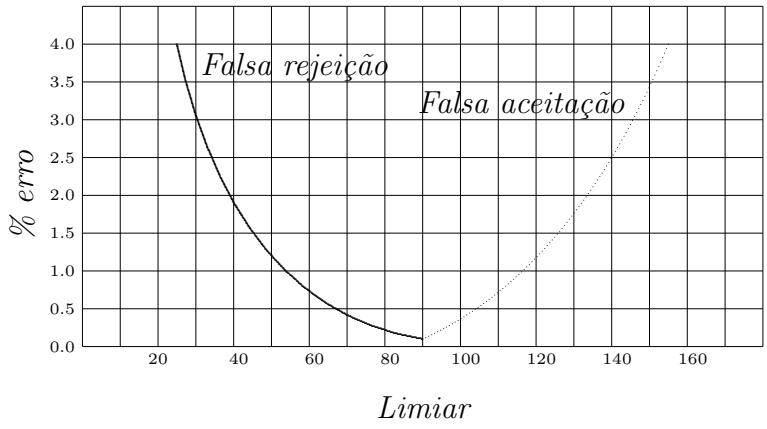


Figura 8: Uma curva de desempenho ideal.

O critério de desempenho de um sistema automático de verificação é baseado na taxa de falsas rejeições (TFR) e na taxa de falsas aceitações (TFA)[4]. Uma nova falsa rejeição será contabilizada quando o sistema não autentica corretamente um usuário autorizado. Por outro lado, uma falsa aceitação significa a autenticação de usuário não autorizado. A Figura 8 mostra a curva de desempenho ideal de um sistema de verificação. Já a Figura 9 apresenta uma curva de desempenho típica, onde TFR e TFA se igualam em 0,6%.

Quando a tarefa é a identificação, a TFA é obtida através da contagem das vezes em que o sistema não apresenta a correta impressão digital ao usuário. Além disso, nesse caso, é crucial a taxa de emparelhamento por segundo (TES), que mede o número de digitais comparadas por segundo. No planejamento de sistema de identificação deve-se levar em conta a performance geral, que é a TES média, quando leva-se em conta as

[4]Em processamento digital de imagens TFR e TFA são chamados de erros de inclusão e exclusão, respectivamente

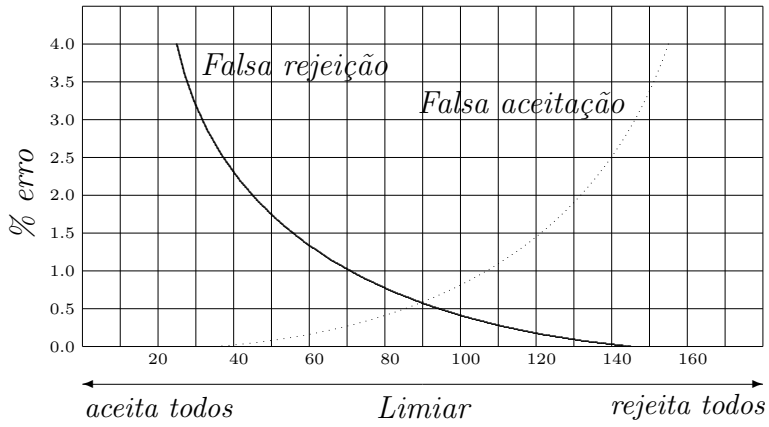


Figura 9: Taxas de falsa aceitação e falsa rejeição. O limiar pode ser ajustado de acordo com o nível de falsas rejeições desejado. O nível de falsas rejeições é zero quando o limiar é ajustado para 140.

características da rede e do hardware envolvido. Para uma discussão detalhada sobre critérios de performance em sistemas biométricos podemos consultar [6].

2 *Processamento Digital de Imagens*

Para manipularmos imagens de impressões digitais em um sistema automático de verificação ou identificação ou ainda de classificação, precisamos saber lidar com imagens de forma a extrairmos as características que precisamos para realizar uma determinada tarefa. Uma pessoa com um pouco de paciência e instrumentos adequados consegue classificar, verificar e até mesmo identificar impressões digitais a olho nú. Entretanto, para um sistema automático a imagem por si só não diz muito e é necessário um processamento da imagem para coletarmos os atributos que interessam. A área da ciência da computação que trata dos algoritmos sobre imagens é chamada de *Processamento Digital de Imagens*. O assunto é muito abrangente e vai muito além dos limites desse texto. Nos concentramos nas definições fundamentais e nas técnicas necessárias para a realização desse trabalho.

2.1 Realce de Imagem

O principal objetivo de técnica da realce é processar uma imagem de modo que a imagem resultante seja mais adequada do que a imagem original para uma aplicação específica. A palavra *específica* é importante, porque ela estabelece no início que as técnicas de realce de imagem são orientadas ao problema[13].

As abordagens para realce de imagens são agrupadas em duas categorias: métodos de domínio espacial e métodos de domínio freqüência. O *domínio espacial* refere-se a imagem plana e abordagens nessa categoria são baseadas na manipulação direta de pixels em uma imagem. As técnicas de processamento no *domínio freqüência* são baseadas na modificação da transformada de Fourier de uma imagem. Técnicas de realce baseadas em várias combinações de métodos dessas categorias não são incomuns[13].

2.1.1 Métodos no Domínio Espacial

O termo *domínio espacial* refere-se ao agregado de pixels que compõe uma imagem e métodos de domínio espacial são procedimentos que operam diretamente sobre esses pixels. Funções de processamento de imagem no domínio espacial podem ser expressas como

$$g(x,y) \quad = \quad T[f(x,y)] \tag{2.1}$$

onde $f(x,y)$ é uma imagem, $g(x,y)$ é a imagem processada e T é um operador sobre f, definido sobre alguma vizinhança de (x,y). Ainda, T pode operar também sobre a entrada de um *conjunto* de imagens, tal como executar a soma pixel-a-pixel de M imagens para redução de ruído [13].

A principal abordagem para definir uma vizinhança sobre (x,y) é usar uma área de sub-imagem quadrada ou retangular centrada em (x,y), como mostrado na Figura 10. O centro da sub-imagem é movido pixel-a-pixel começando, por exemplo, do canto superior esquerdo e um operador é aplicado em cada localização (x,y) para produzir g nesta localização. Embora outras formas de vizinhança, tal como aproximação a um círculo, às vezes é usada, vetores quadrados e retangulares são predominantes porque são de fácil implementação [13].

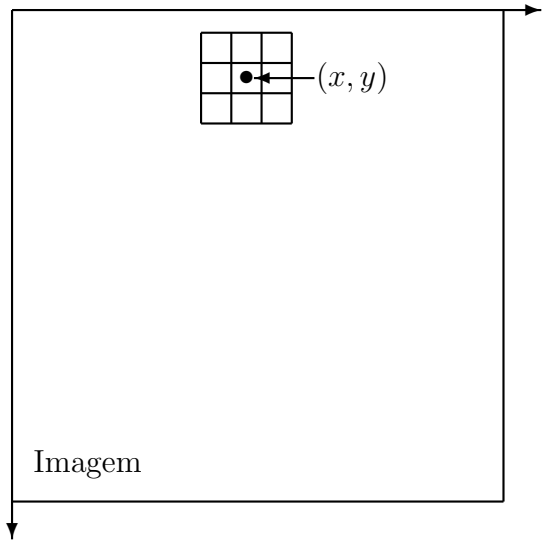


Figura 10: Um vizinhança 3×3 em torno de um ponto (x,y) em uma imagem.

A forma mais simples de T é quando a vizinhança é 1×1. Neste caso, g depende so-

mente do valor de f em (x, y) e T torna-se uma *função (ou mapeamento) de transformação em nível de cinza* da forma

$$s = T(r) \tag{2.2}$$

onde, para simplicidade da notação, r e s são variáveis denotando o nível de cinza de $f(x, y)$ e $g(x, y)$ no ponto (x, y). Por exemplo, se $T(r)$ tem a forma mostrada na Figura 11(a), o efeito desta transformação é produzir uma imagem de alto contraste da imagem original por escurecer os níveis de cinza abaixo de m e clarear os níveis de cinza acima de m na imagem original. Esta técnica, conhecida como *estiramento de contraste* ou *realce*[1], os valores de r abaixo de m são comprimidos pela função de transformação em um intervalo estreito de s em direção ao preto; o efeito oposto ocorre para valores de r acima de m. No caso limite mostrado na Figura 11(b), $T(r)$ produz uma imagem de dois níveis (binária). Devido ao realce, qualquer ponto de uma imagem depende apenas do nível de cinza neste ponto. Técnicas nesta categoria freqüentemente são referenciadas como *processamento de ponto* [13].

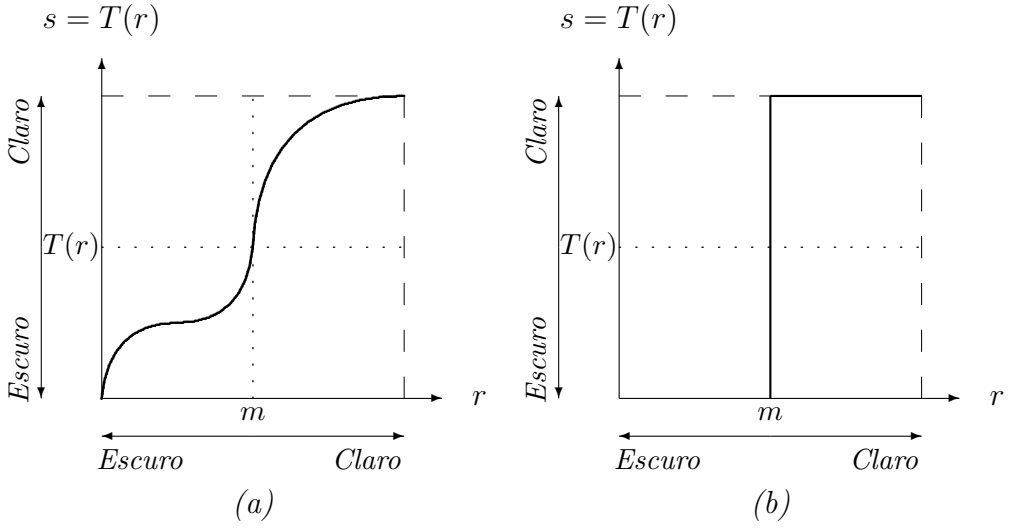


Figura 11: Funções de transformação de nível de cinza para realce de contraste.

Vizinhanças grandes permitem uma variedade de funções de processamento que vão além de realce de imagem. Indiferente da aplicação específica, entretanto, a abordagem geral é deixar os valores de f em uma vizinhança predefinida de (x, y) e determinar o valor de g em (x, y). Uma das principais abordagens nesta formulação é baseada no uso de *máscaras*, também denominadas como *gabaritos, janelas* ou *filtros*. Basicamente, uma máscara é um vetor bidimensional pequeno (por exemplo, 3×3), como mostrado na Figura 13, na qual os valores dos coeficientes determinam a natureza do processo, como imagem

[1] *Do original, contrast stretching*

"*sharpening*". Técnicas de realce baseadas neste tipo de abordagem freqüentemente são referenciadas como *processamento de máscaras* ou *filtragem* [13].

2.1.2 Métodos no Domínio Freqüência

O fundamento das técnicas de domínio freqüência é o teorema da convolução. Seja $g(x, y)$ uma imagem formada pela convolução de uma imagem $f(x, y)$ e um operador linear invariante a posição $h(x, y)$[2], isto é

$$g(x, y) = h(x, y) * f(x, y). \tag{2.3}$$

Então, a partir do teorema da convolução, a seguinte relação de domínio freqüência ocorre:

$$G(u, v) = H(u, v)F(u, v) \tag{2.4}$$

onde G, H e F são transformadas de Fourier de g, h e f, respectivamente, na terminologia de teoria de sistemas lineares, a transformação $H(u, v)$ é chamada de *função transferência* do processo. Em ótica, $H(u, v)$ é chamada de *função transferência ótica* e sua magnitude é chamada a *função de modulação de transferência* [13].

Inúmeros problemas de realce de imagem podem ser expressos na forma da equação (2.4). Em uma aplicação típica de realce de imagem, $f(x, y)$ é dada e o objetivo, depois do cálculo de $F(u, v)$, é encontrar uma $H(u, v)$ tal que a imagem desejada,

$$g(x, y) = \mathcal{F}^{-1}[H(u, v)F(u, v)] \tag{2.5}$$

exiba alguma característica evidenciada de $f(x, y)$. Por exemplo, arestas em $f(x, y)$ podem ser acentuadas usando uma função $H(u, v)$ que enfatiza os componentes de alta freqüência de $F(u, v)$ [13].

A equação (2.3) descreve um processo espacial que é análago ao uso de máscaras discutido na seção 2.1.1. De fato, a expressão de convolução discreta dada na equação (2.3) basicamente é uma representação matemática da mecânica envolvida em implementar o processo de deslocamento de máscara explicado na Figura 10. Por esta razão, $h(x, y)$ é freqüentemente referenciado como uma *máscara de convolução espacial*. O mesmo termo é freqüentemente usado em conexão com a máscara discutida na seção 2.1.1 [13].

[2]Um operador invariante a posição é aquele cujo resultado depende somente do valor de $f(x, y)$ num ponto da imagem e não da posição do ponto.

2.2 Filtragem Espacial

O uso de máscaras espaciais para processamento de imagens geralmente é chamado *filtragem espacial*, e as máscaras são chamadas de *filtros espaciais*.

Os filtros lineares introduzidos nas seções 2.1.1 e 2.1.2 podem ser divididos em filtros *"passa-baixa"*, *"passa-alta"* e *"passa-banda"*. Os filtros passa-baixa atenuam ou eleminam componentes de freqüência alta no domínio de Fourier enquanto deixam freqüências baixas intocadas, isto é, o filtro deixa passar as freqüências baixas. Componentes de freqüência alta caracterizam arestas e outros detalhes nítidos em uma imagem, então o efeito de filtros passa-baixa é borrar a imagem. Similarmente, filtros passa-alta atenuam ou eliminam os componentes de freqüência baixa. Porque esses componentes são responsáveis pela lenta variação de características de uma imagem, tais como contraste geral e intensidade média, a rede de resultados da filtragem passa-alta é a redução dessas características e uma nitidez aparente de arestas e outros detalhes. O terceiro tipo de filtragem, chamado filtragem passa-banda, remove regiões de freqüência selecionados entre freqüências baixa e alta. Esses filtros são usados para restauração de imagens e são de interesse raro em realce de imagem. Nesse trabalho, entretanto, nos interessamos por filtros de passa-baixa e passa-alta, pois estes reduzem imperfeições e realçam arestas respectivamente.

2.2.1 Filtros Média da Vizinhança

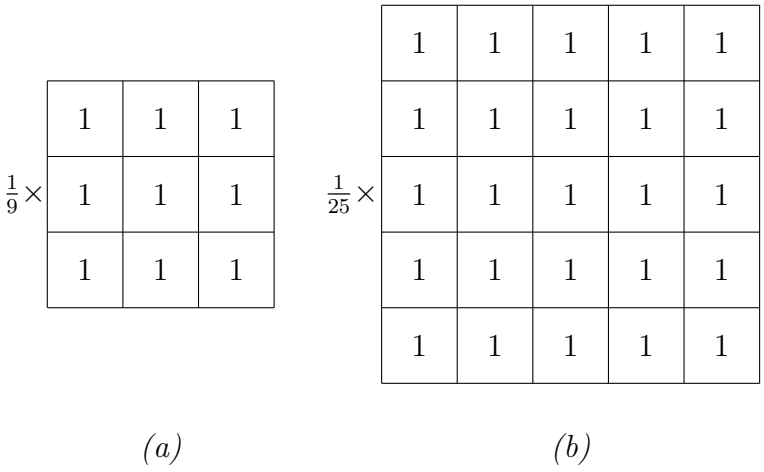


Figura 12: Filtros Média da Vizinhança.

2.2.2 Filtros Derivada

Os filtros passa-alta de derivada ou de diferenciação são usados para tornar os detalhes de uma imagem mais nítidos.

O método mais comum de diferenciação utilizado em aplicações de processamento de imagem é o gradiente. Para uma função $f(x, y)$, o gradiente de f nas coordenadas (x, y) é definido como o vetor

$$\nabla \mathbf{f} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} \tag{2.6}$$

A magnitude desse vetor,

$$\nabla f = \text{mag}(\nabla \mathbf{f}) = \left[\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2 \right]^{1/2} \tag{2.7}$$

é a base para várias abordagens para diferenciação de imagem. Considere a região imagem mostrada na Figura 13(a), onde o z's são os valores de níveis de cinza. A equação (2.7) pode ser aproximada *no ponto* z_5 de inúmeras formas. A mais simples é usar a diferença $(z_5 - z_8)$ no direção x e $(z_5 - z_6)$ na direção y, combinada como

$$\nabla f \approx [(z_5 - z_8)^2 + (z_5 - z_6)^2]^{1/2}. \tag{2.8}$$

Ao contrário de usar quadrados e raízes quadradas, podemos obter resultados similares usando valores absolutos:

$$\nabla f \approx |z_5 - z_8| + |z_5 - z_6|. \tag{2.9}$$

Outra abordagem para aproximar a equação (2.7) é usar as diferenças:

$$\nabla f \approx [(z_5 - z_9)^2 + (z_5 - z_8)^2]^{1/2} \tag{2.10}$$

ou, usando valores absoluto,

$$\nabla f \approx |z_5 - z_9| + |z_5 - z_8|. \tag{2.11}$$

As equações (2.8), (2.9), (2.10) e (2.11) podem ser implementadas usando máscaras de tamanho 2×2. Por exemplo, a equação (2.11) pode ser implementada tomando o valor absoluto da resposta das duas máscaras mostradas na Figura 13(b) e somando o resultado. Essas máscaras são chamadas de *operadores gradiente cruzado de Roberts* [13].

Máscaras de tamanhos par são inadequadas para implementação. Uma aproximação

para a equação (2.7), ainda no ponto z_5, mas agora usando uma vizinhança 3×3, é

$$\nabla f \approx |(z_7 + z_8 + z_9) - (z_1 + z_2 + z_3)| + |(z_3 + z_6 + z_9) - (z_1 + z_4 + z_7)|. \qquad (2.12)$$

A diferença entre a terceira e a primeira linha da região 3×3 aproxima a derivada na direção x e a diferença entre a terceira e a primeira coluna aproxima a derivada na direção y. As máscaras mostradas na Figura 13(c), chamadas de *operadores Prewitt*, podem ser usadas para implementar a equação (2.12). Finalmente, a Figura 13(d) mostra outro par de máscaras (chamado de *operadores Sobel*) para aproximar a magnitude do gradiente [13].

A Figura 14 mostra o resultado de uma operação com os filtros Sobel, Prewitt e Roberts sobre uma imagem de impressão digital.

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

(a) Uma sub-image 3×3

1	0
0	−1

0	1
−1	0

(b) Operadores de Roberts

−1	−1	−1
0	0	0
1	1	1

−1	−1	−1
0	0	0
1	1	1

(c) Operadores de Prewitt

−1	−2	−1
0	0	0
1	2	1

−1	0	1
−2	0	2
−1	0	1

(d) Operadores de Sobel

Figura 13: Uma região 3×3 de uma imagem (os valores de tom de cinza são os z's) e várias máscaras usadas para calcular a derivada de um ponto rotulado com z_5. Observe que todas os coeficientes de máscaras somam 0, indicando uma resposta nula em áreas constantes, como esperado de um operador de derivada
[13].

2.3 Segmentação de Imagens

Esta seção trata das técnicas para extrair informação de uma imagem, também conhecida como *análise de imagem*.

O primeiro passo da análise de imagem geralmente é segmentar a imagem. A segmentação subdivide uma imagem em suas partes constituintes ou objetos. O nível com o

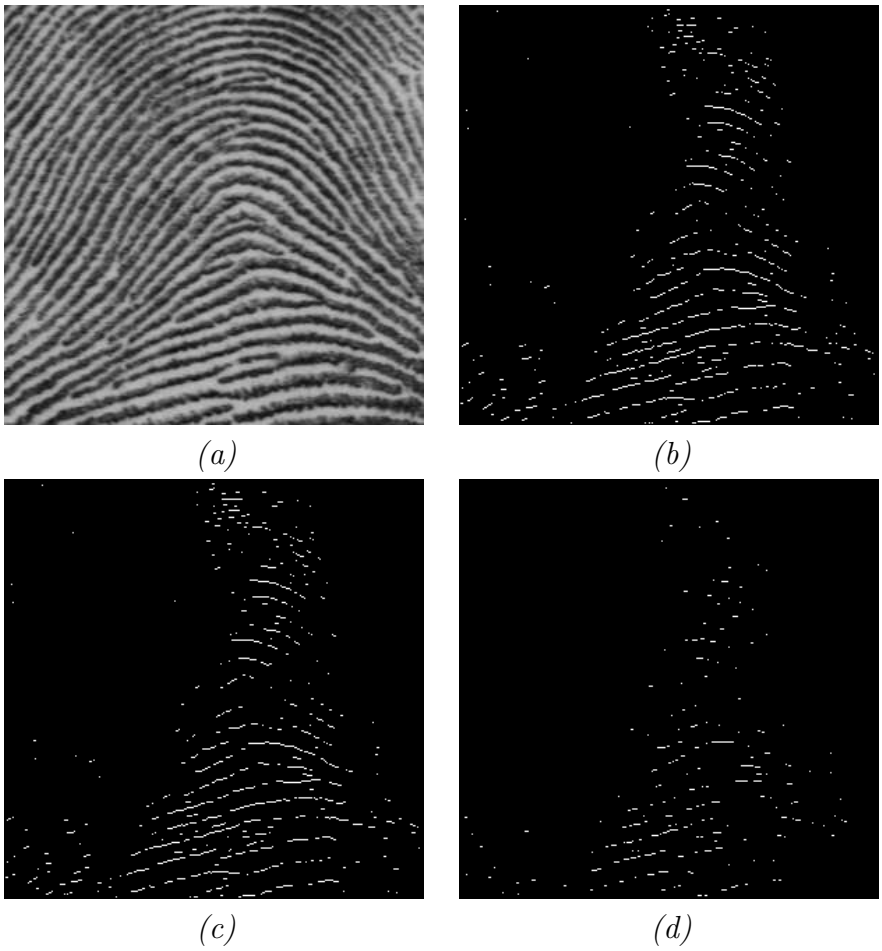


Figura 14: Exemplos de detecção de arestas usando (b) Sobel, (c) Prewitt e (d) Roberts.

qual essa subdivisão é tratada depende do problema a ser resolvido. Isto é, a segmentação deve parar quando o objeto de interesse em uma aplicação for isolado. Por exemplo, em aplicações automáticas de aquisição de alvo do ar-para-a-terra um dos interesses é identificar veículos em uma estrada. O primeiro passo é segmentar a estrada de uma imagem e então segmentar o conteúdo da estrada em objetos com tamanhos pertencentes a um certo intervalo que correspondem a tamanhos de veículos em potencial [13].

Em geral, segmentação autônoma é uma das tarefas mais difíceis em processamento de imagem[3]. Este passo no processo determina o eventual sucesso ou fracasso da análise. De fato, segmentação efetiva raramente falha e leva a uma solução bem-sucedida. Por esta razão, um cuidado considerável deve ser tomado para melhorar a probabilidade de sucesso de uma segmentação difícil. Em aplicações, tais como aquisição de alvo, o projetista do sistema não tem controle sobre o ambiente. Então, a abordagem usual é focar na seleção de tipos de sensores que provavelmente realcem os objetos de interesse enquanto diminuem

[3]Não é considerado da mesma forma em morfologia matemática.

a contribuição de componentes irrelevantes da imagem. Um bom exemplo é o uso de captura de imagem infravermelha para detectar objetos com uma forte transmissão de calor, tais como tanques de guerra em movimento [13].

Algoritmos de segmentação para imagens monocromáticas geralmente são baseados sobre duas propriedades básicas de valores de tom-cinza: descontinuidade e similaridade. Na primeira categoria, a abordagem é particionar uma imagem baseada em mudanças abruptas de nível de cinza. As áreas principais de interesse dentro dessa categoria são detecção de pontos isolados e detecção de linhas e arestas em uma imagem. As abordagens principais na segunda categoria são baseadas em limiar, crescimento da região e divisão e fusão de região. O conceito de segmentação de uma imagem baseada em descontinuidade ou similaridade dos valores de tom de cinza de seus pixels é aplicável a imagens estáticas ou dinâmicas (variantes com o tempo). No último caso, entretanto, o movimento pode freqüentemente ser usado como um sinal poderoso para melhorar o desempenho de algoritmos de segmentação [13].

2.3.1 Detecção de Descontinuidades

Em geral, há três tipos básicos de descontinuidades: pontos, linhas e arestas. Na prática, a forma mais comum de procura por uma descontinuidade é passar uma máscara através da imagem da maneira descrita na seção 2.1. Para a máscara 3×3 mostrada na Figura 15, este procedimento envolve calcular a soma dos produtos dos coeficientes com os níveis de cinza contidos na região cercada pela máscara. Isto é, a resposta da máscara em qualquer ponto da imagem é

$$R = w_1 z_1 + w_2 z_2 + \cdots + w_9 z_9 = \sum_{i=1}^{9} w_i z_i \tag{2.13}$$

onde z_i é o nível de cinza do pixel associado com o coeficiente w_i da máscara. Usualmente, a resposta da máscara é definida com respeito a seu ponto central. Quando a máscara é centrada sobre uma fronteira de pixel, a resposta é calculada pelo uso da vizinhança parcial apropriada [13].

2.3.1.1 Detecção de arestas

Uma aresta é a fronteira entre duas regiões com propriedades distintas de nível de cinza. Assume-se que as regiões em questão são suficientemente homogêneas de forma que a transição entre duas regiões pode ser determinada baseada somente em descontinuidades

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

Figura 15: Uma máscara geral 3×3.

de nível de cinza [13].

Basicamente, a idéia por trás de muitas técnicas de detecção de arestas é a computação de um operador local de derivada. A Figura 16 ilustra esse conceito. A Figura 16(a) mostra a imagem de uma listra clara sobre um pano de fundo escuro, o perfil de nível de cinza de uma linha horizontal da imagem e a primeira e segunda derivada do perfil. Observe a partir do perfil que uma aresta (transição de escuro para claro) é modelada como uma suave, às vezes abrupta, mudança de nível de cinza. Esse modelo reflete o fato de que arestas em imagens digitais são geralmente um pouco embaçadas após um processo de amostragem [13].

A Figura 16(a) mostra que a primeira derivada do perfil de nível de cinza é positiva no início da aresta de uma transição, negativa no final da aresta, e, como esperado, zero em áreas de nível de cinza constante. A derivada segunda é positiva para a parte da transição associada com o lado escuro da aresta, negativo para a parte da transição associada com o lado claro da aresta, e nula (zero) em áreas de nível de cinza constante. Portanto a magnitude da derivada primeira pode ser usada para detectar a presença de uma aresta em uma imagem, e o sinal da segunda derivada pode ser usado para determinar se um pixel de aresta cai sobre o lado escuro ou claro de uma aresta. Observe que a derivada segunda tem valor nulo (zero) cruzando o ponto médio de uma transição em nível de cinza. A derivada segunda fornece uma abordagem poderosa para localizar arestas em uma imagem [13].

Embora a discussão até aqui tenha se limitado a um perfil de dimensão horizontal, um argumento similar aplica-se a uma aresta de qualquer orientação em uma imagem. A primeira derivada de qualquer ponto em uma imagem é obtida usando a magnitude do gradiente naquele ponto. A derivada segunda é similarmente obtida usando o Laplaciano.

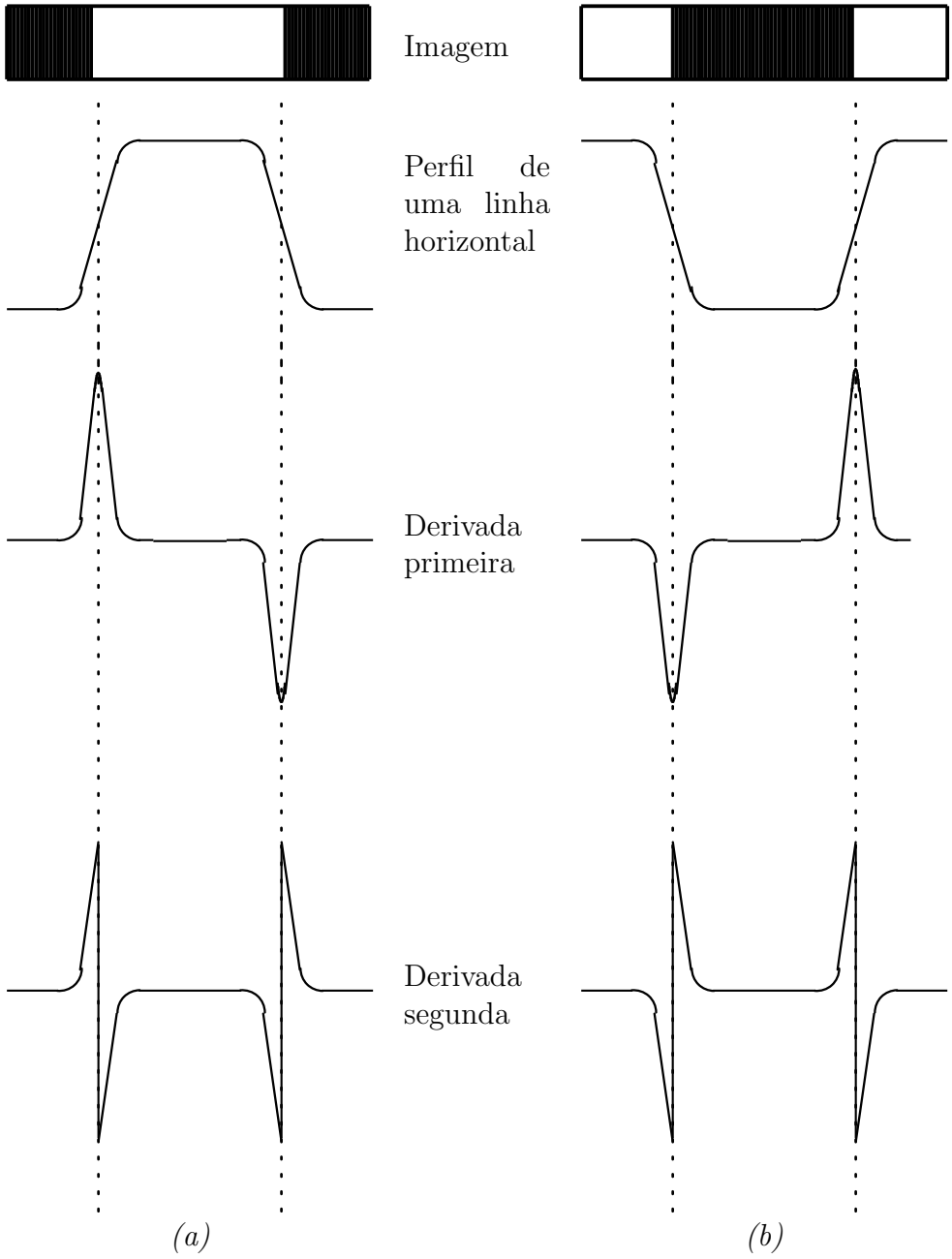


Figura 16: Detecção de arestas pelos operadores de derivada: (a) uma listra clara sobre um fundo escuro; (b) uma listra escura sobre um fundo claro. Observe que a derivada segunda assume valor nulo (zero) ao cruzar a localização de cada aresta.

2.3.1.2 Operadores Gradiente

A seção 2.2.2 introduziu brevemente o conceito do uso do gradiente para diferenciação de imagem. Da equação 2.6, o gradiente de uma imagem $f(x, y)$ no local (x, y) é o vetor

$$\nabla \mathbf{f} = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} \tag{2.14}$$

Em detecção de arestas uma quantidade importante é a magnitude deste vetor, geralmente referenciado como o *gradiente* e denotado por ∇f, onde:

$$\nabla f = \text{mag}(\nabla \mathbf{f}) = [G_x^2 + G_y^2]^{1/2}. \tag{2.15}$$

Esta quantidade é igual a taxa máxima de incremento de $f(x, y)$ por unidade de distância na direção de $\nabla \mathbf{f}$. A prática comum é aproximar o gradiente para valores absolutos:

$$\nabla f \approx |G_x| + |G_y| \tag{2.16}$$

o que é muito mais simples de implementar, particularmente com um hardware dedicado.

A *direção* do vetor gradiente também é uma quantidade importante. Seja $\alpha(x, y)$ representação do ângulo da direção do vetor $\nabla \mathbf{f}$ em (x, y). Então, da análise vetorial,

$$\alpha(x, y) = \tan^{-1}\left(\frac{G_y}{G_x}\right) \tag{2.17}$$

onde o ângulo é medido com respeito ao eixo x.

Observe que das equações 2.14 e 2.15 que o cálculo do gradiente de uma imagem é baseado na obtenção das derivadas parciais $\partial f/\partial x$ e $\partial f/\partial y$ em cada localização de pixel. As derivadas podem ser implementadas em computador de várias formas. Entretanto, os operadores de *Sobel* têm a vantagem de fornecer um efeito de diferenciação e de suavização ao mesmo tempo. Pelo fato de derivadas realçarem imperfeições, o efeito de suavização uma característica de interesse particular dos operadores de *Sobel*. Da Figura 17, as derivadas baseadas nas máscaras operador dos operadores de *Sobel* são

$$G_x = (z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3) \tag{2.18}$$

e

$$G_y = (z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7) \tag{2.19}$$

onde os z's são os níveis de cinza dos pixels sobrepostos pela máscara em qualquer

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

(a)

-1	-2	-1
0	0	0
1	2	1

(b)

-1	0	1
-2	0	2
-1	0	1

(c)

Figura 17: (a) região 3×3 da imagem; (b) máscara usada para calcular G_x, no ponto central da região 3×3; (c) máscara usada para calcular G_y naquele ponto. Essas máscaras são freqüentemente referenciadas como operadores de Sobel.

localização de uma imagem. O cálculo do gradiente na localização do centro da máscara utiliza as equações 2.16 ou 2.17, que dá um valor do gradiente. Para obter o próximo valor, as máscaras são movidas para a próxima localização de pixel e o procedimento é repetido. Assim, depois do procedimento ter sido completado para todas as possíveis localizações, o resultado é uma imagem gradiente do mesmo tamanho da imagem original. Como sempre, os operadores tipo máscara atuando nas bordas de uma imagem são implementados por usar uma vizinhança parcial apropriada.

2.3.1.3 Laplaciano

O *Laplaciano* de uma função de duas dimensões $f(x, y)$ é a derivada segunda definida como

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}. \tag{2.20}$$

Como no caso do gradiente, a equação 2.20 pode ser implementada em computador de várias formas. Para uma região 3×3, a forma mais freqüentemente de gradienate encontrada na prática é

$$\nabla^2 f = 4z_5 - (z_2 + z_4 + z_6 + z_8) \tag{2.21}$$

onde os z's são os níveis de cinza dos pixels sobrepostos pela máscara em qualquer localização de uma imagem. O requerimento básico na definição do *Laplaciano* computacional

é que o coeficiente associado com o pixel central seja positivo e o coeficiente associado com os pixels externos seja negativo. Pelo fato do *Laplaciano* ser uma derivada, a soma dos coeficinetes tem que ser igual a zero. Logo a resposta é zero sempre que o ponto em questão e sua vizinhança têm o mesmo valor. A Figura 18 mostra a máscara espacial que pode ser usada para implementar a equação 2.21.

0	-1	0
-1	4	-1
0	-1	0

Figura 18: Máscara usada para calcular o Laplaciano.

Embora o *Laplaciano* responda a transição em intensidade, ele é raramente usado na prática para detecção de arestas por muitas razões. Como uma derivada de segunda ordem, *Laplaciano* tipicamente é inaceitavelmente sensível a ruído. Além disso, o *Laplaciano* produz arestas duplicadas (veja Figura 16) e é incapaz de detectar a direção da aresta. Por essas razões, o *Laplaciano* geralmente exerce um papel secundário de detector para determinar se um pixel está sobre o lado escuro ou claro da aresta.

Um uso mais elaborado do *Laplaciano* é encontrar a *localização* de arestas usando sua propriedade de cruzando-zero (veja Figura 16). Este conceito é baseado sobre a convolução de uma imagem com o *Laplaciano* de uma função Gausseana de duas dimensões de forma

$$h(x, y) = \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right) \tag{2.22}$$

onde σ é o desvio padrão (também conhecido como operador *Marr-Hildreth*). Seja $r^2 = x^2 + y^2$. Então, da equação 2.20, o *Laplaciano* de h (isto é, a derivada segunda de h com respeito a r) é

$$\nabla^2 h = \left(\frac{r^2 - \sigma^2}{\sigma^4}\right) \exp\left(-\frac{r^2}{2\sigma^2}\right). \tag{2.23}$$

A Figura 19 mostra uma seção de cruzamento desta função circularmente simétrica. Observe a suavidade da função, seu cruzamento em zero quando $r = \pm\sigma$, o centro positivo e as saias negativas. Essa forma é o modelo com o qual a equação 2.21 e a máscara na Figura 18 são baseadas. Quando vista em uma perspectiva tridimensional com o eixo vertical correspondendo a intensidade, a equação 2.23 tem a forma de chapéu mexicano (sombreiro).

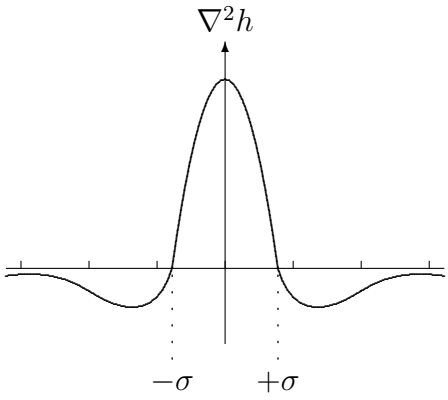


Figura 19: A seção de cruzamento de $\nabla^2 h$, também chamada de Chapéu Mexicano.

3 Campo Direcional

O *campo direcional*[1] descreve a estrutura global, ou forma básica, de uma impressão digital. O campo direcional é definido como a orientação local das estruturas de *l*inha e *v*ale. Ele é usado para *classificação* de impressões digitais [17].

Inúmeras abordagens para estimar o campo direcional de uma impressão digital são conhecidas na literatura. Dentre elas, métodos baseados em *gradiente* fornecem melhor acurácia [14–17]. O vetor gradiente $[\ G_x(x,y) \quad G_y(x,y)\ ]^T$ é definido como:

$$\begin{bmatrix} G_x(x,y) \\ G_y(x,y) \end{bmatrix} = \nabla I(x,y) = \begin{bmatrix} \frac{\partial I(x,y)}{\partial x} \\ \frac{\partial I(x,y)}{\partial y} \end{bmatrix}$$

onde $I(x,y)$ representa a imagem em tom de cinza. A notação de gradiente pode ser melhor explicada quando os valores de pixel são usados para indicar altura num cenário contínuo bidimensional. Neste caso, o vetor gradiente é o vetor que aponta na direção descendente mais íngrime e o comprimento do vetor gradiente é a medida para o declive. Gradientes podem ser considerados como orientações elementares em cada pixel da imagem [17].

O campo direcional, a princípio, é perpendicular aos gradientes. Entretanto, os gradientes são orientações em escala de pixel, enquanto o campo direcional descreve a orientação das estruturas linha e vale, que é uma escala maior ou menos detalhista. Por essa razão, o campo direcional pode ser derivado dos gradientes pela execução de alguma operação de *média* sobre os gradientes, envolvendo pixels em alguma vizinhança. A Figura 20(a) mostra os gradientes de uma parte de uma impressão digital e a Figura 20(b) mostra a média do campo direcional [17].

[1] Chamado de Imagem Orientação em [14, 15] e Imagem Direcional em [16]

3.1 Média dos Gradientes Quadrados

A média dos gradientes não pode simplesmente ser feita em alguma vizinhança local, desde que vetores gradientes opostos se cancelarão, embora eles indiquem a mesma orientação linha-vale. Isso é causado pelo fato de que estruturas locais de linha-vale permanecem indiferentes quando rotacionadas em 180 graus. Desde que as orientações dos gradientes são distribuídas em um espaço cíclico de 0 a π e a orientação média tem que ser encontrada, uma outra formulação para esse problema, 'média do ciclo π-periódico', tem que ser calculada [17].

Uma solução para esse problema é dobrar os ângulos dos vetores gradiente antes de fazer a média. Após dobrar os ângulos, os vetores gradiente opostos apontarão para a mesma direção e por isso se reforçarão enquanto os gradientes perpendiculares se anularão. Depois de feita a média, os vetores gradiente têm que ser convertidos de volta para a representação de ângulo. A orientação de linha-vale principal é então perpendicular à direção do vetor gradiente médio [17].

Não somente o ângulo dos gradientes é dobrado, mas também o comprimento dos vetores gradiente é elevado ao quadrado, como se os vetores gradiente fossem considerados números complexos que são elevados ao quadrado. Isso tem o efeito de que orientações fortes têm um peso maior na orientação média do que as orientações mais fracas. Além disso, essa abordagem resulta em expressões muito enxutas. Entretanto, outras escolhas, tais como atribuir todos os elementos a identidade são encontradas na literatura [17].

O cálculo dos vetores gradiente quadrado $[\ G_{s,x} \quad G_{s,y}\ ]^T$ pode ser obtido por:

$$\begin{bmatrix} G_{s,x} \\ G_{s,y} \end{bmatrix} = \begin{bmatrix} G_x^2 - G_y^2 \\ 2G_xG_y \end{bmatrix}$$

Agora, o gradiente quadrado médio $[\ \overline{G_{s,x}} \quad \overline{G_{s,y}}\ ]^T$ pode ser calculado. A média é feita em alguma vizinhança, possivelmente usando uma janela não uniforme W, e no plano cartesiano:

$$\begin{bmatrix} \overline{G_{s,x}} \\ \overline{G_{s,y}} \end{bmatrix} = \begin{bmatrix} \sum_W G_x^2 - \sum_W G_y^2 \\ \sum_W 2G_xG_y \end{bmatrix}$$

A direção gradiente média Φ, com $-\frac{1}{2}\pi < \Phi \leq \frac{1}{2}\pi$, é dada por

$$\Phi = \frac{1}{2}\angle(G_{xx} - G_{yy}, 2G_{xy})$$

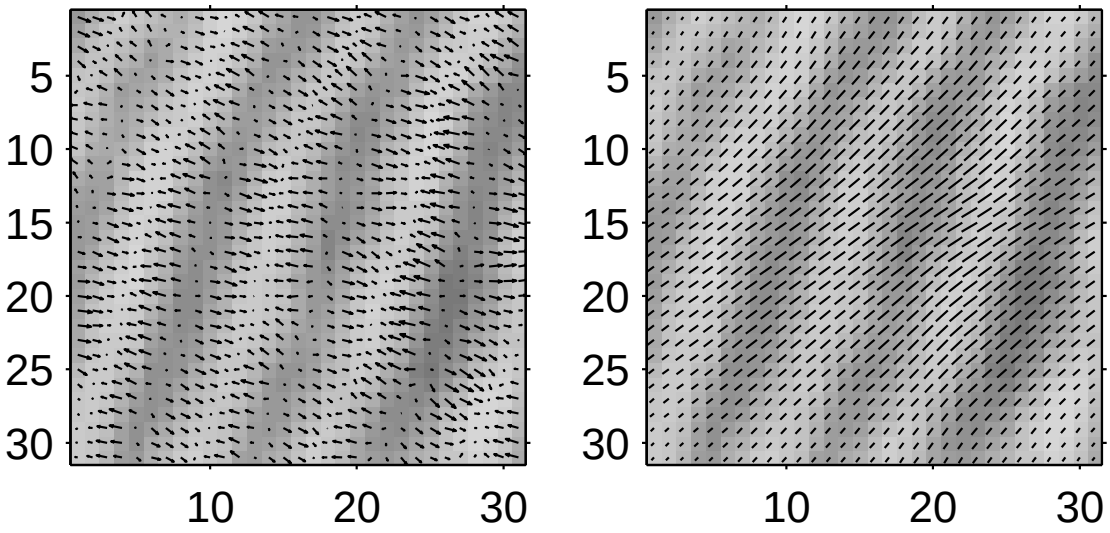
onde $\angle(x, y)$ é definido como:

$$\angle(x,y) = \begin{cases} \arctan(y/x) & \text{, para} \quad x \geq 0 \\ \arctan(y/x) + \pi & \text{, para} \quad x < 0 \wedge y \geq 0 \\ \arctan(y/x) - \pi & \text{, para} \quad x < 0 \wedge y < 0 \end{cases}$$

e a direção média linha-vale θ, com $-\frac{1}{2}\pi < \theta < \frac{1}{2}\pi$, é perpendicular a Φ:

$$\theta = \begin{cases} \Phi + \frac{1}{2}\pi & \text{, para} \quad \Phi \leq 0 \\ \Phi - \frac{1}{2}\pi & \text{, para} \quad \Phi > 0. \end{cases}$$

O algoritmo para calcular o campo direcional é descrito a seguir: [17, 18].



(a) *(b)*

Figura 20: (a) Gradientes e (b) Campo Direcional.

Algorithm 1 Extração do Campo Direcional

1. Calcule os gradientes $G_x = \partial_x(i,j)$ e $G_y = \partial_y(i,j)$ em cada pixel (i,j) usando o operador gradiente *Sobel* (veja Figura 17).

2. Calcule os vetores gradiente quadrado para cada pixel (i,j)

$$\begin{bmatrix} G_{s,x} \\ G_{s,y} \end{bmatrix} = \begin{bmatrix} G_x^2 - G_y^2 \\ 2G_xG_y \end{bmatrix}$$

3. Para cada bloco ou janela de tamanho $W = 16 \times 16$, calcule o gradiente quadrado médio:

$$\begin{bmatrix} \overline{G_{s,x}} \\ \overline{G_{s,y}} \end{bmatrix} = \begin{bmatrix} \sum_W G_x^2 - \sum_W G_y^2 \\ \sum_W 2G_xG_y \end{bmatrix}$$

4. Calcule a direção gradiente média Φ de cada bloco, com $-\frac{\pi}{2} < \Phi \leq \frac{\pi}{2}$:

$$\Phi(x,y) = \begin{cases} \frac{1}{2}\arctan(y/x) & \text{, para} \quad x \geq 0 \\ \frac{1}{2}\arctan(y/x) + \pi & \text{, para} \quad x < 0 \wedge y \geq 0 \\ \frac{1}{2}\arctan(y/x) - \pi & \text{, para} \quad x < 0 \wedge y < 0 \end{cases}$$

5. Devido a presença de ruído, estruturas de linhas e vales corrompidas, minúcias etc na imagem de entrada, a orientação local de linha estimada, $\Phi(x,y)$, pode nem sempre estar correta. Desde que a orientação local de linha varia lentamente em uma vizinhança local onde nenhum ponto singular aparece, um filtro de passa-baixa pode ser usado para modificar a orientação local de linha incorreta. De fato, para executar a filtragem passa-baixa, a imagem orientação precisa ser convertida em um vetor de campo contínuo, que é definido como segue:

$$\begin{aligned} \Phi_x(i,j) &= \cos(2\Phi(i,j)), \\ \Phi_y(i,j) &= \sin(2\Phi(i,j)), \end{aligned}$$

onde Φ_x e Φ_y são as componentes x e y do vetor campo, respectivamente. Com o vetor campo resultante, o filtro de passa-baixa pode então ser executado como segue:

$$\begin{aligned} \Phi'_x(i,j) &= \sum_{u=-\frac{w_\Phi}{2}}^{\frac{w_\Phi}{2}} \sum_{v=-\frac{w_\Phi}{2}}^{\frac{w_\Phi}{2}} W(u,v)\Phi_x(i-uw, j-vw), \\ \Phi'_y(i,j) &= \sum_{u=-\frac{w_\Phi}{2}}^{\frac{w_\Phi}{2}} \sum_{v=-\frac{w_\Phi}{2}}^{\frac{w_\Phi}{2}} W(u,v)\Phi_y(i-uw, j-vw) \end{aligned}$$

onde W é um filtro de bidimensional passa-baixa e $w_\Phi \times w_\Phi$ especifica o tamanho do filtro. Observe que a operação de suavização é executada em nível do bloco. O tamanho padrão do filtro é 5×5 (veja Figura 12(b)).

6. Calcule a orientação local de linha em (i,j) usando

$$O(i,j) = \frac{1}{2}\arctan\left(\frac{\Phi'_y(i,j)}{\Phi'_x(i,j)}\right).$$

Com isso, uma suave estimativa do campo orientação pode ser obtida.

4 Máquinas de Vetores Suporte

Máquinas de Vetores Suporte (SVMs) foram introduzidas recentemente como uma técnica para resolver problemas de reconhecimento de padrões [19]. Esta estratégia de aprendizagem introduzida por Vapnik e colaboradores é um método muito poderoso que em poucos anos desde sua introdução tem superado a maioria dos sistemas em uma ampla variedade de aplicações [20]. De acordo com a teoria de SVMs, enquanto técnicas tradicionais para reconhecimento de padrões são baseadas na minimização do *risco empírico*—isto é, tenta-se otimizar o desempenho sobre o conjunto de treinamento—, SVMs minimizam o *risco estrutural*—isto é, a probabilidade de classificar de forma errada padrões ainda não vistos por uma distribuição de probabilidade dos dados fixa e desconhecida. Este novo princípio de indução, que é equivalente a minimizar um limite superior do erro de generalização, depende da teoria de convergência uniforme de probabilidade. O que fazem SVMs atrativas é *(a)* a habilidade de condensar a informação contida no conjunto de treinamento e *(b)* o uso de famílias de superfícies de decisão de dimensão VC relativamente baixa [19].

O objetivo de classificação Vetor Suporte é elaborar uma forma computacionalmente eficiente de aprender 'bons' hiperplanos de separação em um espaço de características de alta dimensão, onde 'bons' hiperplanos entenderemos por aqueles que otimizam os limites de generalização e por 'computacionalmente eficiente' significará algoritmos capazes de tratar com amostras de tamanho da ordem de 100.000 instâncias. A teoria da generalização dá uma orientação clara sobre como controlar a capacidade e logo como prevenir modelos ruins[1] controlando as medidas das margens dos hiperplanos, enquanto a teoria da otimização fornece as técnicas matemáticas necessárias para encontrar hiperplanos otimizando essas medidas. Existem diferentes limites de generalização, o que motiva algoritmos diferentes: alguém pode por exemplo otimizar a margem máxima, a distribuição das margem, o número de vetores suporte etc. A abordagem mais comum é tratar o problema como minimizando a norma do vetor peso [20].

[1] *Do original, overfitting*

No caso linearmente separável, a idéia chave de uma Máquina de Vetor Suporte pode ser explicada em simples palavras. Dado um conjunto de treinamento S que contém pontos de duas classes, uma SVM separa as classes através de um hiperplano determinado por certos pontos de S, denominados de *vetores suporte*. No caso separável, este hiperplano maximiza a *margem*, ou duas vezes a distância mínima de cada classe ao hiperplano e todos os vetores suporte caem na mesma distância mínima a partir do hiperplano e são assim chamados de *vetores margem*. Em casos reais, as duas classes podem não ser separáveis e tanto o hiperplano quanto os vetores suporte são obtidos da solução de um problema de otimização com restrições. A solução é um compromisso controlado por um parâmetro de regularização entre a maior margem e o menor número de erros [19].

4.1 Classificação Linearmente Separável

A classificação linear é freqüentemente implementada pelo uso de uma função real $f : X \subseteq \Re^n \rightarrow \Re$ na seguinte forma: a entrada $\mathbf{x} = (x_1, \ldots, x_n)'$ é atribuída a uma classe positiva, se $f(x) \geq 0$ e é atribuída a uma classe negativa caso contrário. Considera-se o caso onde $f(\mathbf{x})$ é uma função linear de $\mathbf{x} \in X$, então ela pode ser escrita como

$$\begin{aligned} f(\mathbf{x}) &= \langle \mathbf{w} \cdot \mathbf{x} \rangle + b \\ &= \sum_{i=1}^{n} w_i x_i + b \end{aligned}$$

onde $(\mathbf{w}, b) \in \Re^n \times \Re^n$ são parâmetros que controlam a função e a regra de decisão é dada por sinal$(f(\mathbf{x}))$, onde será usada a convenção que sinal$(0) = 1$. A metodologia de treinamento implica que esses parâmetros devem ser aprendidos a partir dos dados [20].

A interpretação geométrica deste tipo de hipótese é que o espaço de entrada X é dividido em duas partes pelo hiperplano definido pela equação $\langle \mathbf{w} \cdot \mathbf{x} \rangle + b = 0$ (veja Figura 21). Um hiperplano é um subespaço afim de dimensão $n - 1$ que divide o espaço em duas metades que correspondem as entradas das duas classes distintas. Por exemplo, na Figura 21 o hiperplano é a linha escura, com a região positiva acima e a negativa abaixo. O vetor $\mathbf{w}$ define uma direção perpendicular ao hiperplano, enquanto variar o valor de b move o hiperplano paralelamente a ele mesmo. Está claro que uma representação envolvendo $n + 1$ parâmetros livres é necessária, se alguém deseja representar todos os possíveis hiperplanos em $\Re^n$ [20].

Pesquisadores estatísticos e de redes neurais[2] costumavam usar esse tipo simples de

[2]Existem vários outros paradigmas importantes de redes neurais artificiais (RNAs) e, além disso, SVM é um importante método, no entanto, também poderia ser útil para otimizar algum tipo de RNA.

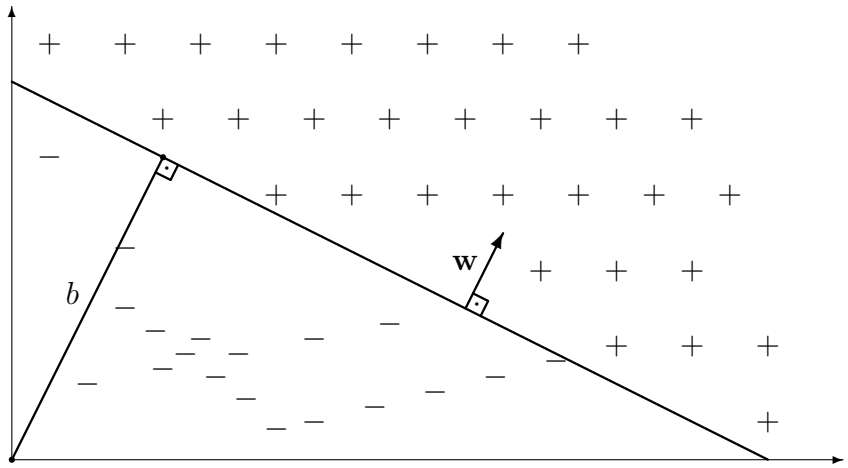


Figura 21: Um hiperplano separando $\mathbf{w}, b$ para um conjunto de treinamento de duas dimensões.

classificador, chamado-o respectivamente de *discriminante linear* e *perceptrons*. As quantidades $\mathbf{w}$ e b serão referenciadas como *vetor peso*[3] e *tendência*[4], termos tomados emprestados da literatura de redes neurais. Às vezes $-b$ é substituído por θ, uma quantidade conhecida como *limiar*[5] [20].

Como SVM usa aprendizado supervisionado a partir de exemplos, será introduzida alguma notação para referenciar entradas, saídas, conjuntos de treinamento etc.

Tipicamente usa-se X para denotar um espaço de entrada e Y para denotar o domínio de saída. Tipicamente tem-se $X \subseteq \Re^n$, enquanto para classificação binária $Y = \{-1, +1\}$, para classificação m-classes $Y = \{1, 2, \ldots, m\}$ e para regressão[6] $Y \subseteq \Re$. Um *conjunto de treinamento* é uma coleção de *exemplos de treinamento*, que são também chamados de *dados de treinamento*. É geralmente denotado por

$$S = ((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_l, y_l)) \subseteq (X \times Y)^l,$$

onde l é o número de exemplos. Trata-se $\mathbf{x}_i$ como *exemplos* ou *instâncias* e y_i como *rótulos*. O conjunto de treinamento S é **trivial** se os rótulos de todos os exemplos são iguais. Observe que se X é um espaço vetor, os vetores de entradas são vetores colunas

[3] *Do original, weight vector*
[4] *Do original, bias*
[5] *Do original, threshold*
[6] *Não usado nesse trabalho*

assim como são os vetores pesos. Quando se deseja trabalhar com vetores linha de $\mathbf{x}_i$ então toma-se a transposta $\mathbf{x}_i'$.

4.1.1 Hiperplano de Separação Ótimo

Para o caso linearmente separável, o algoritmo de vetor suporte simplesmente procura o hiperplano de separação com a maior margem. Este pode se formulado como segue: suponha que todos os dados de treinamento satisfazem as seguintes restrições:

$$\langle \mathbf{x_i} \cdot \mathbf{w} \rangle + b \geq +1, \text{ para } y_i = +1 \tag{4.1}$$

$$\langle \mathbf{x_i} \cdot \mathbf{w} \rangle + b \leq -1, \text{ para } y_i = -1 \tag{4.2}$$

Essas equações podem ser combinadas nas seguintes inequações:

$$y_i(\langle \mathbf{x_i} \cdot \mathbf{w} \rangle + b) - 1 \geq 0, \text{ para } i = \{1, \ldots, l\} \tag{4.3}$$

O conjunto de vetores é dito estar otimamente separado pelo hiperplano se ele é separado sem erro e a distância entre o vetor mais próximo ao hiperplano é maximal. Sem perda de generalidade, pode-se considerar os *hiperplanos canônicos* $(\mathbf{w}, \mathbf{x_i})$ todos hiperplanos que satisfazem a equação (4.4) e onde os parâmetros $\mathbf{w}$, b são restringidos por,

$$\min_{i=1,\ldots,l} |\langle \mathbf{w} \cdot \mathbf{x_i} \rangle + b| = +1. \tag{4.4}$$

Esta restrição sobre a parametrização é uma alternativa preferível para simplificar a formulação do problema. Em palavras o que ela define é: *a norma do vetor peso deve ser igual ao inverso da distância do ponto mais próximo no conjunto de dados ao hiperplano.*

Um hiperplano de separação em forma canônica deve satisfazer as seguintes restrições,

$$y_i(\langle \mathbf{w} \cdot \mathbf{x_i} \rangle + b) \geq 1, \ i = 1, \ldots, l. \tag{4.5}$$

O hiperplano representado pelo par $(\mathbf{w}, b)$ define a equação

$$\langle \mathbf{w} \cdot \mathbf{x} \rangle + b = 0 \tag{4.6}$$

chamado de *hiperplano de separação*, onde $\mathbf{w}$ é normal ao hiperplano, $\frac{|b|}{\|\mathbf{w}\|}$ é a distância perpendicular do hiperplano à origem e $\|\mathbf{w}\| = \sqrt{\langle \mathbf{w} \cdot \mathbf{w} \rangle} = \sqrt{\sum_{i=1}^{l} x_i^2}$ é a norma Euclidiana de $\mathbf{w}$. Seja d_+ (d_-) a menor distância a partir do hiperplano de separação ao

exemplo positivo (negativo) mais próximo. Define-se a *margem* ρ de um hiperplano de separação como sendo a maior margem geométrica entre todos os hiperplanos, ou seja, $\rho = (d_+ + d_-)$. A distância $d_i(\mathbf{w}, b; \mathbf{x_i})$ de um ponto $\mathbf{x_i}$ ao hiperplano $(\mathbf{w}, b)$, ou seja, sua margem, é,

$$d_i(\mathbf{w}, b; \mathbf{x_i}) = \frac{|\langle \mathbf{w} \cdot \mathbf{x_i} \rangle + b|}{\|\mathbf{w}\|} = \frac{y_i(\langle \mathbf{w} \cdot \mathbf{x_i} \rangle + b)}{\|\mathbf{w}\|}. \tag{4.7}$$

Combinando as equações (4.3) e (4.7), para todo $\mathbf{x_i} \in S$, temos que:

$$d_i(\mathbf{w}, b; \mathbf{x_i}) \geq \frac{1}{\|\mathbf{w}\|}. \tag{4.8}$$

Portanto, $\frac{1}{\|\mathbf{w}\|}$ é o limite inferior da distância entre os pontos $\mathbf{x_i}$ e o hiperplano de separação $(\mathbf{w}, b)$. As distância d_+ e d_- ficam

$$d_+ = d_- = \frac{1}{\|\mathbf{w}\|}. \tag{4.9}$$

Como a margem é dada por $\rho = (d_+ + d_-)$, então

$$\rho = \frac{2}{\|\mathbf{w}\|}. \tag{4.10}$$

O hiperplano de separação ótimo (HSO) é dado pela maximização da margem, ρ, sujeita às restrições da equação (4.5). Logo o hiperplano que divide otimamente os dados é aquele que minimiza[7]

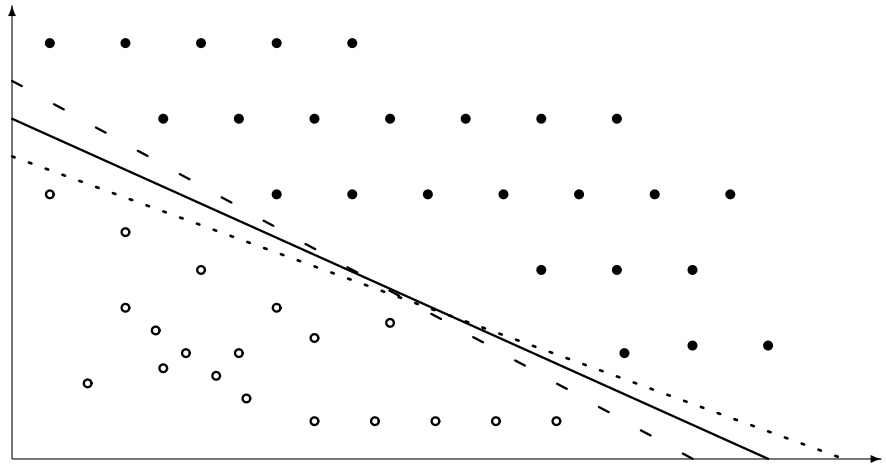
$$\Phi(\mathbf{w}) = \frac{1}{2}\langle \mathbf{w} \cdot \mathbf{w} \rangle = \frac{1}{2}\|\mathbf{w}\|^2. \tag{4.11}$$

Mais formalmente temos,

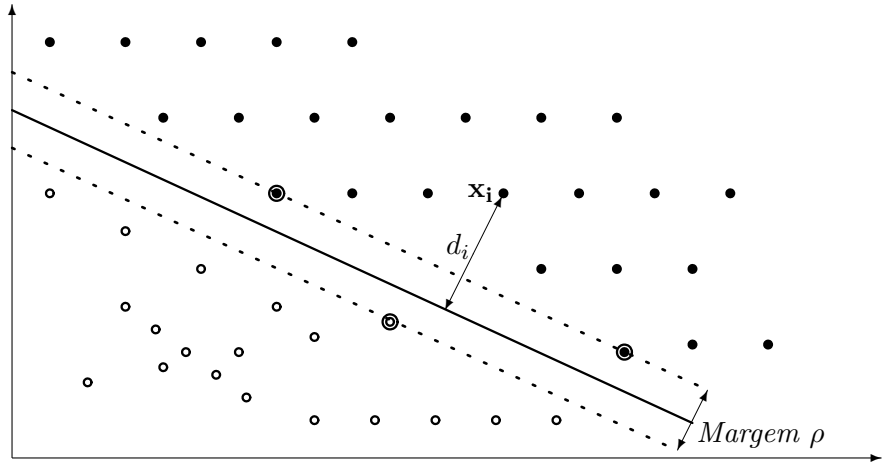
Problema	**P1**
Minimize	$\frac{1}{2}\|\mathbf{w}\|^2$
Sujeito a	$y_i(\langle \mathbf{x_i} \cdot \mathbf{w} \rangle + b) \geq 1$, para $i = \{1, \ldots, l\}$

Desta formulação, observa-se que se o par $(\mathbf{w}, b)$ resolve **P1**, então para pelo menos um $\mathbf{x_i} \in S$ tem-se $y_i(\langle \mathbf{x_i} \cdot \mathbf{w} \rangle + b) = 1$. Em particular, isso implica que a solução de **P1** é sempre um hiperplano de separação na representação canônica. Ainda, o parâmetro b entra nas retrições, mas não na função objetivo a ser minimizada.

[7] *Os artigos não explicam o porquê da fração* $\frac{1}{2}$ *na função* $\Phi(\mathbf{w})$.



(a)



(b)

Figura 22: (a) Exemplos de hiperplanos de separação. (b) Margem geométrica de um ponto $\mathbf{x_i}$ e a margem ρ do hiperplano de separação ótimo. Os círculos fechados são os exemplos positivos e os círculos abertos são os exemplos negativos. Os círculos que caem sobre as margens (linhas tracejadas) são os vetores suporte para esse conjunto de treinamento. Os vetores suporte são realçados com um círculo mais externo.

4.1.2 Vetores Suporte

O problema **P1** pode ser resolvido pelo método clássico de multiplicadores de Lagrange. [21] cita duas razões para trocarmos para uma formulação Lagrangiana do problema:

1. As restrições (4.5) serão substituídas por restrições de multiplicadores de Lagrange, que são muito mais fáceis de manipular.

2. Nesta reformulação do problema, os dados de treinamento somente aparecerão (nos algoritmos de treinamento e teste) na forma de produtos internos entre vetores. Esta é uma propriedade crucial que permitirá generalizar o procedimento para o caso não linear.

Assim, seja $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_l)$ os l multiplicadores de Lagrange não negativos associados com as restrições (4.5), a solução do problema **P1** é equivalente a determinar o *ponto sela*[8] da função

$$L(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{l} \alpha_i \{y_i(\langle \mathbf{x_i} \cdot \mathbf{w} \rangle + b) - 1\}) \tag{4.12}$$

sujeita a $\alpha_i \geq 0$ para $i = 1, \ldots, l$. No ponto sela, L tem um valor mínimo para $\mathbf{w} = \mathbf{w}^*$ e $b = b^*$, e um valor máximo para $\boldsymbol{\alpha} = \boldsymbol{\alpha^*}$, e assim pode-se escrever

$$\frac{\partial L}{\partial b} = \sum_{i=1}^{l} y_i \alpha_i = 0, \tag{4.13}$$

$$\frac{\partial L}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{l} y_i \alpha_i \mathbf{x_i} = 0 \tag{4.14}$$

com

$$\frac{\partial L}{\partial \mathbf{w}} = \left(\frac{\partial L}{\partial w_1}, \frac{\partial L}{\partial w_2}, \ldots, \frac{\partial L}{\partial w_l} \right) \tag{4.15}$$

Substituindo as equações (4.13) e (4.14) no lado direito de (4.12), pode-se ver que o problema **P1** reduz-se a maximização da função

$$\mathcal{L}(\boldsymbol{\alpha}) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j \langle \mathbf{x_i} \cdot \mathbf{x_j} \rangle, \tag{4.16}$$

[8] *Do original, saddle point*

sujeito a restrição (4.13) com $\boldsymbol{\alpha} \geq 0$. Este novo problema é chamado *problema dual* e pode ser formulado como

$$\begin{array}{ll} \text{Problema} & \mathbf{P2} \\ \text{Maximize} & \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j \langle \mathbf{x_i} \cdot \mathbf{x_j} \rangle \\ \text{Sujeito a} & \sum_{i=1}^{l} y_i \alpha_i = 0 \\ & \boldsymbol{\alpha} \geq 0 \end{array}$$

Para o par $(\mathbf{w}^*, b^*)$, da equação (4.14) segue que

$$\mathbf{w}^* = \sum_{i=1}^{l} \alpha_i^* y_i \mathbf{x_i}, \tag{4.17}$$

enquanto b^* pode ser determinado pela equações Karush-Kuhn-Tucker (KKT)

$$\alpha_i^* \left(y_i \left(\langle \mathbf{w}^* \cdot \mathbf{x_i} \rangle + b^* \right) - 1 \right) = 0, \ i = 1, 2, \ldots, l. \tag{4.18}$$

Observe que somente os α_i^*'s que podem ser diferente de zero na equação (4.18) são aqueles para os quais as restrições (4.5) são satisfeitas com o sinal de igualdade. Os pontos $\mathbf{x_i}$ correspondentes, chamados de *vetores suporte*, são os pontos de S mais próximos do HSO.

Dado o vetor suporte $\mathbf{x_j}$, o parâmetro b^* pode ser obtido da condição de KKT por

$$b^* = y_j - \langle \mathbf{w}^* \cdot \mathbf{x_j} \rangle. \tag{4.19}$$

O problema de classificar um novo ponto de dado $\mathbf{x}$ é agora simplesmente resolvido calculando

$$\text{sinal} \left(\langle \mathbf{w}^* \cdot \mathbf{x_i} \rangle + b^* \right). \tag{4.20}$$

Enfim, os vetores suporte condensam toda a informação contida no conjunto de treinamento S que é necessário para classificar novos pontos de dados.

4.2 Classificação Não Lineamente Separável

Se um conjunto S não é linearmente separável ou alguém simplesmente ignora se o conjunto S é ou não linearmente separável, o problema de buscar por um HSO fica sem sentido (pode não haver hiperplano de separação). Felizmente, a análise anterior pode ser

generalizada pela introdução de l variáveis não negativas $\boldsymbol{\xi} = (\xi_1, \xi_2, \ldots, \xi_l)$ tais como

$$y_i(\langle \mathbf{w} \cdot \mathbf{x_i} \rangle + b) \geq 1 - \xi_i, \ i = 1, \ldots, l. \tag{4.21}$$

Se o ponto $\mathbf{x_i}$ satisfaz a inequação (4.5), então ξ_i é nulo e (4.21) reduz-se a (4.5). Em caso contrário, se o ponto $\mathbf{x_i}$ não satisfaz a inequação (4.5), o termo $-\xi_i$ é adicionado ao lado direito de (4.5) para obter a equação (4.21). O HSO generealizado é então considerado como a solução para

Problema	**P3**
Minimize	$\frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{l} \xi_i$
Sujeito a	$y_i(\langle \mathbf{x_i} \cdot \mathbf{w} \rangle + b) \geq 1 - \xi_i$, para $i = \{1, \ldots, l\}$
	$\boldsymbol{\xi} \geq 0$.

O termo $C\sum_{i=1}^{l} \xi_i$ pode ser pensado como alguma medida de erro de classificação. Ele faz o HSO menos sensível à presença de exemplos "mal comportados" no conjunto de treinamento. O parâmetro C pode ser considerado como um parâmetro de regularização. O HSO tende a maximizar a distância mínima $\frac{1}{\|\mathbf{w}\|}$ para um C pequeno e minimizar o número de pontos classificados errados para um C grande. Para valores intermediários de C a solução do problema **P3** compensa o erro para uma margem grande.

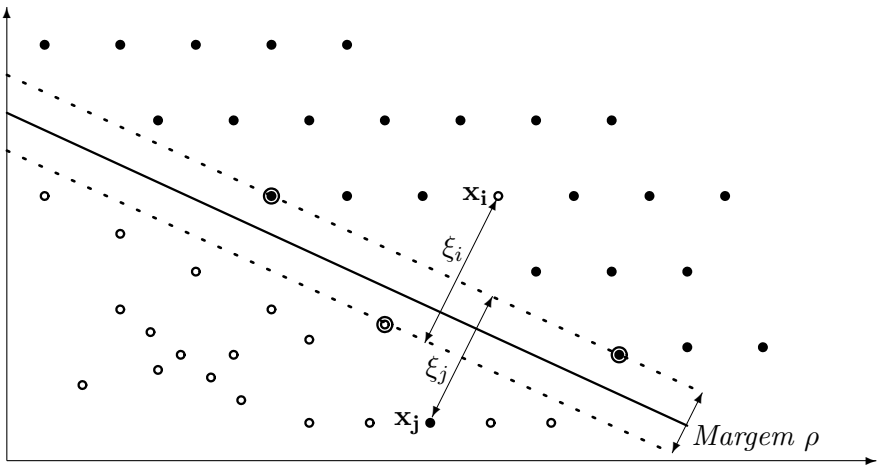


Figura 23: As variáveis de folga ξ_i e ξ_j para o problema de classificação. Eles medem (informalmente) quanto um ponto falhou em ter uma margem de $\rho/2$ a partir do hiperplano. Se $\xi_i > \rho/2$, então $\mathbf{x_i}$ é classificado de forma errada por $(\mathbf{w}, b)$.

Em analogia com o que foi feito para o caso linearmente separável, o problema **P3**

pode ser transformado no *dual*

$$\begin{array}{ll} \text{Problema} & \mathbf{P4} \\ \text{Maximize} & \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j \langle \mathbf{x_i} \cdot \mathbf{x_j} \rangle \\ \text{Sujeito a} & \sum_{i=1}^{l} y_i \alpha_i = 0 \\ & 0 \leq \alpha_i \leq C,\ i = 1, \ldots, l \end{array}$$

Das restrições do problema **P4** segue que se C é suficientemente grande e o conjunto S é linearmente separável, o problema **P4** reduz-se a **P2**.

Para o par $(\mathbf{w}^*, b^*)$, é fácil encontrar que

$$\mathbf{w}^* = \sum_{i=1}^{l} \alpha_i^* y_i \mathbf{x_i}, \tag{4.22}$$

enquanto b^* pode novamente ser determinado a partir de $\boldsymbol{\alpha}$, solução do problema **P4**, e pelas novas condições de Karush-Kuhn-Tucker

$$\alpha_i^* \left(y_i \left(\langle \mathbf{w}^* \cdot \mathbf{x_i} \rangle + b^* \right) - 1 + \xi_i^* \right) = 0,\ i = 1, 2, \ldots, l \tag{4.23}$$

$$\left(C - \alpha_i^* \right) \xi_i^* = 0,\ i = 1, 2, \ldots, l \tag{4.24}$$

onde ξ_i^* são os valores dos ξ_i's no ponto sela. Similarmente ao caso separável, os pontos $\mathbf{x_i}$ para os quais $\alpha_i^* > 0$ são chamados de *vetores suporte*. A principal diferença é que agora se tem que distinguir entre os vetores suporte para os quais $\alpha_i^* < C$ e aqueles para os quais $\alpha_i^* = C$. No primeiro caso, da condição (4.24) segue que $\xi_i^* = 0$, e logo, da condição (4.23), que os vetores suporte caem a uma distância $\frac{1}{\|\mathbf{w}^*\|}$ do HSO. Esses vetores suporte são chamados *vetores margem*. Os vetores suporte para os quais $\alpha_i^* = C$, ao contrário, são pontos classificados errados se $\xi_i > 1$. Se $0 < \xi \leq 1$, então são pontos corretamente classificados, entretanto mais próximo que $\frac{1}{\|\mathbf{w}^*\|}$ do HSO ou, em alguns casos degenerados, pontos caindo sobre a margem quando $\xi_i = 0$. Em qualquer evento, refere-se a todos os vetores suporte para os quais $\alpha_i = C$ como *erros*. Um exemplo de HSO generalizado com os respectivos vetores margem e erros é mostrado na Figura 24. Todos os pontos que não são vetores suporte são corretamente classificados e caem fora da margem divisora.

4.3 Superfícies Não Lineares

Geralmente quanto se trabalha com aplicações do mundo real, os dados não são linearmente separáveis. Uma das propriedades mais fortes de Máquinas de Vetores Suporte é que elas são capazes de aprender também num espaço não linear. A idéia básica é

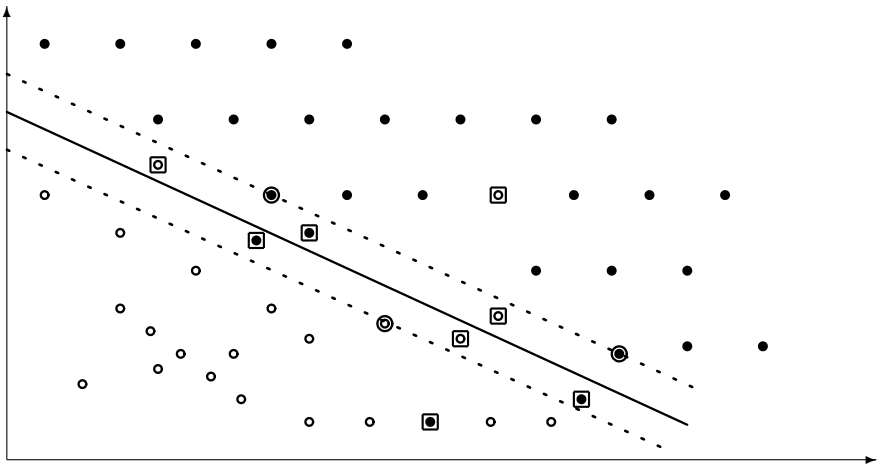


Figura 24: Hiperplano de separação ótimo generalizado. Os dois conjuntos de círculos (abertos e fechados) são não linearmente separáveis. A linha sólida é o hiperplano de separação ótimo, as linhas tracejadas são as margens, os círculos duplos são os vetores margem ($\alpha_i^* < C$) e os exemplos envolvidos por um quadrado são os erros ($\alpha_i^* = C$).

fazer um mapeamento dos dados para um espaço onde os dados possam ser linearmente separáveis. A forma da superfície de separação dos dados no espaço original pode ser completamente diferente de um hiperplano.

4.3.1 Espaço de Características

Os algoritmos de aprendizagem são influenciados pelos dados e, mais especificamente, por seus *atributos*. O número de atributos nos dados pode degradar o desempenho computacional de um algoritmo de aprendizagem, se esse número é muito grande, ou a acurácia do mesmo, se o número de atributos é muito pequeno ou não são significantes (redundantes ou impuros). De qualquer forma, a complexidade da função objetivo a ser aprendida depende da forma como é representada e a dificuldade da tarefa de aprendizagem pode variar de acordo. Então uma representação que se adequa ao problema específico a ser aprendido deve ser escolhida[9]. Uma técnica comum em aprendizagem de máquina é mudar a representação dos dados [20]:

$$\mathbf{x} = (x_1, \ldots, x_n) \longmapsto \phi(\mathbf{x}) = (\phi_1(\mathbf{x}), \ldots, \phi_m(\mathbf{x})).$$

[9]Outra forma de ver esta problema é que freqüentemente o conceito alvo não pode ser expresso como uma combinação linear simples dos atributos dados, mas em geral requer que mais características abstratas dos dados sejam exploradas.

Isso é equivalente a mapear o espaço de entrada $\mathcal{X}$ em um novo espaço,

$$\mathcal{F} = \{\phi(\mathbf{x})|\mathbf{x} \in \mathcal{X}\}$$

chamado de *espaço de características*.

Por exemplo, um algoritmo de separação linear não consegue aprender uma função não linear como

$$f(x_1, x_2, x_3) = C\frac{x_1 x_2}{x_3^2}.$$

Um simples mapeamento do tipo

$$(x_1, x_2, x_3) \longmapsto (y_1, y_2, y_3) = (\ln x_1, \ln x_2, \ln x_3)$$

resulta na seguinte representação:

$$g(y_1, y_2, y_3) = \ln f(x_1, x_2, x_3) = \ln C + \ln x_1 + \ln x_2 - 2\ln x_3 = c + y_1 + y_2 - 2y_3,$$

o que pode ser aprendido por um algoritmo linear. Isso pode simplificar uma tarefa bastante conhecida em aprendizagem de máquina como *seleção de características*.

A Figura 25 mostra um exemplo de um mapeamento de características de um espaço de entrada bidimensional para um espaço de características bidimensional, onde os dados não podem ser separados por uma função linear no espaço de entrada, mas podem ser no espaço de característica. O que uma Máquina de Vetor Suporte faz é mapear os dados para espaços de dimensões muito alta onde a separação linear torna-se fácil.

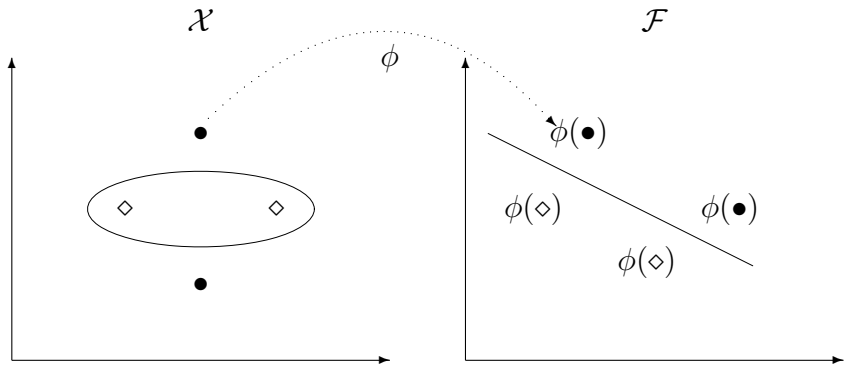


Figura 25: Um mapeamento de características pode simplificar a tarefa de classificação.

Freqüentemente, quando se deseja selecionar características, pode-se procurar por

um conjunto menor de características que ainda contém a informação essencial contida nos atributos originais, também conhecido como *redução da dimensionalidade*. Isso traz benefícios tanto computacionais, quanto na acurácia de generalização, pois, geralmente, esses dois parâmetros degradam quando o número de características cresce. Esse fenômeno é referenciado, às vezes, como a *maldição da dimensionalidade*.

4.3.2 Mapeamento Implícito

Para um algoritmo linear aprender funções não lineares, necessita-se selecionar um conjunto de características não lineares e reescreve-las (mapeá-las) em outra representação. Isso é equivalente a aplicar um mapeamento não linear e fixo dos dados para um espaço de características, no qual um algoritmo linear pode ser usado [20]. O problema **P4**

$$\begin{array}{ll} \text{Problema} & \mathbf{P4} \\ \text{Maximize} & \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j \langle \mathbf{x_i} \cdot \mathbf{x_j} \rangle \\ \text{Sujeito a} & \sum_{i=1}^{l} y_i \alpha_i = 0 \\ & 0 \leq \alpha_i \leq C,\, i = 1, \ldots, l \end{array}$$

usa o produto interno $\langle \mathbf{x_i} \cdot \mathbf{x_j} \rangle$ dos dados de treinamento para encontrar o Hiperplano de Separação Ótimo. Entretanto, se somos capazes de fazer o mapeado do espaço de entrada no espaço de características usando a transformação ϕ, então o problema **P4** muda para **P5**

$$\begin{array}{ll} \text{Problema} & \mathbf{P5} \\ \text{Maximize} & \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j \langle \phi(\mathbf{x_i}) \cdot \phi(\mathbf{x_j}) \rangle \\ \text{Sujeito a} & \sum_{i=1}^{l} y_i \alpha_i = 0 \\ & 0 \leq \alpha_i \leq C,\, i = 1, \ldots, l \end{array}$$

onde $\phi : \mathcal{X} \longmapsto \mathcal{F}$ é um mapeamento não linear do espaço de entrada no espaço de características.

Entretanto, calcular o produto interno $\langle \phi(\mathbf{x_i}) \cdot \phi(\mathbf{x_j}) \rangle$ diretamente no espaço de características pode ser computacionalmente inviável. Por exemplo, suponha que desejamos mapear atributos em $\Re^2$ para o espaço de características formado por todos os produtos possíveis entre os atributos, ou seja:

$$\begin{array}{rcl} \phi : \Re^2 & \longrightarrow & \mathcal{F} = \Re^3 \\ (x_1, x_2) & \longmapsto & (x_1^2, x_2^2, x_1 x_2). \end{array}$$

Com isso, pode-se coletar todas as características de monômios de grau 2 nesse mapeamento não linear [22, 23]. Essa abordagem é adequada para poucos atributos, mas torna-se

inviável para problemas de tamanhos reais. Para um espaço de entrada de dimensão n existem

$$N_{\mathcal{F}} = \frac{(n+d-1)!}{d!(n-1)!}$$

monômios diferentes, formando um espaço de características de dimensão $N_{\mathcal{F}}$. Imagens de 16×16 pixels e um monômio de grau 5 produz uma dimensionalidade de grau 10^{10}! Então, o mapeamento *explícito* no espaço de características torna-se inviável. Para resolver esse problema usamos uma função *"kernel"* que *implicitamente* faz o (1) mapeamento no espaço de características e (2) depois usa um algoritmo linear para classificar tal espaço.

4.3.3 Funções "Kernel"

Um *"kernel"* é uma função $\mathcal{K}$, tal que para todo $\mathbf{x}, \mathbf{z} \in \mathcal{X}$

$$\mathcal{K}(\mathbf{x}, \mathbf{z}) = \langle \phi(\mathbf{x}) \cdot \phi(\mathbf{z}) \rangle.$$

onde ϕ é um mapeamento de $\mathcal{X}$ em um espaço de características produto interno $\mathcal{F}$.

Então o problema **P5** agora pode ser transformado em

$$\begin{array}{ll} \text{Problema} & \mathbf{P6} \\ \text{Maximize} & \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j \mathcal{K}(\mathbf{x}_i, \mathbf{x}_j) \\ \text{Sujeito a} & \sum_{i=1}^{l} y_i \alpha_i = 0 \\ & 0 \leq \alpha_i \leq C,\ i = 1, \ldots, l \end{array}$$

Por exemplo, para um mapeamento

$$\phi : (x_1, x_2) \quad \longmapsto \quad (x_1^2, x_2^2, x_1 x_2, x_2 x_1),$$

o produto interno em $\mathcal{F}$ tem a forma de

$$\langle \phi(\mathbf{x}) \cdot \phi(\mathbf{y}) \rangle \quad = \quad (x_1^2 y_1^2, x_2^2 y_2^2, 2 x_1 x_2 y_1 y_2) \quad = \quad \langle \mathbf{x} \cdot \mathbf{y} \rangle^2.$$

Isso significa que a função *"kernel"* $\mathcal{K}$ que substitui o produto interno do mapeamento ϕ é simplesmente o quadrado do produto interno do espaço de entrada.

Alguns benefícios diretos se obtêm disso:

1. Não precisamos conhecer diretamente o mapeamento ϕ;
2. O mapeamento ϕ é computado *implicitamente*;

3. A dimensão do espaço de características não necessariamente afeta o desempenho computacional;

4. As propriedades de representações de *"kernel"* são auto-contidas e podem ser usadas com diferentes teorias de aprendizagem;

5. O método *"kernel"* desacopla os algoritmos e a teoria de aprendizagem das especificidades da área de aplicação, que deve ser codificada no projeto de uma função *"kernel"* apropriada.

Com isso, todo algoritmo linear que usa somente produtos escalares pode implicitamente ser executado em um espaço de característica $\mathcal{F}$ (de dimensão potecialmente alta) usando *"kernels"*, isto é, alguém pode construir de forma elegante uma versão não linear de um algoritmo linear [22].

4.3.4 Produto Interno em Espaço de Características

Nas seções anteriores, percebemos que as funções *"kernel"* aumentam o poder de classificação de Máquinas de Vetores Suporte por flexibilizar a forma da superfície de separação. Entretanto, nem todo *"kernel"* é útil para Máquinas de Vetores Suporte. Os mais interessantes são aqueles formados pelo produto interno do mapeamento ϕ no espaço de características, ou seja,

$$\mathcal{K}(\mathbf{x}, \mathbf{z}) = \langle \phi(\mathbf{x}) \cdot \phi(\mathbf{z}) \rangle \tag{4.25}$$

onde o mapeamento ϕ é da forma

$$\phi : \Re^d \mapsto \mathcal{H}$$

e $\mathcal{H}$ é um espaço Euclidiano de dimensão finita ou infinita chamado de *espaço Hilbert*. De acordo com [24], qualquer função simétrica $\mathcal{K}(\mathbf{x}, \mathbf{z}) \in L_2$ pode ser expandida na forma

$$\mathcal{K}(\mathbf{x}, \mathbf{z}) = \sum_{i=1}^{\infty} \lambda_i \langle \phi(\mathbf{x}) \cdot \phi(\mathbf{z}) \rangle, \tag{4.26}$$

onde $\lambda_i \in \Re$ e ϕ_i são autovalores[10] e autovetores[11]

$$\int \mathcal{K}(\mathbf{x}, \mathbf{z}) \phi(\mathbf{u}) d\mathbf{u} = \lambda_i \phi(\mathbf{v})$$

[10] *Do inglês, eigenvalues.*
[11] *Do inglês, eigenfuntions.*

do operador de integral definido pelo *"kernel"* $\mathcal{K}(\mathbf{x}, \mathbf{z})$. Uma condição suficiente para garantir que (4.25) define um produto interno no espaço de características é que todos os autovalores na expansão (4.26) sejam positivos. Para garantir que esses coeficientes sejam positivos, é necessário e suficiente (Teorema de Mercer) que a condição

$$\int \int \mathcal{K}(\mathbf{x}, \mathbf{z}) g(\mathbf{u}) g(\mathbf{v}) d\mathbf{u} d\mathbf{v} > 0$$

seja satisfeita para todo g tal que

$$\int g^2(\mathbf{u}) d\mathbf{u} < \infty.$$

Assim, funções que satisfazem o teorema de Mercer podem ser usadas como produtos internos.

4.3.5 Exemplos de Funções "Kernel"

Vimos que Máquinas de Vetores Suporte podem aprender a separar dados que são tanto linearmente separáveis quanto não linearmente separáveis através de funções *"kernel"*. Quando se aplica Máquinas de Vetores Suporte à dados do mundo real, então o que necessitamos fazer é encontrar a configuração de parâmetros que melhor generaliza os dados. Basicamente esses parâmetros são:

- C: o peso que o erro exerce na função objetivo e que também limita os multiplicadores de Lagrange;
- *"Kernel"*: a função *"kernel"* que dá forma à superfície que melhor separa os dados;
- *Parâmetros do "kernel"*: geralmente um *"kernel"* possui parâmetros que influenciam o poder de generalização da superfície.

A literatura cita alguns exemplos de *"kernels"* mais usados [19–21, 24–26]:

Função "Kernel"	**Tipo de Classificador**
$K(\mathbf{x}, \mathbf{y}) = (\langle \mathbf{x} \cdot \mathbf{y} \rangle + 1)^d$	Polinômio de grau d
$K(\mathbf{x}, \mathbf{y}) = \exp(-\frac{\|\langle \mathbf{x}-\mathbf{y} \rangle\|^2}{2\sigma^2})$	Gaussiano
$K(\mathbf{x}, \mathbf{y}) = \tanh(\langle \mathbf{x} \cdot \mathbf{y} \rangle - \theta)$	Perceptron Multi Camadas
$K(\mathbf{x}, \mathbf{y}) = 1 + \langle \mathbf{x} \cdot \mathbf{y} \rangle + \frac{1}{2}\langle \mathbf{x} \cdot \mathbf{y} \rangle \min(\mathbf{x} \cdot \mathbf{y}) - \frac{1}{6}\min(\mathbf{x} \cdot \mathbf{y})^3$	Splines Linear

Tabela 1: Os *"kernels"* mais populares: polinomial, gaussiano RBF e *"perceptron"* de multiplas camadas.

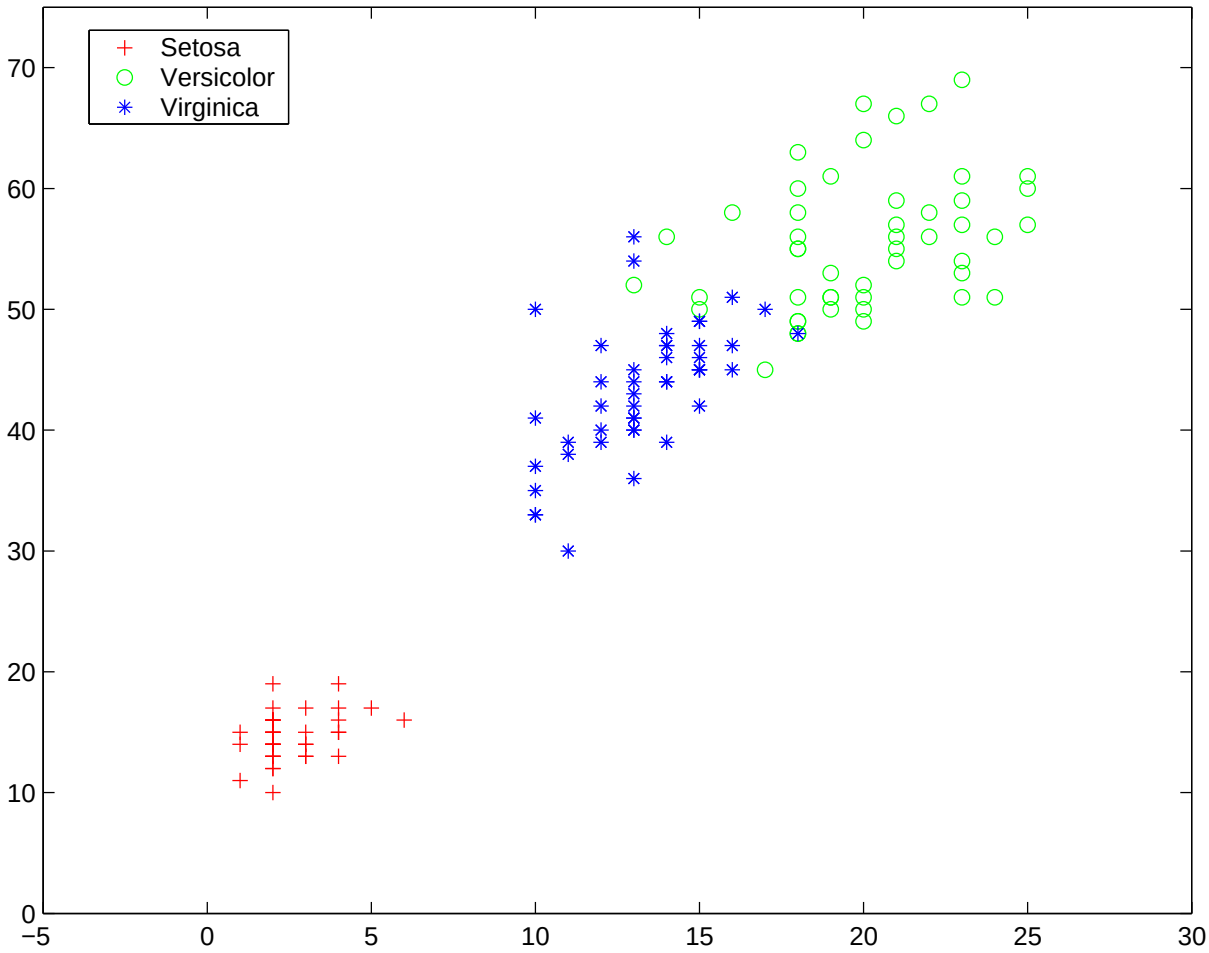
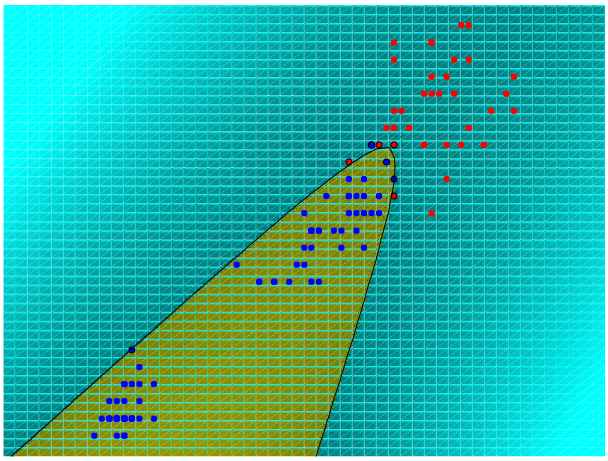
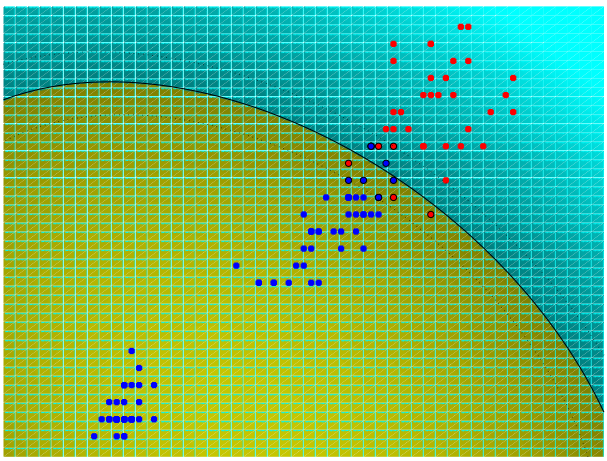


Figura 26: Classes dos dados de flores iris. Todas as figuras foram construídas com a toolbox SVM para MATLAB desenvolvida por [2].

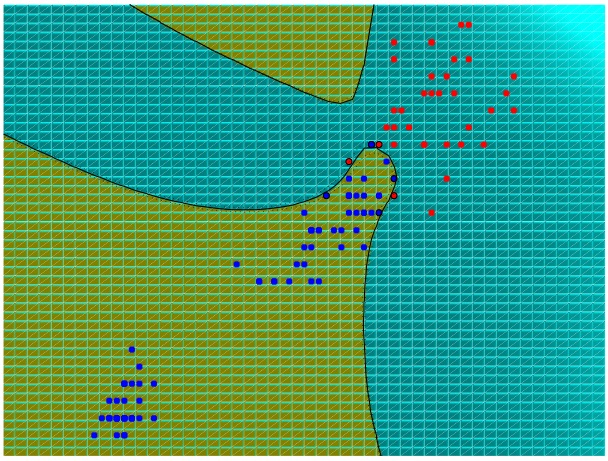
Steve Gunn [2, 26] desenvolveu a toolbox SVM para MATLAB na qual podemos visualizar o poder de separação e a flexibilidade que *"kernels"* dão ao SVM. Nas Figuras 27 e 28, usamos o conjunto de dados de flores iris, bastante conhecido, que contém as classes *Setosa*, *Versilcolor*, *Viginica* e os atributos comprimento da pétala e largura da pétala. Na Figura 27, consideramos *Setosa* e *Versilcolor* uma classe e *Viginica* outra. Então variamos os *"kernels"* e seus parâmetros, e o limitante C. Na Figura 27, nos gráficos de (a) a (d) foi utilizado um *"kernel"* polinomial e nos gráficos (e) a (f) foi utilizado um *"kernel"* gaussiano RBF. Os gráficos (c) e (e) mostram que a região superior na forma de parábola pode indicar *"overfitting"* devido a dimensão alta. Na Figura 28, podemos perceber a versatilidade do *"kernel"* *"spline"* linear.



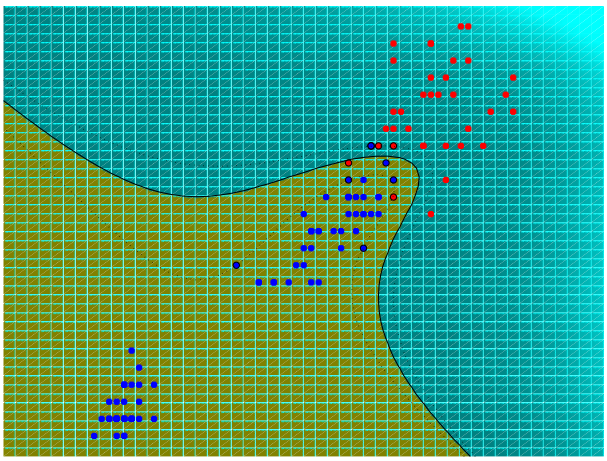
(a) "Kernel"=polinomial, $d = 2$ e $C = \infty$



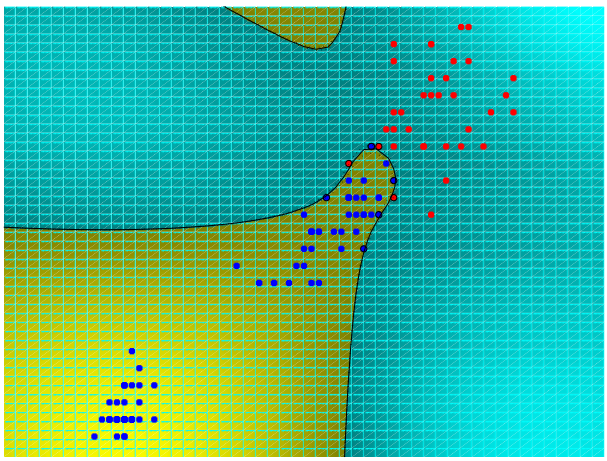
(b) "Kernel"=polinomial, $d = 2$ e $C = 10$



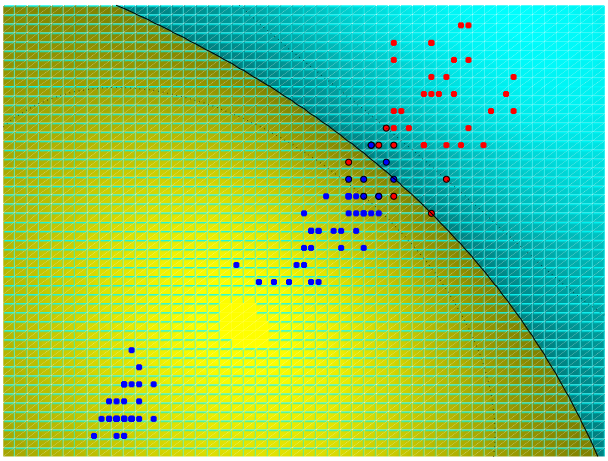
(c) "Kernel"=polinomial, $d = 10$ e $C = \infty$



(d) "Kernel"=polinomial, $d = 10$ e $C = 10$

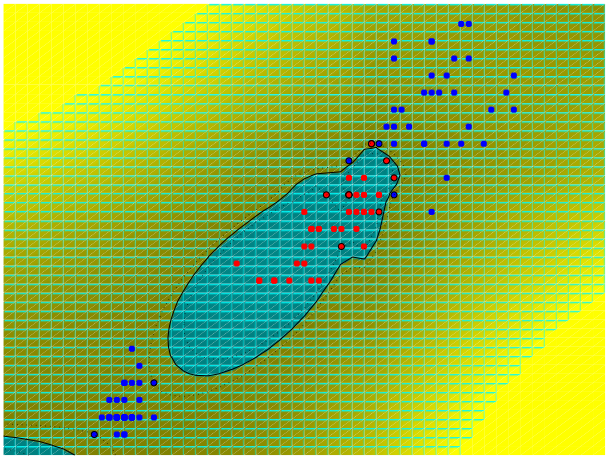


(e) "Kernel"=RBF, $\sigma = 1$ e $C = \infty$

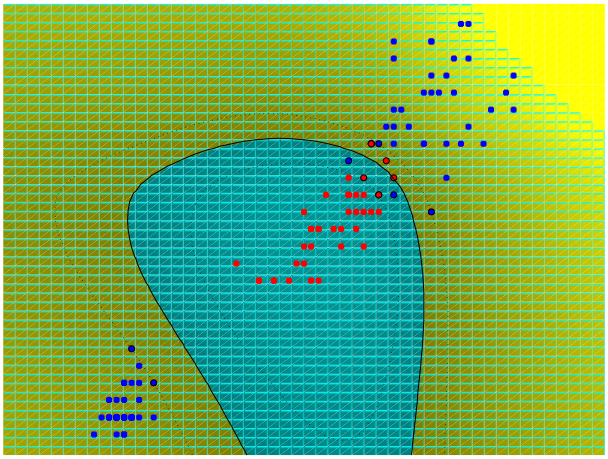


(f) "Kernel"=RBF, $\sigma = 1$ e $C = 10$

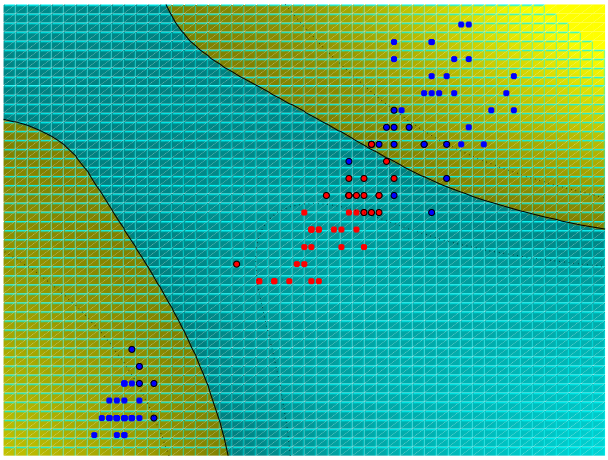
Figura 27: "Kernels" polinomias e gaussianos aplicados aos dados de flores iris. Todas as figuras foram construídas com a toolbox SVM para MATLAB desenvolvida por [2].



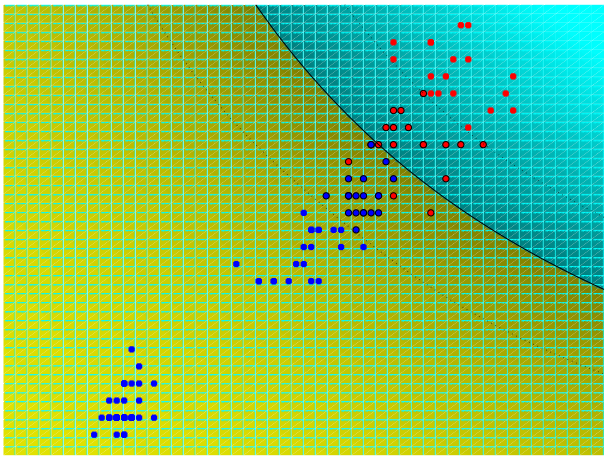
"Kernel"=Spline Linear e $C = \infty$



"Kernel"=Spline Linear e $C = 1000$



"Kernel"=Spline Linear e $C = 10$



"Kernel"=Spline Linear e $C = 1$

Figura 28: "Kernel" spline linear aplicado aos dados de flores iris. Todas as figuras foram construídas com a toolbox SVM para MATLAB desenvolvida por [2].

5 Resultados

Nossos testes foram realizados sobre o conjunto de dados NIST-4 que contém 4.000 imagens de impressões digitais (2.000 dedos), igualmente distribuídas entre cinco classes - *Plain Arch* (A), *Tented Arch* (T), *Left Loop* (L), *Right Loop* (R) e *Whorl* (W). As imagens têm tamanho de 512×512 pixels e foram classificadas manualmente. Cada dedo tem duas impressões chamadas de f e s [1] (Figura 29). Separamos esses dados em dois conjuntos de 2000. Para dados de treinamento, usamos os arquivos de $f0001$ a $f2000$ equivalentes a primeira impressão dos dedos e, para os dados de teste, os arquivos de $s0001$ a $s2000$ equivalentes a segunda impressão dos dedos.

f s

Figura 29: Duas impressões digitais de um mesmo dedo.

Máquinas de Vetores Suporte foram projetadas para problemas que possuem duas classes. Entretanto, para problemas com mais de duas classes, *múltiplas classes*, o conjunto de dados de treinamento deve ser combinado para formar problemas de duas classes. As abordagens mais comuns são *um-contra-todos* e *todos-os-pares*. Na abordagem *um-contra-todos*, para um problema com c classes, c problemas são separados dos dados

[1] f de *"first"* em inglês e s de *"second"* em inglês

de treinamento. Por exemplo, a classe *Plain Arch* considerada como exemplo positivo e as outras classes como exemplos negativos: $(A)^+(TLRW)^-$. Em outras palavras, cada classe é separada das demais classes num problema. Usando o NIST-4, esta abordagem dividiu os dados em 5 problemas de 2 classes, sendo cada problema composto de 2000 exemplos. No caso, todo o conjunto de treinamento é usado. Já na abordagem *todos-os-pares*, $\frac{c(c-1)}{2}$ pares de problemas (combinação dois-a-dois das classes) são separados a partir dos dados de treinamento. Essa forma de gerar os problemas usa uma quantidade menor dos dados de treinamento. Por exemplo, a classe *Plain Arch* considerada como exemplo positivo e a classe *Tented Arch* considerada como exemplo negativo: $(A)^+(T)^-$. Usando o NIST-4, esta abordagem dividiu os dados em 10 problemas de 2 classes, sendo cada problema composto de 800 exemplos.

Os resultados mais significantes em classificação de impressões digitais usando Máquinas de Vetores Suporte foram alcançados por Yao, *et. al.* [4, 5]. Em [5], eles usaram SVM com um *"kernel"* gaussiano, $\sigma = 1$, $C = 10$ e alcançaram 89,3% de acurácia (com cinco classes). Foi usado o esquema de representação *"FingerCode"* (que usa filtros de *Gabor*) ao contrário de campo direcional, como características de uma impressão digital. Este esquema consiste de um vetor de 192 características. Entretanto, em [4], os autores usaram Máquinas de Vetores Suporte em conjunto com *"FingerCode"* e redes neurais recursivas, como extrator de características, e eles elevaram a acurácia para 90,0% (com cinco classes). É importante citar que em *"FingerCode"* um dos principais passos é a localização do ponto *core*. Em algumas imagens de qualidade baixa, este passo pode falhar e então o exemplo seria rejeitado. De fato, isso foi usado como critério de qualidade da imagem. Em ambos os artigos, eles trabalharam com o conjunto de dados NIST-4 que dispõe de 4.000 imagens de impressões digitais e usaram a implementação SVMFu [27] .

Em nosso trabalho, usamos campo direcional com vetores de 1.024 atributos como características para impressões digitais e Máquinas de Vetores Suporte[2] para classificar os dados do NIST-4. Nós treinamos o SVM usado um *"kernel"* gaussiano $K(\mathbf{x}, \mathbf{y}) = \exp(-\frac{\|\langle \mathbf{x}-\mathbf{y} \rangle\|^2}{2\sigma^2})$, com $\sigma = 8$ e $C = 10$, pois este obteve erro zero de treinamento para cada classificador e alcançou os melhores resultados na fase de teste. Os experimentos foram conduzidos em duas formas: sem rejeição de imagens e com rejeição de imagens. No conjunto de dados NIST-4, existem imagens de má qualidade que influenciam nos resultados, levando um classificador a uma acurácia menor. É comum na literatura a rejeição de imagens de má qualidade (difíceis até de classificar a olho nú) para observar o ganho em acurácia nos resultados. Assim, conduzimos os experimentos com e sem critério

[2]Usamos a implementação SVM^{light} [28].

de rejeição. Os resultados são discutidos nas seções a seguir.

5.1 Resultados sem rejeição de imagens

Os resultados da fase de teste estão resumidos nas Tabelas 2 e 3. Obtivemos 91,18% de acurácia média na abordagem *todos-os-pares* (Tabela 2) e 90,43% de acurácia média na abordagem *um-contra-todos* (Tabela 3). Observamos pelos resultados da abordagem *todos-os-pares*, como era de se esperar, que as classes *Plain Arch* e *Tented Arch* são mais difíceis de separar (menor acurácia de classificador) devido a semelhança de características. Já as classes *Left Loop* e *Right Loop* são mais fáceis de classificar por causa de suas características simétricas (recorra a Figura 7 para detalhes). Já na abordagem *um-contra-todos*, observamos que a classe *Tented Arch* é a mais semelhante às outras classes, obtendo assim um resultado de acurácia menor. Entretanto, a classe *Whorl* é a que apresenta o maior número de características disjuntas sendo assim a mais diferente em relação às outras classes.

Classe	*Precisão (%)*	*Classe*	*Precisão (%)*
AL	92,38	*LT*	87,74
AR	91,37	*LW*	93,66
AT	73,18	*RT*	87,96
AW	95,99	*RW*	94,16
LR	98,84	*TW*	96,49

Média	**91,18**
Desvio Padrão	6,88

Tabela 2: Estratégia *Todos os Pares* sem rejeição de imagens.

Classe	*Precisão (%)*
$(A)LRTW$	87,83
$(T)ALRW$	83,95
$(L)ARTW$	93,52
$(R)ALTW$	91,94
$(W)ALRT$	94,96
Média	**90,43**
Desvio Padrão	4,01

Tabela 3: Estratégia *Um contra Todos* sem rejeição de imagens.

5.1.1 Unindo *Plain Arch* e *Tented Arch* em uma classe

Outra abordagem para testar os classificadores, é unir as classes *Plain Arch* e *Tented Arch* em uma única classe por serem muito semelhantes. Agora o número de classes fica reduzido a apenas quatro. Para a abordagem *um-contra-todos* com quatro classes, apenas o classificador para o problema $(AT)LRW$ precisa ser treinado. Os classificadores e resultados anteriores para os problemas $(L)ARTW$, $(R)ALTW$, $(W)ALRT$ permanecem os mesmos. Para a abordagem *todos-os-pares* com quatro classes, os classificadores para os problemas $(AT)L$, $(AT)R$ e $(AT)W$ precisam ser treinados. Os classificadores e resultados anteriores para os problemas LR, LW e RW permanecem os mesmos. Então uma nova fase de treinamento do classificador e uma nova fase de teste foram realizadas. Os resultados estão catalogados nas Tabelas 4, 5, 6, 7.

Na Tabela 4, comparamos os resultados de acurácia do problema $(AT)LRW$ com $(A)TLRW$, $(T)ALRW$ e $(\overline{AT})LRW$ (corresponde a média de acurácia entre $(A)TLRW$, $(T)ALRW$), respectivamente. A última coluna mostra o ganho em porcentagem da acurácia em relação a união das duas classes. A Tabela 5 mostra que o ganho da média de acurácia dos problemas com quatro classes em relação aos problemas com cinco classes foi de **2,1%**.

Classe	*Acurácia*	*Classe*	*Acurácia*	*Ganho (%)*
$(AT)LRW$	88,91	$(A)TLRW$	87,83	**1,2**
$(AT)LRW$	88,91	$(T)ALRW$	83,95	**5,9**
$(AT)LRW$	88,91	$(\overline{AT})LRW$	85,89	**3,5**

Tabela 4: Estratégia *Um contra Todos* sem rejeição de imagens e unindo *Plain Arch* e *Tented Arch* .

um-contra-todos	*4 classes*	*5 classes*	*Ganho*
Média	92,33	90,43	**2,1**
Desvio Padrão	2,25	4,01	–

Tabela 5: Ganho médio em acurácia.

Na Tabela 6, comparamos os resultados de acurácia do problema $(AT)L$ com AL e TL, do problema $(AT)R$ com AR e TR, e do problema $(AT)W$ com AW e TW. A última coluna mostra o ganho em porcentagem da acurácia em relação à união das duas classes. A Tabela 7 mostra que o ganho da média de acurácia dos problemas com quatro classes em relação aos problemas com cinco classes foi de **4,1%**. Ainda na Tabela 7, na terceira linha, observa-se uma anomalia no ganho de acurácia. O ganho foi negativo, ou seja, perdeu-se precisão. Isso, provavelmente, deve-se ao fato de algumas instâncias de *Tented*

Arch serem parecidas com *Right Loop* , o que deve ter levado o classificador a cometer erros.

Classe	*Acurácia (%)*	*Classe*	*Acurácia (%)*	*Ganho (%)*
$(AT)L$	95,44	AL	92,38	**3,3**
$(AT)L$	95,44	TL	87,74	**8,8**
$(AT)R$	90,38	AR	91,37	**-1,1**
$(AT)R$	90,38	TR	87,96	**2,8**
$(AT)W$	97,24	AW	95,99	**1,3**
$(AT)W$	97,24	TW	96,49	**0,8**

Tabela 6: Estratégia *Todos os Pares* sem rejeição de imagens e unindo *Plain Arch* e *Tented Arch* .

todos-os-pares	*4 classes*	*5 classes*	*Ganho*
Média	94,95	91,18	**4,1**
Desvio Padrão	2,70	6,88	–

Tabela 7: Ganho médio em acurácia.

5.2 Rejeitando exemplos ruins

Quando trabalhamos com imagens de impressões digitais os resultados podem ser influenciados pela qualidade das imagens. Imagens parciais ou que não se consegue distinguir as linhas da impressão digital podem levar um classificador de impressões digitais a errar. Ao extrairmos o campo direcional de imagens de má qualidade, o campo direcional fica afetado pelo ruído que a imagem contém e isso leva um classificador a errar. Por exemplo, na Figura 30 mostramos à esquerda a imagem de uma *Left Loop* e de uma *Tented Arch* ou *Right Loop* . À direita mostramos a imagem do campo direcional extraído. Observe que o campo direcional assemelha-se ao campo direcional de uma *Plain Arch* para os dois exemplos.

Para rejeitarmos imagens de má qualidade, precisamos de algum critério de qualidade de imagem e de um limite de qualidade, que quando violado, uma imagem seja rejeitada. O critério de qualidade escolhido foi o mesmo adotado em [16]. Quando calculando o campo direcional de uma imagem com máscaras de 16×16, nós obtemos uma imagem de vetores de 32×32, onde a direção de um vetor corresponde a direção da linha e o comprimento dos vetores dão o grau de confiança[3] da direção de uma linha em particular. Fazemos então a média dos comprimentos dos 32×32 vetores e chamamos isso de qualidade da

[3]Do inglês: confidence value.

imagem. As imagens que estiverem abaixo de um certo limite serão rejeitadas. Então usando esse critério de qualidade, nos rejeitamos as 20 (1,0%) piores imagens do grupo f e as 36 (1,8%) piores imagens do grupo s, como pode ser verificado na Tabela 8.

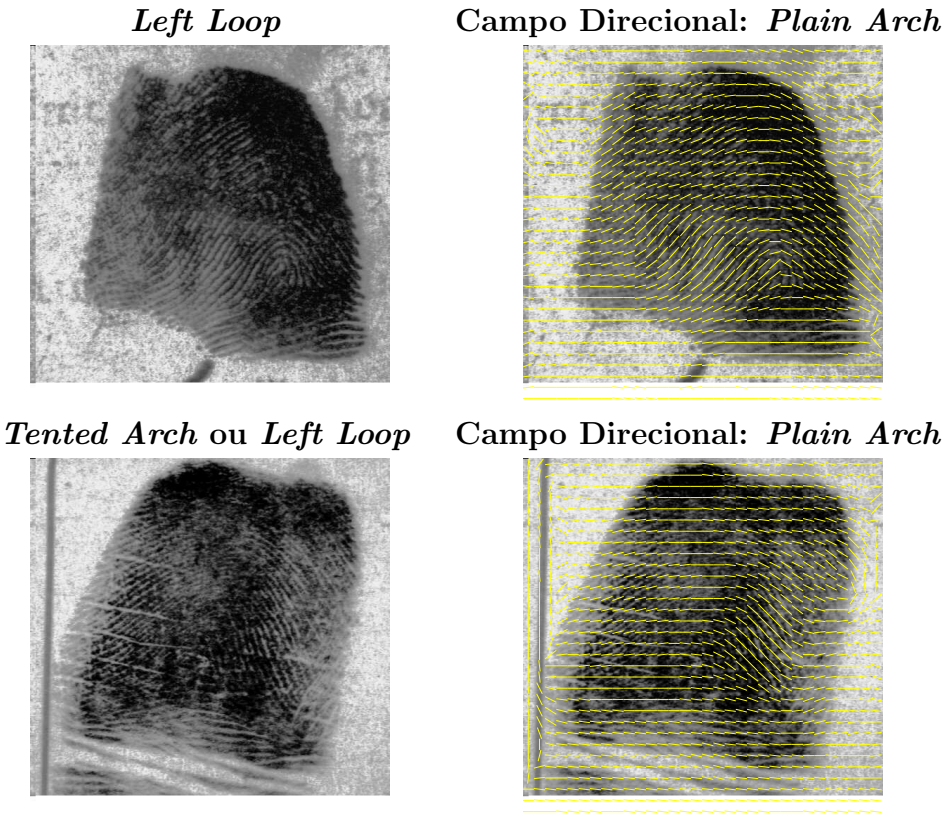


Figura 30: Exemplos de imagens de má qualidade de impressões digitais.

	Plain Arch	*Tented Arch*	*Left Loop*	*Right Loop*	*Whorl*	$\sum$	%
f	6	4	6	2	2	**20**	**1,0**
s	9	7	15	5	0	**36**	**1,8**
$\sum$	**15**	**11**	**21**	**7**	**2**	–	–
%	**0,38**	**0,28**	**0,53**	**0,18**	**0,05**	–	–

Tabela 8: Quantidade de imagens de má qualidade de impressões digitais rejeitadas.

5.3 Resultados após rejeição de imagens

Após a rejeição de 56 (1,4%) imagens do conjunto de dados do NIST-4, executamos novamente as fases de treinamento e teste. Os resultados estão resumidos na Tabelas 9 e 10.

A Tabela 9 compara os resultados antes e após a rejeição das imagens de má qualidade para a estratégia *um-contra-todos*. O ganho médio de **0,66%** não é significativo e mostra que as impressões digitais que atrapalham a classificação ainda continuam entre os dados. Por outro lado, os resultados revelam as suspeitas de que imagens de má qualidade interferem na classificação. Com exceção do problema $(W)ALRT$, os outros tiveram um ganho de classificação. O problema $(W)ALRT$ nos surpreendeu com o resultado: houve perda de acurácia na classificação. A provável causa para isso é porque alguma impressão digital rejeitada poderia estar fazendo parte do conjunto dos vetores suporte e logo o classificador ficou menos eficiente.

Classe	*COM Rejeição Acurácia (%)*	*SEM Rejeição Acurácia (%)*	*Ganho*
$(A)LRTW$	88,14	87,83	**0,35**
$(T)ALRW$	85,54	83,95	**1,89**
$(L)ARTW$	94,03	93,52	**0,54**
$(R)ALTW$	92,65	91,94	**0,77**
$(W)ALRT$	94,80	94,96	**-0,16**
Média	**91,03**	**90,43**	**0,66**
Desvio Padrão	**3,59**	**4,01**	

Tabela 9: Estratégia *um-contra-todos* rejeitando imagens.

A Tabela 10 compara os resultados antes e após a rejeição das imagens de má qualidade para a estratégia *todos-os-pares*. O ganho médio de **0,46%** também não é significativo e mostra que as impressões digitais que atrapalham a classificação ainda continuam entre os dados. Por outro lado, os resultados revelam as suspeitas de que imagens de má qualidade interferem na classificação. Com exceção dos problemas LR, RW e TW, os outros tiveram um ganho de classificação. Para os problemas LR, RW e TW, também houve perda de acurácia na classificação. A causa é idêntica a mencionada no parágrafo anterior ou seja, alguma impressão digital rejeitada poderia estar fazendo parte do conjunto dos vetores suporte.

5.3.1 Unindo *Plain Arch* e *Tented Arch* em uma classe

Novamente unimos as classes de impressões digitais *Plain Arch* e *Tented Arch* para verificarmos o ganho na classificação. As Tabelas 11 e 12 mostram a comparação dos resultados.

Para termos noção do ganho total, elaboramos a Tabela 13 que mostra a o ganho em relação a média de acurácia para 4 e 5 classes com e sem rejeição de imagens. As classes

Classe	*COM Rejeição Acurácia (%)*	*SEM Rejeição Acurácia (%)*	*Ganho*
AL	92,86	92,38	**0,51**
AR	92,67	91,37	**1,42**
AT	73,64	73,18	**0,62**
AW	96,41	95,99	**0,43**
LR	98,79	98,84	**-0,05**
LT	88,64	87,74	**1,02**
LW	93,74	93,66	**0,08**
RT	89,43	87,96	**1,67**
RW	93,55	94,16	**-0,64**
TW	96,26	96,49	**-0,23**
Média	**91,60**	**91,18**	**0,46**
Desvio Padrão	**6,66**	**6,88**	-

Tabela 10: Estratégia *todos-os-pares* rejeitando imagens.

de impressões digitais *Plain Arch* e *Tented Arch* por serem as mais difíceis de classificar, quando unidas em uma única classe, realmente melhoram a acurácia de classificação.

Classe	*Acurácia (%)*	*Classe*	*Acurácia (%)*	*Ganho (%)*
$(AT)LRW$	90,05	$(A)TLRW$	88,14	**2,2**
$(AT)LRW$	90,05	$(T)ALRW$	85,54	**5,3**
$(AT)LRW$	90,05	$(\overline{AT})LRW$	86,84	**3,7**

	4 classes	*5 classes*	*Gain*
Média	92,88	91,03	2,0
Desvio Padrão	1,81	3,59	–

Tabela 11: Estratégia *um-contra-todos* rejeitando imagens e unindo *Plain Arch* e *Tented Arch* .

5.4 Comparando os resultados com outros trabalhos

Nossa primeira comparação é com o trabalho de Jain, Prabhakar e Hong publicado em 1999 [3]. Nesse artigo, eles usaram como banco de dados o NIST-4 e dividiram as impressões digitais em cinco categorias: *Whorl* , *Right Loop* , *Left Loop* , *Plain Arch* e *Tented Arch* . O algoritmo de extração de características consistia em separar o número de linhas existente em uma impressão digital em quatro direções (0°, 45°, 90° e 135°) filtrando a parte central de uma impressão digital com um banco de filtros de Gabor. Essa informação era quantificada para gerar um código da impressão digital (forma de representação), chamado de *"FingerCode"*, que foi usado para classificação. A classificação era baseada em um classificador de dois estágios que usava um classificador *K-vizinhos mais próximos*,

Classe	*Acurácia (%)*	*Classe*	*Acurácia (%)*	*Ganho (%)*
$(AT)L$	95,36	AL	92,86	**2,7**
$(AT)L$	95,36	TL	88,64	**7,6**
$(AT)R$	92,08	AR	92,67	**-0,6**
$(AT)R$	92,08	TR	89,43	**2,9**
$(AT)W$	97,17	AW	96,41	**0,8**
$(AT)W$	97,17	TW	96,26	**0,9**

	4 classes	*5 classes*	*Ganho*
Média	95,12	91,60	**3,8**
Desvio Padrão	2,29	6,66	–

Tabela 12: Estratégia *todos-os-pares* rejeitando imagens e unindo *Plain Arch* e *Tented Arch* .

Todos exemplos	*4 classes*	*5 classes*	*Ganho*
um-contra-todos	92,33	90,43	**2,1**
todos-os-pares	94,95	91,18	**2,6**
Com rejeição	*4 classes*	*5 classes*	*Ganho*
um-contra-todos	92,88	91,03	**1,4**
todos-os-pares	95,12	91,60	**3,8**

Tabela 13: Resumo do ganho em acurácia.

no primeiro estágio, e um conjunto de redes neurais, no segundo estágio. Eles dividiram o problema de cinco classes em um conjunto de dez problemas de duas classes, ou seja, usaram somente a estratégia *todos-os-pares*. O classificador foi testado sobre as 4.000 imagens no banco de dados NIST-4. Para as cinco classes de problema, a acurácia de classificação alcançada foi de $90,0\%$, com critério de rejeição de 1,8%. Para o problema de quatro classes (*Plain Arch* e *Tented Arch* combinadas em uma única classe), eles alcançaram uma acurácia de classificação de $94,8\%$, com critério de rejeição de 1,8%.

A Tabela 14 resume os resultados dos dois trabalhos. Percebemos que a nossa abordagem conseguiu ser um pouco melhor mesmo sem critério de rejeição algum. Isso mostra duas idéias: a força de SVMs em gerar um modelo de classificação e que campos direcionais tem informação suficiente para a extração de um modelo de classificação. Isso significa que o processo de extração de características que usa muitos passos pode ser dispensável.

Estratégia: todos-os-pares	*Rejeição (%)*	*4 classes*	*Ganho*	*5 classes*	*Ganho*
SVM + Campo Direcional	0,0	94,95	**0,15**	91,18	**1,31**
SVM + Campo Direcional	1,4	95,12	**0,33**	91,60	**1,77**
Jain, Prabhakar e Hong, 1999 [3]	1,8	94,80	-	90,00	-

Tabela 14: Comparando resultados com Jain, Prabhakar e Hong, 1999 [3].

Nossa segunda comparação é com o trabalho de Yao, Frasconi e Pontil publicado em 2001 [4]. Nesse artigo, eles usaram como banco de dados o NIST-4 e dividiram as impressões digitais em cinco categorias: *Whorl* , *Right Loop* , *Left Loop* , *Plain Arch* e *Tented Arch* . O algoritmo de extração de características foi o mesmo utilizado por Jain, Prabhakar e Hong, 1999 [3]: filtro de Gabor (*"FingerCode"*). A classificação era baseada em Máquinas de Vetores Suporte - SVMs. Eles dividiram o problema de cinco classes entre as estratégias *todos-os-pares*, *um-contra-todos* e Códigos de Correção de Erros - ECC[4]. O classificador foi testado sobre as 4.000 imagens no banco de dados NIST-4. Para as cinco classes de problema, a acurácia de classificação alcançada foi de $88,04\%$, para a estratégia *um-contra-todos*, de $88,13\%$, para a estratégia *todos-os-pares* e de $89,26\%$, para a estratégia *ECC*, ambos com critério de rejeição de 1,8%. Para o problema de quatro classes (*Plain Arch* e *Tented Arch* combinadas em uma única classe), eles alcançaram uma acurácia de classificação de $93,30\%$, para a estratégia *um-contra-todos* e $93,00\%$, para a estratégia *todos-os-pares* e de $93,49\%$, para a estratégia *ECC*, ambos com critério de rejeição de 1,8%.

A Tabela 15, resume os resultados dos dois trabalhos. Percebemos que a nossa abordagem só não conseguiu ser melhor do que o outro trabalho na estratégia *um-contra-todos* com quatro classes.

	Abordagem	*Rejeição (%)*	*4 classes*	*5 classes*
SVM + Campo Direcional	*um-contra-todos*	0,0	92,33	**90,43**
SVM + Campo Direcional	*todos-os-pares*	0,0	**94,95**	**91,18**
SVM + Campo Direcional	*um-contra-todos*	1,4	92,88	**91,03**
SVM + Campo Direcional	*todos-os-pares*	1,4	**95,12**	**91,60**
Yao, Frasconi e Pontil, 2001	*um-contra-todos*	1,8	93,30	88,04
Yao, Frasconi e Pontil, 2001	*todos-os-pares*	1,8	93,00	88,13
Yao, Frasconi e Pontil, 2001	*ECC*	1,8	93,49	89,26

Tabela 15: Comparando resultados com Yao, Frasconi e Pontil, 2001 [4].

Nossa terceira e última comparação é com o trabalho de Yao, Marcialis, Pontil, Frasconi e Roli publicado em 2001 [5]. Nesse artigo, eles usaram como banco de dados o NIST-4 e dividiram as impressões digitais em cinco categorias: *Whorl* , *Right Loop* , *Left Loop* , *Plain Arch* e *Tented Arch* . O algoritmo de extração de características foi o mesmo utilizado por Jain, Prabhakar e Hong, 1999 [3]: filtro de Gabor (*"FingerCode"*). Eles apresentaram algoritmos de classificação de impressões digitais baseados em duas abordagens de aprendizagem de máquinas: máquinas de vetores suporte (SVMs) e redes neurais

[4]Do inglês, *Error-Correcting Codes*

recursivas (RNNs). RNNs são treinadas sobre uma representação estruturada da imagem da impressão digital. Elas são então usadas para extrair um conjunto de características distribuídas que podem ser integradas nas SVMs. SVMs são combinadas então com um novo esquema de código de correção de erro (ECC) que, ao contrário de outros, podem explorar informação contida em imagens de impressões digitais ambíguas. Eles dividiram o problema de cinco classes entre as estratégias *todos-os-pares*, *um-contra-todos* e Códigos de Correção de Erros - ECC[5]. O classificador foi testado sobre as 4.000 imagens no banco de dados NIST-4. Para as cinco classes de problema, a melhor acurácia de classificação alcançada foi $89,97\%$, para a estratégia *ECC, SVM + RNN*, com critério de rejeição de 1,8%. Para o problema de quatro classes (*Plain Arch* e *Tented Arch* combinadas em uma única classe), a melhor acurácia de classificação alcançada foi de $93,49\%$, para a estratégia *ECC*, com critério de rejeição de 1,8%.

Percebemos, pela Tabela 16, que a nossa abordagem só não conseguiu novamente ser melhor do que o outro trabalho na estratégia *um-contra-todos* com quatro classes.

	Abordagem	*Rejeição (%)*	4 classes	5 classes
SVM + Campo Direcional	*um-contra-todos*	0,0	92,33	**90,43**
SVM + Campo Direcional	*todos-os-pares*	0,0	**94,95**	**91,18**
SVM + Campo Direcional	*um-contra-todos*	1,4	92,88	**91,03**
SVM + Campo Direcional	*todos-os-pares*	1,4	**95,12**	**91,60**
Yao et al, 2001	*um-contra-todos*	1,8	93,30	88,04
Yao et al, 2001	*todos-os-pares*	1,8	93,00	88,13
Yao et al, 2001	*ECC*	1,8	93,49	89,26
Yao et al, 2001	*ECC, SVM + RNN*	1,8	93,09	89,97

Tabela 16: Comparando resultados com Yao et al, 2001 [5].

5.5 Conclusão

Como conclusão, percebemos que Máquinas Vetores Suporte - SVMs fornecem uma infra-estrutura robusta e rápida tanto para a geração de modelos (fase de treinamento) como para a classificação de impressões digitais (fase de testes). Todavia, a técnica de SVM depende da qualidade dos dados de treinamento para extrair bons modelos. Muitas técnicas de extração de características de impressões digitais usam inúmeros passos de pré-processamento. A maioria delas tem como primeiro passo o campo direcional. A nossa intuição, antes de realizar esse trabalho, era que o campo direcional já trazia informações suficientes para a classificação, sendo desnecessários outros passos. Os resul-

[5]Do inglês, *Error-Correcting Codes*

tados mostraram que valores equivalentes foram obtidos, até mesmo superados, apenas usando as informações de campo direcional. Como relatado em outros trabalhos, um critério de rejeição de imagens de má qualidade deve ser usado para melhorar a acurácia de um classificador de impressão digital. Quando a imagem não dispõe de qualidade suficiente, o campo direcional extraído é deturpado e o classificador gera um modelo de menor eficiência. O critério de rejeição usado foi simplório e mostrou que deveria ser melhorado, pois em alguns casos o classificador obteve acurácia menor do que sem o critério de rejeição. Isso revela que a rejeição removeu alguma instância que fazia parte dos vetores suporte dos hiperplanos de separação das classes, induzindo o classificador a errar.

5.6 Trabalhos futuros

Para trabalhos futuros os seguintes itens são intencionados:

- **Estudo do tamanho da janela:** para os experimentos realizados, usamos uma janela de 16×16. Entretanto um estudo sobre o tamanho adequado de janela para a extração de características pode ser feito. Se conseguíssemos resultados de acurácia próximos aos obtidos usando uma janela maior, por exemplo 32×32, o tamanho do vetor de características seria menor e o processo de geração de modelo (treinamento) seria mais rápido. Por outro lado, se diminuíssemos o tamanho da janela, por exemplo 8×8, poderíamos extrair maiores detalhes da imagem. O vetor de características seria maior, mas a acurácia de classificação poderia ser maior também. É claro que ao extrairmos mais detalhes levaríamos também ruídos que a imagem por ventura pudesse ter.

- **Códigos de Correção de Erro (ECC):** ECC são uma técnica para melhorar o desempenho de algoritmos de classificação em problemas de aprendizagem de múltiplas classes. Alguns algoritmos, como Máquinas de Vetores Suporte, somente conseguem manipular problemas com duas classes. Para aplicar tais algoritmos a conjuntos de dados de múltiplas classes, os dados são decompostos em vários problemas independentes de duas classes. Então o algoritmo é executado com cada um dos problemas e os resultados dos classificadores são combinados. Códigos de Correção de Erro são um esquema para construir a maior parte dessa transformação. O método parece trabalhar tão bem que é freqüentemente vantajoso usá-lo mesmo quando o algoritmo de aprendizagem manipula múltiplos conjuntos de dados diretamente. [29]

  No trabalho de [4], ECC foram usados para melhorar a acurácia do classificador.

Os resultados encorajam a incorporação dessa técnica, pois podemos melhorar a acurácia em algumas classes de problemas.

- **Outros métodos e taxas de rejeição:** Como critério de qualidade da imagem usamos a média dos vetores gradiente obtidos do campo direcional. Existem outros critérios que podem avaliar melhor a qualidade da imagem como relatado em [4, 5]. Usamos uma taxa de rejeição de $1,4\%$ onde outros trabalhos usaram taxas mais elevadas variando de $1,8\%$ até $30,0\%$. Um complemento seria estudar nos intervalos de rejeição relacionar com o ganho de acurácia.

Referências Bibliográficas

1 HALICI, U.; JAIN, L. C.; EROL, A. Introduction to Fingerprint Recognition. In: JAIN, L. C. et al. (Ed.). *Intelligent Biometric Techniques in Fingerprint and Face Recognition*. [S.l.]: The CRC Press, 1999, (International Series on Computational Intelligence). cap. 1, p. 3–34.

2 GUNN, S. *MATLAB Support Vector Machine Toolbox (version 2.0, Aug-1998)*. 1998. Image Speech and Intelligent Systems Group University of Southampton. Disponível em: <http://www.isis.ecs.soton.ac.uk/resources/svminfo/>.

3 JAIN, A. K.; PRABHAKAR, S.; HONG, L. A multichannel approach to fingerprint classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 21, n. 4, p. 348–359, 1999.

4 YAO, Y.; FRASCONI, P.; PONTIL, M. Fingerprint classification with combinations of support vector machines. *Lecture Notes in Computer Science*, v. 2091, p. 253–??, 2001. Disponível em: <citeseer.nj.nec.com/yao01fingerprint.html>.

5 YAO, Y. et al. A new machine learning approach to fingerprint classification. *Lecture Notes in Computer Science*, v. 2175, p. 57–??, 2001. Disponível em: <citeseer.nj.nec.com/yao01new.html>.

6 ASHBOURN, J. *Biometrics: Advanced Identity Verification, The Complete Guide*. [S.l.]: Springer-Verlag London, 2000.

7 JAIN, A. K.; HONG, L.; BOLLE, R. *On-line Fingerprint Verification*. East Lansing, Michigan, March 1996. 35 p.

8 WAYMAN, J. L. Error Rate Equations for the General Biometric System. *IEEE Robotics & Automation*, v. 6, n. 9, January 1999.

9 STOSZ, J. D.; ALYEA, L. A. Automated Systems for Fingerprint Authentication Using Pores and Ridge Structure. *Proceedings of SPIE , Automatic Systems for the Identification and Inspection of Humans*, v. 2277, p. 210–223, 1994.

10 RODDY, A. R.; STOSZ, J. Fingerprint Feature Processing Techniques and Poroscopy. In: JAIN, L. C. et al. (Ed.). *Intelligent Biometric Techniques in Fingerprint and Face Recognition*. [S.l.]: The CRC Press, 1999, (International Series on Computational Intelligence). cap. 2, p. 37–105.

11 DRETS, G. A.; LILJENSTRöM, H. G. Fingerprint Sub-classification: A Neural Network Approach. In: JAIN, L. C. et al. (Ed.). *Intelligent Biometric Techniques in Fingerprint and Face Recognition*. [S.l.]: The CRC Press, 1999, (International Series on Computational Intelligence). cap. 3, p. 109–134.

12 CAPPELLI, R. et al. Fingerprint Classification by Directional Image Partitioning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 21, n. 5, p. 402 – 421, Maio 1999.

13 GONZALEZ, R. C.; WOODS, R. E. *Digital Image Processing*. [S.l.]: Addison Wesley, 1992.

14 RATHA, N. K.; CHEN, S.; JAIN, A. K. *Adaptive Flow Orientation Based Feature Extraction In Fingerprint Images*. East Lansing, Michigan, March 1999.

15 HONG, L. et al. *Identity Authentication Using Fingerprints*. East Lansing, Michigan, January 1996. 5 p.

16 KARU, K.; JAIN, A. K. *Fingerprint Classification*. East Lansing, Michigan, April 1999.

17 BAZEN, A. M.; GEREZ, S. H. Directional Field Computation for Fingerprints Based on the Principal Component Analysis of Local Gradients. In: *ProRISC 2000 Workshop on Circuits, Systems and Signal Processing*. Veldhoven, The Netherlands: [s.n.], 2000.

18 HONG, L.; WAN, Y.; JAIN, A. Fingerprint Image Enhancement: Algorithm and Performance Evaluation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 20, n. 8, p. 777 – 789, Agosto 1998.

19 PONTIL, M.; VERRI, A. *Properties of Support Vector Machines*. [S.l.], 1997. 17 p. Disponível em: <citeseer.nj.nec.com/pontil97propertie.html>.

20 CRISTIANINI, N.; SHAWE-TAYLOR, J. *An Introduction to Support Vector Machines and other kernel-based learning methods*. [S.l.]: Cambridge University Press, 2000.

21 BURGES, C. J. C. A Tutorial on Support Vector Machines for Pattern Recognition. *Data Mining and Knowledge Discovery*, v. 2, n. 2, p. 121–167, 1998. Disponível em: <citeseer.nj.nec.com/burges98tutorial.html>.

22 MüLLER, K.-R. et al. An Introduction to Kernel-Based Learning Algorithms. *IEEE Transactions on Neural Networks*, v. 12, n. 2, p. 181–201, Março 2001.

23 SCHöLKOPF, B. *Support Vector Learning*. Tese (Doutorado) — Universität Berlin, 1997.

24 CORTES, C.; VAPNIK, V. Support-Vector Networks. *Machine Learning*, v. 20, n. 3, p. 273–297, 1995. Disponível em: <citeseer.nj.nec.com/cortes95supportvector.html>.

25 ROOBAERT, D.; HULLE, M. M. V. *View-Based 3d Object Recognition With Support Vector Machines*. 1999. Disponível em: <citeseer.nj.nec.com/roobaert99viewbased.html>.

26 GUNN, S. R. *Support Vector Machines for Classification and Regression*. [S.l.], 1998.

27 RIFKIN, R. *SVMFu v3.0*. 2000. MIT's Center for Biological and Computational Learning, and at Compaq's Cambridge Research Lab. Disponível em: <http://five-percent-nation.mit.edu/SvmFu/>.

28 JOACHIMS, T. *SVM^{light} v5.0*. 2002. Cornell University, Department of Computer Science. Disponível em: <http://svmlight.joachims.org/>.

29 WITTEN, I.; FRANK, E. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. [S.l.]: Morgan Kaufmann, 2000.