Diego Alonso Fernández Merjildo

Algoritmo *AdaBoost* robusto ao ruído: Aplicação à detecção de faces em imagens de baixa resolução

Campinas
2013

Universidade Estadual de Campinas
Faculdade de Engenharia Elétrica e Computação

Diego Alonso Fernández Merjildo

Algoritmo *AdaBoost* robusto ao ruído: Aplicação à detecção de faces em imagens de baixa resolução

Dissertação de mestrado apresentada à Faculdade de Engenharia Elétrica e Computação da Universidade Estadual de Campinas, como parte dos requisitos exigidos para a obtenção do título de Mestre em Engenharia Elétrica, na Àrea de Telecomunicações e Telemática.

Orientador: Prof. Dr. Lee Luan Ling

ESTE EXEMPLAR CORRESPONDE À VERSÃO FINAL DISSERTAÇÃO DEFENDIDA PELO ALUNO DIEGO ALONSO FERNANDEZ MERJILDO, E ORIENTADA PELO PROF. DR. LEE LUAN LING

Campinas
2013

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca da Área de Engenharia e Arquitetura
Elizangela Aparecida dos Santos Souza - CRB 8/8098

F391a

Fernandez Merjildo, Diego Alonso, 1982-
Algoritmo AdaBoost robusto ao ruído : aplicação à detecção de faces em imagens de baixa resolução / Diego Alonso Fernandez Merjildo. – Campinas, SP : [s.n.], 2013.

Orientador: Luan Ling Lee.
Dissertação (mestrado) – Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Aprendizado de máquina. I. Lee, Luan Ling,1956-. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

Informações para Biblioteca Digital

Título em outro idioma: Noise robust AdaBoost algorithm : Applying to face detection in low resolution images.
Palavras-chave em inglês:
Machine learning
Área de concentração: Telecomunicações e Telemática
Titulação: Mestre em Engenharia Elétrica
Banca examinadora:
Luan Ling Lee [Orientador]
Marco Antonio Garcia de Carvalho
Romis Ribeiro de Faissol Attux
Data de defesa: 06-12-2013
Programa de Pós-Graduação: Engenharia Elétrica

COMISSÃO JULGADORA - TESE DE MESTRADO

Candidato: Diego Alonso Fernández Merjildo

Data da Defesa: 6 de dezembro de 2013

Título da Tese: "Algoritmo AdaBoost Robusto ao Ruído: Aplicação à Detecção de Faces em Imagens de Baixa Resolução"

Prof. Dr. Lee Luan Ling (Presidente): _____

Prof. Dr. Marco Antonio Garcia de Carvalho: _____

Prof. Dr. Romis Ribeiro de Faissol Attux: _____

Resumo

O presente trabalho propõe um algoritmo *AdaBoost* modificado, que minimiza o efeito do *overfitting* no treinamento produzido por amostras ruídosas. Para este fim, a atualização da distribuição de pesos é feita baseado na fragmentação do erro de treinamento, o qual permite atualizar efetivamente as amostras classificadas incorretamente para cada nivel de taxa de erro. Subsequentemente, o algoritmo desenvolvido é aplicado no processo de detecção de faces, utilizando os Padrões Binários Locais Multi-Escala em Blocos (*Multiscale Block Local Binary Patterns (MB-LBP)*) como padrões característicos para formação de uma cascata de classificadores. Os resultados experimentais mostram que o algoritmo proposto é simples e eficiente, evidenciando vantagens sobre os algoritmos *AdaBoost* clássicos, em termos de maior capacidade de generalização, prevenção de overfitting e maiores taxas de acerto em imagens de baixa resolução.

Palavras-chave: Algoritmos de *Boosting*, Detecção de faces, *Overfitting*, Aprendizado de máquinas, Erros fragmentários, Fator de frequência

Abstract

This work aims a modification to the AdaBoost algorithm applied to face detection. Initially, we present the approaches used in face detection, highlighting the success of methods based on appearance. Then, we focus on the AdaBoost algorithm, its performance and the improvements realized by author as published literature. Despite the indisputable success of Boosting algorithms, it is affected by the high sensitivity to noisy samples. In order to avoid overfitting of noisy samples, we consider that the error rate is divided into fragmentary errors. We introduce a factor based on misclassified samples,to update the weight distribution in the training procedure. Furthermore, the algorithm developed is applied to face detection procedure, for which it is used Block Multiscale Local Binary Patterns (MB-LBP) in feature extraction as well as a cascade of classifiers. The experimental results show that the proposal to include a factor based on the frequency of misclassified samples, is simple and efficient, showing advantages over classical AdaBoost algorithms, which include ability to generalize, preventing overfitting and higher hit rates in images of low resolution.

Keywords: *Boosting* Algorithms, Face Detection, Overffiting, Machine Learning, Boosting, Fragmentary Errors, Frequency Factor

SUMÁRIO

DEDICATÓRIA

Dedicado a

mi querida, admirada, respetada, luchadora y cariñosa abuela Nicolaza Puente Silverio.

AGRADECIMENTOS

Esta dissertação é resultado de muito esforço e dedicação, e não poderia deixar de registrar aqui minha sincera gratidão a todos que estiveram ao meu lado durante o desenvolvimento deste trabalho.

Primeiramente gostaria de agradecer a Deus por ter me dado esta oportunidade e também saúde, coragem, persistência e sabedoria para poder chegar a mais esta conquista.

Aos meus pais, pela sólida formação que me foi dada, pelo incentivo e apoio incondicional que sempre me ajudaram em minhas conquistas. Aos meus irmãos Diana e Rodrigo, pelo amor, admiração e respeito.

Ao Prof. Dr. Lee Luan Ling, professor e orientador, pelo seu permanente apoio e atenção dispensada no decorrer deste trabalho. À sua disponibilidade irrestrita, sua paciência, sua forma amiga, exigente e crítica, fundamental contribuição no meu crescimento enquanto pesquisador.

Agradeço a todos professores do mestrado, amigos e funcionários da Universidade Estadual de Campinas, pelo auxílio e atenção ao longo destes meses.

À minha namorada, amiga e companheira, Laurita, pelo incansável apoio durante o desenvolvimento deste trabalho, por sua paciência e compreensão reveladas, fundamental nesta trajetória.

Obrigado aos colegas do Laboratório de Reconhecimento de Padrões e Comunicações e de mestrado, em especial aos amigos: Rodrigo "Digão" Alves do Nascimento, Yulios Zavala, Ana Chipana, Jose Huaman, Jeferson Stenico e Matheus Passos, pelas amizades então demonstradas, que, com uma perfeita mistura de dever e descontração, me ajudaram a cumprir minhas obrigações acadêmicas.

Por fim, deixo aqui minha sincera gratidão a todas as pessoas que, direta ou indiretamente, contribuíram para a concretização deste trabalho.

LISTA DE FIGURAS

LISTA DE TABELAS

ABREVIAÇÕES

AdaBoost Adaptive Boosting

AM Aprendizado de Máquina

ANN Artificial neural network

AS Almost Surely

Bagging Bootstrap aggregating

EMargin Equilibrium margin

ERM Empirical risk minimization

HOG Histogram of oriented gradients

IA Inteligência Artificial

LBP Local Binary Pattern

LP Linear Program

MB-LBP Multi-Scale Local Binary Pattern

PAC Probably Approximately Correct

PCA Principal component analysis

SRM Structural risk minimization

SVM Support Vector Machine

VC Vapnik-Chervonenkis

TRABALHOS PUBLICADOS PELO AUTOR

1. **Merjildo e Ling (2011)** Diego A. Fernandez Merjildo e Lee Luan Ling. Uma melhora para detecção de faces baseado no algoritmo adaboost. *Anais do Simpósio de Processamento de Sinais da UNICAMP (SPS 2011)*. Campinas, Brasil, 24 – 25 de Outubro, 2011.

2. **Merjildo e Ling (2012)** Diego Merjildo e Lee Ling. A robust adaboost-based algorithm for low-resolution face detection. Editors Hujun Yin, José Costa e Guilherme Barreto, *Intelligent Data Engineering and Automated Learning - IDEAL 2012*, volume 7435 of *Lecture Notes in Computer Science*, páginas 366-373. Springer Berlin / Heidelberg. ISBN 978-3-642-32638-7. URL http://dx.doi.org/10.1007/978-3-642-32639-4_45. 10.1007/978-3-642-32639-4_45. 28 – 31 de Agosto, 2012. Natal, Brasil.

3. **Merjildo e Luan Ling (2012)** Diego Merjildo e Lee Luan Ling. Enhancing the performance of adaboost algorithms by introducing a frequency counting factor for weight distribution updating. Editors Luis Alvarez, Marta Mejail, Luis Gomez e Julio Jacobo, *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*, volume 7441 of *Lecture Notes in Computer Science*, páginas 527-534. Springer Berlin / Heidelberg. ISBN 978-3-642-33274-6. URL http://dx.doi.org/10.1007/978-3-642-33275-3_65. 10.1007/978-3-642-33275-3_65. 3 – 6 de Setembro, 2012. Buenos Aires, Argentina.

1 INTRODUÇÃO E JUSTIFICATIVA

Nos últimos anos, os computadores têm adquirido maiores recursos de processamento, ganhando novas habilidades, bem como aumento de capacidade de processamento e possibilidades de interagir com os humanos. A detecção automática de faces pelos computadores tem sido uma área muito pesquisada, sendo vista como uma etapa inicial de muitas aplicações e técnicas de processamento de imagens de face, como rastreio de faces, reconhecimento de expressões faciais, sistemas de interação humano-computador, sistemas de identificação através de faces ou íris.

Dada uma imagem ou vídeo, um detector de faces ideal deve ser capaz de identificar e localizar todas as faces presentes, independentemente da sua posição, escala, idade e expressão (Li e Jain , 2011). Nesse sentido, os algoritmos de maior sucesso nesta área são os métodos baseados na aparência, sendo o *AdaBoost (Boosting Adaptativo)* (Zhang e Zhang , 2010) o algoritmo mais usado. Os algoritmos baseados na aparência processam a imagem de entrada, detectando características locais em diferentes escalas para treinamento do classificador. Em outras palavras, um classificador treinado analisa o conteúdo de algumas partes da imagem e as codifica em características específicas. Estas características retornam uma resposta afirmativa ou negativa, caso o classificador determine a existência ou ausência de uma face, respectivamente.

As abordagens baseadas nos algoritmos de *Boosting* Adaptativo (*AdaBoost*) (Freund e Schapire , 1995) têm sido os mais eficientes em termos de taxas de detecção e falsos positivos, entre os algoritmos baseados na aparência (Viola e Jones , 2004), (Li e Zhang , 2004), (Lienhart *et al.* , 2003). Mais especificamente, Viola e Jones (2004) desenvolveram seu algoritmo para detecção de faces baseados no conceito de Imagem Integral, características do tipo *Haar*, o algoritmo *AdaBoost* clássico, e uma cascata de classificadores, conseguindo atingir taxas de detecção bastante competitivas.

Apesar deste sucesso, a detecção de faces em imagens ou vídeo continua sendo um problema

desafiador (Zhang *et al.* , 2011), especificamente quando as imagens apresentam baixa resolução, os objetos estão distantes ou as câmeras precisam cobrir grandes ângulos de vista. Nesse sentido, Hayashi e Hasegawa (2006) estudaram a relação entre a resolução e a taxa de detecção em imagens de baixa resolução, propondo uma técnica de detecção dedicada.

O algoritmo *AdaBoost* foi um dos primeiros algoritmos de *Boosting* a ser amplamente usado em sistemas de tempo real, por causa da sua simplicidade e adaptabilidade. No entanto, a propriedade do algoritmo *AdaBoost* de maximizar as margens de classificação nem sempre conduz à melhor generalização (Rätsch *et al.* , 1998). Grove e Schuurmans (1998) e Quinlan (1996) observaram que o algoritmo *AdaBoost*, o qual cria uma combinação linear de classificadores, frequentemente apresenta um desempenho inferior do que um classificador individual na presença de ruído. Além disso, Dietterich (2000), Rätsch *et al.* (2001) e Servedio (2003) mostraram que o algoritmo *AdaBoost* pode conduzir ao sobreajuste (*overfitting*) das amostras, na presença de dados altamente ruidosos. Hawkins (2004) indica que o sobreajuste aparece quando o número de classificadores aumenta de forma excessiva, ou seja, produz-se uma combinação de classificadores muito complexa, o que leva a uma deterioração do desempenho.

Durante o processo de treinamento, o algoritmo *AdaBoost* modifica os pesos das amostras, focando naquelas que são classificadas incorretamente e atribui pesos maiores para elas. Este procedimento pode provocar efeitos indesejáveis quando as amostras são altamente ruidosas. Isto é evidenciado quando os classificadores possuem um bom desempenho com amostras de treinamento, no entanto, o desempenho é inferior com amostras de teste.

Neste trabalho, com o propósito de evitar os problemas de *overfitting* descritos anteriormente, apresentamos um algoritmo *AdaBoost* modificado. Em particular, visamos a um desempenho melhorado na detecção de faces em imagens de baixa resolução. Analisamos e comparamos a abordagem desenvolvida com outras abordagens convencionais para validar a metodologia proposta.

1.1 Objetivos

O objetivo principal deste trabalho é desenvolver um novo algoritmo *AdaBoost*, menos sensível ao ruído. O algoritmo desenvolvido será aplicado na detecção de faces em imagens de baixa resolução, evitando efeito do sobre-ajuste de amostras. Os estudos técnicos desenvolvidos foram:

- Investigar as abordagens relacionadas com o processo de atualização de pesos nos algoritmos

de *Boosting* Adaptativo (*AdaBoost*), a fim de equacionar possíveis soluciones aos problemas de sobre-ajuste de amostras.

- Desenvolver e implementar um algoritmo de *Boosting* Adaptativo com maior resistência ao ruído, que atualize os pesos da amostras em função de novos fatores.
- Teste do método desenvolvido na tarefa de detecção de faces com imagens de baixa resolução.
- Avaliar o desempenho do algoritmo desenvolvido e compará-lo com os algoritmos *AdaBoost* clássicos, tais como *Real AdaBoost*, *GentleBoost*, e *Modest AdaBoost*.

1.2 Estrutura da dissertação

A presente dissertação é organizada em 7 capítulos e a descrição de cada um segue abaixo.

- **Capítulo 1.** Apresenta o panorama global do trabalho constituído por uma introdução, a motivação e os objetivos.
- **Capítulo 2.** Nesse capítulo, introduzimos os conceitos principais do Aprendizado de Máquinas, explicamos seus princípios, propriedades e características.
- **Capítulo 3.** Apresentamos os métodos de ensemble, seus princípios e propriedades. Explicamos os principais algoritmos e suas variações. Apresentamos também algumas variações no *AdaBoost* e as dividimos de acordo com a atualização de pesos, maximização da margem e a importância atribuída ao classificador.
- **Capítulo 4.** Apresentamos uma revisão dos principais método de detecção de faces. Discutimos os métodos baseados em conhecimento, em *templates*, em características (faciais, textura e cor da pele) e na aparência.
- **Capítulo 5.** Apresentamos uma descrição detalhada da abordagem desenvolvida nesta dissertação de Mestrado. Assim, introduzimos um algoritmo *AdaBoost* modificado com o propósito de superar os problemas de sobre-ajuste de amostras ruidosas. Esta abordagem visa a melhorar o desempenho do algoritmo no processo de detecção de faces em imagens de baixa resolução.

- **Capítulo 6.** São apresentados os resultados dos experimentos realizados para comprovação do aperfeiçoamento obtido na abordagem proposta. Além disto, é proporcionado um estudo comparativo entre a abordagem proposta (descrita no Capítulo anterior) e uma seleção de trabalhos representativos, que foram apresentados na revisão bibliográfica (apresentada nos Capítulos 2 e 3).
- **Capítulo 7.** Neste capítulo se apresentam as conclusões e contribuições do trabalho.

2 APRENDIZADO DE MÁQUINA

O matemático britânico Alan Turing criou o primeiro modelo abstrato de um computador que conseguiria avaliar entradas, modificá-las e gerar saídas. Ele foi um dos pioneiros em estudar Inteligência Artificial (IA) e, em 1950, chegou a postular que: "Podemos esperar que as máquinas compitam com os humanos em todos os campos puramente intelectuais" (Turing , 1995). Mais tarde, no seminário de Dartmouth de 1956, John McCarthy criou o termo "Inteligência Artificial" (McCarthy *et al.* , 2006), descrevendo o conceito como "a ciência e a engenharia capazes de dar inteligência às máquinas".

No entanto, o objetivo da Inteligência Computacional não é criar uma inteligência, mas sim adicionar comportamento inteligente a um processo, adicionar comportamento semelhante ao de um humano e compreender como o computador pode adquirir conhecimento a partir da observação do ambiente (processo de aprendizado).

Dentro da Inteligência Computacional, o Aprendizado de Máquina (do inglês, *Machine Learning*) é um campo de pesquisa que estuda o desenvolvimento de algoritmos de predição eficientes e precisos, capazes de extrair informação a partir de amostras de dados (Mitchell , 1997). O Aprendizado de Máquina (AM) é responsável pelo desenvolvimento de teorias computacionais focadas na criação do conhecimento artificial. Os algoritmos desenvolvidos possuem a habilidade de tomar decisões baseados em conhecimento prévio acumulado através da interação com o ambiente. As diversas técnicas de AM podem ser divididas segundo o paradigma utilizado na análise de dados. Os dois principais paradigmas são o aprendizado supervisionado e o aprendizado não supervisionado.

No aprendizado supervisionado, o objetivo é obter conhecimento a partir de amostras que estão pré-classificadas, ou seja, amostras rotuladas com uma classe conhecida. Assim, um supervisor utiliza esses rótulos para treinar um algoritmo de Aprendizado de Máquina, visando aproximar uma função desejada. Quando as classes possuírem valores discretos, o problema é categorizado

como de classificação. Caso as classes possuam valores contínuos, o problema é categorizado como de regressão.

No aprendizado não supervisionado, a informação acerca de qual seria a resposta é praticamente nula. Não existe um supervisor e, ao contrário do aprendizado supervisionado, apenas possuímos amostras de dados sem rótulos. O objetivo é encontrar as regularidades nessas amostras. Esse aprendizado, considera que dentro do conjunto de amostras existem estruturas que ocorrem com alta frequência. Um exemplo é *clustering*, onde o objetivo é encontrar *clusters* ou agrupar amostras com características similares.

Neste capítulo, são apresentados alguns conceitos da teoria de aprendizado estatístico, com destaque para o risco funcional, a minimização do risco empírico e a minimização do risco estrutural. Em seguida, são descritas uma série de definições utilizadas no aprendizado de máquina, além de serem abordados alguns algoritmos comumente utilizados. Na seção 3.4, apresentamos as técnicas de seleção de classificadores conhecidas pelo algoritmo *Boosting*. Em seguida (seção 3.5), apresentamos o algoritmo *AdaBoost*, e, finalmente (seção 3.6), suas variações.

2.1 Definições preliminares

O aprendizado a partir de dados suscita uma grande variedade de tópicos, tais como regressão, classificação, estimação de densidade e *clustering*. Neste trabalho, estuda-se principalmente classificação ou reconhecimento de padrões. Assim, na continuação, apresentamos alguns conceitos, ideias e resultados teóricos, que possuem grande importância e aplicabilidade.

- **Dados de treinamento,** $\mathcal{D}_n = \{(x_i, y_i), i = 1, \cdots, n\}$, onde as amostras são denotados por $x_i \in \mathcal{X} = \mathbb{R}^d$ e os rótulos das classes são $y_i \in \mathcal{Y} = \{0, 1\}$. O conjunto de pares (x_i, y_i) é linearmente independente com distribuição desconhecida $\mathcal{P}_{x,y}$. De forma alternativa, usa-se P(·) como uma notação genérica para qualquer outra função de probabilidade.

- **Regra de decisão** g(·). Um algoritmo de AM procura um mapa $g : \mathcal{X} \to \mathcal{Y}$ baseado nos dados $\mathcal{D}_n$. O mapa g depende dos dados de treinamento $\mathcal{D}_n$, considerando $g(x)$ como uma variável aleatória. O termo regra de decisão ou função de decisão pode ser associado a qualquer configuração de decisão. Neste trabalho, se usam os termos *regra de classificação* e *função de classificação*.

- **A função de perda** $L : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ define um critério de desempenho para uma escolha da função de classificação g. A perda *1-0*, conhecida também como *erro de classificação incorreta*, é uma das funções de perda mais usadas em problemas de classificação. Esta é dada por:

$$L(g(x), y) := I(g(x) \neq y) \tag{2.1}$$

  em que $I(q) = 1$ se q é verdadeiro e $I(q) = 0$ se q é falso.

- **O funcional de risco** ou perda esperada é definido como a probabilidade teórica com que uma função de classificação g fique exposta ao erro. O risco tem a seguinte forma:

$$R(g) := E_{\mathcal{P}} L(g(x), y) = \mathcal{P}(g(x) \neq y). \tag{2.2}$$

Frequentemente, procura-se encontrar uma função de classificação g com um risco $R(f)$ mínimo. Na seguinte seção esses conceitos são aprofundados.

2.2 Teoria do Aprendizado Estatístico

A teoria do aprendizado estatístico, ou teoria Vapnik-Chervonenkis (teoria VC) , foi introduzida no final da década de 60 e inícios da década de 70 por Vapnik e Chervonenkis (1971), os quais, de forma puramente teórica, analisaram o problema de encontrar uma função de estimação a partir de alguma coleção de dados. A teoria VC continuou sendo desenvolvida na década de 80 por Vapnik (1982). Por outra lado, Valiant (1984) introduziu um modelo teórico chamado de *Probably Approximately Correct* (PAC) , base dos algoritmos de *Boosting*. Na década de 90, novos algoritmos de aprendizado foram apresentados, tais como *Support Vector Machine* (SVM) (Cortes e Vapnik , 1995) e *Boosting* (Schapire , 1990), baseados no principio de minimização do risco estrutural (*Structural risk minimization* - SRM) de Vapnik (1991).

2.2.1 Funcional de Risco

Consideremos um espaço de entrada $\mathcal{X}$ e um espaço de saída $\mathcal{Y}$ restrito para classificação binária. Por tanto, temos que $\mathcal{Y} = \{-1, 1\}$. Formalmente, assumimos que os pares $(x, y) \in \mathcal{X} \times \mathcal{Y}$ são variáveis independentes e identicamente distribuídas, de acordo com uma distribuição desconhecida $\mathcal{P}$. Observamos uma sequência de n pares (x_i, y_i) amostrados de acordo com $\mathcal{P}$, onde nosso propósito é construir uma função $g() : \mathcal{X} \to \mathcal{P}$ que prediga Y a partir de X.

No entanto, precisamos de um critério para escolher esta função $g(x)$. Este critério foi definido por Vapnik (1995) como uma probabilidade de classificação incorreta $P(g(X) \neq Y)$, também chamada de funcional de risco $R(g)$, tendo como argumento a função $g(x)$, conforme apresentado na Equação 2.2. Esta definição pode ser generalizada para argumentos na forma de funções de regressão e de decisão limitada.

Dado que a distribuição $\mathcal{P}$ é desconhecida, medir o funcional de risco diretamente não é uma tarefa trivial. Além disso, não podemos obter diretamente o valor de alguma função $t(\cdot)$ que relacione X e Y ($Y = t(X)$). Nesse contexto, unicamente podemos medir o custo (função de perda) de uma função candidata $g(x)$ relacionada às amostras. A somatória destes custos chama-se de *funcional de risco empírico* (R_{emp}), e é dado pela Equação 2.3. Para isso, consideramos uma função de custo $c(g(x_i),y_i)$ relacionada à previsão $g(x_i)$ com saída desejada y_i. Para funções de decisão limitada, na forma $g : R^n \rightarrow \{-1,1\}$, o tipo de função de custo é a *perda 0/1*. Este tipo de função é definida por $c(g(x),y) = 1$ se $(yg(x)) < 0$ ou $c(g(x),y) = 0$ se $(yg(x)) \geq 0$ (Muller *et al.* , 2001).

$$R_{emp}(g) = \frac{1}{n}\sum_{i=1}^{n} c(g(x_i),y_i) \tag{2.3}$$

onde n é o número de amostras.

O desempenho de uma função g pode ser definido como a probabilidade de cometer um erro frente a uma nova amostra gerada por $\mathcal{P}$. A partir desta afirmação, define-se o *Funcional de Risco* que quantifica a capacidade de classificação da função g, como descrito na Equação 2.4 :

$$R(g) = \int c(g(x),y)dP(x,y) \tag{2.4}$$

Os algoritmos de aprendizado procuram minimizar o risco empírico $R_{emp}(g)$, esperando que isto leve a um valor mínimo do risco $R(g)$. Assim, na seleção de g, se opta por $\hat{g}$ (função que minimiza o risco empirico), tal que:

$$R_{emp}(\hat{g}) = \min_{g \in G} R_{emp}(g). \tag{2.5}$$

No entanto, é desejável que a função possua uma boa capacidade de generalização, o que significa dizer que se tenha g^* (função que minimiza o risco), tal que:

$$R(g^*) = \min_{g \in G} R(g). \tag{2.6}$$

O funcional de risco empírico corresponderá ao erro médio de treinamento, onde a minimização do funcional de risco desconhecida será substituída pela minimização do funcional de risco empírico que pode ser conhecida. Porém, dado que o conjunto de dados de treinamento é finito, obter $R_{emp}(g) = 0$ não implica sempre $R(g) = 0$.

A Teoria de Aprendizado Estatístico (TAE) apresenta uma variedade de limites no funcional de risco, os quais são usados na seleção de funções classificadoras. Assim, na próximas seções são apresentadas considerações específicas da TAE, bem como conceitos relacionados à minimização do risco funcional e empírico.

2.3 Considerações da Teoria de Aprendizado Estatístico

Vapnik (1998) afirma que o propósito da teoria de aprendizado estatístico é obter um modelo a partir de um espaço de funções H, o qual está relacionado à função $g_p(x)$ no espaço objetivo através de alguma medida de erro. Nesse contexto, existem dois tipos de erros: o erro de aproximação e o erro de estimação.

- **Erro de Aproximação (Bias).** Para um espaço H de funções candidatas, o erro de aproximação $E(g_H)$ é obtido quando o espaço H não contém todo o espaço objetivo, de tal forma que a função fundamental $g_P(\cdot)$ pode-se encontrar fora de H. Então, o erro de aproximação $E(g_H)$ dependerá do espaço H e da probabilidade desconhecida P. Portanto, se $g_P \in H$ então $g_H = g_P$ e o erro de aproximação será zero.

  No aprendizado estatístico, é importante fazer uma boa escolha do espaço H, dado que não fazer isto pode resultar num erro de aproximação grande, também conhecido como discordância do modelo.

- **Erro de Estimação (Variância).** Consideremos que $z \in Z^n$ é uma amostra, onde uma função objetivo empírica g_z minimiza o erro empírico $E_z(g)$ (Risco Empírico, Equação 2.3) com $g \in H$. Esta função é empírica porque depende da amostra z e não depende de P. Para um espaço de funções H, o erro de estimação E_H em H de uma função $g \in H$ é descrita da seguinte forma:

$$E_H(g_z) = E(g_z) - E(g_H) \tag{2.7}$$

Este erro é devido ao aprendizado que pode levar à seleção de um modelo ineficiente dentro de H. Os ditos erros formam o erro de generalização.

- **Margem**. A margem ρ com que uma amostra x_i é classificada mede a distância duma amostra em relação à fronteira $g(x) = 0$. Dado um par (x_i, y_i), Schapire e Freund (2012) definem a margem simplesmente como:

$$\rho(x_i, y_i) = y_i g(x_i) \tag{2.8}$$

A margem ρ_g da função g é definida como a margem miníma observada em todo o conjunto de dados, assim:

$$\rho_g = \min_{1 \leq i \leq n} \rho(x_i, y_i) = \min_{1 \leq i \leq n} y_i g(x_i) \tag{2.9}$$

A *margem de erro* é dada pela proporção de amostras de treinamento que têm margem menor que ρ. A estimativa empírica de risco conta as amostras como um erro nos casos em que a classificação é correta, mas com margem menor do que ρ, se a classificação é incorreta. Nesse caso a função de custo assume a seguinte forma:

$$c(g(x_i), y_i) = \begin{cases} 1 & \text{se } |g(x_i) - y_i| \geq \rho \\ 0 & \text{se } |g(x_i) - y_i| < \rho \end{cases} \tag{2.10}$$

A teoria da convergência uniforme (Vapnik , 1982) indica que, antes de empregar um algoritmo de aprendizado para classificação, todas as funções de classificação devem ser igualmente prováveis. Assim, durante a execução do algoritmo de aprendizado, a plausibilidade das funções de classificação $g(x)$ aumentam de acordo com as amostras de treinamento $(x_i, y_i)_{i \leq i \leq N}$. No entanto, a convergência uniforme do funcional de risco empírico $R_{emp}(g)$ para o funcional de risco $R(g)$ inclui limitantes na razão de convergência, que são baseados em um importante parâmetro denominado *dimensão VC*.

2.4 Dimensão VC

A Dimensão Vapnik e Chervonenkis (VC), definida por Vapnik e Chervonenkis (1971), é uma medida quantitativa da capacidade de classificação de um conjunto de funções classificadoras (hipóteses), que foram selecionadas por um algoritmo de aprendizado.

Seja um conjunto H de hipóteses $h_i(x) = sgn(g(x))$, sendo $g(x)$ uma função contínua qualquer. Para definir a dimensão VC de um conjunto de hipóteses H, deve-se definir os conceitos de dicotomia e fragmentado, como segue:

- **Dicotomia**.

  Dado um conjunto de hipóteses H, uma dicotomia de um espaço de entradas S é uma das possíveis formas de etiquetar os pontos de S usando uma hipótese em H. Por exemplo, se considerarmos que a hipótese $h(x) = sgn(g(x))$ divide o espaço de entradas em dois subconjuntos disjuntos, a dicotomia obtida através de $h(x)$ produz um subconjunto com padrões x_i para os quais $g(x_i) > 0$ e outro subconjunto de padrões x_i com $g(x_i) < 0$.

- **Fragmentado**.

  Um espaço de entradas S de m pontos ($m \geq 1$) se diz fragmentado por um conjunto de hipóteses H quando as hipóteses contidas em H são capazes de induzir todas as possíveis dicotomias em S. Seja $\Delta_H(S)$ o número de dicotomias que as hipóteses em H induzem sobre S, e seja $|S| = m$ a cardinalidade do conjunto S. Diz-se que S é fragmentado por H se $\Delta_H(S) = 2^{|S|}$.

A dimensão VC de um conjunto H de hipóteses (dicotomias) é então definida como a cardinalidade do maior conjunto de pontos S, que pode ser fragmentado pelas dicotomias obtidas através do conjunto H, ou seja, o maior N tal que $\Delta_H(S) = 2^N$ com $N = |S|$. Caso o valor de N não possa ser estimado, então assume o valor ∞.

Na Figura 2.1 apresentamos um exemplo de três amostras em R^2. Para qualquer padrão de rotulação binária que as amostras venham a assumir, é possível determinar retas capazes de fragmentá-las (separá-los de acordo com suas classes). Porém, o exemplo da Figura 2.2 apresenta quatro amostras em R^2 com um padrão de rotulação particular, nas quais não existe reta alguma que seja capaz de fragmentá-las. Portanto, a dimensão VC de retas no R^2 é 3, dado que 3 é o número máximo de amostras que podem ser corretamente classificadas por uma reta.

Baseada na complexidade das amostras, a dimensão VC proporciona uma limitante superior de um espaço de hipóteses. Se o numero de amostras N é pequeno, então preferiremos um espaço H com dimensão VC baixa. Na continuação, apresentamos a minimização do risco empírico e do risco estrutural, que são baseados na dimensão VC do espaço de hipóteses.

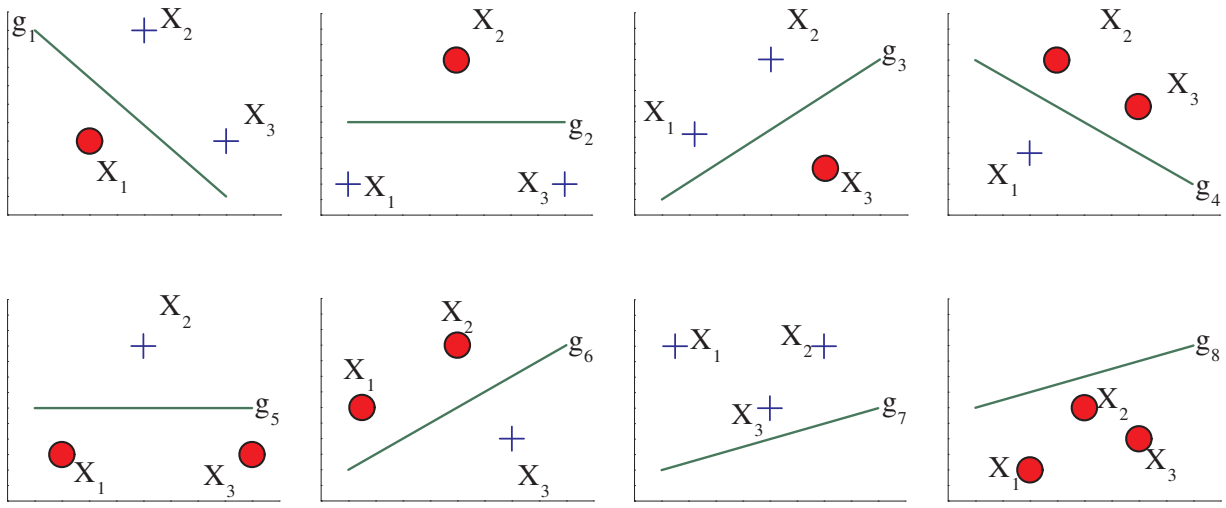


Figura 2.1: Separação de três pontos no espaço bidimensional por meio de retas

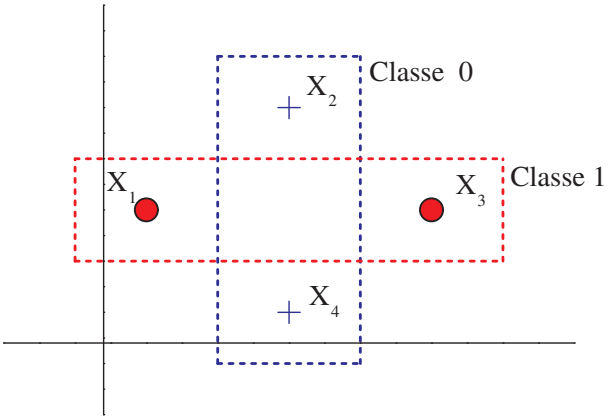


Figura 2.2: Distribuição particular de quatro pontos no espaço bidimensional com duas classes diferentes.

2.4.1 Minimização do Risco Empírico

Nos problemas de classificação binária, a tarefa de aprendizagem a partir de amostras de treinamento é formulada como segue:

Seja uma classe de funções de decisão:

$$\{g_\alpha(x) : \alpha \in A\}, \text{ com } g_\alpha : R^n \to \{-1, +1\} \tag{2.11}$$

onde A é um conjunto de parâmetros que permite uma correspondência entre $g_\alpha(x)$ e o espaço das hipóteses H. No entanto, precisamos também de um conjunto de amostras:

$$(x_1,y_1),\cdots,(x_n,y_n)\text{ , com } x_i \in R^n, y_i \in \{-1,+1\}, i=1,\cdots,n, \tag{2.12}$$

que são obtidas de uma distribuição desconhecida $P(x,y)$. Procura-se encontrar uma função g_α que forneça o menor valor possível para o funcional de risco:

$$R(\alpha) = \int |g_\alpha(x) - y| P(x,y) dxdy \tag{2.13}$$

O risco esperado mede a eficiência com que uma hipótese candidata prediz o valor correto de y para uma amostra x. Porém, como a distribuição P(x,y) é desconhecida, não é possível obter diretamente o valor do funcional de risco $R(\alpha)$, razão pela qual é usado o funcional de risco empírico:

$$R_{emp}(\alpha) = \frac{1}{n}\sum_{i=1}^{n} |g(x_i) - y_i| \tag{2.14}$$

De acordo com a lei dos grandes números (James, 1981), a frequência empírica da ocorrência de um evento converge na forma AS (*Almost Surely*) para a verdadeira probabilidade do evento quando o número de amostras cresce para infinito. Assim, esta lei de convergência garante que $R_{emp}(\alpha)$ converge em probabilidade para $R(\alpha)$. Em outras palavras, a minimização do Risco Empírico $R_{emp}(\alpha)$ deve proporcionar o mínimo de $R(\alpha)$. Isto valida-se quando substitui-se a teoria de convergência em probabilidade de $R_{emp}(\alpha)$ para $R(\alpha)$ pela teoria de convergência uniforme (Vapnik e Chervonenkis , 1971; Vapnik , 1991, 1995). Baseado na dimensão VC, a teoria da convergência uniforme proporciona uma limitante que expressa o risco esperado $R(\alpha)$ em função do risco empírico $R_{emp}(\alpha)$ e a um *termo de capacidade* (segundo termo na Equação 2.15). Esse limitante, apresentado na Equação 2.15, é garantido com probabilidade $1-\eta \in [0,1]$,

$$R(\alpha) \leqslant R_{emp}(\alpha) + \sqrt{\frac{d(\ln(2n/d)+1) - \ln(\eta/4)}{n}}, \tag{2.15}$$

onde d denota a dimensão Vapnik-Chervonenkis (VC) de $\{g_\alpha(x) : \alpha \in A\}$, n representa o número de amostras no conjunto de treinamento e a parcela de raiz na soma é referenciada como *termo de capacidade*.

Este limitante mostra que a minimização do risco esperado é diretamente proporcional à mi-

nimização de dois fatores: por um lado o risco empírico e, por outro, a razão entre a dimensão VC (d) e o número de amostras de treinamento (n). Então, para um número dado de amostras, a escolha para um valor apropriado para a dimensão VC será de suma importância para se obter um desempenho ótimo, sobretudo quando temos um número de amostras pequeno.

Em termos práticos, o algoritmo de aprendizado deve escolher um classificador g que pertença a um conjunto de funções G e possua baixa dimensão VC. Este classificador deve minimizar o risco empírico e este por sua vez deve minimizar o risco esperado. A fim de atingir esses objetivos, definiu-se um princípio de indução denominado minimização do risco estrutural (Vapnik , 1998).

2.4.2 Principio da Minimização do Risco Estrutural

Vapnik (1982) desenvolveu a técnica de minimização do risco estrutural (*Structural Risk Minimization (SRM)*), que procura resolver o problema da escolha de uma dimensão VC apropriada. Vapnik *et al.* (1996) demonstraram que o princípio da minimização do risco estrutural é superior em precisão ao princípio de minimização do risco empírico.

A partir do limitante apresentado em 2.15, pode-se inferir que valores pequenos para o risco empírico $R_{emp}(\alpha)$ não necessariamente implicam valores pequenos para o risco esperado $R(\alpha)$. O princípio de minimização do risco estrutural é baseado na observação de que ambos riscos deveriam ser minimizados, o que implica em minimizar de forma simultânea a dimensão VC e o risco empírico.

No entanto, se o *termo de capacidade* diz respeito à classe de funções G e o risco empírico refere-se a um classificador particular g; então, para minimizar ambas as parcelas divide-se inicialmente G em subconjuntos de funções com dimensão VC crescente (Vapnik , 1998). Esse procedimento é conhecido como introdução de estruturas em G, considerando-se os subconjuntos definidos também como estruturas (Scholkopf e Smola , 2001). Minimiza-se então a limitante sobre as estruturas introduzidas.

A fim de implementar o princípio de minimização do risco estrutural, consideram-se subconjuntos G_i com $G_0 \subset G_1 \subset \ldots \subset G_q \subset G$, como apresentado na Figura 2.3.

Como cada G_i fica maior com o crescimento do índice i, a capacidade do conjunto de funções que ele representa também é maior à medida que i cresce, ou seja, $d_0 < d_1 < \cdots < d_q < d$. Seja g_k uma função classificadora com menor risco empírico dentro de um subconjunto G_k. Conforme

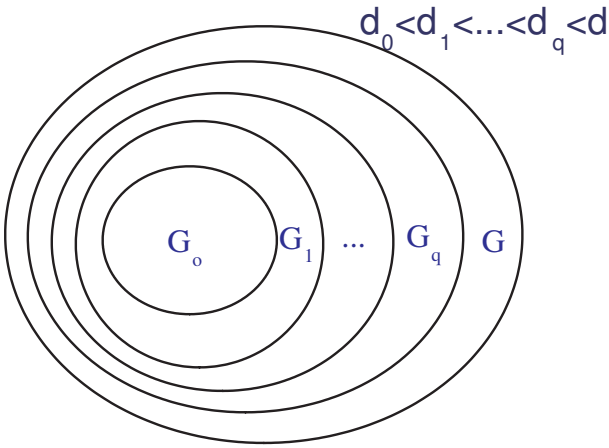


Figura 2.3: Introdução de sub-estruturas em numa classe de funções G

aumenta o valor de k, o risco empírico de g_k diminui, uma vez que a complexidade do conjunto de classificadores é maior. Porém, o termo de capacidade também aumenta com k. Assim, procura-se um valor ótimo de k que produza uma soma mínima do risco empírico e do termo de capacidade, a qual minimize a limitante sobre o risco esperado. A escolha da função classificadora g_k constitui o princípio da minimização do risco estrutural, como se ilustra na Figura 2.4.

Embora muitas extensões sejam admitidas, aqui consideramos que o processo fundamental é determinístico e que existem múltiplas entradas a partir das quais é desejado predizer uma única saída. O princípio de minimização do risco estrutural possui uma fundamentação matemática consistente, porém pode ser difícil de ser implementado pelas seguintes razões:

- Pode ser difícil calcular a dimensão VC de G_k, além do fato de que há somente um número pequeno de classes de funções para as quais é sabido como calcular a dimensão VC.
- Aceitando a viabilidade de obtenção da dimensão VC de G_k, o problema de minimização pode não ser trivial.

Em consequência, modelos de aprendizado de máquina baseados no princípio SRM tendem a apresentar uma maior habilidade para generalizar frente a dados não observados, que é um dos principais propósitos do aprendizado estatístico.

Para funções de decisão lineares do tipo $g(x) = \mathbf{w} \cdot \mathbf{x}$, entretanto, existem resultados alternativos que relacionam o risco esperado ao conceito de margem (Smola e Bartlett , 2000).

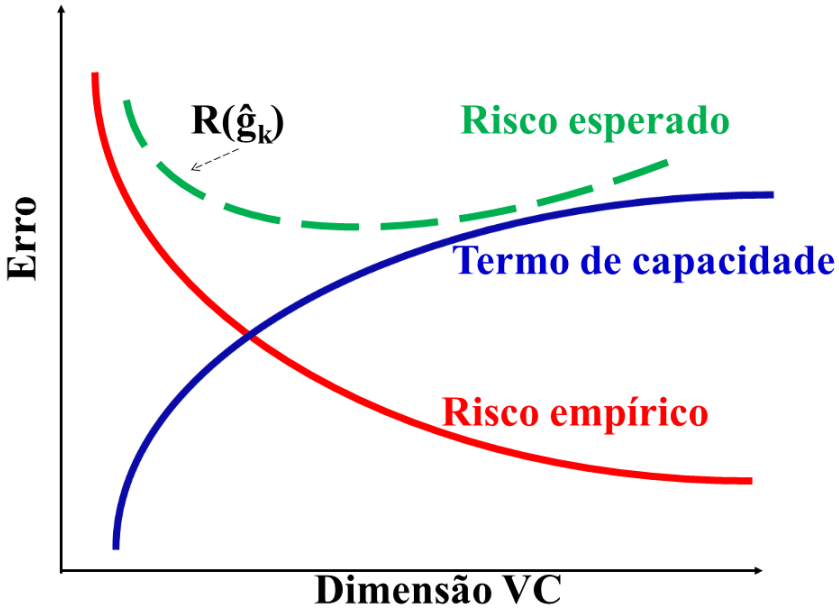


Figura 2.4: Princípio de minimização do risco estrutural

2.4.3 Minimização do funcional de risco baseado na margem de separação

A minimização do risco usando a margem de separação é uma abordagem que difere em respeito aos princípios apresentados anteriormente. Esta abordagem relaciona a dimensão VC de uma função classificadora linear à margem obtida pela mesma na separação de dados.

A *margem* de uma amostra x_i está relacionada com sua distância em relação à fronteira de separação e mede a confiança de previsão de uma função classificadora g. Nos problemas de classificação binária, onde $y_i \in \{-1,+1\}$, a margem $\rho(g(x_i),y_i)$ com que a amostra x_i pode ser classificada por g é calculada usando a Equação 2.8. Logo, um valor negativo de $\rho(x_i,y_i)$ denota uma classificação incorreta.

A *margem geométrica* de uma amostra x_i em relação a uma função linear $g(x) = w \cdot x + b$ que mede efetivamente a distância de x_i frente à fronteira de decisão, é obtida através da divisão pela norma $||w||$. Em amostras classificadas de forma incorreta, esta margem equivale à distância com sinal negativo.

O *erro (risco) marginal* de uma função $g(R_\rho(g))$ (Smola e Bartlett , 2000), sobre um conjunto de treinamento, proporciona o número de amostras com margem inferior a uma determinada constante $\rho > 0$, como apresentado na Equação 2.16:

$$R_\rho(g) = \frac{1}{n}\sum_{i=1}^{n} I(y_i g(x_i) < \rho), \quad I(\cdot) = \begin{cases} 1, & y_i g(x_i) < \rho \\ 0, & \text{caso contrário} \end{cases} \tag{2.16}$$

Como visto anteriormente, pode-se definir uma limitante para o risco funcional baseado na soma dos erros marginais e um termo de capacidade. Considere-se uma constante **c** com probabilidade $1-\eta \in [0,1]$, para todo $\rho > 0$, uma classe G de funções lineares $g(x) = \mathbf{w}\cdot\mathbf{x}$ com $||x|| \leq R$ e $||w|| \leq 1$, então temos a seguinte limitante (Smola e Bartlett , 2000):

$$R(g) \leq R_\rho(g) + \sqrt{\frac{c}{n}\left(\frac{R^2}{\rho^2}\log^2\left(\frac{n}{\rho}\right) + log\left(\frac{1}{\eta}\right)\right)} \tag{2.17}$$

Uma margem maior implica em um menor termo de capacidade. Este procedimento pode levar a um aumento na taxa de erro marginal. Um baixo valor de ρ leva a um erro marginal menor, porém aumenta o termo de capacidade

Em consequência, o procedimento para escolher uma função classificadora deve considerar duas propriedades. Primeiro, a função classificadora deve possuir um hiperplano com margem alta para minimizar o erro sobre as amostras de teste. Segundo, o procedimento deve escolher a função classificadora que cometa poucos erros marginais, minimizando o erro sobre as amostras de treinamento.

2.4.4 Erro de generalização

O erro de generalização é definido como a frequência do erro obtido para um conjunto de hipóteses específico quando testado num conjunto de amostras não apresentado anteriormente (amostras de teste).

Logo, baseados na razão de convergência uniforme desenvolvida por Vapnik (1991, 1998), a qual afirma que, para um número de amostras de treinamento $n > d$ (onde d é a dimensão VC), com probabilidade $1-\eta$, para todas as funções classificadoras $g(x,y)$ (hipóteses), o erro de generalização deve ser menor do que o risco esperado ($\varepsilon_{gen}(\alpha) < R(\alpha)$).

Haykin (1998) afirma que a teoria do aprendizado estatístico procura resolver o problema de controlar a habilidade de generalização de um algoritmo de aprendizado em termos matemáticos. Quando essa habilidade de generalização é avaliada, pode-se ter dois fenômenos:

- **Sub-ajuste**. Um algoritmo de aprendizado será considerado como sub-ajustado (*underfitting*) quando a sua dimensão VC ou capacidade seja muito pequena para as n amostras de treinamento e o ponto mínimo do risco garantido ainda não seja atingido.

- **Sobre-ajuste** Um algoritmo de aprendizado será considerado como sobre-ajustado (*overfitting*), quando o ponto mínimo de erro seja ultrapassado e a complexidade do algoritmo seja maior do que o número de amostras disponíveis para o treinamento.

Finalmente, o objetivo de um algoritmo de aprendizado é obter o melhor desempenho de generalização equilibrando a dimensão VC do algoritmo com o número das amostras de treinamento. Nesse sentido, o princípio *structural risk minimization* (SRM) envolve a minimização de um limite superior sobre o erro de generalização, enquanto o princípio *empirical risk minimization* (ERM) envolve a minimização do erro sobre os dados de treinamento.

Diversas técnicas foram propostas utilizando os limites providos pela Teoria de Aprendizado Estatístico na obtenção de classificadores com baixo risco funcional. Entre elas incluem-se generalizações de algoritmos de Aprendizado de Máquina amplamente difundidos, como o Perceptron de Rosenblatt (Rosenblatt , 1962), e modelos de AM formulados diretamente a partir dos limites da Teoria VC, como as Máquina de Suporte Vetorial (Cortes e Vapnik , 1995) e *Boosting*, sendo que esta última utiliza implicitamente os princípios de maximização de margem apresentados anteriormente.

3 MÉTODOS DE ENSEMBLE

Quando projetamos e analisamos algoritmos que aprendem a partir de amostras, é necessário considerar alguns desafios importantes, tais como: encontrar o número de amostras necessárias para aprender com sucesso, procurar um modelo geral de aprendizado, fazer com que o algoritmo aprenda eficientemente, descobrir o que pode ser aprendido eficientemente. Esses assuntos são tratados através do modelo teórico chamado de aprendizado Provavelmente Aproximadamente Correto (*Probably Approximately Correct* - PAC) (Valiant , 1984). Este modelo permite explicar o conceito de aprendibilidade em termos do número de amostras necessárias para obter uma solução aproximada, da complexidade das amostras e da complexidade em tempo e espaço do algoritmo de aprendizado.

Os métodos de *ensemble* são técnicas gerais no aprendizado de máquina. Esses métodos são compostos de vários classificadores ou hipóteses, os quais, quando combinados, produzem uma maior precisão. Neste capítulo apresenta-se uma importante família de métodos de ensemble conhecidos como *Bagging* e *Boosting*, que têm suas raízes no modelo teórico PAC. Estuda-se com maior detalhe o algoritmo *AdaBoost (Adaptive Boosting)*, que tem mostrado um desempenho relevante na resolução de vários problemas práticos e possui uma base teórica robusta.

3.1 Algoritmo Provavelmente Aproximadamente Correto (PAC)

No algoritmo PAC, um *conceito* é uma função computada eficientemente dentro de um domínio. Os *elementos* do domínio podem ser considerados como objetos e, por sua vez, os *conceitos* podem ser considerados como uma classificação desses objetos. Uma classe de conceitos é uma coleção de conceitos. O algoritmo PAC faz uso de um conjunto de treinamento, que consiste de pares $(elemento, valor)$, onde cada *elemento* pertence ao domínio e o *valor* é um conceito avali-

ado naquele elemento. Quando um determinado conceito produz uma hipótese que proporciona valores corretos na maior parte do domínio e é consistente com o conjunto de treinamento, então considera-se que o algoritmo pode aprender esse *conceito*. A diferença entre o valor de uma determinada hipótese e o valor verdadeiro do elemento sendo estimado é considerada como *bias* (também chamada de heurística). Todo algoritmo de aprendizado exibe algum tipo de bias, dado que sempre existe um número muito grande de hipóteses consistentes com o conjunto de treinamento. O principio de *Ockham*, que sempre opta pela hipótese mais simples, é um típico exemplo de bias. Assim sendo, o desafio consiste em descobrir a hipótese mais próxima ao *valor*, principalmente quando este *valor* é desconhecido.

Valiant (1984) introduziu o princípio de aprendizado provavelmente aproximadamente correto (PAC). Este princípio afirma que qualquer hipótese que esteja errada será, com alta probabilidade, identificada depois de se avaliar um determinado número de amostras de treinamento. Desta forma, qualquer hipótese que seja suficientemente consistente com um conjunto de treinamento é improvável de estar errada, ou seja, deve ser Provavelmente Aproximadamente Correta. De maneira formal, vamos denominar $\mathcal{X}$ ao conjunto de todas as amostras ou instâncias, que também é chamado de espaço de entradas. O conjunto de todos os possíveis valores de rótulos é denominado $\mathcal{Y}$, que possui dois elementos ($\mathcal{Y} = \{0,1\}$). Um conceito $c : \mathcal{X} \to \mathcal{Y}$ é um mapeamento de $\mathcal{X}$ em $\mathcal{Y}$, onde o conjunto de conceitos que desejamos aprender é considerado uma classe e é denotado por $\mathcal{C}$. A ideia de *aprendizado* é equivalente ao mapeamento de $\mathcal{X}$ em $\{0,1\}$.

Consideremos um conjunto de possíveis conceitos H, chamado de conjunto de hipóteses. O aprendizado PAC recebe duas entradas. A primeira é o conjunto de amostras $S = \{x_1, \cdots, x_n\}$, que obedece a uma distribuição desconhecida $\mathcal{D}$. A segunda entrada é o conjunto de rótulos $(c(x_1), \cdots, c(x_n))$, baseado em determinado conceito $c \in \mathcal{C}$. A tarefa do algoritmo é usar as amostras rótuladas S para selecionar uma hipótese $h \in H$ que possua um erro de generalização $R(h)$ pequeno em relação ao conceito c. Uma hipótese h é chamada de aproximadamente correta se:

$$R(h) \leq \varepsilon, \tag{3.1}$$

onde ε é uma constante pequena entre 0 e 1.

Como a probabilidade de que um conceito aprendido tenha um erro limitado por ε é desconhecida geralmente, então, estabelece-se um limitante pequeno δ na probabilidade de que esse erro seja maior que ε. Pode-se dizer que:

$$p(R(h) > \varepsilon) < \delta \tag{3.2}$$

Um conceito aprendido será "bom" quando a probabilidade de que seu erro seja maior que a *precisão* ε for menor do que um *grau de confiança* δ. Assim, um algoritmo de aprendizado induz uma hipótese h, que com probabilidade menor do que $1-\delta$ consiga atingir um erro $R(h) \leq \varepsilon$. Esta hipótese será uma boa aproximação (dentro da confiança δ) do conceito real com erro máximo de ε. O aprendizado PAC indica, com alta probabilidade, que todas as hipóteses consistentes serão aproximadamente corretas depois de examinar n amostras. Na seguinte definição, formalizamos os *conceitos* anteriormente explicados (Kearns e Vazirani , 1994).

Definição 1 *Uma classe de conceitos C é **pac-aprendível** se cumpre que todo $c \in C$, o sub-conjunto $\mathcal{D} \in X$, $\delta > 0$, $\varepsilon > 0$, e existe um algoritmo $\mathcal{A}$ com acesso aos valores de ε, δ e o par $(x,c(x))$, que produz, com probabilidade pelo menos de $1-\delta$, uma hipótese $h \in H$ com erro $R(h) \leq \varepsilon$.*

Note que o algoritmo $\mathcal{A}$ pode falhar em obter hipóteses suficientemente precisas. Essa definição apenas garante que a probabilidade será menor que um valor δ suficientemente pequeno. Para garantir que o algoritmo $\mathcal{A}$ termine dentro de um espaço polinomial a definição de "Eficientemente PAC-aprendível" é adotada:

Definição 2 *Uma classe de conceitos C é **eficientemente PAC-aprendível** quando C é PAC-aprendível e existe um algoritmo $\mathcal{A}$, que processa em tempo polinomial em $(1/\varepsilon, 1/\delta, n, tamanho(c))$.*

É importante salientar os seguintes aspectos (Mohri *et al.* , 2012).

- O algoritmo $\mathcal{A}$ será polinomial em $\ln(1/\delta)$ e não apenas em $1/\delta$.
- O aprendizado PAC é um modelo independente da distribuição, no qual se desconsideram suposições particulares sobre a distribuição $\mathcal{D}$.
- As amostras de treinamento e de teste, usadas para definir o erro, são obtidas de acordo com a mesma distribuição $\mathcal{D}$.
- O aprendizado PAC trata a questão da apreensibilidade de uma classe de conceitos C e não de um conceito em particular.

- Em muitos casos, quando a representação computacional dos conceitos é simples ou linear, pode-se omitir a dependência polinomial de n e $tamanho(c)$, focando unicamente na complexidade das amostras.

O aprendizado PAC é uma abordagem que considera o custo da representação computacional e a complexidade do algoritmo de aprendizado. Se omitimos os aspectos computacionais, esta é similar à abordagem apresentada anteriormente por Vapnik e Chervonenkis (Vapnik , 2000). Nas seções posteriores apresentaremos aplicações do aprendizado PAC.

3.2 Combinação de classificadores

Um ensemble consiste de um conjunto de classificadores treinados individualmente, cujas predições são combinadas quando apresentadas a uma nova instância. Um ensemble é considerado eficiente quando cada classificador possui precisão necessária e os eventos de erros são amplamente espalhados no espaço de entrada. Existe uma busca constante pelo aperfeiçoamento dos algoritmos de aprendizado de máquina, tanto para aumentar a acurácia e minimizar os erros, como para otimizar o tempo de aprendizado. Nesse sentido, são propostos diferentes métodos que combinam classificadores trabalhando em conjunto, para que os ensembles obtenham resultados mais precisos se comparados com os obtidos por um único classificador (Schapire *et al.* , 1997).

De acordo com o procedimento de manipulação do conjunto de treinamento, os métodos de criação de ensembles podem ser divididos em duas classes:

- ***Perturb and Combine*:** garante a criação de diferentes classificadores através da alteração do conjunto de treinamento ou do método de construção de classificadores.
- ***Adaptively Resample and Combine*:** envolve um grupo de técnicas que combinam as amostras de treinamento e re-amostragem do conjunto de treinamento adaptativamente.

Dados um conjunto de classificadores, a combinação das predições destes classificadores pode ser feita em várias formas. Entre elas, destacamos a técnica do voto majoritário e a decisão por ponderação das saídas.

- **Voto majoritário:** considera que todos os classificadores são igualmente importantes, sendo que a resposta final é dada de acordo com a maior frequência.

- **Ponderação das saídas:** considera que existem classificadores mais influentes na decisão final. De acordo com o nível de precisão de cada classificador individual, cada predição recebe um maior peso.

3.2.1 Classificadores base

Os classificadores base são funções que, avaliando uma instância ou amostra, retornam um rótulo. Estes classificadores não necessariamente devem apresentar um desempenho ótimo. A seguir apresentam-se alguns dos classificadores base mais usados em aprendizado de máquina.

- **Redes Neurais:** As redes neurais artificiais (ANN) são modelos computacionais baseados no funcionamento do cérebro humano. De forma similar ao aprendizado humano, as redes neurais adquirem o conhecimento a partir do seu ambiente e o armazenam usando as conexões entre neurônios (pesos sinápticos) (Haykin , 1998). Os benefícios de usar redes neurais estão relacionados com uma adequada adaptabilidade às mudanças do ambiente, à robustez frente aos sinais de entrada não lineares e à possibilidade de extrair informações sobre a confiança das decisões.

  Uma rede neural é formada por uma interconexão de elementos computacionais simples chamados de neurônios. Cada neurônio possui quatro componentes principais: os sinais de entrada x_i, os pesos sinápticos w_i, uma junção aditiva e uma função de ativação não linear f, como se ilustra na Figura 3.1.

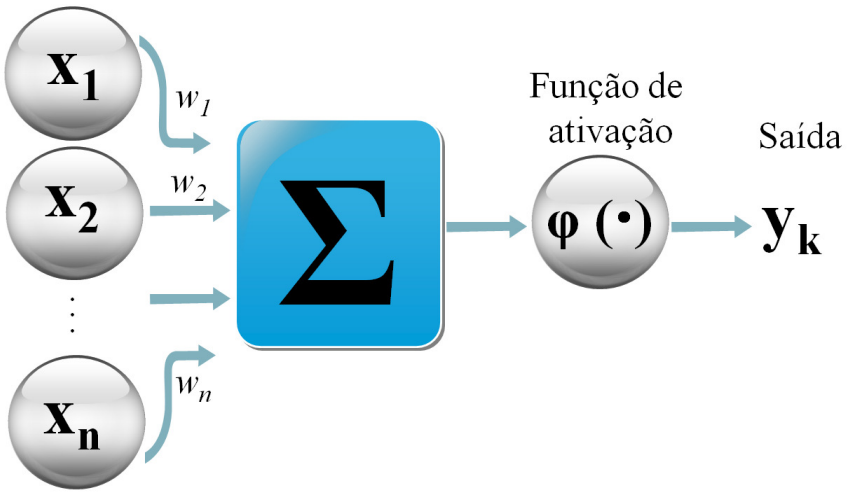


Figura 3.1: Esquema de um neurônio

A modelagem de um neurônio é expressado na Equação 3.3:

$$y = f(\sum_{i=1}^{n} x_i w_i) \tag{3.3}$$

- **Árvores de decisão (Decision Trees):** As árvores de decisão são modelos recursivos que geram estruturas em forma de árvores simplificadas. Este modelo divide um problema complexo em sub-problemas, cada um menos complexo de resolver. Cada sub-problema é novamente dividido em problemas ou tarefas ainda menos complexos, até se atingir problemas suficientemente simples de resolver. O modelo é montado usando as informações obtidas do conjunto de amostras de treinamento, porém a classificação é feita de forma eficiente evitando avaliar todos os elementos da árvore. A Figura 3.2 apresenta um esquema de uma árvore de decisão, os elementos nessa figura são o nó raiz (entrada), os nós de decisão que geram ramificações e os nós folha que apresentam a classificação final do algoritmo.

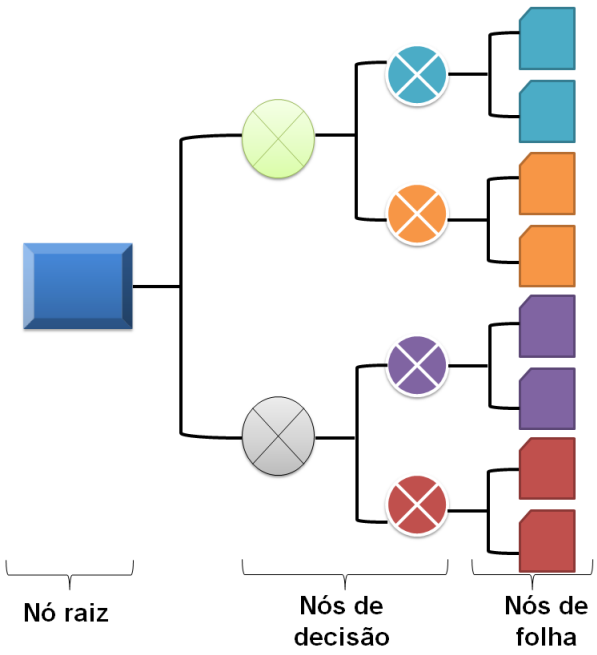


Figura 3.2: Esquema de uma árvore de decisão.

Cada nó avalia um atributo e cada percurso na árvore (da raiz até a folha) é uma regra de classificação. Uma amostra é classificada percorrendo toda a estrutura da árvore, considerando os valores e atributos da amostra analisada.

- ***Decision Stump*:** O método de *Decision Stump* consiste em uma árvore de decisão com apenas duas folhas (Figura 3.3). Cada classificador base (fraco) analisa apenas um determi-

nado problema. *Decision Stump* é muito utilizado em estudos dos métodos de *Boosting* por simplicidade. A análise simplificada que o algoritmo realiza pode não ser tão eficiente com problemas complexos, porém permite maior rapidez na classificação.

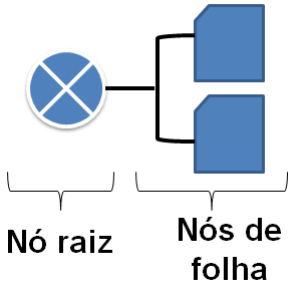


Figura 3.3: Esquema do *Decision Stump*.

A seguir, apresentam-se os métodos de combinação de classificadores denominados *Bagging* e *Boosting*.

3.3 *Bagging*

O método de agregação *bootstrap* ou *Bootstrap aggregating (Bagging)* foi introduzido por Breiman (1996). Usando um conjunto de treinamento com n amostras, o método *Bagging* procura gerar conjuntos independentes chamados de *bootstraps*. Cada *bootstrap* é composto de n amostras aleatoriamente escolhidas com *reposição*. A *reposição* significa que um amostra pode-se repetir no mesmo *bootstrap*. Então, *Bagging* cria k *bootstraps* e treina k classificadores para cada *bootstrap*, como se ilustra na Figura 3.4.

As saídas dos k classificadores são combinadas por meio do voto majoritário usando pesos iguais (Equação 3.4).

$$C = \frac{1}{k}\sum_{i=1}^{k} C_i \tag{3.4}$$

O método foi aplicado com relativo sucesso com *classificadores base* instáveis como redes neurais e árvores de decisão (Freund *et al.* , 2003). Um classificador é considerado instável quando pequenas mudanças no conjunto de amostras causam grandes impactos na decisão final. Além

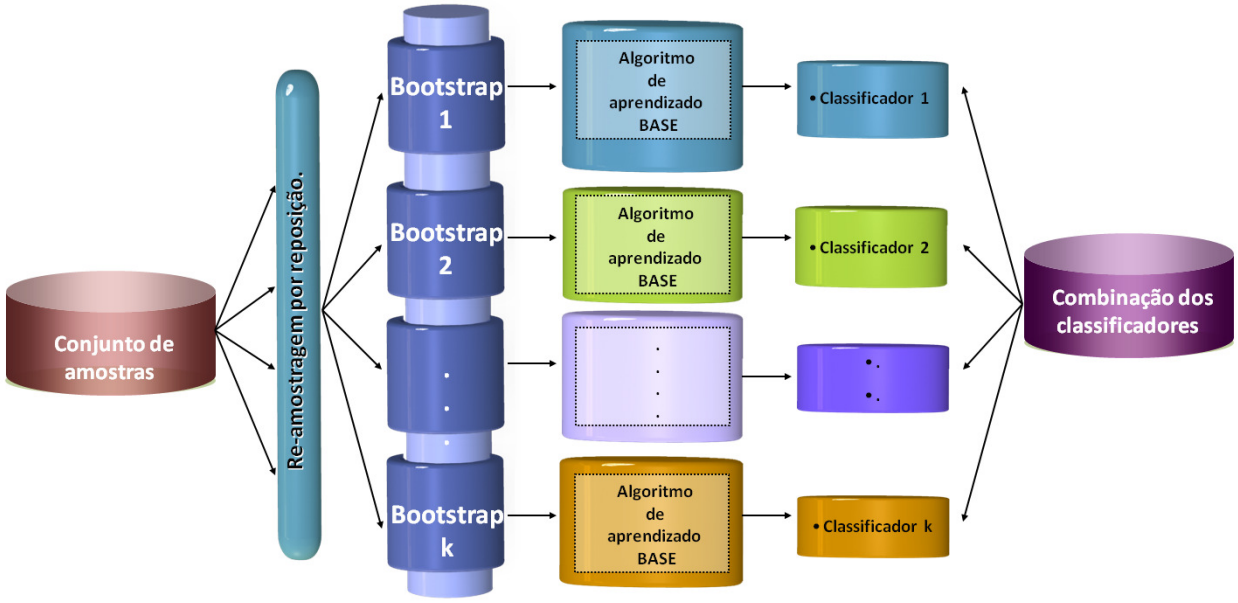


Figura 3.4: Ilustração do método *Bagging*

da re-amostragem com reposição, existem outros mecanismos de amostragem, como a adição de perturbação com ruído nos *bootstraps* (Breiman , 2000; Dietterich , 2000).

3.3.1 Variações do bagging

- **Random Forest:** O método *Random Forest* (Breiman , 2001) tem sua origem no *random decision forests* proposto por Ho (1998) e no *Bagging* (Breiman , 1996). *Random forests* são combinações de métodos de árvores de decisão, cujos parâmetros de treinamento variam de forma aleatória. Normalmente, estes parâmetros são obtidos a partir das amostras de treinamento, similarmente ao que geram com o Bagging; porém, podem apresentar características diferentes, como no método *random decision forests*.

- **Rotation Forest:** *Rotation Forest* (Rodriguez *et al.* , 2006) é outro método de geração de classificadores para ensemble. Nesse método, o conjunto de treinamento é separado de forma aleatória em k subconjuntos, e cada subconjunto de treinamento para um classificador base é formado aplicando o método *análise de componentes principais* (do inglês, *Principal component analysis (PCA)*. Todos os componentes principais são mantidos a fim de conservar a variabilidade nos dados. Então, utilizam-se k eixos para formar os novos atributos para um

classificador base. A principal ideia do método *Rotation Forest* é, de forma simultânea, produzir precisão e diversidade dentro do ensemble. A diversidade é obtida através da extração de características para cada componente principal, escolhendo árvores de decisão por serem sensíveis à rotação dos eixos de características, daí o nome *Forest*. A precisão se obtém usando o conjunto total de dados para cada componente e considerando todas as direções principais.

- **Pasting Small Votes:** O método *pasting small votes* (Breiman , 1999b) foi projetado para treinar classificadores a partir de conjuntos de dados grandes. Um conjunto extremamente grande é dividido em pequenos subconjuntos, chamados de bites. Cada bite é usado para treinar um determinado classificador. O método *pasting small votes* apresenta duas variações. Na primeira, chamada de *Rvotes*, são criados pequenos subconjuntos randômicos para treinar cada classificador. Na segunda, chamada de *Ivotes*, são construídos pequenos subconjuntos de forma consecutiva, baseados na qualidade dos classificadores treinados previamente. Em outras palavras, as amostras classificadas de forma incorreta possuem uma maior probabilidade de serem selecionadas durante cada iteração.

3.4 *Boosting*

O trabalho de Schapire (1990) fez com que os sistemas de ensemble se tornassem centrais em aprendizado de máquina. Nesse trabalho, ele provou que um classificador forte, no sentido Provavelmente Aproximadamente Correto (PAC), pode ser construído pela combinação de classificadores fracos através de um procedimento conhecido como *Boosting*.

O método de *Boosting* procura melhorar o desempenho de algoritmos de aprendizagem através da combinação de classificadores fracos produzidos por outras técnicas, tais como Árvores de Decisão ou *Decision Stumps* (Schapire e Singer , 1999). A escolha de um classificador fraco se justifica pelo seu desempenho levemente superior dentro do conjunto. Este classificador levemente superior, é impulsionado (*boosted*) para formar um classificador forte (Bishop , 2006).

Formalmente, um algoritmo de *Boosting* $\mathcal{B}$ utiliza um algoritmo de aprendizado fraco $\mathcal{A}$ para acessar um conjunto de hipóteses fracas $\mathcal{C}$ *eficientemente PAC-aprendíveis*. O algoritmo $\mathcal{A}$ garante a obtenção de um classificador fraco h com erro $err(h) \leq 1-\gamma$ e probabilidade de pelo menos $1-\delta$. O algoritmo $\mathcal{B}$, como um típico algoritmo de PAC, possui $\varepsilon > 0$, $\delta > 0$ e n amostras rotuladas

$(x_1, c(x_1)), \cdots, (x_n, c(x_n))$ para algum $c \in \mathcal{C}$. Usando seu acesso ao algoritmo $\mathcal{A}$, o algoritmo de *Boosting* pode produzir sua própria hipótese (forte) H de tal forma que:

$$Pr[err(H) > \varepsilon] \leq \delta \tag{3.5}$$

De forma parecida ao algoritmo de *Bagging*, o algoritmo de *Boosting* cria vários conjuntos de treinamento por re-amostragem dos dados. Porém, no algoritmo de *Boosting*, a re-amostragem fornece amostras de treinamento mais informativas para cada classificador fraco obtido de forma consecutiva. A Figura 3.5 ilustra o método de *Boosting*.

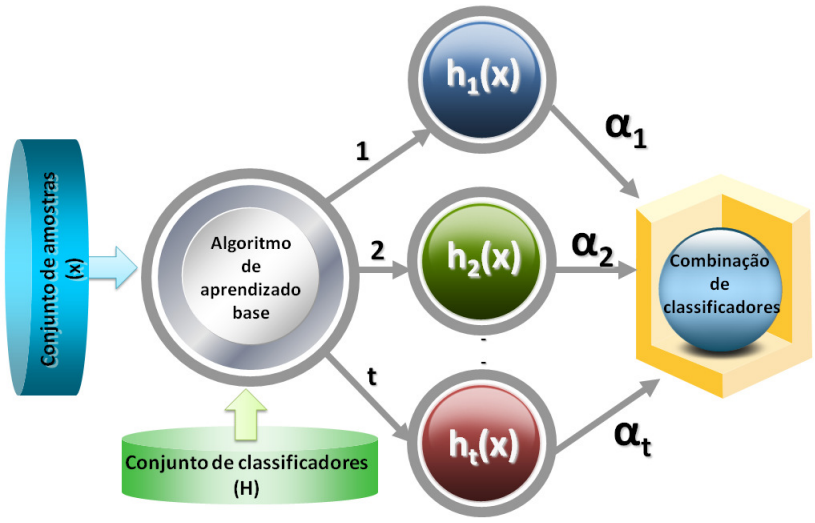


Figura 3.5: Geração de classificadores distintos via *Boosting*.

3.5 O algoritmo *AdaBoost*

O algoritmo de *Boosting* adaptativo (*AdaBoost*) foi introduzido por Freund e Schapire (1995), procurando resolver diversas dificuldades encontradas anteriormente no *Boosting*. *AdaBoost* apresenta várias propriedades que facilitam seu uso e implementação, e podemos mencionar duas. A primeira está relacionada com os parâmetros utilizados para analisar dados de grandes dimensões, e as margens entre classes podem se apresentar mais precisas do que outros métodos. A segunda propriedade é o baixo custo computacional, dado que corresponde a um programa de complexidade linear e evita o uso de componentes computacionais pesados (Freund e Schapire , 1999).

O algoritmo de aprendizado *AdaBoost* toma como entrada uma sequência de amostras de treinamento $(x_1, y_1), \cdots, (x_n, y_n)$ que são usadas para formular hipóteses de classificação e atualizar um

conjunto de pesos W_t. Dado que o foco deste trabalho está relacionado com a classificação binária, consideramos apenas dois possíveis rótulos, de tal forma que cada $y_i \in \{-1,+1\}$.

A forma como uma hipótese h ajusta as amostras de treinamento é determinada através de seu erro de treinamento (*erro empírico*). Este erro é uma fração das n amostras que foram classificadas incorretamente, como se mostra na Equação 3.6:

$$\hat{\varepsilon}(h) = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}\{h(x_i) \neq y_i\}, \tag{3.6}$$

sendo $\mathbf{1}\{\cdot\}$ uma função que retorna *um* se o argumento é verdadeiro e *zero* caso contrário.

Usualmente se assume que as amostras de treinamento e de teste são geradas a partir de uma mesma distribuição $\mathcal{D}$ em pares (x,y). A respeito dessa distribuição, o erro de teste esperado de uma hipótese h é chamado de *erro de generalização*, que é igual à probabilidade de classificar de forma incorreta uma única amostra (x,y), como ilustra a Equação 3.7. Para qualquer hipótese h, a medida monitorada será uma soma de eventos de erro.

$$\varepsilon(h) = \mathbf{Pr}_{(x,y)\sim W_t}[h(x) \neq y] \tag{3.7}$$

No Algoritmo 3.1, é apresentado o procedimento para o treinamento do algoritmo *AdaBoost*, no qual:

- W_t: Distribuição de pesos atribuídas às amostras de treinamento.
- t: O período de aprendizagem para cada classificador fraco "t".
- T: Número total de ciclos de aprendizagem (número de classificadores fracos).
- h_t: Hipóteses geradas a cada unidade de iteração (t).
- ε_t: Erro cometido durante a iteração.
- Z_t: Termo normalizador.
- α_t: Importância associada do classificador, calculada a partir do erro ε_t.

Procuram-se hipóteses fracas h_t que minimizem os erros ε_t ou, pelo menos, que cada hipótese obtenha um erro menor que $\frac{1}{2}$. O Teorema 1 apresenta uma limitante destes erros.

Algoritmo 3.1 *Pseudo-código do AdaBoost*

- Dado o conjunto de dados $S = \{(x_1, y_1), ..., (x_N, y_N)\}$, no qual $y_i \in \{-1, +1\}$
- Inicializar com a distribuição de pesos: $W_t = \frac{1}{N}$
- Para $t = 1, ..., T$

  1. Chamar ao conjunto de classificadores fracos e escolher o melhor classificador fraco $h_t : \chi \to [-1, +1]$, ou seja, com menor erro ε_t:

  $$\varepsilon_t = Pr_{i \sim W_t}[h_t(x_i) \neq y_i] \tag{3.8}$$

  2. Obter :

  $$\alpha_t = \frac{1}{2} \ln \left(\frac{1 - \varepsilon_t}{\varepsilon_t} \right) \tag{3.9}$$

  3. Atualizar a distribuição de pesos.

  $$W_{t+1}(i) = \frac{W_t(i) exp(-y_i \alpha_t h_t(x_i))}{Z_t} \tag{3.10}$$

  no qual $Z_t = \sum W_t(i)$ é um fator de normalização assegurando que:

  $$\sum_{i=1}^{N} W_t(i) = 1$$

- Finalmente o classificador forte é introduzido como a seguinte combinação linear:

$$H(x) = sign(\sum_{t=1}^{T} \alpha_t h_t(x)) \tag{3.11}$$

Teorema 1 *(Freund e Schapire , 1995) Considere o passo $\gamma_t = \frac{1}{2} - \varepsilon_t$ e uma distribuição de pesos inicial W_1 sobre o conjunto de treinamento. Então, a soma de erros da combinação de classificadores H está limitada por:*

$$\boldsymbol{Pr}_{i \sim W_1}[H(x_i) \neq y_i] \leqslant \prod_{t=1}^{T} \sqrt{1 - 4\gamma_t^2} \leqslant \exp\left(-2 \sum_{t=1}^{T} \gamma_t^2 \right) \tag{3.12}$$

Sendo T o número de classificadores fracos.

Baseado no algoritmo *PAC*, o algoritmo *AdaBoost* utiliza classificadores fracos que cumprem as seguintes restrições de desigualdades: dado o passo $\gamma > 0$, um classificador fraco com acesso ao conjunto de amostras retorna um erro de classificação de no máximo $\frac{1}{2} - \gamma$, ou seja, o erro máximo

de cada classificador fraco é $\varepsilon \leqslant \frac{1}{2} - \gamma$, para todo t. O tempo de execução deve ser $1/\varepsilon$, para qualquer $\varepsilon \in [0,1]$, que depende unicamente da performance do classificador fraco.

Dado que *AdaBoost* é adaptativo, γ_t não precisa ser conhecido *a priori*. O teorema 1 garante uma queda rápida no erro de treinamento e seu limitante toma a forma da *Limitante de Chernoff* (Freund e Schapire , 1995). Este limitante considera a probabilidade de que pelo menos a metade dos classificadores fracos classifiquem corretamente durante todos os ciclos de aprendizagem. A suposição de que cada classificador fraco possui um limitante de erro menor que $\frac{1}{2}$, em cada rodada, é uma variação do aprendizado fraco empírico, que é fundamental em todos os algoritmos de *Boosting*.

Por outro lado, o processo de minimização do erro permite obter os coeficientes α_t para cada classificador ótimo. O coeficiente α_t mede a importância atribuída a cada classificador h_t, que se incrementa enquanto o erro de treinamento ε_t diminui. O método para se obter a hipótese final $H(x)$ fornecida pelo *AdaBoost* é a combinação ponderada das diversas saídas (hipóteses) em cada iteração. Assim, realiza-se uma votação ponderada das T hipóteses fracas (Equação 3.11).

A distribuição de probabilidades W_t (pesos) é atualizada durante cada iteração (Equação 3.10). O efeito desta atualização é o aumento dos pesos para as amostras classificadas incorretamente. No entanto, os pesos das amostras classificadas corretamente são diminuídos. Assim, se este procedimento de atualização permite trazer várias vantagens para os algoritmos *Boosting*, também pode gerar outras dificuldades, como veremos a seguir.

3.6 Variações do Algoritmo *AdaBoost*

Com o objetivo de melhorar a eficiência do *AdaBoost*, nos últimos anos, diversos pesquisadores introduziram várias modificações. Nesta seção, mencionaremos as alterações mais relevantes, incluindo atualização das amostras, maximização da margem, importância atribuída a cada classificador e escolha dos classificadores fracos.

3.6.1 Os pesos das amostras

Em cada iteração, o procedimento de atualização aumenta os pesos das amostras classificadas incorretamente, fornecendo desta forma a característica adaptativa do *AdaBoost*. A fim de evitar o fenômeno de sobre-ajuste das amostras, diversos autores propuseram estratégias relacionadas à

variação deste procedimento de atualização.

Freund (1999) propôs o *BrownBoost*, o qual resolve um sistema de duas equações mediante métodos numéricos padrão. O parâmetro $\mathscr{C}$ do *BrownBoost* é o período de tempo em que o algoritmo roda, análogo ao número de iterações no *AdaBoost* clássico, sendo que cada classificador possui um índice de tempo diretamente relacionada com a confidência α_t. Um valor grande para a variável $\mathscr{C}$ indica que as amostras são menos ruidosas. Ao contrário, um valor pequeno para o parâmetro $\mathscr{C}$ implica que as amostras são ruidosas e mais exemplos podem ser descartados. De forma similar, o algoritmo *MadaBoost*, introduzido por Domingo e Watanabe (2000), apresentou uma atualização de pesos limitantes moderada. Este limitante é atribuído a cada amostra tendo como base a probabilidade inicial. Desta forma, os pesos das amostras não podem aumentar demais arbitrariamente. Este crescimento incontrolado é uma das deficiências no algoritmo *AdaBoost* inicialmente elaborado.

Por outro lado, o *LogitBoost* (Friedman *et al.* , 2000) coloca menor ênfase nas amostras classificadas incorretamente. Empiricamente, mostrou-se que o *LogitBoost* apresentava um melhor desempenho em amostras ruidosas. Servedio (2003) introduziu o *SmoothBoost*, o qual é um método que procura reduzir o fenômeno de sobre-ajuste de amostras; para isso, utilizam-se limitações na distribuição de pesos. De uma forma, em cada iteração, o método atribui um peso limitado a cada amostra específica. Assim, as amostras ruidosas são salientadas durante as iterações, sendo a elas atribuídas com pesos extremamente grandes.

O algoritmo *Modest AdaBoost* foi introduzido por Vezhnevets e Vezhnevets (2005). Estes propuseram um esquema que decrementa a contribuição dos classificadores fracos, caso possuam um desempenho "muito" bom durante a classificação das amostras, razão pela qual o método é chamado de *modest* (modesto). Por outro lado, o *Emphasis AdaBoost* (Gómez-Verdejo *et al.* , 2006) mostrou que a função de ênfase do *AdaBoost* podia ser vista como um produto de dois fatores, o primeiro dependente do erro quadrático das amostras e enquanto o segundo interpretável como uma função da proximidade entre uma amostra e a fronteira de classificação. Esta proximidade mede a saída obtida pela combinação linear em construção.

O algoritmo *EAdaBoost* utiliza o método dos vizinhos mais próximos *k nearest neighbors - kNN*. Introduzido por Gao e Gao (2010), esta abordagem no início do procedimento executa o *AdaBoost* clássico, a fim de descobrir alguma regularidade estatística nas amostras. Em seguida, o método dos vizinhos mais próximos ponderados no espaço de características é aplicado, o qual

está composto pelos classificadores produzidos no algoritmo *AdaBoost*. Assim, utiliza-se o *AdaBoost* para aumentar a precisão da classificação e evitar o sobre-ajuste das amostras, editando o conjunto de amostras mediante o método *kNN* ponderado, melhorando a qualidade das amostras de treinamento.

3.6.2 Maximização da margem mínima

Vários trabalhos na literatura mostraram que o algoritmo *AdaBoost* possui uma propriedade bastante particular; isto é, no procedimento de treinamento, o erro de generalização decresce continuamente mesmo após o erro de treinamento ser estabilizado ou após ter atingido o valor de zero. Schapire *et al.* (1997) explicaram este fenômeno mediante a introdução do conceito de *margem* de classificação, que os algoritmos de *Boosting* tendem a maximizar. Baseados nesta obervação, numerosos trabalhos foram apresentados procurando controlar o nível da margem de classificação, alguns tentando maximizá-la enquanto outros procuraram minimizá-la.

Rätsch e Warmuth (2005) introduziram uma nova versão do algoritmo, chamada de $AdaBoost_v^*$, a qual maximiza a margem mínima das amostras com alta precisão. Experimentalmente, mostraram que esta abordagem precisa de uma quantidade consideravelmente menor de períodos de aprendizagem. Rudin *et al.* (2007) analisaram os algoritmos de *Boosting* mediante uma "função da margem suave", a qual é uma aproximação diferenciável da margem dos algoritmos. A partir desta função, apresentaram duas modificações no *AdaBoost* que convergem para a margem máxima assintoticamente.

O algoritmo *LPBoost* foi introduzido por Demiriz *et al.* (2002), e procura maximizar a margem mínima de todas as amostras de treinamento. Através da otimização formulada na base de um programa linear (*Linear Program - LP*), foi observado que as margens rígidas do *LPBoost* não garantem um desempenho ótimo na maioria dos casos e, além disso, apesar de maximizarem as margens mínimas, apresentou altas taxas de erro de generalização. De fato, anteriormente, Breiman (1999a) tinha observado o mesmo fenômeno, quando propôs o algoritmo de *Arc-Gv*. Este maximiza a margem mínima sobre o conjunto de treinamento, mas com erros de generalização altos. Os experimentos realizados no *LPBoost* e *Arc-Gv* colocaram a eficiência da teoria da margem em dúvida.

Com o propósito de verificar o fenômeno e avaliar a teoria da margem, Reyzin e Schapire (2006) reproduziram a principal descoberta do Breiman (1999a). Descobriram que há uma expli-

cação simples para o desempenho ruim do *Arc-Gv*, que está relacionada ao aumento excessivo da complexidade dos classificadores fracos. Isso é verificado experimentalmente e é consistente com a teoria da margem. Assim, os autores concluíram que a maximização das margens é desejável, mas não necessariamente em detrimento de outros fatores, especialmente na complexidade dos classificadores fracos. Além disso, eles sugeriram o uso da *margem média* e *margem mediana* como medidas de comparação da distribuição de margens.

Atualmente, a distribuição de margens é amplamente aceita como margem de classificação com a generalização do algoritmo. Wang *et al.* (2008) introduziram uma média de margem nova, chamada de *Equilibrium margin (EMargin)*. Esta abordagem de margem pode ser vista como uma medida de quão eficiente pode ser a distribuição de margens. A *EMargin* depende da distribuição de margens, embora esteja fracamente relacionada com a margem mínima.

O *MDAdaBoost (margin-distribution boosting)* foi introduzido por Shen e Li (2010a) mediante a otimização da distribuição de margens; o método, otimiza diretamente a distribuição de margens, procurando maximizar a margem média e minimizar a variância.

3.6.3 Os classificadores fracos

Durante a avaliação dos algoritmos de *Boosting*, muitas vezes a importância da confidência α_t foi questionada. O *LocAdaBoost*, introduzido por Meir *et al.* (2000), é um "*Boosting* com localizações" que relaciona os valores das confidências com os dados de entrada. Este algoritmo permite que estas confidências variem em função dos dados de entrada.

Por outro lado, vários autores procuraram o melhor procedimento para fazer a escolha do classificador. O algoritmo *RankBoost* (Freund *et al.* , 2003) combina vários *rankings* ou preferências das amostras, ordenando classificadores de acordo com as características preferidas (com maior *ranking*).

Por outro lado, Li e Zhang (2004) incorporaram o método *Floating Search* (Pudil *et al.* , 1994), obtendo um novo algoritmo chamado de *FloatBoost*. Assim, a fim de melhorar a seleção de características, utilizaram um mecanismo de *backtrack*, removendo aqueles classificadores fracos ineficientes. Isso tudo se dá com a condição de que esta remoção levasse a uma taxa de erro menor que algum limiar predefinido.

O *PAV-AdaBoost*, proposto por Wilbur *et al.* (2005), introduziu o algoritmo *Pool Adjacent*

Violators (PAV) (Härdle , 1990) com o propósito de aplicá-lo nos classificadores fracos, produzindo novos classificadores fracos com uma confidência α_t melhorada. Um classificador fraco poderia inverter incorretamente a ordem dos pesos de duas amostras, bem como poderia atribuir pesos que não fossem consistentes com as amostras. Em contraste com o *AdaBoost*, *PAV-AdaBoost* utiliza estruturas lineares ordenadas.

3.6.4 Otimização da função de custo

A primeira modificação real do algoritmo *AdaBoost* possivelmente tenha sido o *Real AdaBoost* (Schapire e Singer , 1999), o qual otimiza a função de custo com menor taxa de erro, mediante a otimização da função $E[exp(-y(F(x)+h_i(x))]$. Esta modificação é chamada de *Real* porque os classificadores atribuem uma probabilidade de pertencer a uma classe, mediante um valor real que considera a distribuição atual de pesos. Em contraste, o algoritmo *GentleBoost* é uma versão mais robusta e estável. Proposto por Friedman *et al.* (2000), o algoritmo utiliza o método de Newton em cada passo da otimização. Ao invés de ajustar a probabilidade estimada das classes, o *GentleBoost* utiliza uma regressão dos mínimos quadrados ponderados para minimizar a função de custo $E[exp(-yF(x))]$.

Similarmente, Friedman *et al.* (2000) introduziram o *LogitBoost*, substituindo a função de custo exponencial do *AdaBoost* pela função de regressão logística. O *LogitBoost* pode ser visto como uma generalização da regressão logística clássica, dado que adota um modelo aditivo em função de T termos : $F(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$. Em cada estágio de iteração, o *LogitBoost* ajusta uma função de distribuição para cada classe separadamente. Esta é necessária quando implementamos o classificador fraco usando uma árvore de regressão. O *LogitBoost* minimiza : $\sum_i \log(1+exp(-y_i F(x_i)))$.

Shen e Li (2010b) introduziram o $AdaBoost_{\ell 1}$, o qual é uma versão da norma $\ell 1$ regularizada do *AdaBoost*. Eles demonstraram a importância de introduzir uma função logarítmica de custo: $log\left(\sum_{i=1}^{N} exp\left(-y_i\left(\sum_t h_t(x_i)\alpha_t\right)\right)\right)$ sujeito a $\alpha_t \geqslant 0$ com $\sum_t \alpha_t = 1/T$, que maximiza a margem média, oferecendo uma explicação teórica alternativa e consistente com a teoria da margem.

3.6.5 Otimização da entropia

A entropia relativa é um caso especial das *distâncias de Bregman* (Bregman , 1967). Nos algoritmos de *Boosting*, a maioria das discussões referem-se ao estágio de atualização dos pesos das

amostras, que é considerado como um minimizador limitado da entropia relativa. As limitantes dos erros de treinamento baseiam-se na forma específica da função exponencial, e esta opera na atualização de pesos, minimizando a entropia relativa (Kivinen e Warmuth , 1999). De forma similar, Lebanon e Lafferty (2001) e Collins *et al.* (2002) notaram a conexão entre as técnicas de *Boosting* e a maximização da entropia baseados nas *distâncias de Bregman*.

Com o propósito de determinar um limitante superior ótimo, o algoritmo *SoftBoost* (Warmuth *et al.* , 2007) introduziu uma regularização da entropia relativa. Esta produz um limitante superior que é logarítmico com respeito ao número de amostras e, diferentemente do *LPBoost*, o erro de generalização decresce lentamente apenas no início do treinamento. De maneira similar, Warmuth *et al.* (2008) propuseram uma variante do *LPBoost* chamada de *ERLPBoost (Entropy Regularized LPBoost)*. Adicionaram a mesma regularização da entropia relativa, combinando a performance do *LPBoost* para maximizar as margens com uma melhor limitante superior para o erro de generalização.

Shen e Li (2010b) apresentaram o *AdaBoost-QP*, que otimiza diretamente a função de custo assintótica do *AdaBoost*. Demonstraram que aquela função de custo pode ser considerada como um problema de programação convexa (*QP*) nas confidências dos classificadores, quando todos os classificadores são conhecidos.

4 DETECÇÃO DE FACE

Durante a última década, os computadores têm mostrado uma grande capacidade de interagir com os seres humanos, de maneira cada vez mais intensa e natural. A detecção de face é uma área da visão computacional que busca métodos automáticos para extração de informações de faces em imagens. Esta área permite uma interação humano-computador, mediante aplicações como reconhecimento de face, verificação de face, alinhamento de face, modelamento de face, reconhecimento de expressões faciais, entre outros.

O problema da detecção de face foi exposto detalhadamente em Yang *et al.* (2002). Dada uma imagem arbitrária, a detecção de faces deve verificar a possível existência de faces nela, retornando a localização e o tamanho de cada face. Numerosas técnicas de detecção de face têm sido propostas na literatura. Nesse sentido, Yang *et al.* (2002) e Hjelmås e Low (2001) apresentam uma revisão exaustiva das metodologias existentes até o ano 2000. Pontualmente, Yang *et al.* (2002) classificaram os métodos de detecção de face em quatro categorias: métodos baseados no conhecimento, métodos do *template* apropriado, métodos baseados nas características e métodos baseados na aparência. Similarmente, Zhang e Zhang (2010) revisaram várias abordagens desenvolvidas na ultima década, com maior ênfase nos métodos baseados em *Boosting*.

Neste capítulo, descrevemos os principais métodos de detecção de face que se destacaram no passado.

4.1 Métodos Baseados em Conhecimento

A metodologia baseada em conhecimento tenta descrever o nosso conhecimento prévio sobre o padrão das faces através de algumas informações explícitas, como intensidade das faces, contorno elíptico da face, relação triângulo equilátero entre olhos e boca, entre outras.

Yang e Huang (1994) apresentaram um método baseado no conhecimento através de uma hierarquia de imagens multi-resolução: estas imagens foram chamadas de "imagens mosaico", as quais são imagens de baixa resolução divididas em células quadradas de iguais tamanhos, conforme apresentado na Figura 4.1. Este método consiste em três etapas. Na primeira etapa, utiliza-se um conjunto de regras simples, baseada no contraste da face e do fundo da imagem. Por exemplo, a diferença entre a média dos níveis de cinza da parte central da face (parte escura sombreada) e a parte que a rodeia (parte sombreada levemente), como se apresenta na Figura 4.2. A saída da primeira etapa é um conjunto de faces candidatas. Na segunda etapa, usaram uma janela dividida em 64 células (8×8) sobre cada imagem candidata e estabeleceram um conjunto de regras para detecção de características faciais. Estas regras basearam-se nas médias dos níveis de cinza das 8 células em cada linha (horizontalmente). Assim, estas médias são utilizadas para procurar mínimos locais (baixos níveis de cinza) na imagem, os quais poderiam indicar uma região contendo características faciais, tais como olhos, boca ou nariz. Na terceira etapa, inicialmente extraíram as bordas das características faciais (olhos e boca). Aplicaram uma equalização de histograma nas imagens candidatas e obtiveram as bordas usando limiares fixos. Adicionalmente, usaram um método de multi-binarização para detectar bordas com três limiares diferentes. O resultado final foi uma combinação de três conjuntos de bordas indicando as posições dos olhos e da boca.

(a) *n=1* (b) *n=2* (c) *n=4* (d) *n=8*

(e) *n=16* (f) *n=32* (g) *n=64* (h) *n=128*

Figura 4.1: (a), imagem original. (b), (c), (d), (e), (f), (g), (h), imagens de resolução menor. Cada janela possui $n \times n$ píxeis, onde a intensidade de cada pixel é substituído pela intensidade média dos píxeis da janela

A implementação proposta anteriormente é computacionalmente custosa e apenas permite usar

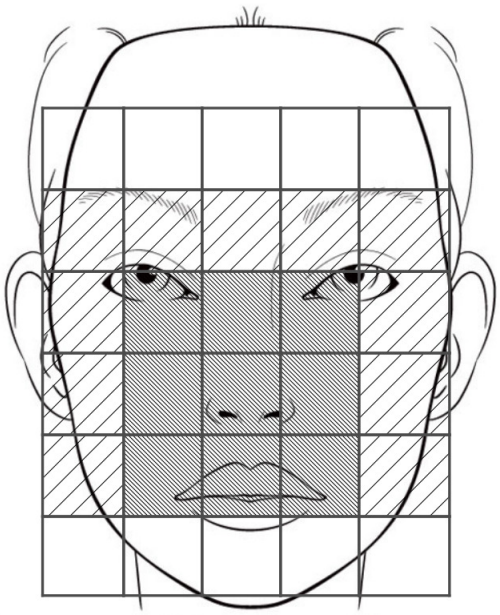


Figura 4.2: Imagem tipica usada em métodos baseado no conhecimento

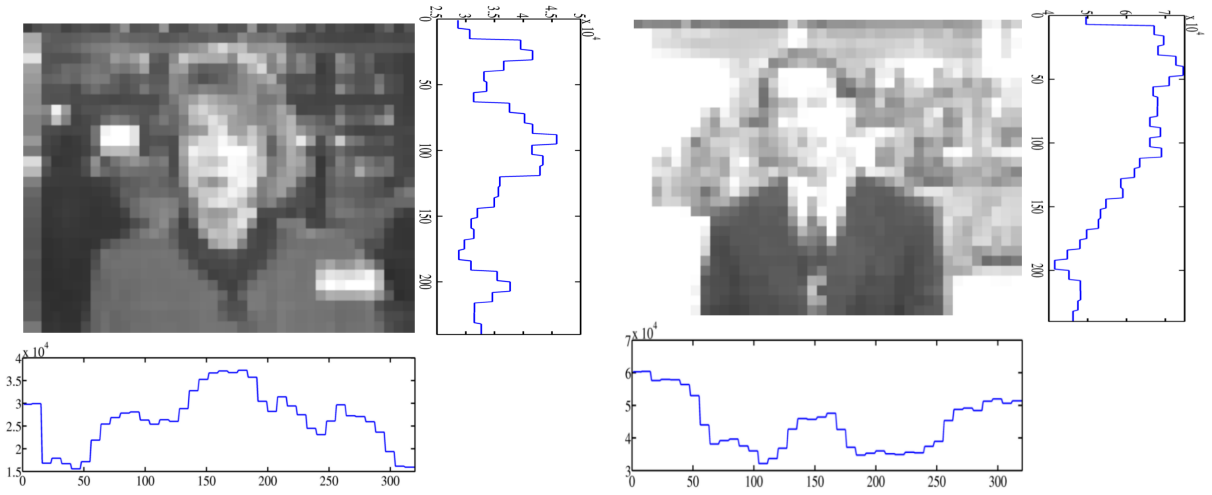
células quadradas. Com o propósito de evitar estas deficiências, Kotropoulos e Pitas (1997) apresentaram uma abordagem baseada em regras hierárquicas e estimação de células através de projeções horizontais e verticais da imagem. Uma projeção horizontal P_{horiz} da face contém a média de todas as intensidades de pixel $I(x,y)$ das colunas, como apresentada na Equação 4.1. Considerando as mudanças abruptas da intensidade, nesta projeção, são determinados dois mínimos locais significativos, os quais correspondem aos lados esquerdo e direito da cabeça (bochechas). Similarmente, a projeção vertical P_{vert} da imagem foi obtida através da média de todas as intensidades de pixel $I(x,y)$ de cada linha, apresentada pela Equação 4.2. Considerando as mudanças abruptas de intensidade, na projeção vertical foram localizados um mínimo local e um máximo local, correspondentes aos olhos ou sobrancelhas e à pele entre a boca e os olhos, respectivamente. Estas características detectadas formam as faces candidatas.

$$P_{horiz}(x) = \sum_{y=1}^{n} I(x,y) \tag{4.1}$$

$$P_{vert}(y) = \sum_{x=1}^{m} I(x,y) \tag{4.2}$$

Na Figura 4.3a, é apresentado um exemplo onde as bordas da face correspondem aos mínimos locais, produto de mudanças abruptas nas projeções horizontal e vertical. Porém, este método

possui baixo desempenho com a presença de fundos complexos e faces rotacionadas, conforme ilustrado na Figura 4.3b.



(a) *n=4* (b) *n=8*

Figura 4.3: Projeções horizontais e verticais em imagens mosaico: (a) $n = 4$ e (b) $n = 8$

4.2 Métodos baseados em *Templates*

Dada por uma função de energia, os métodos baseados em *templates* (moldes) procuram encontrar a melhor correspondência, entre um objeto da imagem e um modelo previamente definido. No caso da detecção de faces, este objeto poderia ser um olho, a boca ou algum outro elemento desejado.

4.2.1 *Templates* Pré-definidos

Sakai *et al.* (1969) usaram *sub-templates* para modelar a face; cada *sub-template* é definido em termos dos contornos dos olhos, boca, nariz e face. Os componentes da face são segmentados baseados na maior mudança do gradiente e em seguida são casados com os *sub-templates*. Inicialmente, as correlações entre os componentes e os *sub-templates* são computados para encontrar localizações candidatas da face. Posteriormente, as imagens candidatas são casadas com outros *templates* especializados para determinar a existência de uma face. As comparações entre amostra candidata e *templates* se-basea em métricas adequadas, normalmente distância euclidiana.

Similarmente, Craw *et al.* (1992) introduziram um método de localização baseado nos contornos frontais da face. Este método consiste em duas etapas. A primeira etapa extrai os contornos da cabeça usando filtros de *Sobel*. Estes contornos são agrupados procurando um casamento com os *templates* predefinidos. Na segunda etapa, o processo de extração de contornos é repetido em diferentes escalas a fim de procurar características locais tais como olhos, lábios e sobrancelhas. Govindaraju *et al.* (1990) e Govindaraju (1996) construíram um modelo da face em termos dos seus contornos usando o detector de bordas de *Marr-Hildreth*, o qual extrai as bordas convoluindo a imagem com o Laplaciano de uma função gaussiana. Os fragmentos dos contornos são conectados em pares com base na sua proximidade e orientação, e os cantos são detectados para segmentar os contornos em curvas características. Em seguida, estas curvas são rotuladas avaliando suas propriedades geométricas e posições relativas na vizinhança. Um grupo de três curvas é considerado como face se, após comparar os raios de cada curva com um raio pré-definido, for obtida uma distância mínima. Estes métodos possuem a vantagem de serem facilmente implementados, mas são susceptíveis a mudanças de escala, pose ou forma.

4.2.2 *Templates* deformáveis

Com o propósito de melhorar o desempenho dos *templates* em termos de variações de escala, inclinação, rotação e iluminação, muitos autores introduziram *templates* deformáveis. O método proposto por Yuille *et al.* (1989) ajusta um modelo elástico com os componentes faciais, tais como olhos, nariz e boca. Estes componentes são detectados usando *templates* parametrizados, os quais utilizam uma função de energia para conectar os contornos, picos e vales da imagem. O melhor ajuste do modelo elástico é encontrado minimizando uma função de energia dos parâmetros, onde estes parâmetros descrevem as características da face.

A Figura 4.4 apresenta o *template* dos olhos contendo um total de nove parâmetros. A função de energia completa $E_c(\vec{x}_e,\vec{x}_c,p_1,p_2,r,a,b,c,\theta)$ é expressa como uma combinação de termos relacionados com as características da face. O círculo de raio r, centrado no ponto $\vec{x}_c$, corresponde ao limite entre a íris e a esclera. O interior do círculo representa um vale ou valores baixos na intensidade de pixeis. O contorno do olho é modelado por duas curvas parabólicas representando as limitantes superior e inferior dos olhos, com centro em $\vec{x}_e$ e largura $2b$. Os tamanhos máximos desde as limitantes superior e inferior até o centro $\vec{x}_e$ são a e c, respectivamente, onde o centro possui um ângulo de orientação θ. Os pontos correspondentes aos centros das escleras são considerados como picos. Estes pontos são etiquetados por $\vec{x}_e + p_1(\cos\theta,\sin\theta)$ e $\vec{x}_e + p_2(\cos\theta,\sin\theta)$,

onde $p_1 \geq 0$ e $p_2 \leq 0$. O ponto $\vec{x}_e$ corresponde ao centro do olho e θ corresponde à orientação do olho.

Esta função de energia garante que o algoritmo convergirá agindo como uma função de *Lyapunov* e mostra uma medida da eficiência do *template*.

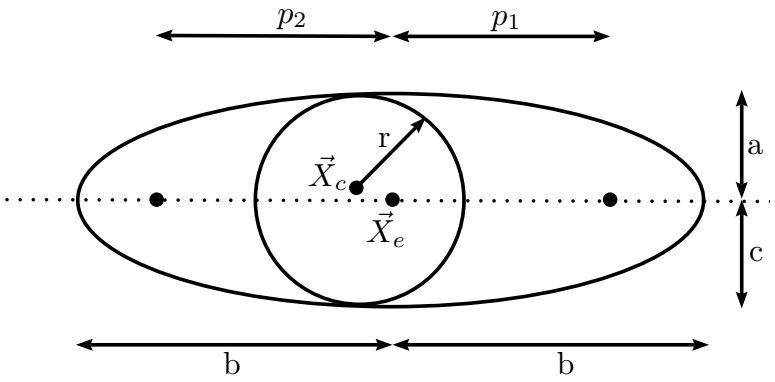


Figura 4.4: Modelo da *template* deformável utilizada por Yuille *et al.* (1989) na detecção dos olhos.

Kwon e da Vitoria Lobo (1994) propuseram um método de detecção de faces combinando *templates* e contornos ativos. O método de contornos ativos (também chamados de *snakes*) procura ajustar uma curva fechada sobre as bordas dos objetos de interesse. O método de detecção proposto inicialmente faz uma convolução entre a imagem e um filtro de suavização gaussiano; em seguida, aplica um operador morfológico a fim de realçar os contornos. Após isto, o método de contornos ativos procura e elimina os segmentos de curva menos representativos, mantendo aqueles contornos com maior probabilidade de representar uma face. Estes contornos são usados para se obter uma transformação de *Hough* através de elipses. Assim, estas elipses são descritas por quatro parâmetros e usadas como localizações candidatas da face. Finalmente, em cada elipse é aplicado um método similar ao proposto por Yuille *et al.* (1989) a fim de obter detalhes característicos. Um grupo de contornos será considerado como face caso um grande número de características faciais sejam encontradas no interior da elipse, e além disso, as distâncias entres estas características satisfaçam as proporções do *template* da face.

4.3 Métodos Baseados em Características

Tendo por base a idéia de que o ser humano é capaz de identificar objetos independentemente do ponto de vista, esta metodologia explora características invariantes presentes em faces, tais como

cor da pele e a textura da face, os quais são altamente influenciados pelos nível de iluminação, ruído e oclusão.

4.3.1 Características Faciais

O método de detecção de faces proposto por Sirohey *et al.* (1993) se baseia na forma elíptica da face. Inicialmente, usa um detector de bordas de *Canny* para extrair segmentos dos contornos da face. Estes segmentos são combinados em pares e ajustado num sistema de equações lineares. Para obter os valores dos parâmetros da elipse (semi-eixo maior, semi-eixo menor e excentricidade), é aplicado o método da pseudo-inversa no sistema de equações. Depois de combinar todos os segmentos dos contornos, estes são ajustados na elipse e classificados para obter parâmetros de interesse. Finalmente, um algoritmo de voto é usado para encontrar os melhores valores dos parâmetros.

Graf *et al.* (1995) introduziram um método para localizar características locais da face em imagens em escala de cinza. Este método, inicialmente, combina filtros passa-faixa e operações morfológicas a fim de realçar aquelas regiões com alta intensidade de cor e que apresentem uma forma de interesse (por exemplo uma forma de olho). Considerando que os histogramas das imagens normalmente possuem um pico alto, aplicam um limiar adaptativo baseados no valor do pico a fim de obter duas imagens binarizadas. Finalmente, obtêm as áreas com características faciais candidatas identificando os componentes conectados nas duas imagens binarizadas. Porém, não fica claro como as características candidatas são combinadas para localizar a face.

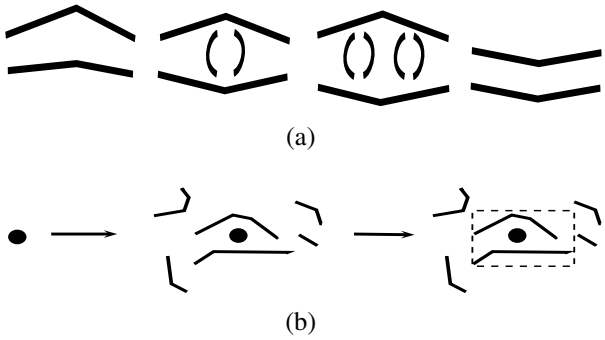
Leung *et al.* (1995) utilizaram cinco características para descrever uma face (dois olhos, duas fossas nasais, e os lábios), como se apresenta na Figura 4.5. Para cada par de características faciais do mesmo tipo (por exemplo o olho esquerdo e o direito) se computa uma distância, e o conjunto de medidas de distância é modelado usando uma distribuição gaussiana. Um *template* da face é definido com base na média das respostas. Porém, este sistema depende de um número muito reduzido de características, que não são extraídas de forma robusta devido ao ruído da imagem ou outros fatores. Dessa forma, com escassez das características faciais o desempenho do algoritmo é significativamente afetado.

Usando um grande número de medidas geométricas, Yow e Cipolla (1997) propuseram um método que detecta pontos característicos da face numa imagem. Inicialmente, aplicaram um filtro gaussiano de segunda ordem indicando as posições prováveis das características faciais. Assim,

(a) (b)

Figura 4.5: (a) Os elipses mostram as áreas com alta probabilidade incluindo as características ausentes. (b) Performance da abordagem do (Leung *et al.* , 1995) mostrando o melhor emparelhamento.

baseados na similaridade, agruparam as bordas em torno dessas posições. Finalmente, usaram uma rede Bayesiana para avaliar cada característica facial e cada agrupamento e detectar rostos em diferentes orientações e poses, como se apresenta na Figura 4.6 .



(a)

(b)

Figura 4.6: (a) Modelos das características faciais proposto por Yow e Cipolla (1997). (b) Processo de seleção de características .

4.3.2 Textura e Cor da pele

As texturas são propriedades estruturais das superfícies, as quais exibem homogeneidade apesar das flutuações do brilho e da cor. Dai e Nakano (1996) usaram um modelo da textura da face baseadas na cor da pele, implementando um esquema de varredura em cenas coloridas e considerando as partes de cor laranja como áreas que incluem faces. Assim, conseguiram detectar faces

rotacionadas ou com características adicionais tais como barba e óculos.

Por outro lado, a cor da pele é considerada como uma característica particular da face humana, embora os seres humanos possuam diferentes cores de pele. Com o propósito de rotular os pixeis como pele, distintos trabalhos usaram múltiplos espaços de cor. Inicialmente Jebara *et al.* (1998) e Satoh *et al.* (1999) usaram o espaço de cores RGB. De forma similar, Qian *et al.* (1998) utilizaram um espaço RGB normalizado. No entanto, Sobottka e Pitas (1996) usaram o espaço HSV (ou HSI) e Chai e Ngan (1998) aplicaram o espaço YCrCb, entre outros.

Os modelos de detecção baseados na cor da pele são ferramentas eficientes, proporcionando robustez em imagens rotacionadas e ocluídas, porém, estes modelos se mostram instáveis e perdem eficiência quando as fontes de iluminação variam significativamente. McKenna *et al.* (1998) apresentaram um modelo adaptativo de mistura de cores, a fim de rastrear faces em distintas condições de iluminação. Eles modelaram as distribuições das cores da pele como uma mistura de Gaussianas, atribuindo uma probabilidade a cada pixel da imagem, de modo que as faces foram detectadas agrupando as áreas com maior probabilidade. Embora este método seja capaz de rastrear as faces em distintas condições de iluminação, são exigidas múltiplas imagens sequenciais no processo de detecção de face.

4.4 Métodos baseados na Aparência

Os métodos baseados na aparência não necessitam de nenhum conhecimento prévio sobre o objeto ou característica a ser detectada e sim algumas imagens para fazer o treinamento. Esses métodos estão normalmente baseados em técnicas de análise estatística e aprendizado de máquina, e apresentam um desempenho superior em comparação com as técnicas anteriores.

4.4.1 Métodos baseados em redes neurais

As Redes Neurais obtiveram bastante sucesso em diversos problemas do reconhecimento de padrões, tais como reconhecimento de caracteres, reconhecimento de objetos, etc. Considerando a detecção de faces como um problema de classificação binária, vários autores propuseram distintas arquiteturas de redes neurais. O método desenvolvido por Rowley *et al.* (1996) e Rowley *et al.* (1998b) utiliza um algoritmo baseado numa rede neural para detectar faces em imagens em escala de cinza. Inicialmente, a imagem inteira é segmentada em regiões da imagem (janelas) e cada janela

é submetida a uma equalização de histograma e uma normalização da iluminação. Posteriormente, cada imagem de janela é enviada para uma rede neural que verifica se a janela possui ou não uma face. A rede neural da Figura 4.7 possui 3 arquiteturas diferentes na camada oculta, as quais avaliam diferentes partes da imagem e procuram características relevantes. A primeira arquitetura possui quatro unidades ocultas que utilizam sub-regiões de 10×10 pixeis, a segunda possui 16 unidades a cada sub-região de 5×5 pixeis e a terceira unidade possui 6 unidades de 20×5 pixeis. Finalmente, a saída da rede neural retorna um único valor real, indicando se a janela possui ou não uma face. No entanto, este sistema apenas é eficiente em imagens de faces frontais.

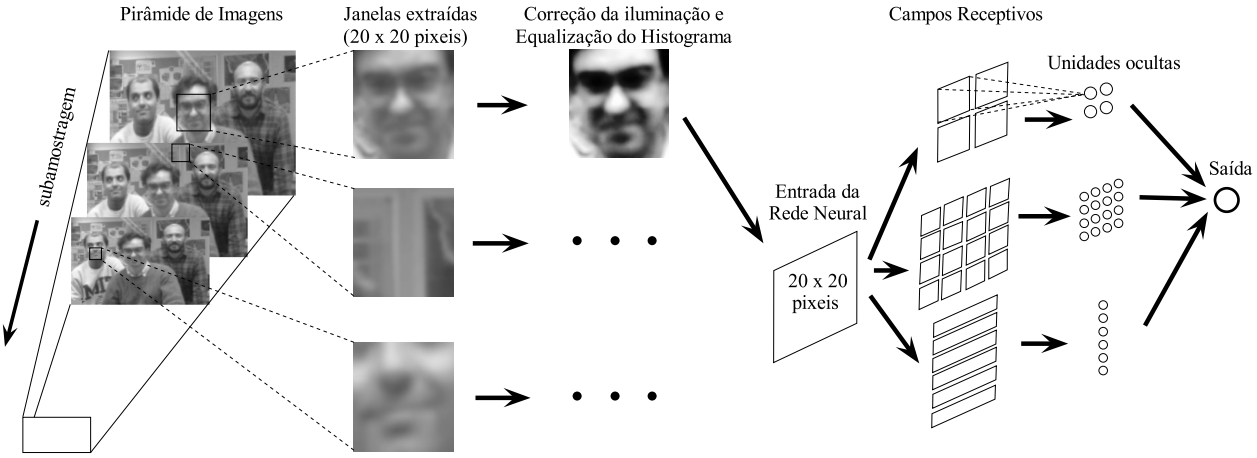


Figura 4.7: O algoritmo utilizado por Rowley *et al.* (1996) para detecção de faces.

Rowley *et al.* (1998a) adicionaram uma rede neural chamada de roteadora (*router network*) a fim de determinar o ângulo de rotação da face. Esta rede continha 400 neurônios na camada de entrada, 15 na camada intermediária e 36 na camada de saída. Estas 36 saídas são colocadas num vetor que indica o ângulo da face, no qual cada elemento contribui com um ângulo de 10° ao vetor de saída da rede roteadora. O resto do algoritmo continua como em Rowley *et al.* (1996).

4.4.2 Métodos baseados em máquinas de suporte vetorial (SVM)

Uma máquina de suporte vetorial (SVM) é um classificador linear, que cria um hiperplano de separação e minimiza o erro de classificação nas amostras de teste, combinando os pesos dos sub-conjuntos de vetores de treinamento (vetores de suporte). Diferente das Redes Neurais, que procuram minimizar o erro de treinamento (*empirical risk minimization*), uma SVM minimiza o erro de teste (*structural risk minimization*).

Osuna *et al.* (1997) provavelmente foram os primeiros em usar um método baseado em máquinas de suporte vetorial para detecção de faces. Li *et al.* (2000) usaram uma SVM para detecção e reconhecimento de faces aproveitando a propriedade simétrica da face e criando um detector de faces para cada ponto de vista da face. Eles estimaram a pose da face, através de técnicas de suporte vetorial para regressão. Sahbi e Geman (2006) introduziram uma melhora na velocidade de teste do SVM baseados em árvores de decisão e formando um conjunto de classificadores binários. Por sua vez, Liu *et al.* (2006) propuseram um método de detecção de faces baseado numa arquitetura em cascata de *kernels*. Eles combinaram uma SVM desbalanceada com um *kernel* originado da PCA (*Principal Component Analysis*) para melhorar a taxa de detecção.

4.4.3 Métodos baseados em Boosting

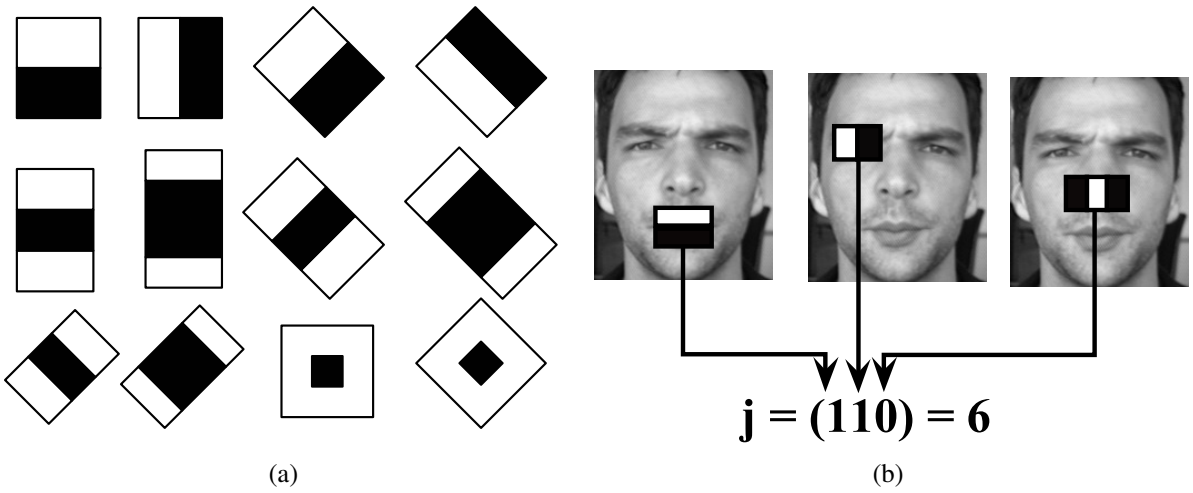
A abordagem apresentada por Viola e Jones (2001) teve sucessos significativos em aplicações reais. O classificador implementado pelo algoritmo proposto é composto de três estágios. O primeiro estágio consiste na criação de uma imagem integral que melhora a velocidade de processamento de imagem. No segundo estágio, o algoritmo *AdaBoost* é usado para obter um classificador robusto, baseado em classificadores simples, chamados de classificadores fracos. Os classificadores fracos operam de maneira específica através das características de *Haar*. O terceiro estágio da abordagem agrupa os classificadores fracos e obtém os chamados classificadores "fortes" (classificador robusto), os quais, por sua vez, são agrupados em forma de cascata.

Vários trabalhos posteriores surgiram e melhoraram este algoritmo inicial. Alguns trabalhos se concentraram no processo de extração de características, enquanto outros focaram em aperfeiçoar os algoritmos de *Boosting*.

Variações na extração de características

Lienhart e Maydt (2002) introduziram características de *Haar* rotacionadas em 45°, aumentando desta forma a quantidade de características como se apresenta na Figura 4.8a. De maneira similar, procurando características mais discriminantes e com o objetivo de melhorar a velocidade de treinamento, Mita *et al.* (2005) propuseram o uso de sub conjuntos de características de *Haar*. As características são agrupadas em função da ocorrência de varias características de *Haar*. De forma diferente ao algoritmo do Viola e Jones (2001), cada classificador fraco J possui F características de *Haar*. Durante o treinamento, estas F características são concatenadas num índice para

2^F possíveis combinações, de forma tal que a característica J represente a ocorrência de diferentes posições, resoluções e orientações, como se apresenta na Figura 4.8b.



(a) (b)

Figura 4.8: (a) Características de *Haar* rotacionadas em 45° (Lienhart e Maydt , 2002). (b) Características de *Haar* conjuntas (Mita *et al.* , 2005).

Uma limitação das características de *Haar* é a falta de robustez na análise de imagens, principalmente quando o ambiente apresenta extremas condições de iluminação. O trabalho realizado por Ojala *et al.* (2002) usa Padrões Binários Locais (*Local binary patterns - LBP*) como características. Estas características foram aplicadas com sucesso na tarefa de detecção de faces por Jin *et al.* (2004) e Zhang *et al.* (2007b). Baseados na combinação das características de *Haar* e os Padrões Binários Locais (LBP), Yan *et al.* (2008) propuseram o uso das características binárias montadas localmente (*Locally Assembled Binary - LAB*), junto com uma cascata de características cêntricas.

Dalal e Triggs (2005) introduziram um esquema que usa os descritores chamados de histogramas de gradientes orientados (*Histograms of Oriented Gradient - HOG*) na obtenção de características. De maneira similar, a fim de controlar oclusões parciais das faces, Wang *et al.* (2009) combinaram histogramas de gradientes orientados (*HOG*) e Padrões Locais Binários (*LBP*) para descrever as características da face. Entretanto, baseados nos padrões locais binários, Nosaka *et al.* (2012) introduziram uma extensão considerando a coocorrência de LBPs adjacentes. A coocorrência foi medida utilizando uma matriz de auto-correlação a partir de vários LBPs.

Variações no algoritmo de aprendizado de *Boosting*

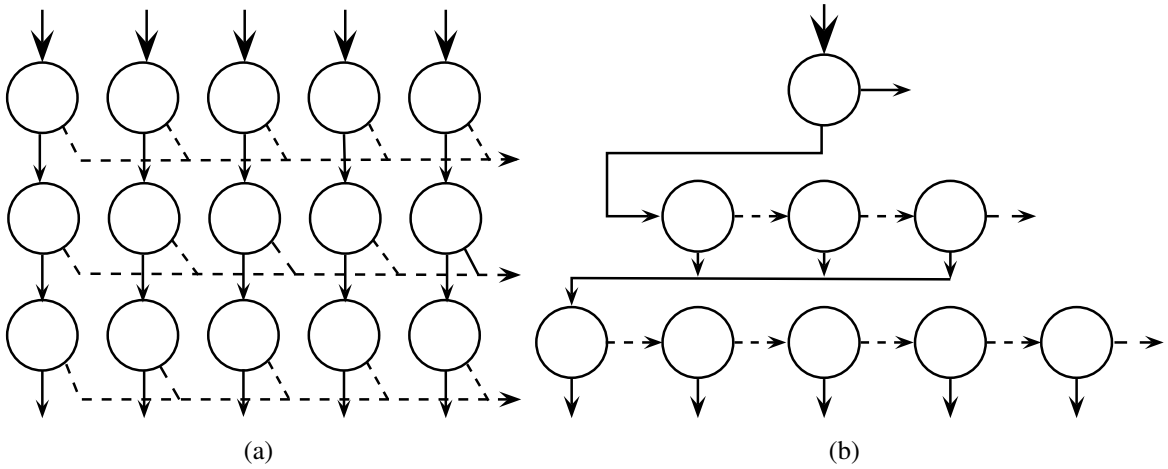
Além das abordagens focadas nas melhorias das características, outra forma de melhorar o desempenho do detector de faces é através da melhora do algoritmo de *Boosting* usando os classificadores dentro de uma cascata. O algoritmo implementado por Viola e Jones (2001) utiliza o algoritmo *AdaBoost* padrão, proposto por Freund e Schapire (1995). A primeira variante do algoritmo *AdaBoost* padrão provavelmente foi o *Real AdaBoost*. Assim, Li *et al.* (2002), Wu *et al.* (2004) e Mita *et al.* (2005) utilizaram o *Real AdaBoost* a fim de melhorar o desempenho do detector robusto em relação à rotação da face.

O algoritmo *GentleBoost* (Friedman *et al.* , 2000) é uma versão considerada mais robusta do *Real AdaBoost*; usando o método de Newton em cada passo da otimização, este algoritmo consegue atingir uma otimização exata em termos do erro de treinamento mínimo. A fim de melhorar a performance do detector, Lienhart *et al.* (2003) utilizaram o algoritmo *GentleBoost* com pequenas árvores CART como classificadores base. A variante *FloatBoost* (Li e Zhang , 2004) procura superar o problema da monotonicidade (crescimento e decrescimento de funções) no aprendizado sequencial do algoritmo *AdaBoost*. Este algoritmo não apenas adiciona novos classificadores na etapa de treinamento, como também re-examina os classificadores previamente selecionados e elimina aqueles menos importantes. Por outro lado, Jang e Kim (2008) propuseram o uso dos algoritmos evolutivos a fim de otimizar o número de classificadores e manter a taxa de detecção inalterada. Este algoritmo permite uma redução do número de classificadores de até 40%.

Contudo, o processo de treinamento de um detector de faces exige uma grande quantidade de tempo, variando de semanas até meses. Com a finalidade de encurtar o tempo de treinamento, o trabalho desenvolvido por Pham e Chain (2007) usa estatísticas da primeira e segunda ordem das características de *Haar* selecionadas das sub-janelas para treinar os classificadores fracos. De maneira similar, Schneiderman (2004) desenvolveu uma cascata de características cêntricas a fim de melhorar a velocidade de detecção. Esta abordagem computa previamente um conjunto de valores das características sobre uma parte da imagem; desta forma, todas as janelas de teste podem usar o valor correspondente da característica para o primeiro estágio da detecção.

Com o propósito de ampliar o algoritmo para detectar multi-poses, Wu *et al.* (2004) propuseram cascatas paralelas que obtém um único classificador para cada pose, e a decisão final é tomada pela cascata com maior pontuação (Figura 4.9a). Li *et al.* (2002) usaram uma estrutura em pirâmide. Esta estrutura consiste em 3 níveis. O primeiro nível da pirâmide avalia as faces em todas

as poses. O segundo nível detecta as faces considerando três poses, a esquerda (-90° até -30°), a frontal (-30° até 30°) e a direita (30° até 90°). O terceiro nível detecta faces em 7 ângulos mais sutis, como se apresenta na Figura 4.9b.



(a) (b)

Figura 4.9: (a) Cascata em Paralelo. (b) Cascata em pirâmide.

Por outro lado, Viola *et al.* (2003) propuseram usar inicialmente um estimador de pose para prever a posição da face de uma janela. Esta abordagem adota uma árvore de decisão para a detecção da pose; dessa forma, uma janela de teste é analisada apenas por uma única cascata até a pose ser estimada, melhorando dessa forma o desempenho do detector. Não obstante, o processo para detecção da orientação da pose pode possuir vários erros, como se apresenta na Figura 4.10a. Com o propósito de evitar estes erros, Huang *et al.* (2005) propuseram um detector de faces multi-pose usando uma estrutura de árvore. Eles dividiram o espaço da face em pequenos sub-espaços e, baseados nessas divisões, cada classificador fraco retorna um vetor de valores. Consequentemente, cada amostra de teste é analisada por várias sub-categorias de classificadores durante a etapa de teste, como se apresenta na Figura 4.10b.

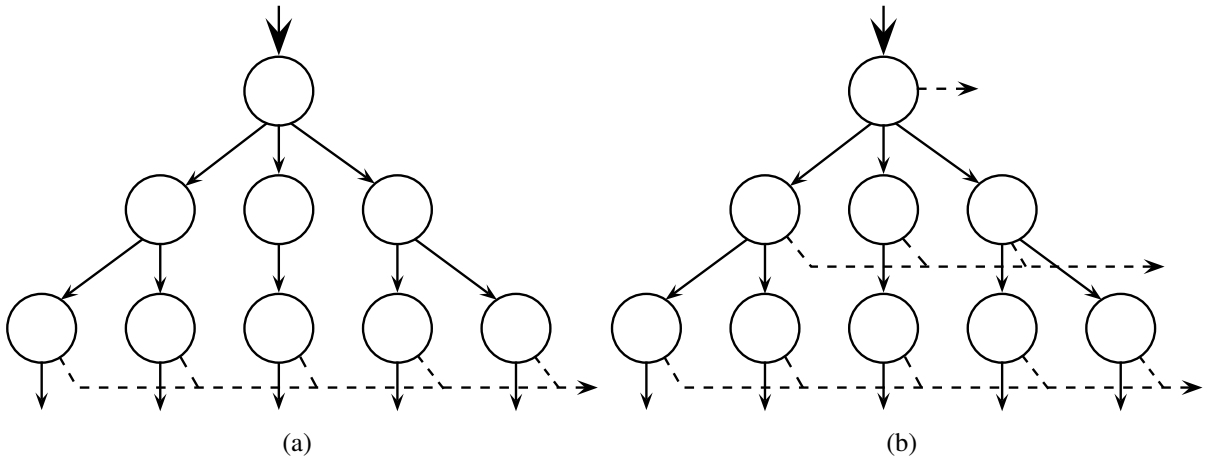


Figura 4.10: (a) Cascata em estrutura de árvore. (b) Cascata em estrutura de árvore não exclusiva.

5 ALGORITMO ADABOOST ROBUSTO AO RUÍDO

Os aplicativos nas áreas de vídeo segurança e análise de vídeo estão baseados na premissa de que a detecção de face funciona de forma precisa. Nestas áreas, as técnicas de detecção com maior sucesso foram as de *Boosting (AdaBoost)*. Porém, essas técnicas ainda apresentam certas deficiências. Em outras palavras, diversos aplicativos em tempo real mostraram que essas técnicas não são totalmente confiáveis, especialmente quando as imagens possuem baixa resolução, os objetos face estão distantes ou as câmeras precisam cobrir ângulos de visualização amplos.

Nesse sentido, o estudo e aperfeiçoamento da tolerância ao ruído no algoritmo *AdaBoost* possui uma importância relevante, principalmente quando o banco de dados é extremamente grande ou ruidoso. Embora os classificadores sejam capazes de aprender comportamento e características do ruído, eles também podem ser sobreajustados.

Neste capítulo, propomos uma modificação do algoritmo *AdaBoost* com uma estratégia diferente de atualização dos pesos e analisamos como esta atualização influencia no erro empírico. O objetivo desta nova abordagem é evitar problemas de sobre-ajuste introduzindo uma atualização de pesos mais robusta. Finalmente, adaptamos o uso deste algoritmo na tarefa de detecção de faces.

5.1 Análise do algoritmo *AdaBoost* considerando o erro empírico

Dietterich (2000), Rätsch *et al.* (2001) e Servedio (2003) demonstraram que o algoritmo *AdaBoost* possui uma alta tendência de sobreajuste às amostras em presença de dados com nível de ruído alto. Hawkins (2004) indica que o sobreajuste aparece quando o número de classificadores aumenta de forma excessiva, ou seja, produz-se uma combinação de classificadores muito com-

plexa, o que leva a uma deterioração do desempenho.

Embora o classificador tenha sido treinado para minimizar o erro empírico, pode acontecer que, quando é aplicado nos dados de teste, um erro de generalização grande seja gerado. Acreditamos que isso seja causado pelos dados de treinamento com alto nível de ruído, que provoca sobreajuste. Vários pesquisadores propuseram estratégias na modificação dos pesos atribuídos às amostras a fim de evitar esse fenômeno.

O algoritmo *AdaBoost* clássico combina vários classificadores fracos a fim de construir um classificador forte. Utiliza-se um conjunto de treinamento $x_i \in \chi, i = 1 \ldots N$, com rótulos $y_i \in \{-1,1\}$ para ajustar estes classificadores. Como apresentado anteriormente no Algoritmo 3.1, uma distribuição de probabilidades $W(i)$ é usada para descrever os pesos das amostras e os pesos são modificados de acordo com a Equação 3.10.

Vapnik (1982) define o erro empírico de um classificador h_t com respeito às amostras χ como:

$$\varepsilon = \sum_{i=1}^{N} |(h_t(x_i) \neq y_i)| \tag{5.1}$$

Esse erro possui o seguinte limitante de erro (Freund e Schapire , 1995):

$$\varepsilon \leqslant \sum_{i=1}^{N} exp(-y_i h_t(x_i)) \tag{5.2}$$

Procura-se minimizar este limitante para se atualizar os pesos das amostras e, desta forma, obter a Equação 5.3, a qual é uma expressão analítica para calcular a importância α_t do classificador selecionado em cada iteração.

$$\alpha_t = \frac{1}{N} \ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right) \tag{5.3}$$

O procedimento de atualização dos pesos poderia causar uma distorção considerável sobre a distribuição de pesos (Freund , 1999). Esta distorção é causada pelo erro acumulativo adquirido em iterações repetidas. Com o objetivo de descobrir que fatores influenciam no erro empírico ε_t, considera-se que este pode ser dividido em pequenos erros, que aqui chamaremos de *erros fragmentários*, de tal forma que a soma de *"V" erros fragmentários* descreva o erro empírico, como se ilustra na Equação 5.4.

$$\varepsilon_t = \varepsilon_{t_1} + \varepsilon_{t_2} + \cdots + \varepsilon_{t_V} = \sum_{v=1}^{V} \varepsilon_{t_v} \tag{5.4}$$

Teorema 1 *Consideremos um algoritmo de aprendizado fraco, de forma tal que, quando chamado pelo AdaBoost em cada iteração sejam gerados classificadores com erros* $\varepsilon_1, \ldots, \varepsilon_T$. *O erro de cada iteração pode ser dividido em "V" erros fragmentários* $\varepsilon_t = \varepsilon_{t_1} + \varepsilon_{t_2} + \cdots + \varepsilon_{t_V}$. *Então o erro de generalização* $\varepsilon_g = Pr_{i\sim W}[F(x_i) \neq y_i]$ *do classificador final F, possui o seguinte limite superior :*

$$\varepsilon_g \leqslant 2^T \prod_{t=1}^{T} \sqrt{\left(\sum_{v=1}^{V} \varepsilon_{t_v}\right)\left(1 - \sum_{v=1}^{V} \varepsilon_{t_v}\right)} \tag{5.5}$$

Prova 1 *Vejamos a Equação 5.2*

$$\begin{aligned} \varepsilon &\leqslant \sum_{i=1}^{N} exp(-y_i F(x_i)) \\ \varepsilon &\leqslant \frac{1}{N} \sum_{i=1}^{N} exp(-y_i \sum_{t=1}^{T} \alpha_t h_t(x_i)) \end{aligned} \tag{5.6}$$

A qual pode ser reescrita da seguinte forma:

$$\varepsilon = \frac{1}{N} \sum_{i=1}^{N} \prod_{t=1}^{T} exp(-y_i \alpha_t h_t(x_i)) \tag{5.7}$$

Assim, de acordo com a Equação 3.10, consideramos:

$$\frac{W_{t+1}(i)}{W_t(i)} = \frac{exp(-y_i \alpha_t h_t(x_i))}{Z_t}$$

Substituindo na equação 5.7, obtemos:

$$\begin{aligned} \varepsilon &= \frac{1}{N} \sum_{i=1}^{N} \prod_{t=1}^{T} Z_t \frac{W_{t+1}(i)}{W_t(i)} \\ &= \frac{1}{N} \sum_{i=1}^{N} \left(\prod_{t=1}^{T} Z_t\right) \frac{W_{T+1}(i)}{W_T(i)} \end{aligned}$$

Considerando que $W_1(i) = \frac{1}{N}$

$$\varepsilon = \left(\prod_{t=1}^{T} Z_t\right)\left(\sum_{i=1}^{N} W_{T+1}(i)\right)$$

Escolhemos Z_t como um fator de normalização, tal que $\sum_{i=1}^{N} W_{T+1}(i) = 1$. Então

$$\varepsilon \leqslant \prod_{t=1}^{T} Z_t$$

Considerando que o erro ε_t é a soma de "V" erros fragmentários, vamos denotar por λ_v os fatores que influenciam em cada erro fragmentário.

$$\varepsilon_t = \varepsilon_{t_1} + \varepsilon_{t_2} + \cdots + \varepsilon_{t_v} = \sum_{v=1}^{V} \varepsilon_{t_v} \tag{5.8}$$

Queremos escolher o α_t que minimize o fator de normalização Z_t, de tal forma que:

$$\begin{aligned} Z_t = & \sum_{i:y=f_t(i)} W_t(i)e^{-\alpha_t} + \\ & \sum_{i:y\neq f_t(i)\wedge\lambda_1} W_t(i)e^{\alpha_t} + \\ & \sum_{i:y\neq f_t(i)\wedge\lambda_2} W_t(i)e^{\alpha_t} + \\ & \vdots \\ & \sum_{i:y\neq f_t(i)\wedge\lambda_v} W_t(i)e^{\alpha_t} \end{aligned} \tag{5.9}$$

Substituindo a soma dos pesos das amostras classificadas de forma incorreta para cada erro fragmentário, obtemos o seguinte:

$$\begin{aligned} Z_t &= \left(1 - \left(\sum_{v=1}^{V} \varepsilon_{t_v}\right)\right) e^{-\alpha_t} + \varepsilon_{t_1} e^{\alpha_t} + \varepsilon_{t_2} e^{\alpha_t} + \cdots + \varepsilon_{t_v} e^{\alpha_t} \\ &= \left(1 - \left(\sum_{v=1}^{V} \varepsilon_{t_v}\right)\right) e^{-\alpha_t} + \left(\sum_{v=1}^{V} \varepsilon_{t_v}\right) e^{\alpha_t} \end{aligned} \tag{5.10}$$

Definindo a derivada de Z_t (Freund e Schapire , 1995), encontra-se que a escolha de α_t que minimiza o erro empírico pode ser expressa assim:

$$\left(1-\sum_{v=1}^{V}\varepsilon_{t_v}\right)e^{-\alpha_t}+e^{\alpha_t}\sum_{v=1}^{V}\varepsilon_{t_v}=0$$

$$\frac{1-\left(\sum_{v=1}^{V}\varepsilon_{t_v}\right)}{\sum_{v=1}^{V}\varepsilon_{t_v}}=e^{2\alpha_t}$$

$$\frac{1}{2}log\left(\frac{1-\left(\sum_{v=1}^{V}\varepsilon_{t_v}\right)}{\sum_{v=1}^{V}\varepsilon_{t_v}}\right)=\alpha_t$$

Substituindo α_t na Equação 5.10 obtemos:

$$Z_t=\left(1-\sum_{v=1}^{V}\varepsilon_{t_v}\right)\sqrt{\frac{\sum_{v=1}^{V}\varepsilon_{t_v}}{1-\left(\sum_{v=1}^{V}\varepsilon_{t_v}\right)}}+\sum_{v=1}^{V}\varepsilon_{t_v}\sqrt{\frac{1-\left(\sum_{v=1}^{V}\varepsilon_{t_v}\right)}{\sum_{v=1}^{V}\varepsilon_{t_v}}}$$

$$Z_t=2\sqrt{\left(\sum_{v=1}^{V}\varepsilon_{t_v}\right)\left(1-\sum_{v=1}^{V}\varepsilon_{t_v}\right)} \tag{5.11}$$

Portanto,

$$\varepsilon\leqslant\prod_{t=1}^{T}2\sqrt{\left(\sum_{v=1}^{V}\varepsilon_{t_v}\right)\left(1-\sum_{v=1}^{V}\varepsilon_{t_v}\right)}$$

O erro de treinamento pode ser composto de vários erros fragmentários, de forma tal que esta modificação deveria apresentar um comportamento similar ao algoritmo *AdaBoost* no sentido adaptativo. A aplicação da soma de *erros fragmentários* beneficia o desempenho do algoritmo, minimizando o erro empírico de forma mais adequada e evitando problemas de sobre-ajuste diante de ruído.

No entanto, usando as Equações 5.4 e 5.9, pode-se descrever uma abordagem que considere novos fatores de influência no erro empírico. A Equação 5.9 indica que podemos procurar e achar muitos eventos para descrever o erro empírico e esses eventos deveriam ser representados pelos *erros fragmentários*.

5.2 Introduzindo um fator de frequência no *AdaBoost*

Rätsch *et al.* (2001) mostraram que *AdaBoost* não consegue ser robusto na presença do ruído e apresenta um comportamento sub-ótimo na generalização. Neste trabalho, consideramos que as amostras ruidosas possuem pelo menos uma das seguintes características: distribuição das probabilidades sobrepostas, amostras atípicas e amostras rotuladas incorretamente. Estes três tipos de fenômeno aparecem com muita frequência na análise das amostras ruidosas.

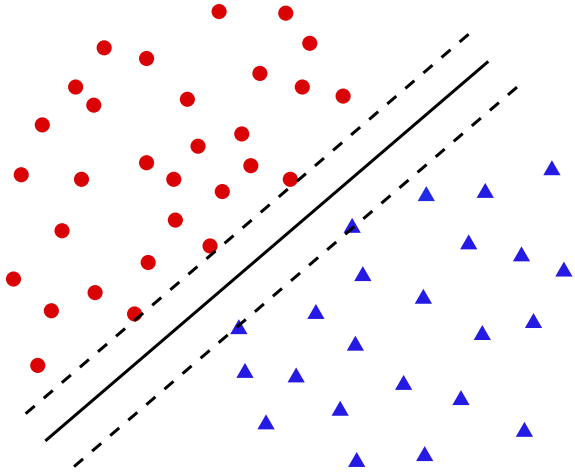
Na Figura 5.1a, observamos dados livres de ruído, nos quais é possível estimar um hiperplano de separação de forma correta. Na Figura 5.1b, ilustra-se uma amostra atípica que corrompe a estimação, de forma tal que o algoritmo se concentrará nessa amostra atípica e a boa estimação obtida na ausência desta será prejudicada. A Figura 5.1c ilustra uma fronteira de decisão mais complexa. O problema do sobre-ajuste é ainda mais acentuado, dado que é possível gerar classificadores mais complexos combinando muitos classificadores fracos. Todas as amostras de treinamento podem ser classificadas de forma correta, incluindo as amostras atípicas e as mal rotuladas, o que pode resultar em uma generalização ineficiente.

Os trabalhos de Grove e Schuurmans (1998), Quinlan (1996), Bauer e Kohavi (1999), Breiman (1999a), Dietterich (2000) e Rätsch *et al.* (2001) demonstraram que o algoritmo *AdaBoost* é sensível ao ruído. Neste trabalho, consideramos que, na presença de dados ruidosos, é necessário diminuir a influência de alguns classificadores fracos. No *AdaBoost*, a fim de evitar um incremento excessivo dos pesos, deve-se procurar alguma variável que expresse a desconfiança nos dados ruidosos. Procurando evitar esses problemas, consideram-se os *erros fragmentários*, de modo que cada um deles é associado a um evento diferente, de acordo com a Equação 5.4.

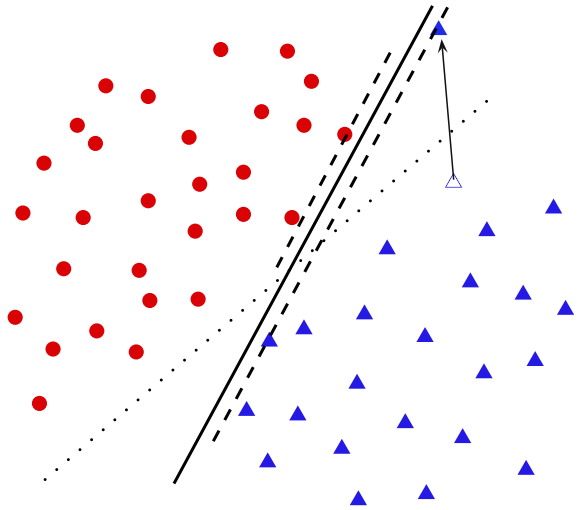
Sugerimos um método diferente para atualizar a distribuição de pesos, considerando a frequência com que uma amostra é classificada de forma incorreta. Esta frequência será denotada por δ_i e estará associada a cada amostra de treinamento x_i. Consideramos também dois eventos (λ_1 e λ_2) relacionados à frequência de amostras classificadas incorretamente δ_i, de forma tal que cada evento seja expresso como se apresenta na Equação 5.12:

$$\begin{aligned} \lambda_1 &= \delta_i \geqslant k \\ \lambda_2 &= \delta_i < k \end{aligned} \tag{5.12}$$

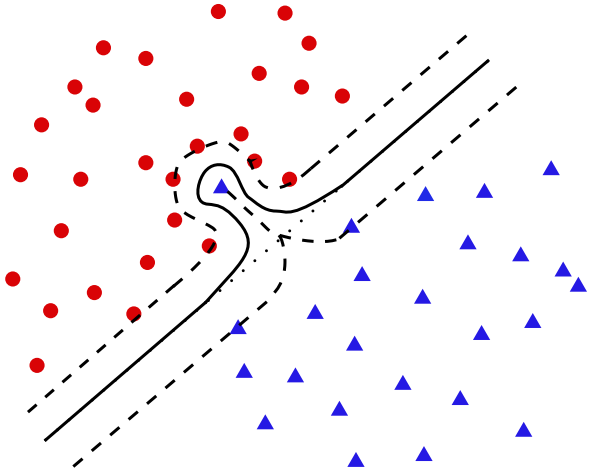
onde k denota o número máximo da frequência de amostras classificadas incorretamente.



(a) Curva de separação em amostras ideais



(b) Curva de separação com um *outlier*



(c) Curva de separação com uma amostra mal rotulada.

Figura 5.1: Curvas de separação.

Assim como Schapire e Singer (1999), vamos supor que o intervalo de cada classificador fraco h_t esteja agora restrito a $\{-1,0,+1\}$, ou seja, um classificador fraco pode fazer uma predição definitiva de tal forma que o rotulo seja -1 ou $+1$, bem como poderia "abster-se", atribuindo um rotulo de 0. Na Equação 5.13, introduzimos uma função $\mu(x)$ que relaciona a predição do classificador com a frequência de classificação incorreta de uma amostra. Esta função permite ao classificador fraco "não decidir" em casos onde as amostras são ruidosas.

$$\mu(x_i) = \begin{cases} +1 & , h_t(x_i) = y_i \\ 0 & , \delta_i < k \wedge h_t(x_i) \neq y_i \\ -1 & , \delta_i \geqslant k \wedge h_t(x_i) \neq y_i \end{cases} \tag{5.13}$$

Os pesos atribuídos às amostras serão incrementados unicamente se estas atingem uma frequência de classificação incorreta igual a k, baseados na função $\mu(x)$, como se apresenta na Equação 5.14.

$$W_{t+1}(i) = \frac{W_{t+1}(i) exp(-\alpha_t \mu(x_i))}{Z_t} \tag{5.14}$$

A frequência de amostras classificadas de forma incorreta δ_i irá produzir dois *erros fragmentários*, os quais são somados para obter o erro total $\varepsilon_t = \varepsilon_{t_1} + \varepsilon_{t_2}$. Por um lado ε_{t_1} contém um *erro fragmentário* acumulado pelas amostras classificadas incorretamente com uma frequência igual a k. Por outro lado, ε_{t_2} contém o *erro fragmentário* daquelas amostras classificadas incorretamente com frequência menor que k.

Em cada iteração t do algoritmo, os *erros fragmentários* ε_{t_1} e ε_{t_2} podem ser expressos da seguinte forma:

$$\varepsilon_{t_1} = \sum_{i:\mu(i)=-1} W_t(i) \tag{5.15}$$

$$\varepsilon_{t_2} = \sum_{i:\mu(i)=0} W_t(i) \tag{5.16}$$

Então, o fator de normalização Z_t pode ser calculado da seguinte forma:

$$\begin{aligned} Z_t &= \sum_{i=1}^{N} W_t(i) e^{-\alpha \mu(x_i)} \\ &= \sum_{b \in \{-1,0,+1\}} \sum_{i:\mu_i(x_i)=b} W_t(i) e^{-\alpha_t b} \\ &= W_t e^0 + W_t e^{\alpha} + W_t e^{-\alpha} \\ &= W_0 + W_{-1} + W_{+1} \end{aligned} \tag{5.17}$$

O fator de normalização Z_t é minimizado quando a importância α_t de cada classificador h_t é:

$$\begin{aligned}\alpha_t &= \frac{1}{2}\ln\left(\frac{W_{+1}}{W_{-1}}\right) \\ &= \frac{1}{2}\ln\left(\frac{1-(\varepsilon_1+\varepsilon_2)}{\varepsilon_1}\right)\end{aligned} \tag{5.18}$$

Com esta definição da importância do classificador, temos que o fator de normalização pode ser expresso da seguinte forma:

$$Z = W_0 + 2\sqrt{W_- W_+} \tag{5.19}$$

Note-se que o conjunto de novas variáveis δ_i procura prevenir o incremento excessivo dos pesos nas amostras ruidosas, através da frequência de ocorrência de classificações erradas. Em outras palavras, os pesos unicamente são incrementados naquelas amostras de fato classificadas de forma incorreta.

Durante o processo de treinamento, a variável δ_i assume seus valores entre $\{0,1,2,\ldots,k\}$, começando em zero e sendo incrementado pela unidade quando a amostra associada é classificada incorretamente. Em contrapartida, a variável δ_i mantém o mesmo valor quando a amostra associada é classificada de forma correta. A Equação 5.20 apresenta o passo de atualização da variável δ_i.

$$\delta_i = \begin{cases} \delta_i + 1 & , h_t(x_i) \neq y_i \wedge \delta_i < k \\ 0 & , \delta_i \geq k \\ \delta_i & , \text{caso contrário} \end{cases} \tag{5.20}$$

Na seção seguinte, introduzimos nossa abordagem na tarefa de detecção de faces.

5.3 Aplicação à detecção de faces em imagens de baixa resolução

Apesar do sucesso atingido pela abordagem do Viola e Jones (2004), esta possui várias limitações. Uma delas ocorre na presença de imagens com baixa resolução, onde o algoritmo perde eficiência e a taxa de detecção de face diminui consideravelmente.

A fim de realizar a tarefa de detecção de faces em imagens de baixa resolução com eficiência, considera-se o algoritmo descrito anteriormente (seção 5.2) e executa-se a extração de caracterís-

ticas na forma eficiente e rápida com uso dos conceitos de imagem integral e classificadores em cascata.

5.3.1 Padrões Binários Locais Multi Escala em Blocos

Ojala *et al.* (1994) introduziram o Padrão Binário Local (*Local Binary Pattern - LBP*) tendo por intento descrever as texturas da superfície bidimensional em imagens. Esta abordagem de descrição esta baseada em duas medidas chamadas padrões espaciais locais e contraste da escala de cinza. O operador envolvido procura formar rótulos para os píxels da imagem, em função de um limiar de detecção, comparando o píxel central com os píxels de vizinhança de 3×3 píxels. Se a intensidade de cinza do píxel central for maior ou igual à do vizinho, então se atribui um valor de um (1) ao píxel central; caso contrário, é atribuído o valor de zero (0). Com oito píxels em torno do pixel central, existem 256 (2^8) possíveis combinações, as quais são chamadas de Padrões Binários Locais. A Figura 5.2 mostra um exemplo do cálculo do valor para um Padrão Binário Local (*LBP*), o qual pode ser usado como um descritor da textura da imagem.

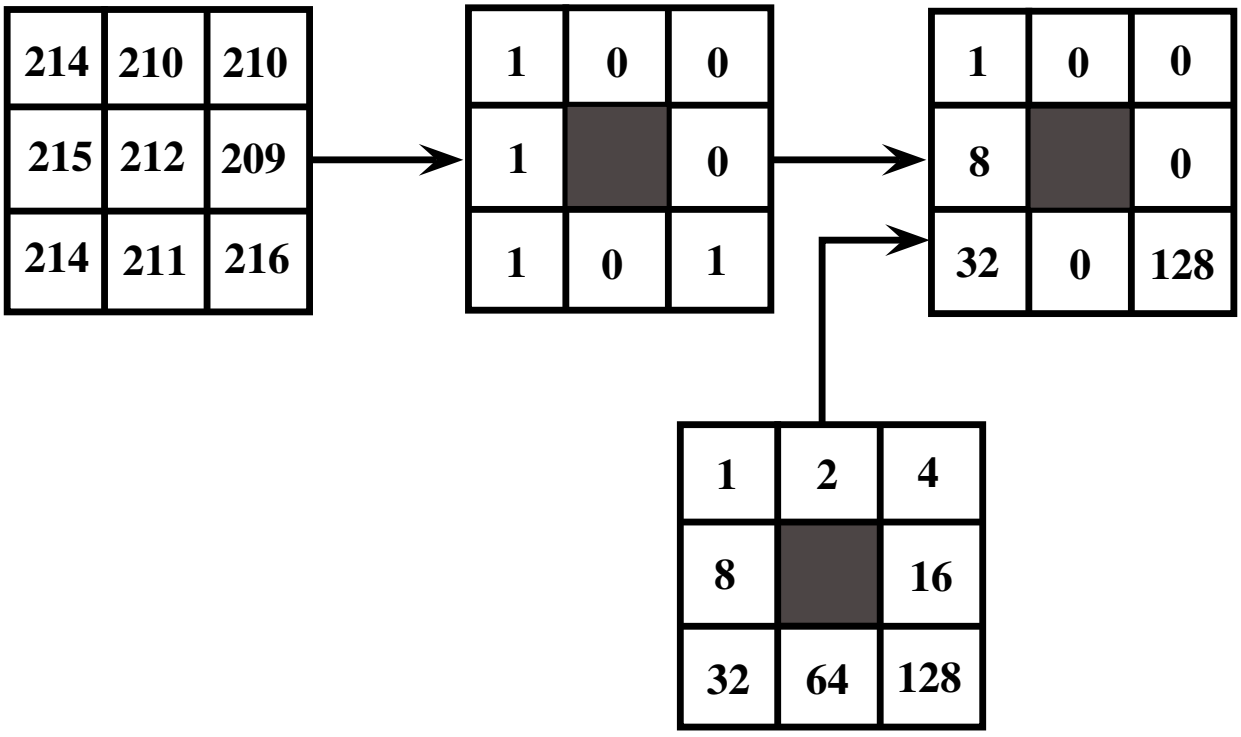


Figura 5.2: Padrão Binário Local (*LBP*) proposto por Ojala *et al.* (1994). $LBP = 1 + 8 + 32 + 128 = 169$.

Uma extensão do Padrão Binário Local (*Local Binary Pattern - LBP*), descrita por Ojala *et al.*

(2002), utiliza a relação entre um píxel e seus píxels vizinhos na discretização da imagem. Esta representação binária da textura local de uma imagem pode não ser eficiente em descrição local detalhada.

Com o propósito de superar as limitações dos Padrões Binários Locais e para aplicá-los na análise de imagens, Liao *et al.* (2007) introduziram os Padrões Binários Locais Multi Escala em Blocos (*Multi-Scale Local Binary Pattern - MB-LBP*). *MB-LBP* são vistos como uma extensão dos Padrões Binários Locais (LBP), utilizando diferentes tamanhos para descrever a região da vizinhança, ou seja, cada píxel é caracterizado por sua região de vizinhança retangular A Figura 5.3 apresenta um filtro *MB-LBP* constituído por nove retângulos, na qual cada retângulo possui 9 píxels.

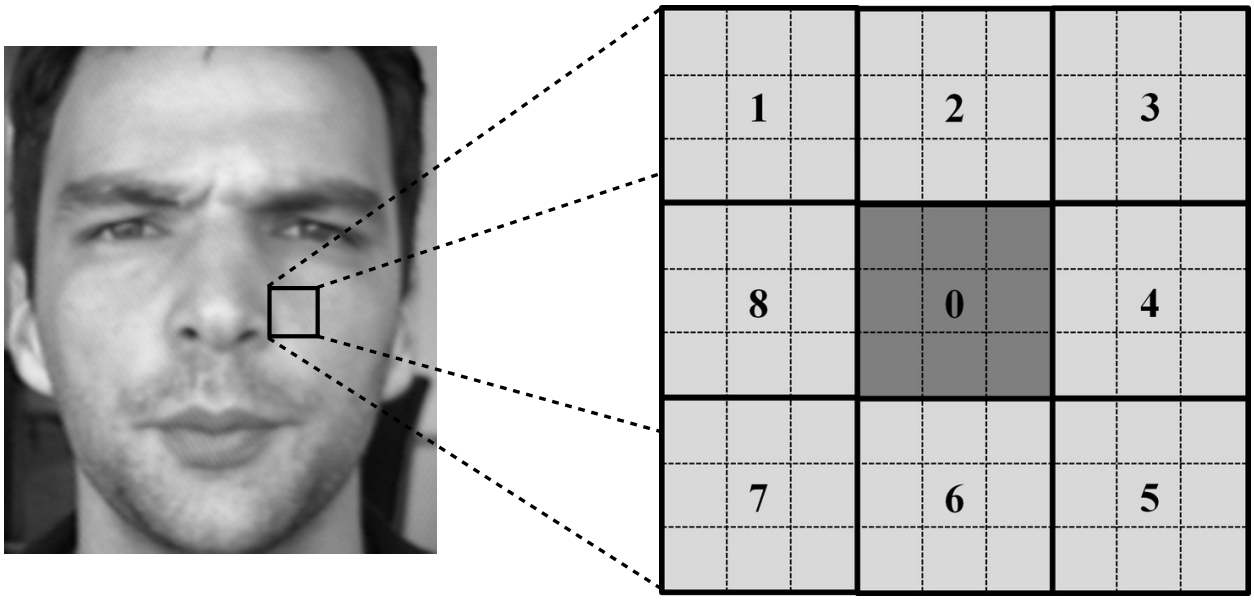


Figura 5.3: O Padrão Local Binário Multi Escala em Blocos (*MB-LBP*) proposto por (Liao *et al.* , 2007).

As características baseadas na abordagem dos *MB-LBP* são calculadas não apenas na média das intensidades de cinza no retângulo central de píxels g_c, como também na média das intensidades de cinza em cada retângulo de píxels vizinho $\{g_1,\ldots,g_{p-1}\}$, onde a notação (p,r) denota uma vizinhança de p pontos amostrados igualmente espaçados, dentro de um círculo de raio r (número de píxels). A Equação 5.21 apresenta a definição matemática do *MB-LBP*, com $p = 9$.

$$MB\text{-}LBP_{(p,r)} = \sum_{p=1}^{p-1} s(g_p - g_c)2^p, \quad s(x) = \begin{cases} 1 & , x \geq 0 \\ 0 & , x < 0 \end{cases} \tag{5.21}$$

As características baseadas no *MB-LBP* podem, também, ser calculadas de maneira veloz através da imagem integral (Zhang *et al.* , 2007a). Estas características capturam mais informação que as características baseadas nas funções de *Haar*, e fornecem uma melhor representação da imagem que os Padrões Locais Binários, posto que as características baseadas no *MB-LBP* codificam, simultaneamente, micro estruturas e macro estruturas.

Com a finalidade de utilizar este tipo de características no processo de treinamento do algoritmo *AdaBoost*, cada classificador fraco $h_i(x)$ implementa uma característica f_i através de um Padrão Binário Local Multi Escala em Blocos (*MB-LBP*). Este classificador usa um limiar θ_i e uma paridade p_i. Quando o produto da paridade e o valor da característica são menores do que o produto do limiar e a paridade, então o resultado do classificador será igual a 1; caso contrário será -1, como se apresenta na Equação 5.22.

$$h_i(x) = \begin{cases} 1 & , p_i f_i \leq p_i \theta_i \\ -1 & , \text{caso contrário} \end{cases} \tag{5.22}$$

5.3.2 Cascata de classificadores

Uma cascata de classificadores permite melhorar a velocidade de processamento das imagens. A cascata implementada neste trabalho divide-se em camadas de classificadores e cada camada é construída treinando classificadores dentro do algoritmo *AdaBoost*. Os classificadores obtidos são colocados de forma sequencial em diferentes camadas e cada uma delas contém um certo número de classificadores *fortes*.

Uma cascata possui a forma de uma árvore de decisão degenerada que percorre a imagem inteira através de sub-janelas. A ideia é que os primeiros estágios da cascata sejam mais genéricos e, ao longo da cascata, os classificadores fiquem mais específicos. Deste modo, acelera-se o processo de avaliação das amostras, uma vez que grande parte das amostras negativas serão eliminados nos primeiros estágios da cascata. Um diagrama representativo de uma cascata com três camadas é apresentado na Figura 5.4.

Uma amostra será classificada como face se for avaliada corretamente em todas as camadas da cascata de classificadores. Se, em qualquer camada, a amostra for classificada como não face, de imediato será excluída, não sendo avaliada nas camadas seguintes. A maioria dos blocos de imagem que não contém faces é rejeitada nas primeiras camadas, avaliando-se um pequeno nú-

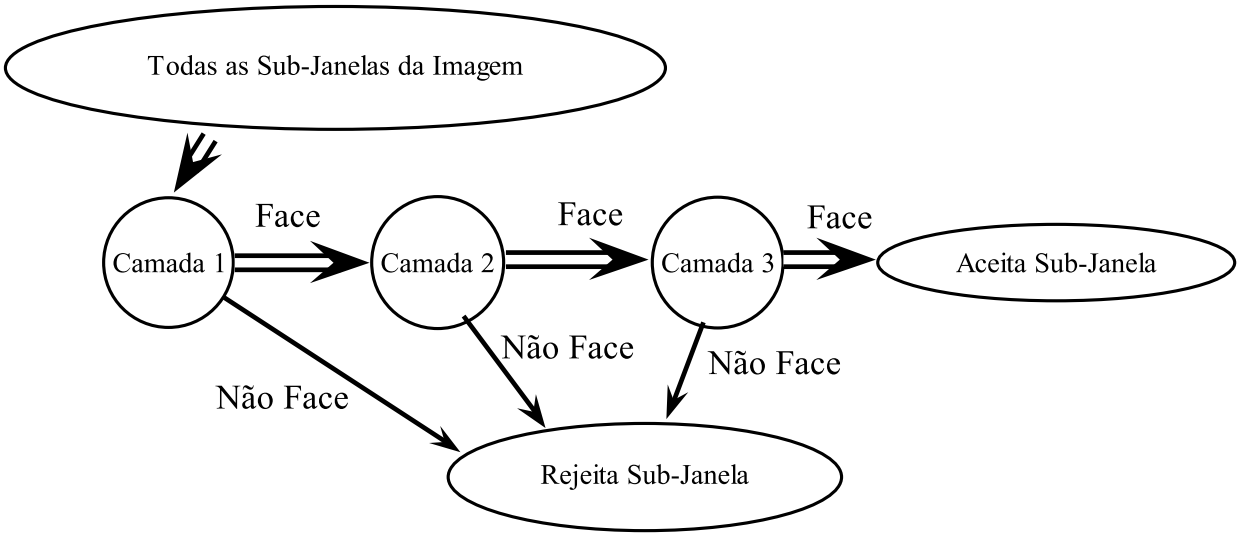


Figura 5.4: Representação de um classificador em cascata com três camadas

mero de características. As últimas camadas são treinadas para diferenciar as amostras falsas mais complexas.

Dada uma cascata de classificadores, a taxa de falsos positivos da cascata é encontrada conforme a Equação 5.23.

$$F = \prod_{i=1}^{K} f_i \tag{5.23}$$

na qual F é a taxa de falsos positivos da cascata, K é o número de classificadores e f_i é a taxa de falsos positivos de cada classificador. A taxa de detecção é encontrada através da Equação 5.24.

$$D = \prod_{i=1}^{K} d_i \tag{5.24}$$

na qual D é a taxa de detecção da cascata, K é o número de classificadores, e d_i é a taxa de detecção para cada classificador.

O Algoritmo 5.1 descreve a construção de uma cascata de classificadores (Viola e Jones , 2001). Este algoritmo permite definir uma taxa mínima de detecção, uma taxa máxima de falsos positivos em cada camada e o número mínimo de classificadores em cada camada.

O número e o tamanho das camadas são suficientes para obter um desempenho eficiente. Viola e Jones (2001) consideraram que o número de características avaliadas em imagens reais é necessariamente um processo probabilístico. Este procedimento constrói um classificador com-

Algoritmo 5.1 O pseudo-código para construir uma cascata de classificadores.

- Escolhe-se um valor para f : a taxa máxima aceitável de falsos positivos por estágio da cascata. d: a taxa mínima aceitável de detecção por estágio da cascata.
- $m =$ conjunto de amostras negativas.
- $l =$ conjunto de amostras positivas.
- $F_0 = 1; D_0 = 1;$ i=0
- Enquanto $F_i \geq F_{final}$:
  1. $i = i + 1$
  2. $n_i = 0; F_i = F_{i-1}$
  3. Enquanto $F_i \geq f \cdot F_{final}$
     - Utiliza-se l e m para treinar um classificador com n_i características usando o *AdaBoost*.
     - Avalia-se a cascata de classificadores no conjunto de validação para determinar D_i e F_i.
     - Diminui-se o limiar para o i-ésimo classificador até que a cascata atual tenha uma taxa de detecção de pelo menos $d \cdot D_{i-1}$, o que também afeta F_i.
  4. $m \leftarrow 0$.
  5. Se $F_i \geq F_{final}$
     - Avalia-se o classificador em cascata no conjunto de imagens negativas e coloca-se qualquer detecção falsa no conjunto m.

plexo com um número maior de características, alta taxa de detecção e baixas taxas de falsos positivos. Embora classificadores com muitas características requeiram mais tempo de processamento, este modelo é otimizado ajustando o número de camadas, o número de classificadores em cada camada e o limiar de cada camada. Trabalhos subsequentes demonstraram que a cascata de classificadores pode ser extremamente eficiente e rápida na detecção de objetos.

6 RESULTADOS E DISCUSSÕES

Este capítulo tem como objetivo descrever os experimentos realizados para comprovação do aperfeiçoamento do algoritmo obtido na abordagem proposta introduzindo o fator de frequência δ (Seção 5.2). Além disto, será apresentado um estudo comparativo entre a abordagem proposta (descrita no Capítulo 5) e uma seleção de trabalhos representativos, que foram apresentados na revisão bibliográfica (apresentada nos Capítulos 2 e 3).

Na Seção 6.1, será exposta uma avaliação do desempenho da abordagem proposta utilizando diferentes conjuntos de dados gerados artificialmente. A Seção 6.2 resume o experimento comparativo envolvendo diferentes níveis de resolução das imagens e imagens de diferentes tamanhos. De igual forma, apresenta-se um comparativo do desempenho entre a abordagem proposta na detecção de faces e outras abordagens existentes.

6.1 Avaliação do algoritmo *AdaBoost* proposto

Os algoritmos *Real AdaBoost*, *Modest AdaBoost* e *GentleBoost* são comparados com o algoritmo desenvolvido neste trabalho, o qual chamamos de *Frequency AdaBoost*. Como classificadores base, usamos *Decision Stumps* e árvores de classificação (*Classification and Regression Trees - CART*), devido a sua simplicidade.

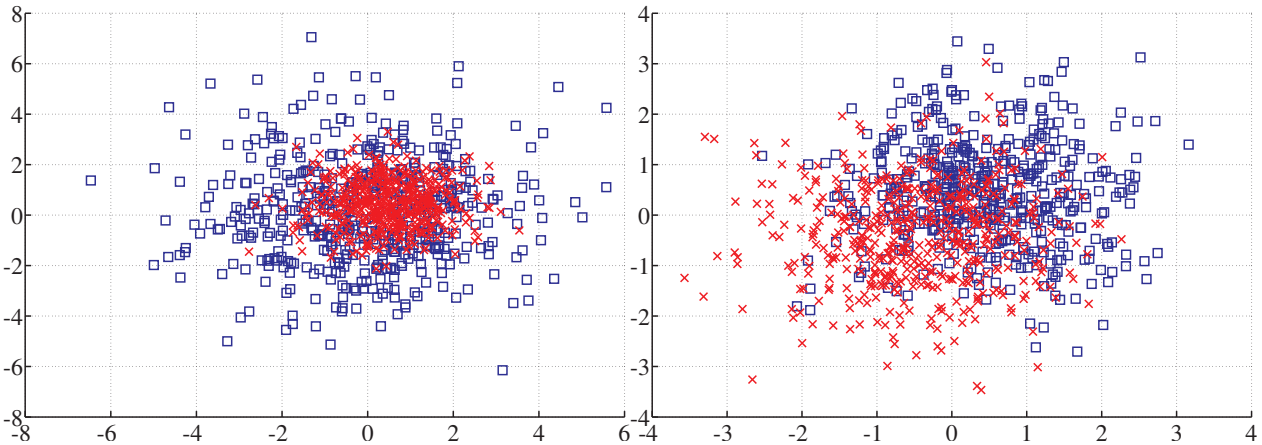
A fim de avaliar o desempenho da nossa abordagem, foram utilizados distintos conjuntos de dados. Por um lado, um grupo de conjuntos foi gerado artificialmente; outro grupo de dados foram obtidos de diferentes sites da internet. Cada conjunto de dados foi dividido aleatoriamente em conjuntos de treinamento e teste numa razão de 80:20. O valor máximo atribuído à variável δ_i é limitado pelo valor de $k = 20$.

6.1.1 Conjunto de dados *TwoNorm* e *RingNorm*

O primeiro grupo de dados consiste de dois conjuntos, que foram utilizados por Breiman (1998) na avalização do desempenho do seu algoritmo (Arc-fs).

- **RingNorm**, possui duas classes de dados. A primeira classe contém 500 dados rotulados (positivos) que possuem uma distribuição gaussiana multivariada, com média zero e matriz de covariância 4 vezes da matriz identidade. De forma similar, a segunda classe contém 500 dados rotulados (negativos), os quais seguem uma distribuição gaussiana multivariada com matriz de covariância igual à identidade e média (a,a, ..., a), com $a = \frac{2}{\sqrt{500}}$, como se apresenta na Figura 6.1a.

- **TwoNorm**, possui duas classes de dados. A primeira classe contém 500 dados rotulados (positivos), os quais seguem uma distribuição gaussiana com uma matriz de covariância igual à identidade e média (a,a, ...,a). A segunda também contém 500 dados rotulados (negativos), seguindo uma distribuição gaussiana multivariada com matriz de covariância igual à identidade e média (-a,-a, ..., -a), com $a = \frac{2}{\sqrt{500}}$, como se apresenta na Figura 6.1b.

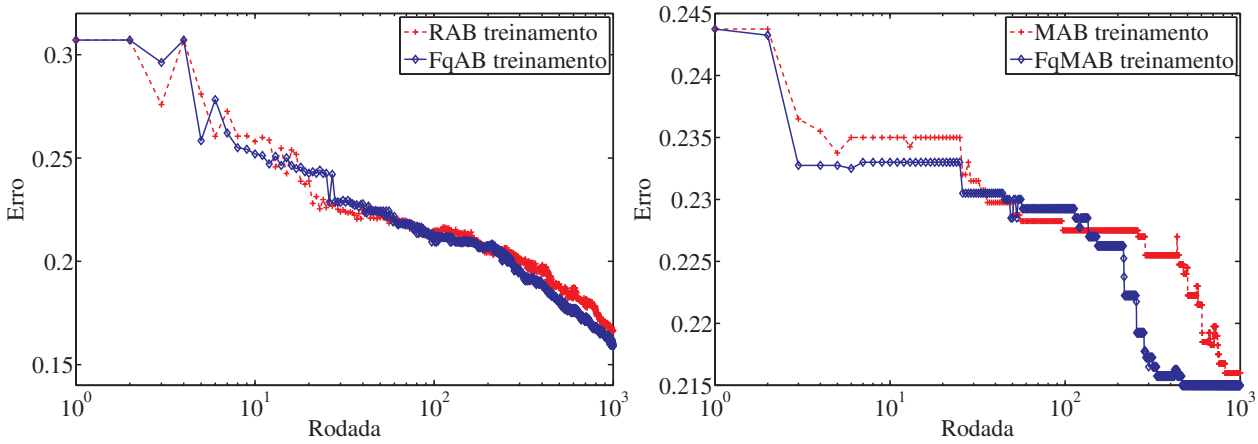


(a) *Ringnorm* (b) *Twonorm*

Figura 6.1: Conjunto de dados gerados artificialmente (Breiman , 1998).

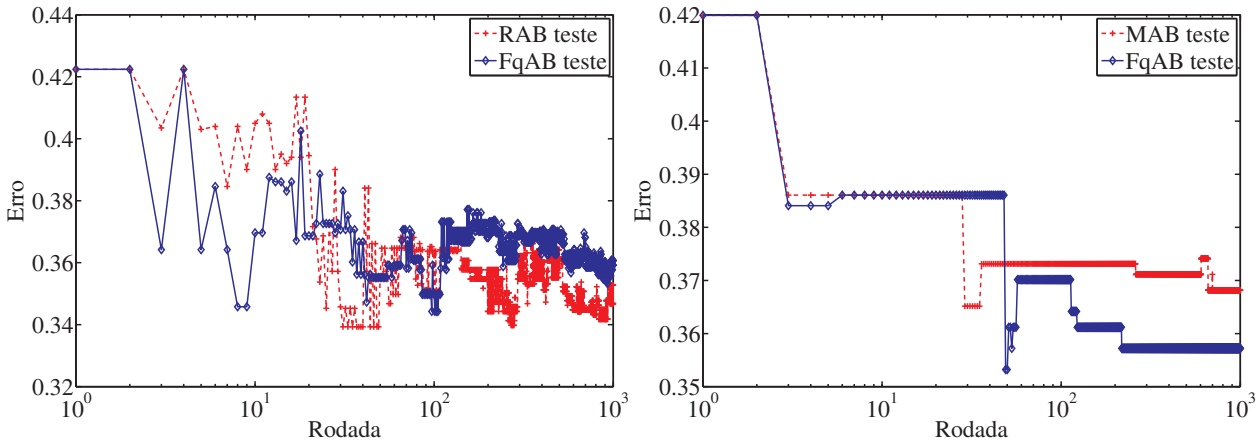
Nas Figuras 6.2a e 6.2b, apresentam-se as curvas de desempenho na fase de treinamento produzidas pelo *Real AdaBoost*, *Modest AdaBoost* e nossa abordagem, utilizando *Decision Stump* como classificadores fracos no primeiro caso, e *CART* no segundo caso. As Figuras 6.2c e 6.2d ilustram

o desempenho diante de amostras de avaliação. Comparando as curvas, é possível perceber o efeito benéfico da introdução do fator de frequência: ele leva a um menor erro no conjunto de dados de teste.



(a) Treino no *RingNorm* usando *Decision Stump*.

(b) Treino no *RingNorm* usando *CART*.



(c) Teste no *RingNorm* usando *Decision Stump*.

(d) teste no *RingNorm* usando *CART*.

Figura 6.2: Curvas de treinamento e de teste para o conjunto de dados *RingNorm*.

Os resultados obtidos para o conjunto de dados *RingNorm* são sumarizados na Tabela 6.1. A primeira coluna indica o número de alternativas ou combinação de classificadores fracos adotados. Na segunda coluna é indicado o tipo de algoritmo utilizado para o treinamento dos classificadores fracos. Nas colunas restantes, são apresentadas: a taxa de erro no treinamento, a taxa de erro no teste e, finalmente, a média de margem, cada uma com o respectivo desvio padrão obtido em cada cenário do experimento.

Tabela 6.1: Resultado de *Modest AdaBoost (MAB)* e *Frequency Modest Adaboost* (FqAB) usando *CART* como classificador fraco. Conjunto de Dados *RingNorm*.

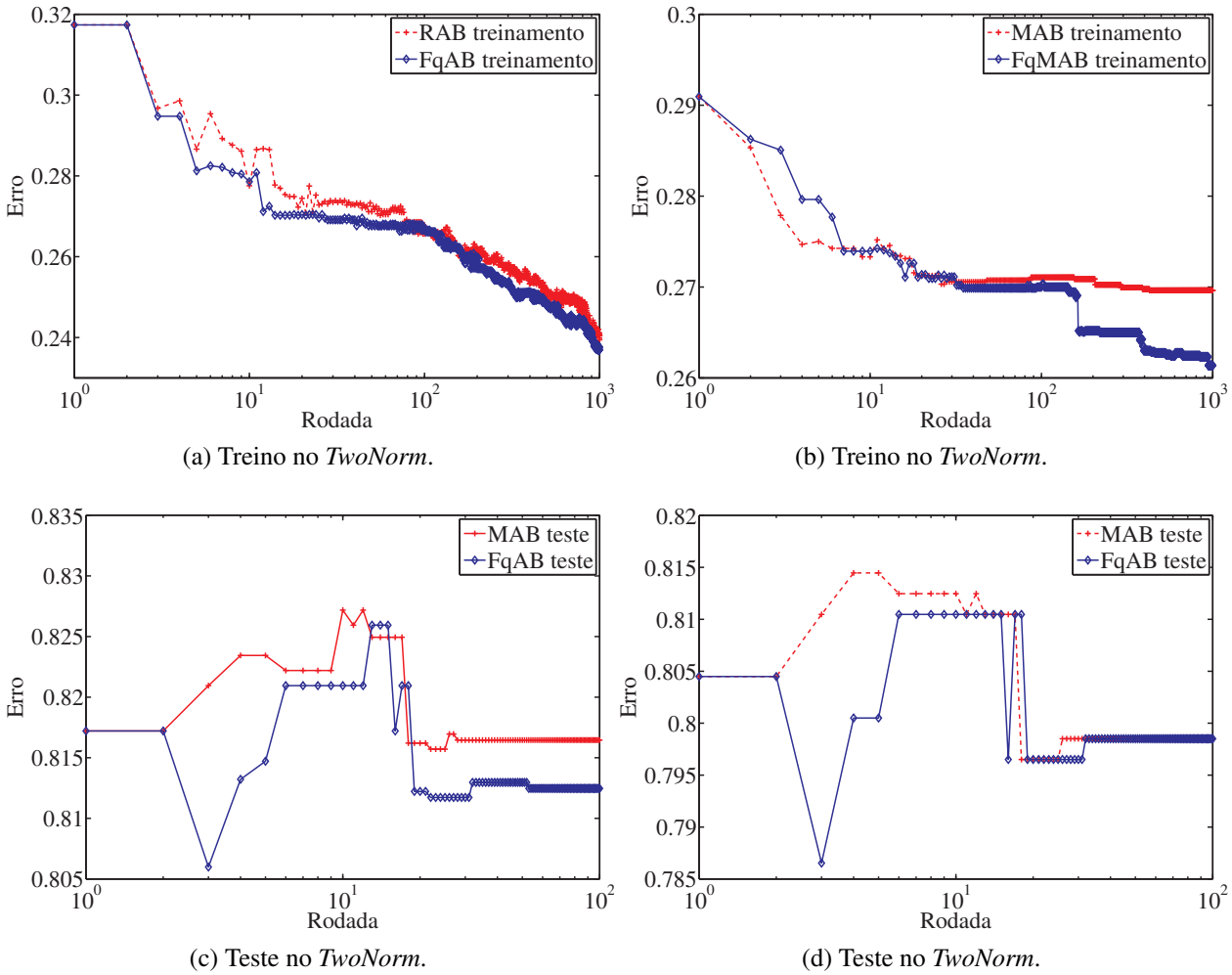
Classificadores	*Algoritmo*	*Erro de Treinamento*	*Erro de Teste*	*Margem Média*
100	MAB	**0,2019 ± 0,0042**	0,2364 ± 0,0083	**0,2284 ± 1,1348**
	FqAB	0,2027 ± 0,0028	**0,2359 ± 0,0049**	-0,5296 ± 8,6169
500	MAB	0,2119 ± 0,0029	0,3027 ± 0,0043	**0,1418 ± 0,2509**
	FqAB	**0,2077 ± 0,0058**	**0,2939 ± 0,0079**	0,0993 ± 1,9211
1000	MAB	0,2227 ± 0,0051	0,3712 ± 0,0039	0,0685 ± 0,0275
	FqAB	**0,2186 ± 0,0057**	**0,3599 ± 0,0071**	**0,105 ± 0,267**

Similarmente, o conjunto de dados *TwoNorm* (Breiman , 1998) também foi utilizado para avaliar o desempenho da nossa abordagem. A Tabela 6.2 apresenta os resultados produzidos neste experimento. A alternativa de combinação de diferentes classificadores é apresentada na primeira coluna, mostrando o desempenho para três tipos de combinação diferentes (100, 500 e 1000 classificadores fracos). Neste caso, foram avaliadas duas modificações do *AdaBoost* (*Real AdaBoost* e *Modest AdaBoost*), sendo que nossa abordagem foi introduzida de maneira separada nas duas modificações, as quais foram chamadas de *Frequency Real AdaBoost* e *Frequency Modest AdaBoost*. Nas colunas restantes são apresentadas a taxa de erro, de teste e a margem média.

Tabela 6.2: Resultados de *Modest AdaBoost (MAB)*, *Real AdaBoost(RAB)* e *Frequency Modest Adaboost* (FqAB) usando *CART* como classificador fraco. Conjunto de Dados *TwoNorm*.

Classificadores	*Algoritmo*	*Erro de Treinamento*	*Erro de Teste*	*Margem Média*
100	MAB	0,2713 ± 0,0037	0,8005 ± 0,0049	-0,0452 ± 6,7523
	FqAB	**0,2703 ± 0,0047**	**0,7997 ± 0,0043**	**1,2778 ± 16,2862**
500	MAB	0,2715 ± 0,0013	**0,8314 ± 0,0013**	-0,0595 ± 5,2109
	FqAB	**0,2661 ± 0,0045**	0,8395 ± 0,0098	**0,2565 ± 9,083**
1000	MAB	0,2701 ± 0,0011	0,817 ± 0,0009	0,1667 ± 5,1604
	FqAB	**0,2644 ± 0,0032**	**0,814 ± 0,005**	**0,3333 ± 5,4997**
100	RAB	0,2675 ± 0,0106	**0,8197 ± 0,0066**	**-0,588 ± 0,1305**
	FqAB	**0,2656 ± 0,0102**	0,8213 ± 0,0078	-0,8146 ± 0,1051
500	RAB	0,2526 ± 0,0094	**0,8611 ± 0,0062**	**-0,3733 ± 0,1166**
	FqAB	**0,2499 ± 0,0099**	0,8663 ± 0,0063	-0,5137 ± 0,1499
1000	RAB	0,2541 ± 0,0095	0,7944 ± 0,0035	-0,3287 ± 0,1336
	FqAB	**0,2505 ± 0,0098**	**0,7913 ± 0,0042**	**-0,32517 ± 0,1828**

Alguns resultados produzidos no comparativo entre o desempenho da nossa abordagem e as abordagens clássicas são apresentados na Figura 6.3. Nas Figuras 6.3a e 6.3b são apresentados os resultados obtidos no treinamento das diferentes abordagens e nas Figuras 6.3c e 6.3d são ilustrados os resultados produzidos na etapa de teste. A comparação entre as classificações está de acordo com o comportamento descrito na Tabela 6.2 e a taxa de erro na etapa de treinamento apresenta uma ligeira diminuição. Porém, esta diminuição fica mais evidente na etapa de teste, na qual as curvas de erro possuem quedas maiores.



(a) Treino no *TwoNorm*.

(b) Treino no *TwoNorm*.

(c) Teste no *TwoNorm*.

(d) Teste no *TwoNorm*.

Figura 6.3: Curvas de treinamento e de teste para o conjunto de dados *TwoNorm*

6.1.2 Conjunto de dados em espiral

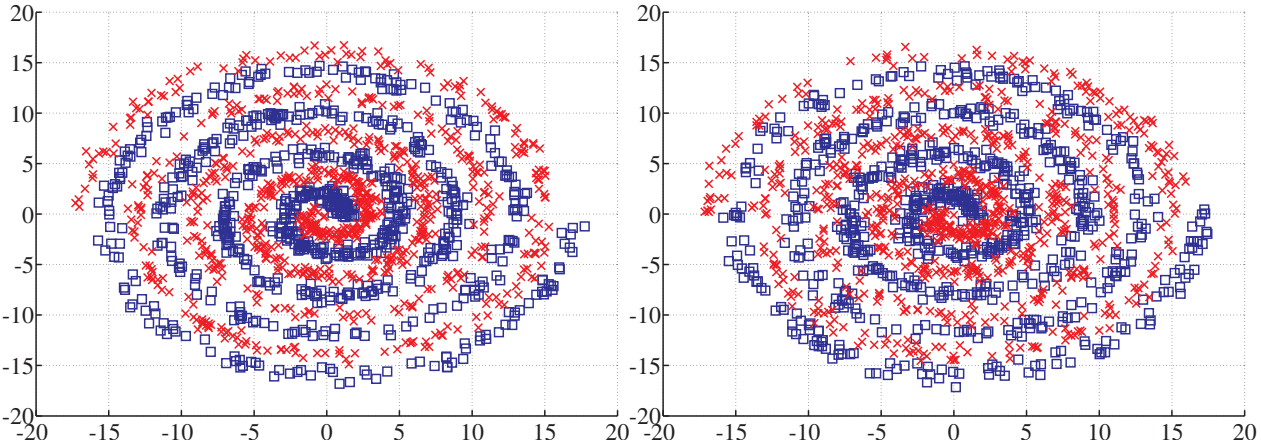
Este experimento comparativo utilizou o conjunto de dados *Espiral*, visando a analisar o desempenho da nossa abordagem em situações nas quais as amostras apresentam ruído ou existem *outliers*. Na Figura 6.4, são apresentados os conjuntos de dados gerados artificialmente. As espirais foram geradas de acordo com Yuan-Cheng e Jing-Yu (2010), considerando os números aleatórios $n_1, n_2 \in [0,N]$, ângulos aleatórios ϕ_1 e $\phi_2 \in [0, 2\pi]$, raios locais aleatórios r_1 e $r_2 \in [0,k], k > 0$ e o passo D da espiral sobre o anel. O conjunto de dados positivos foi gerado da seguinte forma:

$$\begin{cases} x_p = & (D.n_1 + 1).\cos(2\pi.n_1) + r_1 \cos\phi_1 \\ y_p = & (D.n_1 + 1).\sin(2\pi.n_1) + r_1 \sin\phi_1 \end{cases} \tag{6.1}$$

Por outro lado, o conjunto de dados negativos foi gerado assim:

$$\begin{cases} x_n = & (D.n_2 + 1).\cos(2\pi.n_2 + \pi) + r_2 \cos\phi_2 \\ y_n = & (D.n_2 + 1).\sin(2\pi.n_2 + \pi) + r_2 \sin\phi_2 \end{cases} \tag{6.2}$$

O ajuste de D e K pode determinar a existência ou não de sobreposição das amostras positivas e das amostras negativas; bem como ajustar N (o número de rotações da espiral) pode determinar a complexidade do conjunto de dados. Nas Figuras 6.4a e 6.4b, apresenta-se um conjunto de dados sem ruído e outro com 20% de ruído, respectivamente.



(a) *Espiral sem ruído* (b) *Espiral com* 20% *de ruído*

Figura 6.4: Dados gerados artificialmente, *Espiral*.

A fim de avaliar o desempenho da nossa proposta, foram geradas 1600 amostras rotuladas (800 positivas e 800 negativas). Este conjunto de dados segue a distribuição das Equações 6.1 e 6.2. Nos experimentos, utilizam-se *CART* e *Decision Stumps* como classificadores fracos e cada treinamento foi executado 10 vezes. Os resultados destes experimentos se encontram nas Tabelas 6.3 e 6.4. A organização das colunas segue a seguinte ordem: ruído inserido nos dados, algoritmo utilizado para o treinamento, taxa de erro no treinamento, taxa de erro produzida no teste e a margem média da combinação linear de classificadores.

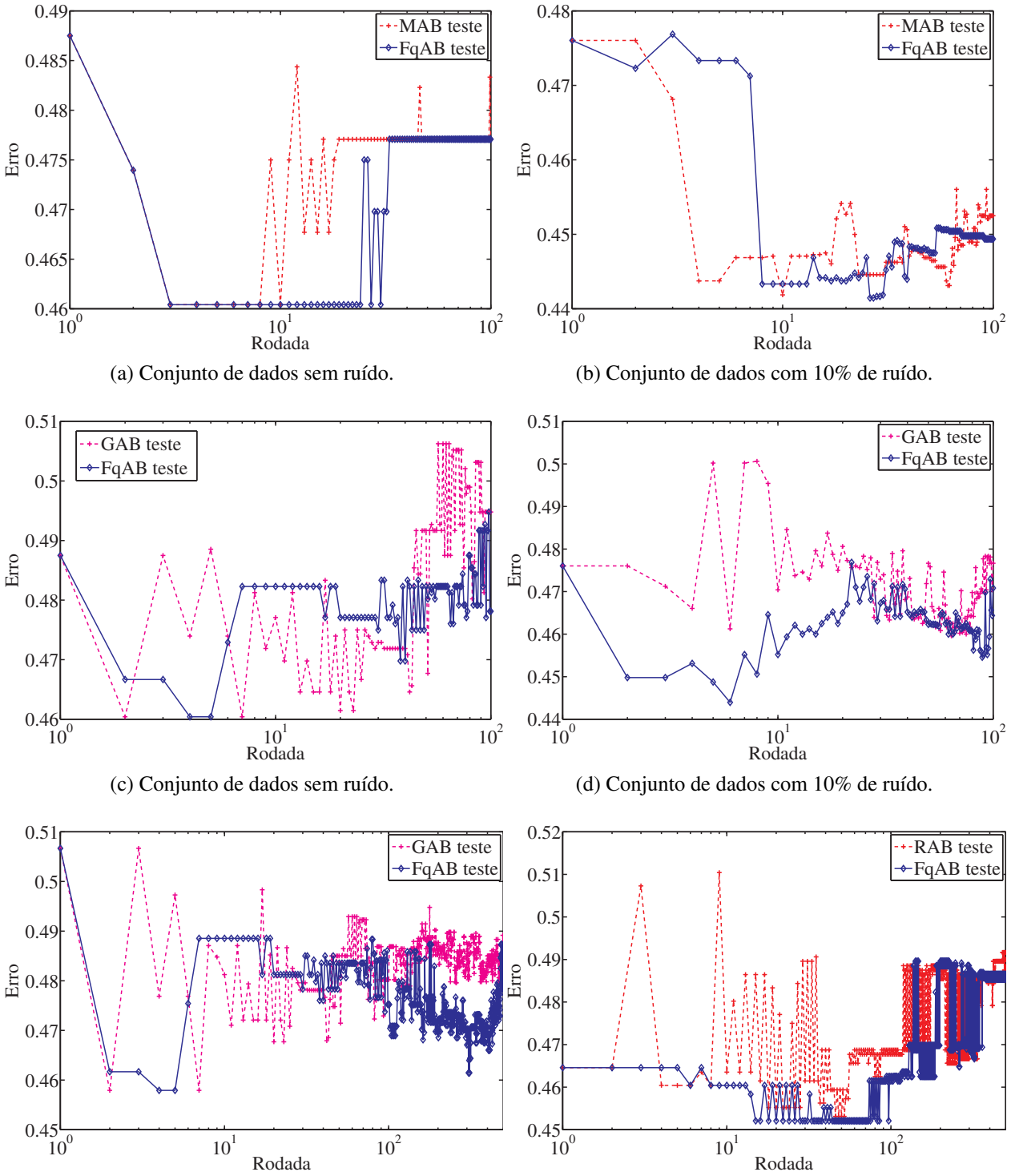
Tabela 6.3: Resultado de *Modest Adaboost* (MAB) e *Frequency Modest Adaboost* (FqAB) usando *CART*. Conjunto de Dados *Espiral*.

Ruído	*Algoritmo*	*Erro Treinamento*	*Error Teste*	*Margem Media*
0	MAB	0,2025 ± 0,0517	0,4942 ± 0,0066	**-0,0038 ± 0,0023**
	FqAB	**0,1974 ± 0,065**	**0.4922 ± 0,0055**	-0,004 ± 0,0061
10%	MAB	0,2802 ± 0,0324	**0,4579 ± 0,0102**	**0,184 ± 0,2841**
	FqAB	**0,2782 ± 0,0413**	0,4512 ± 0,0125	0,0012 ± 0,4775
20%	MAB	0,2247 ± 0,0467	0,4559 ± 0,0078	-0,5381 ± 24,6476
	FqAB	**0,2222 ± 0,0563**	**0,4559 ± 0,0062**	**-0,0553 ± 0,9179**

Tabela 6.4: Resultado de *Real Adaboost* (RAB) e *Frequency Real Adaboost* (FqAB) usando *decision stump*. Conjunto de Dados *Espiral*.

Ruído	*Algoritmo*	*Erro Treinamento*	*Error Teste*	*Margem Media*
0	RAB	0,2526 ± 0,0422	**0,4862 ± 0,0038**	0,0009 ± 0,0032
	FqAB	**0,2527 ± 0,0495**	0.4886 ± 0,0048	**0,0011 ± 0,0031**
10%	RAB	0,296 ± 0,0305	0,4697 ± 0,0073	0,0 ± 0,004
	FqAB	**0,2921 ± 0,042**	**0,468 ± 0,0075**	**0,0099 ± 0,0033**
20%	RAB	**0,2666 ± 0,0365**	0,4598 ± 0,0069	0,0069 ± 0,005
	FqAB	0,2761 ± 0,0437	**0,4531 ± 0,0047**	**0,0082 ± 0,0068**

Alguns exemplos de curvas produzidas no experimento de avaliação de desempenho com o uso do fator de frequência são apresentados na Figura 6.5. As Figuras 6.5a e 6.5b comparam o desempenho da nossa abordagem frente ao desempenho do *Modest AdaBoost*, utilizando dois tipos de classificadores fracos diferentes (*Decision Stumps e CART*). As Figuras de 6.5c a 6.5e apresentam o desempenho da nossa abordagem frente à do *GentleBoost*. Nos três casos, é utilizado *CART* como classificador fraco. Finalmente, a Figura 6.5f apresenta o desempenho da nossa abordagem frente ao *Real AdaBoost* utilizando *Decision Stumps* como classificador fraco.



(a) Conjunto de dados sem ruído.

(b) Conjunto de dados com 10% de ruído.

(c) Conjunto de dados sem ruído.

(d) Conjunto de dados com 10% de ruído.

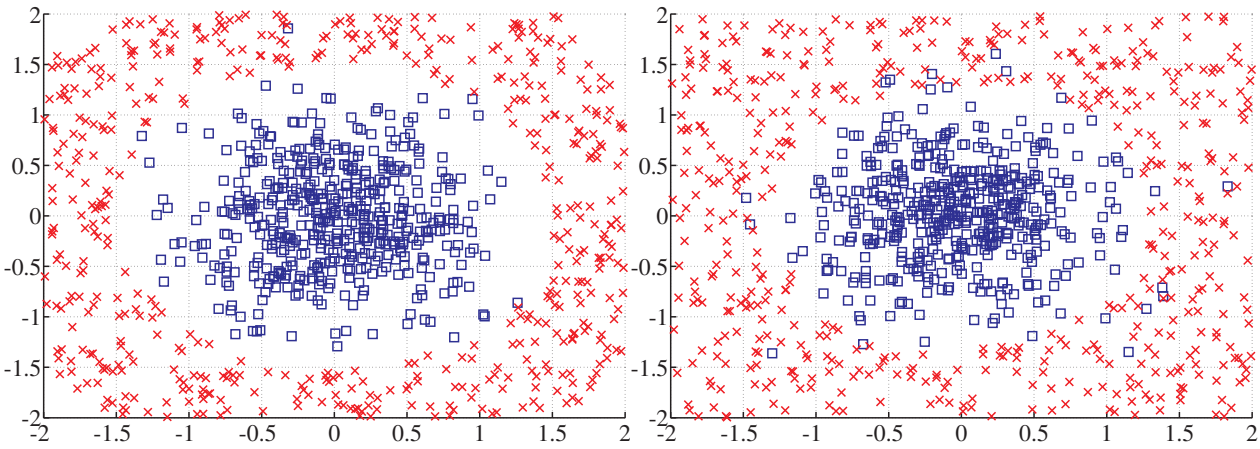
(e) Conjunto de dados sem ruído.

(f) Conjunto de dados com 10% de ruído.

Figura 6.5: Curvas de teste para o conjunto de dados *Espiral*

6.1.3 Conjunto de dados em círculo e gaussiana

De maneira similar às subseções 6.1.1 e 6.1.2, a fim de mostrar a diferença entre os algoritmos de *AdaBoost* clássicos e nossa abordagem, construímos um conjunto de dados de duas dimensões. Este conjunto de dados contém 1000 amostras rotuladas (500 positivas e 500 negativas). As amostras rotuladas como positivas seguem uma distribuição gaussiana multivariada ($\mathcal{N}(0, 0.25)$) e as amostras rotuladas como negativas seguem um círculo com raio 2.0, como se apresenta na Figura 6.6.



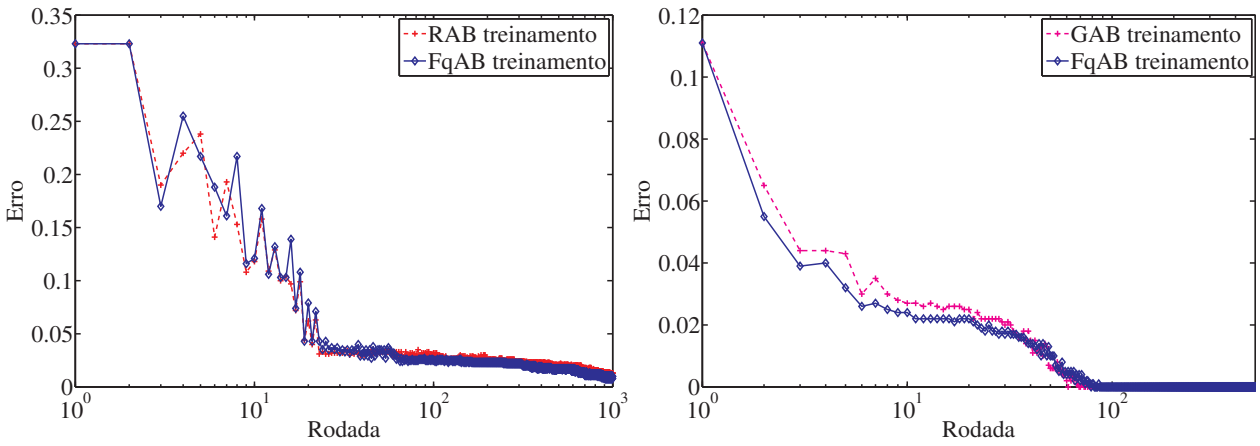
(a) *Gaussiana e círculo sem ruído* (b) *Gaussiana e círculo com* 20% *de ruído*

Figura 6.6: Dados gerados artificialmente, círculo e gaussiana.

Este experimento consistiu na aplicação da nossa abordagem frente aos algoritmos *GentleBoost* e *Modest AdaBoost*. A Tabela 6.5 apresenta alguns resultados produzidos neste experimento. Os ruídos inseridos nos dados são de três níveis: 10%, 20% e 30%, os quais são indicados na primeira coluna. De forma similar às tabelas anteriores, o algoritmo utilizado é apresentado na segunda coluna e nas colunas restantes é apresentado o erro de treinamento, o erro de teste e a margem média, com as respetivas desviações padrão. Neste experimento, o classificador fraco utilizado foi o *CART* e o número de rodadas foi de 1000. Na Figura 6.7 são apresentadas as comparações das curvas de erro no treinamento. Na Figura 6.7a são utilizados *Decision Stumps* como classificadores fracos, a combinação linear de classificadores possui mil classificadores fracos ótimos. Similarmente, na Figura 6.7b também são utilizados mil classificadores fracos; porém, do tipo *CART*. No primeiro caso fazemos a comparação da nossa abordagem com o algoritmo *Real AdaBost*, enquanto que no segundo caso comparamos com o algoritmo *GentleBoost*.

Tabela 6.5: Resultado de *GentleBoost* (GAB) e *Frequency GentleBoost* (FqAB) usando *CART*. Conjunto de Dados *circle*.

Ruído	*Algoritmo*	*Erro Treinamento*	*Error Teste*	*Margem Media*
0	GAB	0,0003 ± 0,0034	0,1766 ± 0,0041	-0,1354 ± 0,0107
	FqGAB	**0,0003 ± 0,0035**	**0,1764 ± 0,0041**	**-0,1272 ± 0,0084**
10%	GAB	0,0003 ± 0,0043	**0,2268 ± 0,0047**	**0,106 ± 0,0076**
	FqAB	**0,0003 ± 0,0048**	0,2276 ± 0,0047	0,1083 ± 0,0085
20%	GAB	0,0005 ± 0,0041	0,1954 ± 0,0042	0,05827 ± 0,0083
	FqAB	**0,0005 ± 0,0042**	**0,1554 ± 0,0052**	**0,0543 ± 0,0059**
30%	GAB	0,0026 ± 0,0089	0,1646 ± 0,0078	**-0,1727 ± 0,0088**
	FqAB	**0,0024 ± 0,0081**	**0,1633 ± 0,0076**	-0,1798 ± 0,0087

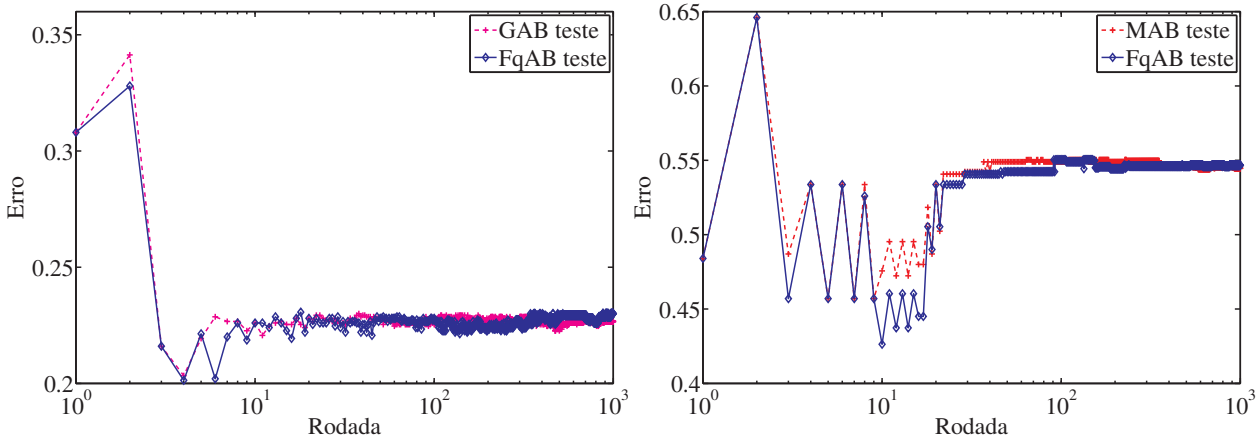


(a) Conjunto de dados com 30% de ruído usando *Decision Stump*.

(b) Conjunto de dados com 30% de ruído usando *CART*.

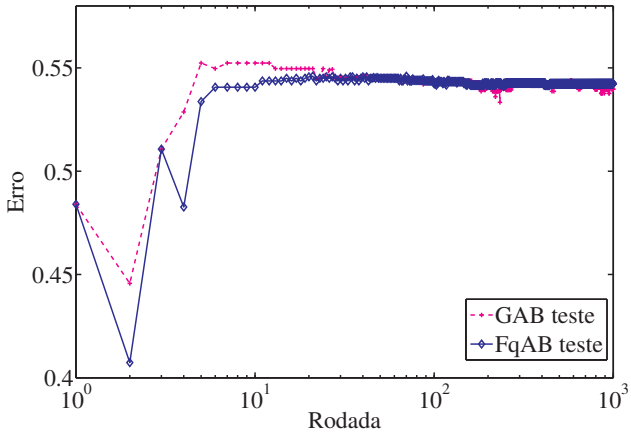
Figura 6.7: Curvas de treinamento usando o conjunto de dados círculo e gaussiana.

Na Figura 6.8, são apresentadas algumas curvas produzidas no teste do desempenho, utilizando o conjunto de dados círculo e gaussiana. A Figura 6.8a avalia o desempenho da nossa abordagem quando as amostras apresentam 10% de ruído, utilizando *GentleBoost* e classificadores do tipo *CART*. As Figuras de 6.8b a 6.8d apresentam as comparações do desempenho da nossa abordagem no conjunto de dados com 20% de ruído, utilizando *Modest AdaBoost*, *GentleBoost* e *Real AdaBoost* respectivamente. Finalmente, as Figuras 6.8e e 6.8f utilizam *GentleBoost* e *Real AdaBoost* para um conjunto de amostras com 30% de ruído. Comparando estas curvas, as quais são baseadas na informação presente na Tabela 6.5; intuitivamente, pode-se inferir que a aplicação do fator de frequência melhora o desempenho dos algoritmos *AdaBoost*.

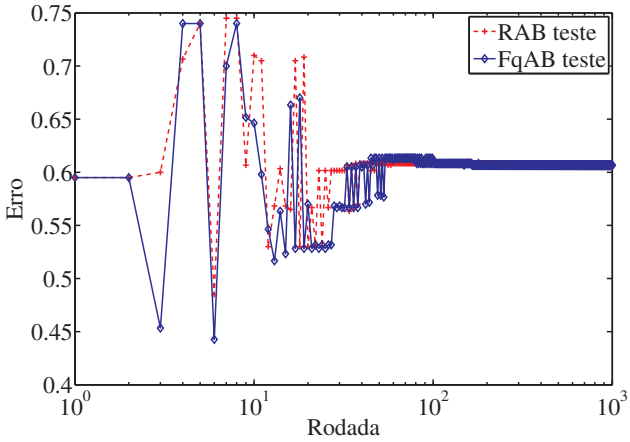


(a) Conjunto de dados com 10% de ruído.

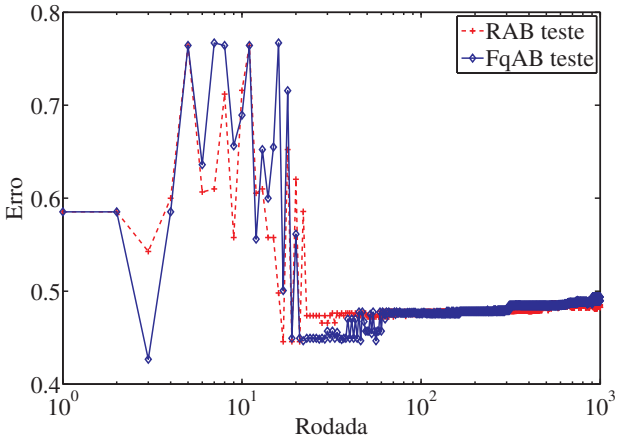
(b) Conjunto de dados com 20% de ruído.



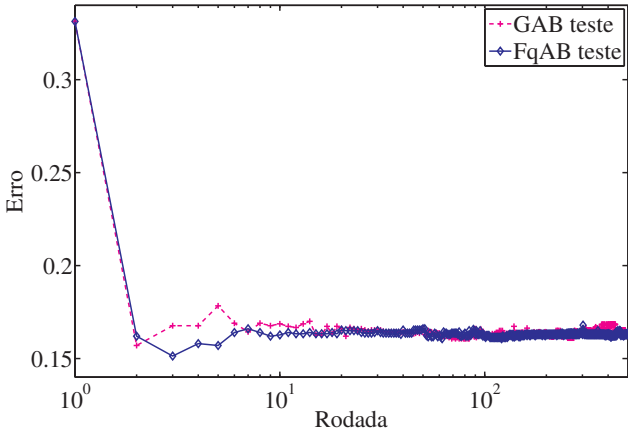
(c) Conjunto de dados com 20% de ruído.



(d) Conjunto de dados com 20% de ruído.



(e) Conjunto de dados com 30% de ruído.



(f) Conjunto de dados com 30% de ruído.

Figura 6.8: Curvas de teste para o conjunto de dados círculo e gaussiana

6.1.4 Conjunto de dados do UCI

Do mesmo modo, outros 12 conjuntos de dados (Chang e Lin , 2011) são utilizados. Cada conjunto de dados é dividido num conjunto de treinamento e outro de teste, numa proporção de 80:20. O valor máximo atribuído à variável δ_i é limitado pelo valor de $k = 20$. Os detalhes destes conjuntos de dados são apresentados na Tabela 6.6.

Tabela 6.6: Descrição do conjunto de dados (Chang e Lin , 2011)

Dados	*# características*	*# amostras*	*Dados*	*# características*	*# amostras*
australian	14	690	**liver**	6	345
b-cancer	10	683	**sonar**	60	208
diabetes	8	768	**splice**	60	1000
fourclass	2	862	**svmguide1**	4	3089
heart	13	270	**svmguide3**	21	1243
ionosphere	34	347	**spam**	1899	4000

Todos estes conjuntos de dados foram rotulados como positivos ou negativos ($[-1,1]$) e todos os testes foram rodados para 1000 iterações em 10 vezes contínuas. Os resultado produzidos neste experimento são resumidos na Tabela 6.7. A primeira coluna indica os 12 conjuntos de dados utilizados no treinamento dos classificadores. Na segunda coluna, é indicado o algoritmo utilizado. Nas colunas restantes, apresentamos: o erro médio obtido após 10 repetições do treinamento, o erro produzido na etapa de teste e a margem média produzida durante o treinamento. Cada valor é acompanhado pelo respectivo desvio padrão.

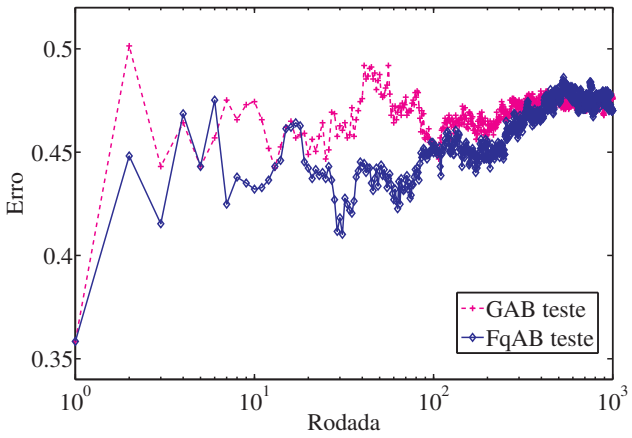
A Figura 6.9 apresenta distintas comparações das curvas de erro no teste. Nas Figuras 6.9a, 6.9c, 6.9d e 6.9e, se apresentam os desempenhos da nossa abordagem frente ao *GentleBoost* utilizando classificadores do tipo *CART*. Similarmente, na Figura 6.9f a nossa abordagem é comparada com o *Modest AdaBoost*, utilizando classificadores do mesmo tipo. Na Figura 6.9b, o algoritmo *Real AdaBoost* é comparado quando introduzimos o fator de frequência. Em termos de erro de treinamento e rapidez de convergência: a abordagem proposta é melhor em 76,9% dos conjuntos de dados, na etapa de treinamento, e em 92,3% dos conjuntos de dados, na etapa da de teste. Olhando o resultados dos erros de treinamento da Tabela 6.7, podemos apreciar que a abordagem proposta possui um desempenho superior na maioria das rodadas, perdendo por muito ligeiramente apenas dois conjuntos de dados na etapa de treinamento e apenas em um na etapa de teste.

Tabela 6.7: Resultado de *GentleBoost* (GAB), *Modest AdaBoost* (MAB) , *Real AdaBoost* (RAB) e *Frequency AdaBoost* (FqAB). Conjuntos de Dados do UCI.

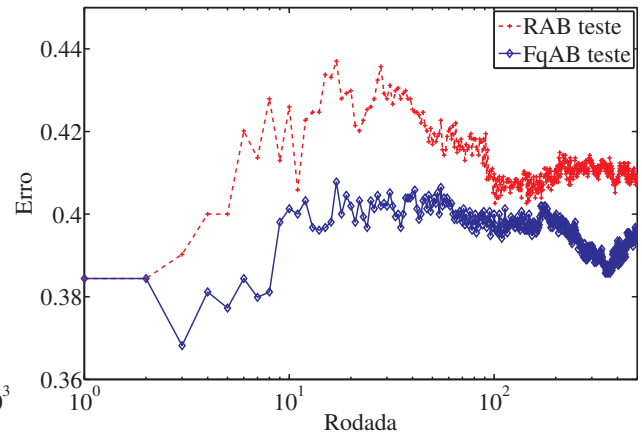
Dados	*Algoritmo*	*Erro de Treinamento*	*Erro de Teste*	*Margem Média*
australian	MAB	0,0843 ± 0,0123	0,8496 ± 0,0178	-90,81 ± 11,2856
	FqAB	**0,0746 ± 0,0141**	**0,8385 ± 0,0157**	**-0,3987 ± 15,1514**
bcancer	GAB	0,0002 ± 0,0025	0,4716 ± 0,0071	-0,3378 ± 8,4556
	FqAB	**0,0002 ± 0,0024**	**0,4669 ± 0,0143**	**-0,2393 ± 3,91**
diabetes	MAB	0,1799 ± 0,0107	0,3827 ± 0,0042	-0,1382 ± 0,0652
	FqAB	**0,1773 ± 0,0102**	**0,3711 ± 0,0041**	**-0,1834 ± 0,2090**
fourclass	GAB	0,0005 ± 0,0081	**0,6411 ± 0,003**	0,1061 ± 0,0273
	FqAB	**0,0005 ± 0,0069**	0,6412 ± 0,0045	**0,1066 ± 0,0321**
german num.	GAB	**0,0152 ± 0,0412**	0,2997 ± 0,0112	-2,7526 ± 66,6704
	FqAB	0,016 ± 0,0403	**0,2901 ± 0,0054**	**-0,9414 ± 12,707**
heart	GAB	**0,014 ± 0,0122**	0,4521 ± 0,0139	-1,0384 ± 41,1325
	FqAB	0,015 ± 0,012	**0,436 ± 0,0185**	**-0,2719 ± 9,3181**
ionosphere	GAB	0,0003 ± 0,0038	0,2363 ± 0,0165	-0,1309 ± 0,0045
	FqAB	**0,0003 ± 0,0039**	**02319 ± 0,0154**	**-0,1266 ± 0,0073**
liver	MAB	0,0768 ± 0,0429	0,5076 ± 0,0196	**0,0589 ± 15,9791**
	FqAB	**0,072 ± 0,0429**	**0,469 ± 0,0255**	-0,1006 ± 5,8698
sonar	MAB	**0,0014 ± 0,0122**	0,4142 ± 0,0017	-0,0575 ± 4,6096
	FqAB	0,0017 ± 0,0137	**0,4119 ± 0,0175**	**0,1609 ± 12,87**
splice	MAB	0,0780 ± 0,0148	0,3800 ± 0,0064	0,1533 ± 0,0719
	FqAB	**0,0771 ±0,0162**	**0,3797 ± 0,0061**	**0,1563 ± 0,0741**
svmguise1	MRAB	0,0241 ± 0,002	0,6028 ± 0,0062	-0,2534 ± 1,2238
	FqAB	**0,021 ± 0,0034**	**0,597 ±0,0061**	**0,4217 ±5,7635**
svmguide3	MAB	0,1267 ± 0,0106	0,6491 ± 0,0489	-0,0226 ± 0,0777
	FqAB	**0,1251 ± 0,0126**	**0,6248 ± 0,0292**	**1,3833 ± 45,0335**
w1a	GAB	0,0052 ± 0,0029	0,1067 ± 0,0188	0,0 ± 0,0
	FqAB	**0,0052 ±0,0027**	**0,1063 ± 0,0197**	**0,0 ± 0,0**

6.2 Avaliação do algoritmo proposto para Detecção de Faces

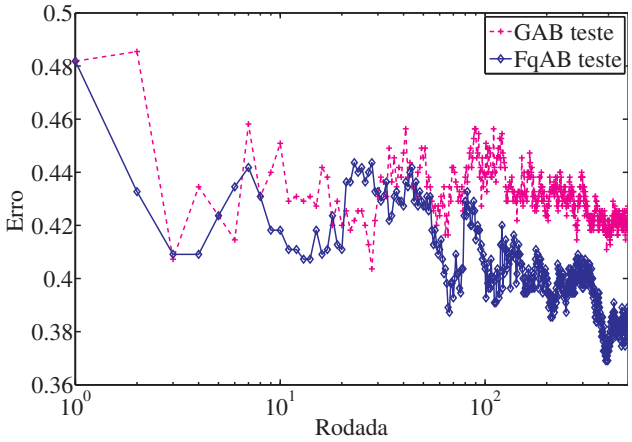
A tarefa de detecção de faces é um processo computacional que determina a localização e o tamanho de uma ou várias faces em imagens arbitrárias, onde o procedimento detecta características faciais e ignora qualquer outra coisa (prédios, carros ou corpos). Viola e Jones (2001) propuseram um algoritmo baseado no algoritmo *AdaBoost*, conseguindo ótimos resultados. No entanto, esse método também possui várias deficiências e a tarefa de detecção de faces tem sido afetada por deficiências em acurácia e rapidez.



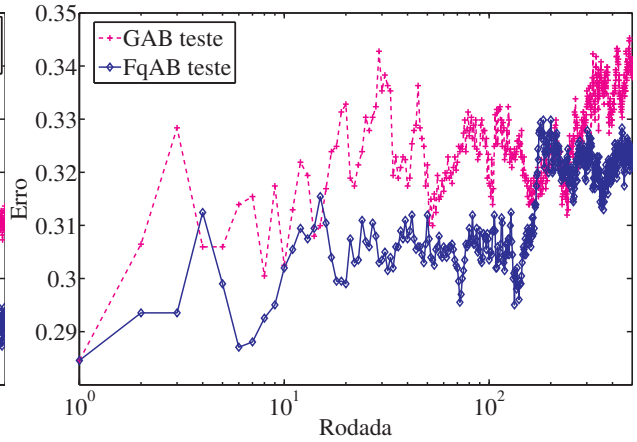
(a) Conjunto de dados *breast cancer*.



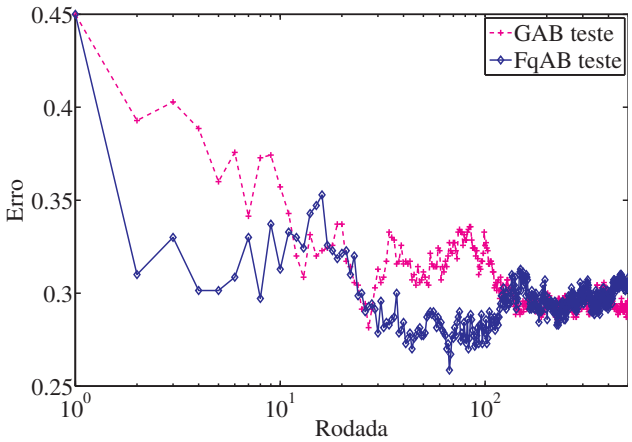
(b) Conjunto de dados *diabetes*.



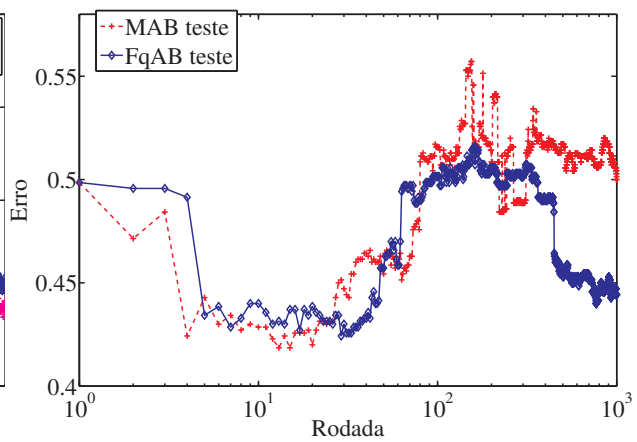
(c) Conjunto de dados *heart*.



(d) Conjunto de dados *german number*.



(e) Conjunto de dados *ionosphere*.



(f) Conjunto de dados *liver disorder*.

Figura 6.9: Curvas de teste para o conjunto de dados UCI (Chang e Lin , 2011).

Nesse experimento foram utilizadas um total de 8094 amostras no treinamento, as quais variam em idade, gênero, etnia entre outros. O conjunto de imagens foi criado a partir de uma coleção de imagens reunidas de diferentes conjuntos de dados, entre os quais podemos mencionar os seguintes: (PICS , 2011), (BioID , 2011), (UMIST , 2011), (FEI , 2011) e (Samaria *et al.* , 1994). As imagens estão divididas da seguinte forma: 2526 amostras positivas (faces) e 5568 amostras negativas (não faces). O ambiente de desenvolvimento foi uma CPU de 2.93GHz com 7935 MB de memória, a aplicação foi implementada na linguagem de programação C++, na plataforma Linux usando o compilador G++, com ajuda da biblioteca aberta para uso acadêmico e comercial de Visão Computacional *OpenCV (Open Source Computer Vision Library)* (Intel , 2011), desenvolvida pela Intel desde 1999. Na tarefa de detecção de objetos, o *OpenCV* utiliza o método de Viola-Jones, o qual é usado neste trabalho para comparar frente a nossa abordagem.

Este teste procura avaliar o desempenho da nossa abordagem frente ao algoritmo *GentleBoost*. Para isso, foram escolhidas características do tipo *MB-LBP* na maioria dos casos, e apenas no primeiro caso com propósitos de avaliação foram utilizados características do tipo *Haar* nos classificadores fracos. Construímos uma cascata de classificadores de 15 camadas e, arbitrariamente, o valor máximo atribuído à variável δ_i é limitado pelo valor de $K = 2$, e consideramos *Decision Stumps* dentro do *AdaBoost*. A fim de realizar o treinamento e teste, cada conjunto de dados foi dividido aleatoriamente em conjuntos de treinamento e teste, numa razão de 80 : 20.

6.2.1 Avaliação do desempenho num ambiente adequado

Neste experimento, consideramos um ambiente adequado dentro das imagens, tendo as faces isoladas e sem perda de resolução; além disso, as características utilizadas nos classificadores fracos foram do tipo *Haar*. O conjunto de amostras de teste foram obtidas do mesmo conjunto das imagens de treinamento; porém, de pessoas diferentes. Todas estas imagens foram redimensionadas para 24×24 pixeis, constituindo um conjunto de 500 imagens de faces e 2000 imagens de não faces.

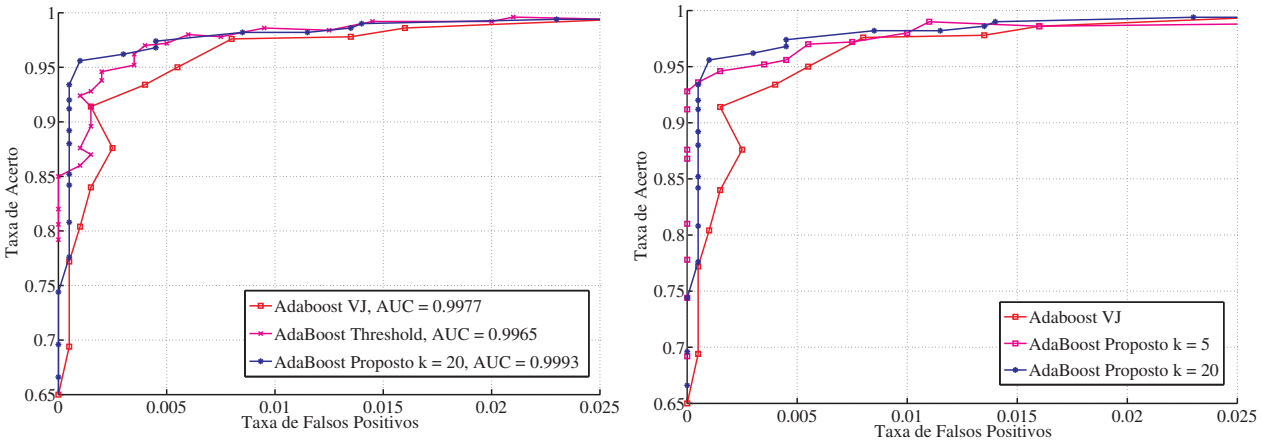
A Tabela 6.8, apresenta os resultados produzidos neste experimento. Na primeira coluna indicamos as abordagens utilizadas para comparar o desempenho da nossa abordagem. Nas seguintes colunas são indicados os resultados obtidos na etapa de teste: o número de amostras etiquetadas como face (Faces), o número de amostras classificadas de forma correta e a taxa de acerto (Acertos). Do mesmo modo, são indicados também o número de amostras etiquetadas negativamente (Não Faces), a taxa de falsos positivos e, finalmente, a acurácia produzida pelas distintas aborda-

gens.

Tabela 6.8: Resultados produzidos num ambiente produzido.

Parâmetros / *Método*	*Faces*	*Acertos*	*Não Faces*	*Falsos Positivos*	*Acurácia*
Método Clássico	500	420 = 84,0%	2000	14	96,64 %
Adaboost (Li *et al.* , 2010)	500	479 = 95,9%	2000	16	98,01 %
Adaboost Proposto	500	484 = 96,8%	2000	18	99,04 %

As curvas *ROC* apresentadas na Figura 6.10 correspondem a detecções obtidas com o uso do fator de frequência nas amostras classificadas de forma incorreta. Na Figura 6.10b são apresentadas três curvas *ROC* correspondentes à abordagem proposta por (Li *et al.* , 2010), Viola e Jones (2004) e a nossa proposta. De forma análoga, na Figura 6.10b são apresentados os resultados com dois valores diferentes de frequência máxima na abordagem proposta ($k = 5$ e $k = 20$).



(a) *Comparação Viola-Jones, (Li et al. , 2010) e Proposto*

(b) *Comparação Viola-Jones e Proposto*

Figura 6.10: Curvas ROC comparando o desempenho dos classificadores para o AdaBoost clássico, AdaBoost (Li *et al.* , 2010) e AdaBoost proposto.

Com base nos dados das curvas *ROC*, apresentadas nos gráficos das Figuras 6.10a e 6.10b, é possível representar a relação entre a taxa de falsos positivos (FPR) e a taxa de detecção (TP). No processo de detecção, é muito importante a análise destes gráficos, pois, dependendo do erro aceito no classificador final, terá que se ajustar o limiar de decisão. Com a análise destes gráficos somos capazes de prever qual a taxa de erro que se irá obter.

6.2.2 Avaliação em imagens em baixa resolução mediante redimensionamento

Neste experimento, em contraste com o anterior, foram utilizados *MB-LBP* na extração de características, a fim de avaliar o desempenho da nossa abordagem. Analogamente, comprovamos a obtenção de melhores resultados na detecção de faces, principalmente em imagens que perderam resolução no processo de redimensionamento. Para isso, o conjunto de imagens de teste foram obtidas do *BAO dataset* (Frischholz , 2012), as quais são redimensionadas e agrupadas em sete grupos diferentes, baseados na porcentagem de diminuição. O redimensionamento se faz utilizando um filtro de alta qualidade (*Antialias*).

A Tabela 6.9 apresenta os resultados obtidos pela abordagem clássica (*GentleBoost*). Na primeira coluna é indicada a porcentagem na qual a imagem foi redimensionada; no resto das colunas se indicam o número de faces utilizadas no teste, o número de faces que foram classificadas corretamente em cada situação, os falsos positivos associados à taxa de detecção e, finalmente, a taxa de acerto.

Tabela 6.9: Resultados produzidos pelo *GentleBoost* em imagens redimensionadas.

Parâmetros / *Redução*	*Faces*	*Acertos*	*Falsos Positivos*	*Taxa de Acerto*
10%	349	0	0	0,0 %
20%	349	0	1	0,0 %
30%	349	33	6	9,45 %
40%	349	101	8	29,93 %
50%	349	133	14	38,10 %
60%	349	181	20	51,86 %
70%	349	217	32	62,17 %

Similarmente, na Tabela 6.10, são apresentados os resultados associados à nossa abordagem. Na primeira coluna, do mesmo modo que na Tabela anterior, são indicados os porcentagens de redimensionamento. No resto das colunas indicamos os valores obtidos no teste do conjunto de imagens. Note-se que em ambos casos a taxa de acerto diminui, enquanto as dimensões das imagens são diminuídas.

Na Figura 6.13, apresenta-se uma comparação entre o algoritmo proposto *Frequency AdaBoost* e o *GentleBoost*. As curvas representam a relação entre a porcentagem de redimensionamento

Tabela 6.10: Resultados produzidos pelo *Frequency AdaBoost* em imagens redimensionadas.

Parâmetros / *Redução*	*Faces*	*Acertos*	*Falsos Positivos*	*Taxa de Acerto*
10%	349	0	0	0,0 %
20%	349	0	4	0,0 %
30%	349	42	5	12,04 %
40%	349	108	17	30,94 %
50%	349	150	23	42,97 %
60%	349	194	25	55,58 %
70%	349	218	31	62,46 %

e a taxa de acerto. Quando apresentadas imagens com diferente grau de compressão, é possível observar que nossa abordagem apresenta uma maior resistência, que varia entre 2,59% e 4,87%,

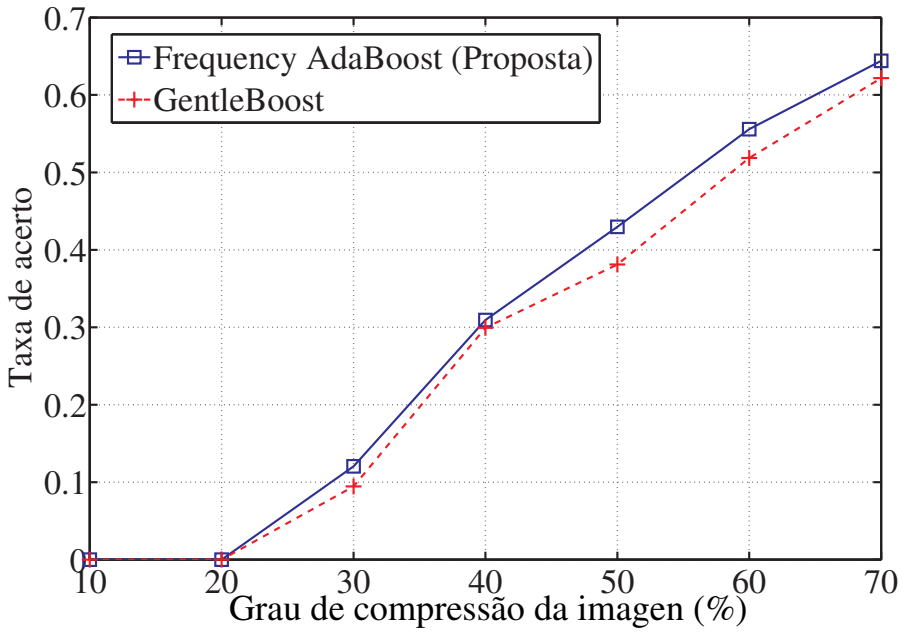
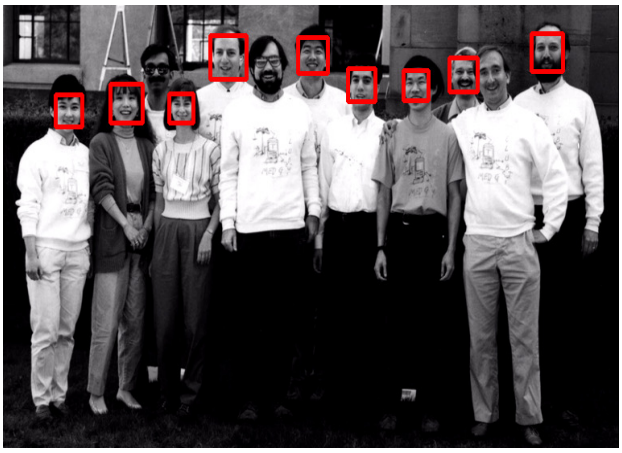


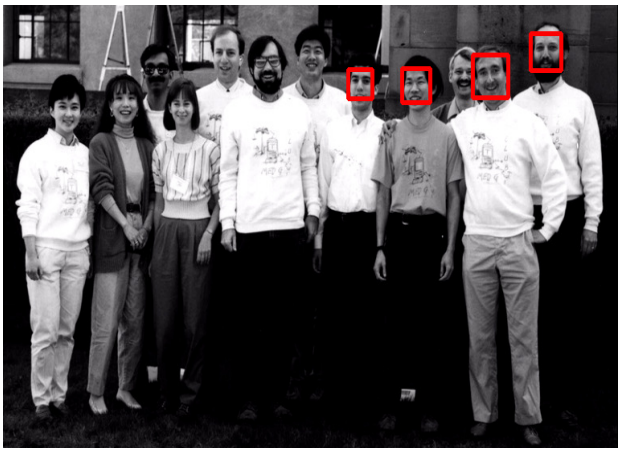
Figura 6.11: Comparação do desempenho da nossa abordagem e da abordagem clássica para imagens redimensionadas.

Nas figuras que compõem a Figura 6.12 são apresentados alguns exemplos do desempenho da nossa abordagem, naquelas imagens redimensionadas em 50%, 60% e 70% do seu comprimento original. Como se pode notar nas Figuras 6.12a, 6.12c e 6.12e, quando é utilizado o fator de frequência, conseguiu-se detectar corretamente quase todas as faces, obtendo-se apenas três falsos

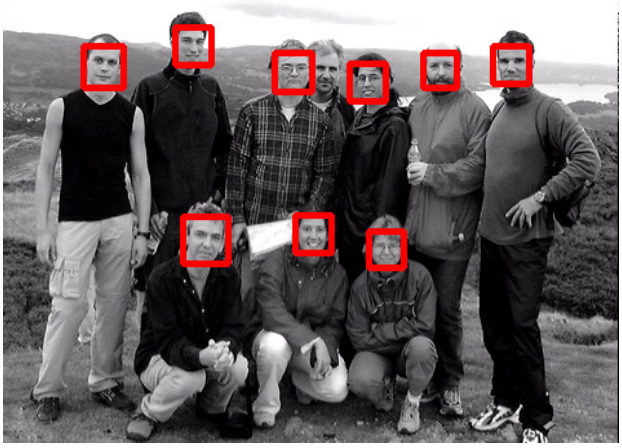
negativos.



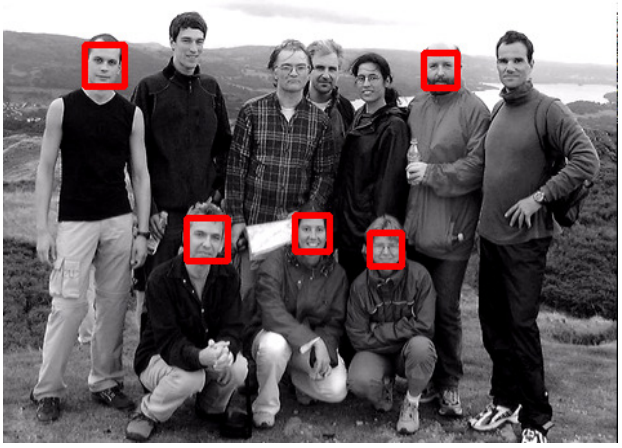
(a) 50% da imagem original, *Frequency AdaBoost*



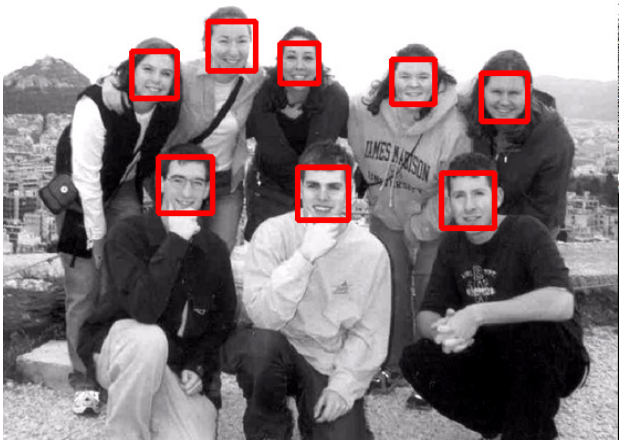
(b) 50% da imagem original, *GentleBoost*



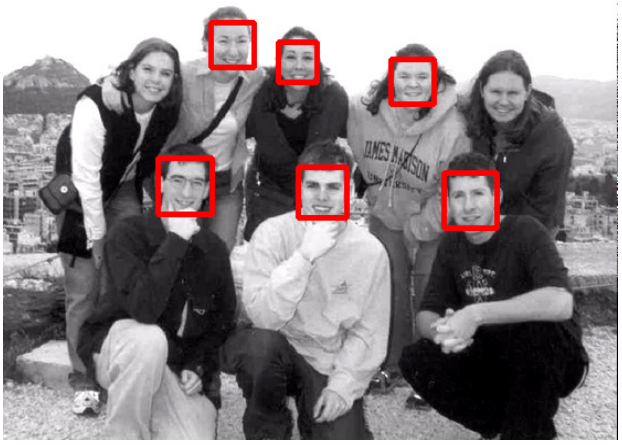
(c) 60% da imagem original, *Frequency AdaBoost*



(d) 60% da imagem original, *GentleBoost*



(e) 70% da imagem original, *Frequency AdaBoost*



(f) 70% da imagem original, *GentleBoost*

Figura 6.12: Faces detectadas em imagens redimensionadas.

6.2.3 Avaliação em imagens mosaico

Neste experimento, utilizamos imagens em baixa resolução que são obtidas aplicando as chamadas *imagens mosaico*. Uma *imagem mosaico*, como foi explicado no Capitulo 4, está dividida em células quadradas de igual tamanho para facilitar seu processamento. Em cada célula existem $n \times n$ pixeis, onde n é o tamanho de um lado da célula. Desta forma, o nível de cinza de cada célula é igual ao valor médio dos níveis de cinza de todos os $n \times n$ pixeis incluídos nesta célula. O conjunto de dados para teste são obtidos do *BAO dataset* (Frischholz , 2012) e a extração de características foi realizada mediante *MB-LBP*.

De forma parecida com o experimento anterior, nesta subseção, visamos avaliar o desempenho do detector de faces com a aplicação do fator de frequência. As resoluções do conjunto de imagens são diminuídas em quatro grupos, variando de acordo com o comprimento n da célula. A Tabela 6.11 apresenta os resultados produzidos; na primeira coluna é indicado o comprimento de cada célula, nas colunas restantes se apresentam o número de faces utilizadas no teste, o número de acertos atingido, a taxa de falsos positivos e a taxa de acerto para à abordagem clássica.

Tabela 6.11: Resultados produzidos pelo *GentleBoost* em imagens mosaico.

Parâmetros / *Celula*	*Faces*	*Acertos*	*Falsos Positivos*	*Taxa de Acerto*
n =2	349	107	17	30,65 %
n = 4	349	185	21	53,00 %
n = 8	349	103	18	29,51 %
n = 16	349	2	3	0,57 %

Na Tabela 6.12, são apresentados os resultados produzidos pelo *Frequency AdaBoost*. Na primeira coluna é indicado o tamanho n da célula, a qual é usada na redução da resolução. Nas colunas restantes indicamos os valores para o número de faces, acertos, taxa de falsos positivos e as taxas de acerto. Neste caso também podemos observar que a taxa de acerto diminui de acordo com a perda de resolução nas imagens. Note-se a perturbação quando usado células de tamanho 4 ($n = 4$), isto é devido a que as imagens apresentam uma perda de características menor, fato que não influencia na analise dos experimentos.

Na Figura 6.13, apresenta-se uma comparação de duas curvas de desempenho. Estas curvas representam a relação entre o comprimento da célula, utilizada na redução da resolução, e a taxa de

Tabela 6.12: Resultados produzidos pelo *Frequency AdaBoost* em imagens mosaico

Parâmetros / *Célula*	*Faces*	*Acertos*	*Falsos Positivos*	*Taxa de Acerto*
n =2	349	143	55	40,97 %
n = 4	349	208	60	59,59 %
n = 8	349	134	51	38,39 %
n = 16	349	21	17	6,01 %

acerto produzida. Assim, podemos notar que o experimento de avaliação de desempenho, usando o fator de frequência em imagens de baixa resolução, proporciona um ganho de 8,0% aproximadamente, na taxa de acerto. Neste caso quando comparado com a aplicação da abordagem clássica *GentleBoost*, o qual fica mais notório quando as imagens perdem resolução.

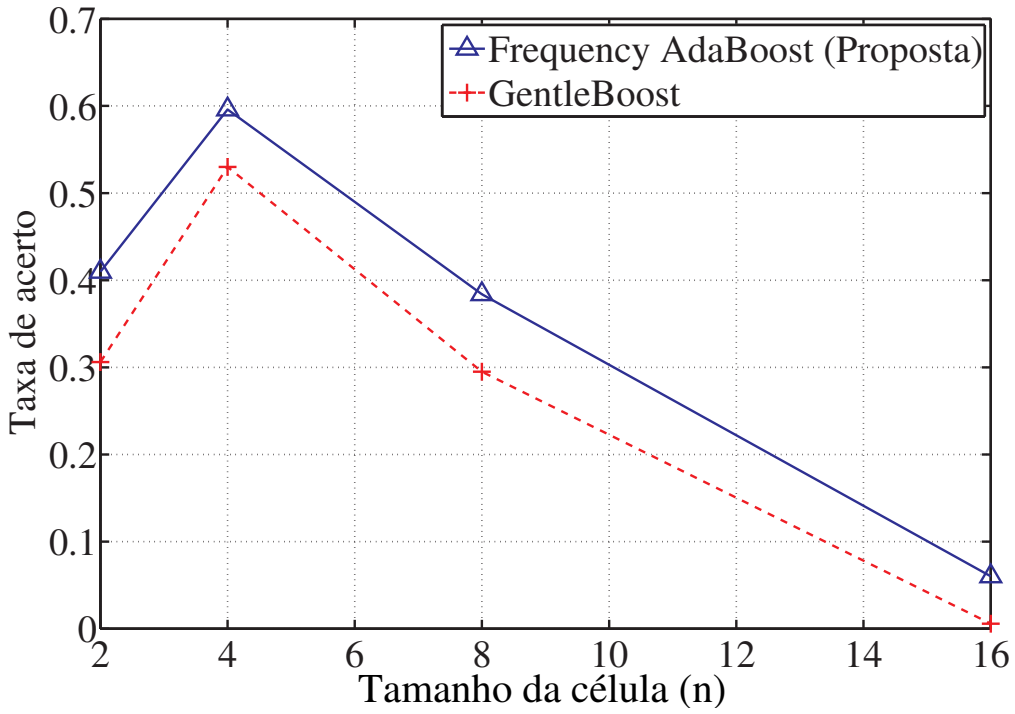


Figura 6.13: Comparação do desempenho da nossa abordagem e da abordagem clássica para imagens reduzidas na resolução.

Alguns exemplos de imagens produzidas no experimento de avaliação de desempenho com o fator de frequência são apresentados nas Figuras 6.14a, 6.14c e 6.14e, nas quais as células de diminuição de resolução possuem tamanhos 2, 8 e 16, respectivamente. De forma parecida com o

experimento anterior, esta abordagem apresenta uma melhoria bastante significativa, sendo que no ultimo exemplo (Fig. 6.14e), apesar de mostrar uma resolução extremamente pobre, o *Frequency AdaBoost* consegue detectar a metade das faces na imagem.

6.2.4 Avaliação em imagens do *CMU e BAO dataset*

De forma análoga aos experimentos anteriores, neste experimento utilizamos três conjuntos de dados, os quais são chamados de *CMU (newtest), CMU (test-low)* (UMIST , 2011) e *BAO dataset* (Frischholz , 2012). Este teste se concentra na comparação do desempenho entre o algoritmo *GentleBoost* e nossa abordagem (*Frequency AdaBoost*), utilizando *MB-LBP* na extração de características.

A Tabela 6.13 apresenta os resultados da comparação. É importante mencionar que as imagens do conjunto de dados *BAO* mostram uma resolução maior do que as do conjunto de dados *CMU (newtest) e CMU (test-low)*. Assim, o desempenho da nossa abordagem é superior em apenas 2,01%, possivelmente por causa da alta resolução das imagens.

Por outro lado, quando as imagens do conjunto de dados *CMU (newtest)* são usadas, nossa abordagem apresenta um desempenho superior de 10,22% . Neste caso, é importante salientar que essas imagens possuem uma resolução pobre. Além disso, o comportamento do classificador tende a ser ainda menos efetivo quando o conjunto de dados *CMU (test-low)* é testado, no qual as imagens apresentam uma resolução bastante pobre. Neste caso, podemos observar uma diminuição drástica na taxa de acerto, porém nossa abordagem apresenta uma robustez maior de 4,27%.

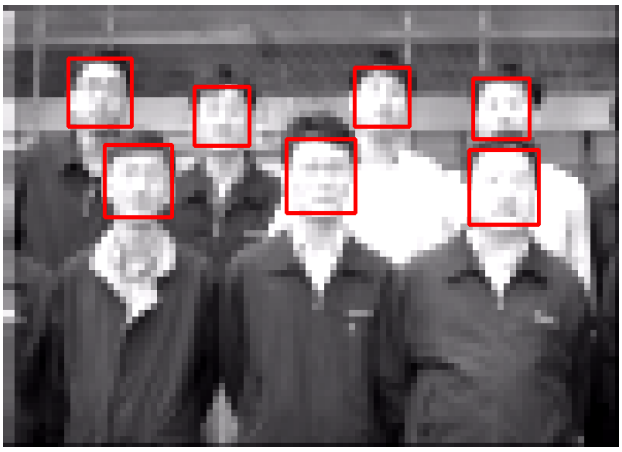
Tabela 6.13: Comparação das taxas de detecção e falsos positivos no *Frequency AdaBoost*.

Parâmetros / *Dataset*	*Algoritmo*	*Faces*	*Acertos*	*Falsos Positivos*	*Taxa de Acerto*
BAO dataset	GB	349	257	38	73,63 %
	FqAB	349	264	53	75,64 %
CMU (newtest) dataset	GB	186	120	18	64,51 %
	FqAB	186	139	23	74,73 %
CMU (test-low) dataset	GB	157	45	13	38,66 %
	FqAB	157	54	18	34,39 %

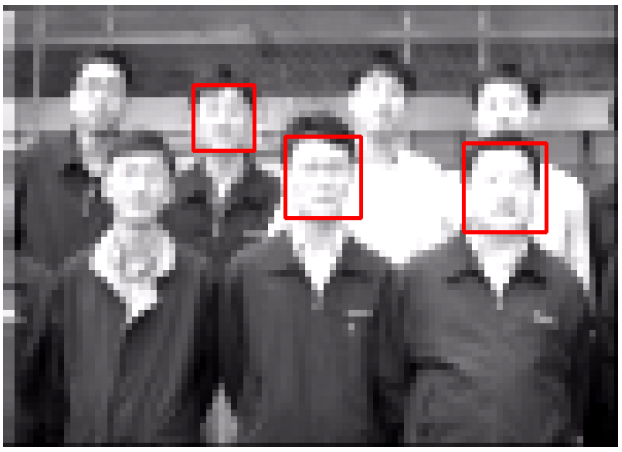
Com propósitos de ilustração, a Figura 6.15 apresenta as faces detectadas pelos algoritmos

(a) Célula $n = 2$, *Frequency AdaBoost*

(b) Célula $n = 2$, *GentleBoost*



(c) Célula $n = 8$, *Frequency AdaBoost*



(d) Célula $n = 8$, *GentleBoost*

(e) Célula $n = 16$, *Frequency AdaBoost*

(f) Célula $n = 16$, *GentleBoost*

Figura 6.14: Faces detectadas em imagens mosaico.

(a) *Frequency AdaBoost* (b) *GentleBoost*

(c) *Frequency AdaBoost* (d) *GentleBoost*

Figura 6.15: Faces detectadas em imagens de diferentes *datasets*.

Frequency AdaBoost e *GentleBoost*, nas imagens do *CMU (test-low)* e do *BAO dataset*.

7 CONCLUSÕES

Neste trabalho, foi apresentado e introduzido um novo procedimento de atualização no algoritmo *AdaBoost*, visando melhorar seu desempenho em dados ruidosos e na detecção de faces em imagens de baixa resolução. Os resultados experimentais demonstram e validam o método proposto, assim, as principais conclusões que podem ser destacadas são:

- **Em relação à investigação das abordagens relacionadas com a atualização de pesos:**
  A atualização da distribuição de pesos no algoritmo *AdaBoost* possui uma importância relevante no processo de seleção de características. Quando a distribuição de pesos associados às amostras são atualizadas adequadamente pode-se obter resultados que melhoram o desempenho do algoritmo *AdaBoost*, como se mostrou na subseção 3.6.1.

- **Em relação ao desenvolvimento e implementação de um novo algoritmo baseado na atualização de pesos:**
  Neste trabalho, mostramos que o erro de treinamento pode ser dividido nos chamados *erros fragmentários* (seção 5.1). Este fato abre muitas possibilidades para avaliar diferentes tipos de fatores que poderiam afetar a performance do algoritmo. Consequentemente, o assim chamado *Frequency AdaBoost* utilizou um fator de frequência para obter erros fragmentários e, dessa forma, dividir o erro empírico. Esta abordagem apresentou vantagens na rapidez de convergência quando comparado com os algoritmos clássicos e, em muitos casos, obteve erros de treinamento mais baixos em menor número de rodadas. Assim, foi possível usar um número menor de classificadores fracos para construir um classificador forte ótimo, o qual é uma propriedade desejável nas aplicações em tempo real, como na detecção de faces, onde a velocidade de teste é crítica.

  A efetividade e precisão do procedimento foram comprovadas através de vários conjuntos de dados artificiais. Nesse sentido, esta abordagem conseguiu melhorar o desempenho dos algoritmos de AdaBoost na presença de dados ruidosos.

- **Em relação à aplicação do algoritmo desenvolvido na detecção de faces:**
  Em termos de detecção de faces, foi desenvolvida e implementada uma versão do *Frequency AdaBoost* para extrair características das faces, combinando-o com os Padrões Binários Locais Multi Escala em Blocos (*MB-LBP*). O algoritmo foi testado de forma intensiva utilizando um alto número de amostras (conforme descrito na seção 6.2), constituindo-se um algoritmo robusto e preciso, uma vez que os resultados forneceram um detector de face com desempenho mais eficiente em termos da taxa de acerto e uma melhor performance frente ao *overfitting* de amostras. Este desempenho superior se tornou ainda mais evidente na presença de imagens com baixa resolução.

- **Em relação à avaliação do desempenho do novo algoritmo na detecção de faces:**
  Aqui destaca-se a geração de vários conjuntos de imagens em baixa resolução. De acordo com duas abordagem de perda de resolução, por um lado as imagens foram redimensionadas, enquanto que, por outro lado, se utilizaram imagens mosaico (descritas na seção 4.1). O *Frequency AdaBoost* apresentou uma resistência maior em todos esse conjuntos de imagens, o que satisfez o critério de estabilidade, deste algoritmo, frente a dados ruidosos.

REFERÊNCIAS BIBLIOGRÁFICAS

Bauer e Kohavi (1999) Eric Bauer e Ron Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Mach. Learn.*, 36(1-2):105–139. ISSN 0885-6125. doi: 10.1023/A:1007515423169. URL http://dx.doi.org/10.1023/A:1007515423169. Citado na pág. 58

BioID (2011) BioID. Face database. https://www.bioid.com/downloads/software.html, Outubro 2011. Citado na pág. 81

Bishop (2006) C. M. Bishop. *Pattern Recognition and Machine Learning*. Citado na pág. 27

Bregman (1967) L.M. Bregman. The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming. *USSR Computational Mathematics and Mathematical Physics*, 7(3):200 – 217. ISSN 0041-5553. doi: 10.1016/0041-5553(67)90040-7. URL http://www.sciencedirect.com/science/article/pii/0041555367900407. Citado na pág. 35

Breiman (1996) Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140. ISSN 0885-6125. doi: 10.1023/A:1018054314350. URL http://dx.doi.org/10.1023/ACitadonapág. 25, 26

Breiman (2000) Leo Breiman. Randomizing outputs to increase prediction accuracy. *Machine Learning*, 40(3):229–242. ISSN 0885-6125. doi: 10.1023/A:1007682208299. URL http://dx.doi.org/10.1023/ACitadonapág. 26

Breiman (2001) Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32. ISSN 0885-6125. doi: 10.1023/A:1010933404324. URL http://dx.doi.org/10.1023/ACitadonapág. 26

Breiman (1999a) Leo Breiman. Prediction games and arcing algorithms. *Neural Comput.*, 11(7):1493–1517. ISSN 0899-7667. doi: 10.1162/089976699300016106. URL http://dx.doi.org/10.1162/089976699300016106. Citado na pág. 33, 58

Breiman (1999b) Leo Breiman. Pasting small votes for classification in large databases and on-line. *Mach. Learn.*, 36(1-2):85–103. ISSN 0885-6125. doi: 10.1023/A:1007563306331. URL http://dx.doi.org/10.1023/A:1007563306331. Citado na pág. 27

Breiman (1998) Leo Breiman. Arcing classifiers. *Annals of Statistics*, 26(3):801–824. URL http://www.jstor.org/stable/120055. Citado na pág. 68, 70

Chai e Ngan (1998) D. Chai e K. N. Ngan. Locating facial region of a head-and-shoulders color image. Em *Proceedings of the 3rd. International Conference on Face & Gesture Recognition*, FG '98, páginas 124–, Washington, DC, USA. IEEE Computer Society. ISBN 0-8186-8344-9. URL http://dl.acm.org/citation.cfm?id=520809.796104. Citado na pág. 45

Chang e Lin (2011) Chih-Chung Chang e Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm. Citado na pág. 78, 80

Collins *et al.* **(2002)** Michael Collins, Robert E. Schapire e Yoram Singer. Logistic regression, adaboost and bregman distances. *Mach. Learn.*, 48(1-3):253–285. ISSN 0885-6125. doi: 10.1023/A:1013912006537. URL http://dx.doi.org/10.1023/A:1013912006537. Citado na pág. 36

Cortes e Vapnik (1995) Corinna Cortes e Vladimir Vapnik. Support-vector networks. *Mach. Learn.*, 20(3):273–297. ISSN 0885-6125. doi: 10.1023/A:1022627411411. URL http://dx.doi.org/10.1023/A:1022627411411. Citado na pág. 7, 18

Craw *et al.* **(1992)** Ian Craw, David Tock e Alan Bennett. Finding face features. Em G. Sandini, editor, *Computer Vision - ECCV'92*, volume 588 of *Lecture Notes in Computer Science*, páginas 92–96. Springer Berlin / Heidelberg. ISBN 978-3-540-55426-4. URL http://dx.doi.org/10.1007/3-540-55426-2_12. 10.1007/3-540-55426-2_12. Citado na pág. 41

Crow (1984) Franklin C. Crow. Summed-area tables for texture mapping. Em *Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '84, páginas 207–212, New York, NY, USA. ACM. ISBN 0-89791-138-5. doi: 10.1145/800031.808600. URL http://doi.acm.org/10.1145/800031.808600. Citado na pág. 109

Dai e Nakano (1996) Ying Dai e Yasuaki Nakano. Face-texture model based on sgld and its application in face detection in a color scene. *Pattern Recognition*, 29(6):1007 – 1017. ISSN 0031-3203. doi: 10.1016/0031-3203(95)00139-5. URL http://www.sciencedirect.com/science/article/pii/0031320395001395. Citado na pág. 44

Dalal e Triggs (2005) Navneet Dalal e Bill Triggs. Histograms of oriented gradients for human detection. Em *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1 - Volume 01*, CVPR '05, páginas 886–893, Washington, DC, USA. IEEE Computer Society. ISBN 0-7695-2372-2. doi: 10.1109/CVPR.2005.177. URL http://dx.doi.org/10.1109/CVPR.2005.177. Citado na pág. 48

Demiriz *et al.* (2002) Ayhan Demiriz, Kristin P. Bennett e John Shawe-Taylor. Linear programming boosting via column generation. *Mach. Learn.*, 46(1-3):225–254. ISSN 0885-6125. doi: 10.1023/A:1012470815092. URL http://dx.doi.org/10.1023/A:1012470815092. Citado na pág. 33

Dietterich (2000) Thomas G. Dietterich. Ensemble methods in machine learning. Em *Proceedings of the First International Workshop on Multiple Classifier Systems*, MCS '00, páginas 1–15, London, UK. Springer-Verlag. ISBN 3-540-67704-6. URL http://dl.acm.org/citation.cfm?id=648054.743935. Citado na pág. 2, 26, 53, 58

Domingo e Watanabe (2000) Carlos Domingo e Osamu Watanabe. Madaboost: A modification of adaboost. Em *Proceedings of the Thirteenth Annual Conference on Computational Learning Theory*, COLT '00, páginas 180–189, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc. ISBN 1-55860-703-X. URL http://dl.acm.org/citation.cfm?id=648299.755176. Citado na pág. 32

FEI (2011) FEI. Face database. http://fei.edu.br/~cet/facedatabase.html, Outubro 2011. Citado na pág. 81

Freund (1999) Yoav Freund. An adaptive version of the boost by majority algorithm. Em *Proceedings of the twelfth annual conference on Computational learning theory*, COLT '99, páginas 102–113, New York, NY, USA. ACM. ISBN 1-58113-167-4. doi: 10.1145/307400.307419. URL http://doi.acm.org/10.1145/307400.307419. Citado na pág. 32, 54

Freund e Schapire (1999) Yoav Freund e Robert E. Schapire. Large margin classification using the perceptron algorithm. *Machine Learning*, 37(3):277–296. Citado na pág. 28

Freund e Schapire (1995) Yoav Freund e Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. Em *Proceedings of the Second European Conference on Computational Learning Theory*, EuroCOLT '95, páginas 23–37, London, UK, UK. Springer-Verlag. ISBN 3-540-59119-2. URL http://dl.acm.org/citation.cfm?id=646943.712093. Citado na pág. 1, 28, 30, 31, 49, 54, 56

Freund *et al.* **(2003)** Yoav Freund, Raj Iyer, Robert E. Schapire e Yoram Singer. An efficient boosting algorithm for combining preferences. *J. Mach. Learn. Res.*, 4:933–969. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=945365.964285. Citado na pág. 25, 34

Friedman *et al.* **(2000)** J. Friedman, T. Hastie e R. Tibshirani. Additive Logistic Regression: a Statistical View of Boosting. *The Annals of Statistics*, 38(2). Citado na pág. 32, 35, 49

Frischholz (2012) R. Frischholz. Bao face database at the face detection homepage. http://www.facedetection.com, Janeiro 2012. Citado na pág. 83, 86, 88

Gao e Gao (2010) Yunlong Gao e Feng Gao. Edited adaboost by weighted knn. *Neurocomput.*, 73 (16-18):3079–3088. ISSN 0925-2312. doi: 10.1016/j.neucom.2010.06.024. URL http://dx.doi.org/10.1016/j.neucom.2010.06.024. Citado na pág. 32

Gómez-Verdejo *et al.* **(2006)** Vanessa Gómez-Verdejo, Manuel Ortega-Moral, Jerónimo Arenas-García e Aníbal R. Figueiras-Vidal. Boosting by weighting critical and erroneous samples. *Neurocomput.*, 69(7-9):679–685. ISSN 0925-2312. doi: 10.1016/j.neucom.2005.12.011. URL http://dx.doi.org/10.1016/j.neucom.2005.12.011. Citado na pág. 32

Govindaraju (1996) Venu Govindaraju. Locating human faces in photographs. *International Journal of Computer Vision*, 19:129–146. ISSN 0920-5691. URL http://dx.doi.org/10.1007/BF00055801. 10.1007/BF00055801. Citado na pág. 41

Govindaraju *et al.* **(1990)** Venu Govindaraju, Sargur N. Srihari e David Sher. A computational model for face location. Em *Computer Vision, 1990. Proceedings, Third International Conference on*, páginas 718 –721. doi: 10.1109/ICCV.1990.139626. Citado na pág. 41

Graf *et al.* **(1995)** H. P. Graf, T. Chen, E. Petajan e E. Cosatto. Locating Faces and Facial Parts. Em *IEEE International Conference on Automatic Face and Gesture Recognition*. Citado na pág. 43

Grove e Schuurmans (1998) Adam J. Grove e Dale Schuurmans. Boosting in the limit: maximizing the margin of learned ensembles. Em *Proceedings of the fifteenth national/tenth conference on Artificial intelligence/Innovative applications of artificial intelligence*, AAAI '98/IAAI '98, páginas 692–699, Menlo Park, CA, USA. American Association for Artificial Intelligence. ISBN 0-262-51098-7. URL http://dl.acm.org/citation.cfm?id=295240.295766. Citado na pág. 2, 58

Hawkins (2004) Douglas M Hawkins. The problem of overfitting. *Journal of Chemical Information and Computer Sciences*, 44(1):1–12. URL http://www.ncbi.nlm.nih.gov/pubmed/14741005. Citado na pág. 2, 53

Hayashi e Hasegawa (2006) Shinji Hayashi e Osamu Hasegawa. Detecting faces from low-resolution images. Em *ACCV (1)*, páginas 787–796. Citado na pág. 2

Haykin (1998) Simon Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2nd ed. ISBN 0132733501. Citado na pág. 17, 23

Hjelmås e Low (2001) Erik Hjelmås e Boon Kee Low. Face detection: A survey. *Computer Vision and Image Understanding*, 83(3):236 – 274. ISSN 1077-3142. doi: 10.1006/cviu.2001.0921. URL http://www.sciencedirect.com/science/article/pii/S107731420190921X. Citado na pág. 37

Ho (1998) Tin Kam Ho. The random subspace method for constructing decision forests. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20(8):832–844. ISSN 0162-8828. doi: 10.1109/34.709601. URL http://dx.doi.org/10.1109/34.709601. Citado na pág. 26

Huang *et al.* (2005) Chang Huang, Haizhou Ai, Yuan Li e Shihong Lao. Vector boosting for rotation invariant multi-view face detection. Em *Proceedings of the Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1 - Volume 01*, ICCV '05, páginas 446–453, Washington, DC, USA. IEEE Computer Society. ISBN 0-7695-2334-X-01. doi: 10.1109/ICCV.2005.246. URL http://dx.doi.org/10.1109/ICCV.2005.246. Citado na pág. 50

Härdle (1990) Wolfgang Härdle. *Smoothing Techniques With Implementations in S*. Springer-Verlang, New York. Citado na pág. 35

Intel (2011) Intel. Open source computer vision library. http://http://code.opencv.org, 2011. Citado na pág. 81

Jang e Kim (2008) Jun-Su Jang e Jong-Hwan Kim. Fast and robust face detection using evolutionary pruning. *Evolutionary Computation, IEEE Transactions on*, 12(5):562 –571. ISSN 1089-778X. doi: 10.1109/TEVC.2007.910140. Citado na pág. 49

Jebara *et al.* (1998) Tony Jebara, Kenneth Russell e Alex Pentland. Mixtures of eigenfeatures for real-time structure from texture. Em *Proceedings of the Sixth International Conference on Computer Vision*, ICCV '98, páginas 128–, Washington, DC, USA. IEEE Computer Society. ISBN 81-7319-221-9. URL http://dl.acm.org/citation.cfm?id=938978.939137. Citado na pág. 45

Jin *et al.* (2004) Hongliang Jin, Qingshan Liu, Hanqing Lu e Xiaofeng Tong. Face detection using improved lbp under bayesian framework. Em *Proceedings of the Third International Conference on Image and Graphics*, ICIG '04, páginas 306–309, Washington, DC, USA. IEEE Computer Society. ISBN 0-7695-2244-0. doi: 10.1109/ICIG.2004.62. URL http://dx.doi.org/10.1109/ICIG.2004.62. Citado na pág. 48

Kearns e Vazirani (1994) Michael J. Kearns e Umesh V. Vazirani. *An introduction to computational learning theory*. MIT Press, Cambridge, MA, USA. ISBN 0-262-11193-4. Citado na pág. 21

Kivinen e Warmuth (1999) Jyrki Kivinen e Manfred K. Warmuth. Boosting as entropy projection. Em *Proceedings of the twelfth annual conference on Computational learning theory*, COLT '99, páginas 134–144, New York, NY, USA. ACM. ISBN 1-58113-167-4. doi: 10.1145/307400.307424. URL http://doi.acm.org/10.1145/307400.307424. Citado na pág. 36

Kotropoulos e Pitas (1997) Constantine Kotropoulos e Ioanis Pitas. Rule-based face detection in frontal views. Em *Acoustics, Speech, and Signal Processing, 1997. ICASSP-97., 1997 IEEE International Conference on*, volume 4, páginas 2537–2540 vol.4. doi: 10.1109/ICASSP.1997.595305. Citado na pág. 39

Kwon e da Vitoria Lobo (1994) Young Ho Kwon e N. da Vitoria Lobo. Face detection using templates. Em *Pattern Recognition, 1994. Vol. 1 - Conference A: Computer Vision Image Processing., Proceedings of the 12th IAPR International Conference on*, volume 1, páginas 764 –767 vol.1. doi: 10.1109/ICPR.1994.576435. Citado na pág. 42

Lebanon e Lafferty (2001) Guy Lebanon e John Lafferty. Boosting and maximum likelihood for exponential models. Em *In Advances in Neural Information Processing Systems*, páginas 447–454. MIT Press. Citado na pág. 36

Leung *et al.* (1995) T. K. Leung, M. C. Burl e P. Perona. Finding faces in cluttered scenes using random labeled graph matching. Em *Proceedings of the Fifth International Conference on Computer Vision*, ICCV '95, páginas 637–, Washington, DC, USA. IEEE Computer Society. ISBN 0-8186-7042-8. URL http://dl.acm.org/citation.cfm?id=839277.840110. Citado na pág. 43, 44

Li *et al.* (2010) Gang Li, Yinping Xu e Jiaying Wang. An improved adaboost face detection algorithm based on optimizing skin color model. Em *Natural Computation (ICNC), 2010 Sixth*

International Conference on, volume 4, páginas 2013 –2015. doi: 10.1109/ICNC.2010.5582393. Citado na pág. 82

Li e Jain (2011) Stan Z. Li e Anil K. Jain, editors. *Handbook of Face Recognition, 2nd Edition*. Springer. ISBN 978-0-85729-931-4. Citado na pág. 1

Li e Zhang (2004) Stan Z. Li e ZhenQiu Zhang. Floatboost learning and statistical face detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(9):1112–1123. ISSN 0162-8828. doi: 10.1109/TPAMI.2004.68. URL http://dx.doi.org/10.1109/TPAMI.2004.68. Citado na pág. 1, 34, 49

Li *et al.* **(2002)** Stan Z. Li, Long Zhu, ZhenQiu Zhang, Andrew Blake, HongJiang Zhang e Harry Shum. Statistical learning of multi-view face detection. Em *Proceedings of the 7th European Conference on Computer Vision-Part IV*, ECCV '02, páginas 67–81, London, UK, UK. Springer-Verlag. ISBN 3-540-43748-7. URL http://dl.acm.org/citation.cfm?id=645318.649240. Citado na pág. 49

Li *et al.* **(2000)** Yongmin Li, Shaogang Gong e Heather Liddell. Support vector regression and classification based multi-view face detection and recognition. Em *Proceedings of the Fourth IEEE International Conference on Automatic Face and Gesture Recognition 2000*, FG '00, páginas 300–, Washington, DC, USA. IEEE Computer Society. ISBN 0-7695-0580-5. URL http://dl.acm.org/citation.cfm?id=795661.796242. Citado na pág. 47

Liao *et al.* **(2007)** Shengcai Liao, Xiangxin Zhu, Zhen Lei, Lun Zhang e Stan Li. Learning multi-scale block local binary patterns for face recognition. Em Seong-Whan Lee e Stan Li, editors, *Advances in Biometrics*, volume 4642 of *Lecture Notes in Computer Science*, páginas 828–837. Springer Berlin / Heidelberg. ISBN 978-3-540-74548-8. URL http://dx.doi.org/10.1007/978-3-540-74549-5_87. 10.1007/978-3-540-74549-5_87. Citado na pág. 63

Lienhart e Maydt (2002) R. Lienhart e J. Maydt. An extended set of haar-like features for rapid object detection. Em *Image Processing. 2002. Proceedings. 2002 International Conference on*, volume 1, páginas I–900 – I–903 vol.1. doi: 10.1109/ICIP.2002.1038171. Citado na pág. 47, 48

Lienhart *et al.* **(2003)** R. Lienhart, A. Kuranov e V. Pisarevsky. Empirical analysis of detection cascades of boosted classifiers for rapid object detection. Em *Proceedings of the 25th DAGM Symposium on Pattern Recognition*, Magdeburg, Germany. Citado na pág. 1, 49

Liu *et al.* **(2006)** Yi-Hung Liu, Yen-Ting Chen e Shey-Shin Lu. Face detection using kernel pca and imbalanced svm. Em *Proceedings of the Second international conference on Advances in*

Natural Computation - Volume Part I, ICNC'06, páginas 351–360, Berlin, Heidelberg. Springer-Verlag. ISBN 3-540-45901-4, 978-3-540-45901-9. doi: 10.1007/11881070_50. URL http://dx.doi.org/10.1007/11881070_50. Citado na pág. 47

McCarthy *et al.* **(2006)** John McCarthy, Marvin Minsky, Nathaniel Rochester e Claude E. Shannon. A proposal for the dartmouth summer research project on artificial intelligence, august 31, 1955. *AI Magazine*, 27(4):12–14. Citado na pág. 5

McKenna *et al.* **(1998)** Stephen J. McKenna, Shaogang Gong e Yogesh Raja. Modelling facial colour and identity with gaussian mixtures. *Pattern Recognition*, 31(12):1883 – 1892. ISSN 0031-3203. doi: 10.1016/S0031-3203(98)00066-1. URL http://www.sciencedirect.com/science/article/pii/S0031320398000661. Citado na pág. 45

Meir *et al.* **(2000)** Ron Meir, Ran El-Yaniv e Shai Ben-David. Localized boosting. Em *Proceedings of the Thirteenth Annual Conference on Computational Learning Theory*, COLT '00, páginas 190–199, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc. ISBN 1-55860-703-X. URL http://dl.acm.org/citation.cfm?id=648299.755180. Citado na pág. 34

Mita *et al.* **(2005)** Takeshi Mita, Toshimitsu Kaneko e Osamu Hori. Joint haar-like features for face detection. Em *Proceedings of the Tenth IEEE International Conference on Computer Vision - Volume 2*, ICCV '05, páginas 1619–1626, Washington, DC, USA. IEEE Computer Society. ISBN 0-7695-2334-X-02. doi: 10.1109/ICCV.2005.129. URL http://dx.doi.org/10.1109/ICCV.2005.129. Citado na pág. 47, 48, 49

Mitchell (1997) Thomas M. Mitchell. *Machine Learning*. McGraw-Hill, Inc., New York, NY, USA, 1 ed. ISBN 0070428077, 9780070428072. Citado na pág. 5

Mohri *et al.* **(2012)** Mehryar Mohri, Afshin Rostamizadeh e Ameet Talwalkar. *Foundations of Machine Learning*. The MIT Press. ISBN 026201825X, 9780262018258. Citado na pág. 21

Muller *et al.* **(2001)** K. R. Muller, S. Mika, G. Ratsch, K. Tsuda e B. Scholkopf. An introduction to kernel-based learning algorithms. *Trans. Neur. Netw.*, 12(2):181–201. ISSN 1045-9227. doi: 10.1109/72.914517. URL http://dx.doi.org/10.1109/72.914517. Citado na pág. 8

Nosaka *et al.* **(2012)** Ryusuke Nosaka, Yasuhiro Ohkawa e Kazuhiro Fukui. Feature extraction based on co-occurrence of adjacent local binary patterns. Em *Proceedings of the 5th Pacific*

Rim conference on Advances in Image and Video Technology - Volume Part II, PSIVT'11, páginas 82–91, Berlin, Heidelberg. Springer-Verlag. ISBN 978-3-642-25345-4. doi: 10.1007/978-3-642-25346-1_8. URL http://dx.doi.org/10.1007/978-3-642-25346-1_8. Citado na pág. 48

Ojala *et al.* **(1994)** T. Ojala, M. Pietikainen e D. Harwood. Performance evaluation of texture measures with classification based on kullback discrimination of distributions. Em *Pattern Recognition, 1994. Vol. 1 - Conference A: Computer Vision Image Processing., Proceedings of the 12th IAPR International Conference on*, volume 1, páginas 582 –585 vol.1. doi: 10.1109/ICPR.1994.576366. Citado na pág. 62

Ojala *et al.* **(2002)** Timo Ojala, Matti Pietikäinen e Topi Mäenpää. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(7):971–987. ISSN 0162-8828. doi: 10.1109/TPAMI.2002.1017623. URL http://dx.doi.org/10.1109/TPAMI.2002.1017623. Citado na pág. 48, 62

Osuna *et al.* **(1997)** Edgar Osuna, Robert reund e Federico Girosi. Training support vector machines: an application to face detection. Em *Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition (CVPR '97)*, CVPR '97, páginas 130–, Washington, DC, USA. IEEE Computer Society. ISBN 0-8186-7822-4. URL http://dl.acm.org/citation.cfm?id=794189.794466. Citado na pág. 46

Pham e Chain (2007) Minh-Tri Pham e Tat-Jen Chain. Fast training and selection of haar features using statistics in boosting-based face detection. *Computer Vision, IEEE International Conference on*, 0:1–7. doi: http://doi.ieeecomputersociety.org/10.1109/ICCV.2007.4409038. Citado na pág. 49

PICS (2011) PICS. Psychological image collection at stirling. http://pics.stir.ac.uk, Outubro 2011. Citado na pág. 81

Pudil *et al.* **(1994)** P. Pudil, J. Novovičová e J. Kittler. Floating search methods in feature selection. *Pattern Recogn. Lett.*, 15(11):1119–1125. ISSN 0167-8655. doi: 10.1016/0167-8655(94)90127-9. URL http://dx.doi.org/10.1016/0167-8655(94)90127-9. Citado na pág. 34

Qian *et al.* **(1998)** R.J. Qian, M.I. Sezan e K.E. Matthews. A robust real-time face tracking algorithm. Em *Image Processing, 1998. ICIP 98. Proceedings. 1998 International Conference on*, volume 1, páginas 131 –135 vol.1. doi: 10.1109/ICIP.1998.723443. Citado na pág. 45

Quinlan (1996) J. Ross Quinlan. Boosting first-order learning. Em *Proceedings of the 7th International Workshop on Algorithmic Learning Theory*, ALT '96, páginas 143–155, London, UK, UK. Springer-Verlag. ISBN 3-540-61863-5. URL http://dl.acm.org/citation.cfm?id=647714.757187. Citado na pág. 2, 58

Rätsch *et al.* (2001) G. Rätsch, T. Onoda e K.-R. Müller. Soft margins for adaboost. *Mach. Learn.*, 42:287–320. ISSN 0885-6125. doi: 10.1023/A:1007618119488. URL http://dl.acm.org/citation.cfm?id=373423.373433. Citado na pág. 2, 53, 58

Rätsch e Warmuth (2005) Gunnar Rätsch e Manfred K. Warmuth. Efficient margin maximizing with boosting. *J. Mach. Learn. Res.*, 6:2131–2152. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=1046920.1194915. Citado na pág. 33

Reyzin e Schapire (2006) Lev Reyzin e Robert E. Schapire. How boosting the margin can also boost classifier complexity. Em *Proceedings of the 23rd international conference on Machine learning*, ICML '06, páginas 753–760, New York, NY, USA. ACM. ISBN 1-59593-383-2. doi: 10.1145/1143844.1143939. URL http://doi.acm.org/10.1145/1143844.1143939. Citado na pág. 33

Rodriguez *et al.* (2006) Juan J. Rodriguez, Ludmila I. Kuncheva e Carlos J. Alonso. Rotation forest: A new classifier ensemble method. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28(10): 1619–1630. ISSN 0162-8828. doi: 10.1109/TPAMI.2006.211. URL http://dx.doi.org/10.1109/TPAMI.2006.211. Citado na pág. 26

Rosenblatt (1962) Frank Rosenblatt. *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*. Spartan Books. URL http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/B0006AXUII. Citado na pág. 18

Rowley *et al.* (1998a) H. A. Rowley, S. Baluja e T. Kanade. Rotation invariant neural network-based face detection. Em *CVPR '98: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, página 38, Washington, DC, USA. IEEE Computer Society. ISBN 0-8186-8497-6. Citado na pág. 46

Rowley *et al.* (1996) Henry Rowley, Shumeet Baluja e Takeo Kanade. Neural network-based face detection. Em *Computer Vision and Pattern Recognition '96*. Citado na pág. 45, 46

Rowley *et al.* (1998b) Henry A. Rowley, Shumeet Baluja e Takeo Kanade. Neural network-based face detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 20(1):23 –38. ISSN 0162-8828. doi: 10.1109/34.655647. Citado na pág. 45

Rudin *et al.* (2007) Cynthia Rudin, Robert E. Schapire e Ingrid Daubechies. Analysis of boosting algorithms using the smooth margin function. *The Annals of Statistics*, 35(6):2723–2768. Citado na pág. 33

Rätsch *et al.* (1998) Gunnar Rätsch, Takashi Onoda e Klaus Robert Müller. An improvement of adaboost to avoid overfitting. Em *Proc. of the Int. Conf. on Neural Information Processing*, páginas 506–509. Citado na pág. 2

Sahbi e Geman (2006) Hichem Sahbi e Donald Geman. A hierarchy of support vector machines for pattern detection. *J. Mach. Learn. Res.*, 7:2087–2123. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=1248547.1248622. Citado na pág. 47

Sakai *et al.* (1969) T. Sakai, M. Nagao e S. Fujibayashi. Line extraction and pattern detection in a photograph. *Pattern Recognition*, 1(3):233 – 248. ISSN 0031-3203. doi: 10.1016/0031-3203(69)90006-5. Citado na pág. 40

Samaria *et al.* (1994) F. S. Samaria, F. S. Samaria, A.C. Harter e Old Addenbrooke's Site. Parameterisation of a stochastic model for human face identification. Citado na pág. 81

Satoh *et al.* (1999) S. Satoh, Y. Nakamura e T. Kanade. Name-it: naming and detecting faces in news videos. *Multimedia, IEEE*, 6(1):22 –35. ISSN 1070-986X. doi: 10.1109/93.752960. Citado na pág. 45

Schapire (1990) Robert E. Schapire. The strength of weak learnability. *Mach. Learn.*, 5(2):197–227. ISSN 0885-6125. doi: 10.1023/A:1022648800760. URL http://dx.doi.org/10.1023/A:1022648800760. Citado na pág. 7, 27

Schapire e Freund (2012) Robert E. Schapire e Yoav Freund. *Boosting: Foundations and Algorithms*. The MIT Press. ISBN 0262017180, 9780262017183. Citado na pág. 10

Schapire e Singer (1999) Robert E. Schapire e Yoram Singer. Improved boosting algorithms using confidence-rated predictions. *Mach. Learn.*, 37(3):297–336. ISSN 0885-6125. doi: 10.1023/A:1007614523901. URL http://dx.doi.org/10.1023/A:1007614523901. Citado na pág. 27, 35, 59

Schapire *et al.* (1997) Robert E. Schapire, Yoav Freund, Peter Barlett e Wee Sun Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. Em *Proceedings of the Fourteenth International Conference on Machine Learning*, ICML '97, páginas 322–330,

San Francisco, CA, USA. Morgan Kaufmann Publishers Inc. ISBN 1-55860-486-3. URL http://dl.acm.org/citation.cfm?id=645526.657129. Citado na pág. 22, 33

Schneiderman (2004) Henry Schneiderman. Feature-centric evaluation for efficient cascaded object detection. Em *Proceedings of the 2004 IEEE computer society conference on Computer vision and pattern recognition*, CVPR'04, páginas 29–36, Washington, DC, USA. IEEE Computer Society. URL http://dl.acm.org/citation.cfm?id=1896300.1896306. Citado na pág. 49

Scholkopf e Smola (2001) Bernhard Scholkopf e Alexander J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, USA. ISBN 0262194759. Citado na pág. 14

Servedio (2003) Rocco A. Servedio. Smooth boosting and learning with malicious noise. *J. Mach. Learn. Res.*, 4:633–648. ISSN 1532-4435. doi: http://dx.doi.org/10.1162/153244304773936072. URL http://dx.doi.org/10.1162/153244304773936072. Citado na pág. 2, 32, 53

Shen e Li (2010a) Chunhua Shen e Hanxi Li. Boosting through optimization of margin distributions. *Trans. Neur. Netw.*, 21(4):659–666. ISSN 1045-9227. doi: 10.1109/TNN.2010.2040484. URL http://dx.doi.org/10.1109/TNN.2010.2040484. Citado na pág. 34

Shen e Li (2010b) Chunhua Shen e Hanxi Li. On the dual formulation of boosting algorithms. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32(12):2216–2231. ISSN 0162-8828. doi: 10.1109/TPAMI.2010.47. URL http://dx.doi.org/10.1109/TPAMI.2010.47. Citado na pág. 35, 36

Sirohey *et al.* (1993) Saad Ahmed Sirohey, Masooda Begum, Iftikhar A. Sirohey e Zarina Sirohey. Human face segmentation and identification, 1993. Citado na pág. 43

Smola e Bartlett (2000) Alexander J. Smola e Peter J. Bartlett, editors. *Advances in Large Margin Classifiers*. MIT Press, Cambridge, MA, USA. ISBN 0262194481. Citado na pág. 15, 16, 17

Sobottka e Pitas (1996) Karin Sobottka e Ioannis Pitas. Segmentation and tracking of faces in color images. *Automatic Face and Gesture Recognition, IEEE International Conference on*, 0: 236. doi: http://doi.ieeecomputersociety.org/10.1109/AFGR.1996.557270. Citado na pág. 45

Turing (1995) A. M. Turing. Computers & thought. chapter Computing machinery and intelligence, páginas 11–35. MIT Press, Cambridge, MA, USA. ISBN 0-262-56092-5. URL http://dl.acm.org/citation.cfm?id=216408.216410. Citado na pág. 5

UMIST (2011) UMIST. Face database. http://www.sheffield.ac.uk/eee/research/iel/research/face, Outubro 2011. Citado na pág. 81, 88

Valiant (1984) L. G. Valiant. A theory of the learnable. *Commun. ACM*, 27(11):1134–1142. ISSN 0001-0782. doi: 10.1145/1968.1972. URL http://doi.acm.org/10.1145/1968.1972. Citado na pág. 7, 19, 20

Vapnik e Chervonenkis (1971) V. N. Vapnik e A. Ya. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 16(2):264–280. doi: 10.1137/1116025. URL http://link.aip.org/link/?TPR/16/264/1. Citado na pág. 7, 10, 13

Vapnik (1982) Vladimir Vapnik. *Estimation of Dependences Based on Empirical Data: Springer Series in Statistics (Springer Series in Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA. ISBN 0387907335. Citado na pág. 7, 10, 14, 54

Vapnik (1991) Vladimir Vapnik. Principles of risk minimization for learning theory. Em *Neural Information Processing Systems*, páginas 831–838. Citado na pág. 7, 13, 17

Vapnik (1998) Vladimir Vapnik. *Statistical learning theory*. Wiley. ISBN 978-0-471-03003-4. Citado na pág. 9, 14, 17

Vapnik *et al.* (1996) Vladimir Vapnik, Steven E. Golowich e Alex J. Smola. Support vector method for function approximation, regression estimation and signal processing. Em *NIPS*, páginas 281–287. Citado na pág. 14

Vapnik (1995) Vladimir N. Vapnik. *The nature of statistical learning theory*. Springer-Verlag New York, Inc., New York, NY, USA. ISBN 0-387-94559-8. Citado na pág. 8, 13

Vapnik (2000) Vladimir Naoumovitch Vapnik. *The nature of statistical learning theory*. Statistics for engineering and information science. Springer, New York. ISBN 0-387-98780-0. URL http://opac.inria.fr/record=b1097848. Citado na pág. 22

Vezhnevets e Vezhnevets (2005) Alexander Vezhnevets e Vladimir Vezhnevets. 'modest adaboost' - teaching adaboost to generalize better. Em *GRAPHICON 2005*. Citado na pág. 32

Viola *et al.* (2003) M. Viola, Michael J. Jones e Paul Viola. Fast multi-view face detection. Em *Proc. of Computer Vision and Pattern Recognition*. Citado na pág. 50

Viola e Jones (2001) Paul Viola e Michael Jones. Rapid object detection using a boosted cascade of simple features. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 1:511. ISSN 1063-6919. doi: http://doi.ieeecomputersociety.org/10.1109/CVPR.2001.990517. Citado na pág. 47, 49, 65, 79, 109

Viola e Jones (2004) Paul Viola e Michael J. Jones. Robust real-time face detection. *Int. J. Comput. Vision*, 57(2):137–154. ISSN 0920-5691. doi: 10.1023/B:VISI.0000013087.49260.fb. URL http://dx.doi.org/10.1023/B:VISI.0000013087.49260.fb. Citado na pág. 1, 61, 82

Wang *et al.* (2008) Liwei Wang, Masashi Sugiyama, Cheng Yang, Zhi-Hua Zhou e Jufu Feng. On the margin explanation of boosting algorithms. Em Rocco A. Servedio e Tong Zhang, editors, *COLT*, páginas 479–490. Omnipress. Citado na pág. 34

Wang *et al.* (2009) Xiaoyu Wang, Tony X. Han e Shuicheng Yan. An hog-lbp human detector with partial occlusion handling. Em *ICCV*, páginas 32–39. IEEE. URL http://dx.doi.org/10.1109/ICCV.2009.5459207. Citado na pág. 48

Warmuth *et al.* (2007) Manfred K. Warmuth, Karen A. Glocer e Gunnar Rätsch. Boosting algorithms for maximizing the soft margin. Em John C. Platt, Daphne Koller, Yoram Singer e Sam T. Roweis, editors, *NIPS*. Curran Associates, Inc. Citado na pág. 36

Warmuth *et al.* (2008) Manfred K. Warmuth, Karen A. Glocer e S. V. Vishwanathan. Entropy regularized lpboost. Em *Proceedings of the 19th international conference on Algorithmic Learning Theory*, ALT '08, páginas 256–271, Berlin, Heidelberg. Springer-Verlag. ISBN 978-3-540-87986-2. doi: 10.1007/978-3-540-87987-9_23. URL http://dx.doi.org/10.1007/978-3-540-87987-9_23. Citado na pág. 36

Wilbur *et al.* (2005) W. John Wilbur, Lana Yeganova e Won Kim. The synergy between pav and adaboost. *Mach. Learn.*, 61(1-3):71–103. ISSN 0885-6125. doi: 10.1007/s10994-005-1123-6. URL http://dx.doi.org/10.1007/s10994-005-1123-6. Citado na pág. 34

Wu *et al.* (2004) Bo Wu, Haizhou Ai, Chang Huang e Shihong Lao. Fast rotation invariant multi-view face detection based on real adaboost. Em *Proceedings of the Sixth IEEE international conference on Automatic face and gesture recognition*, FGR' 04, páginas 79–84, Washington, DC, USA. IEEE Computer Society. ISBN 0-7695-2122-3. URL http://dl.acm.org/citation.cfm?id=1949767.1949784. Citado na pág. 49

Yan *et al.* **(2008)** Shengye Yan, Shiguang Shan, Xilin Chen e Wen Gao. Locally assembled binary (lab) feature with feature-centric cascade for fast and accurate face detection. Em *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, páginas 1 –7. doi: 10.1109/CVPR.2008.4587802. Citado na pág. 48

Yang e Huang (1994) Guangzheng Yang e Thomas S Huang. Human face detection in a complex background. *Pattern Recognition*, 27(1):53 – 63. ISSN 0031-3203. doi: 10.1016/0031-3203(94)90017-5. URL http://www.sciencedirect.com/science/article/pii/0031320394900175. Citado na pág. 37

Yang *et al.* **(2002)** Ming-Hsuan Yang, David J. Kriegman e Narendra Ahuja. Detecting faces in images: a survey. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(1):34 –58. ISSN 0162-8828. doi: 10.1109/34.982883. Citado na pág. 37

Yow e Cipolla (1997) Kin Choong Yow e Roberto Cipolla. Feature-based human face detection. *Image and Vision Computing*, 15(9):713 – 735. ISSN 0262-8856. doi: 10.1016/S0262-8856(97)00003-6. Citado na pág. 43, 44

Yuan-Cheng e Jing-Yu (2010) Xie Yuan-Cheng e Yang Jing-Yu. Using clustering to speed up adaboost and detecting noisy data. *Journal of Software*, 8:1889–1897. doi: 10.3724/SP.J.1001.2010.03611. Citado na pág. 72

Yuille *et al.* **(1989)** Allan L. Yuille, David S. Cohen e Peter W. Hallinan. Feature extraction from faces using deformable templates. Em *Computer Vision and Pattern Recognition, 1989. Proceedings CVPR '89., IEEE Computer Society Conference on*, páginas 104 –109. doi: 10.1109/CVPR.1989.37836. Citado na pág. 41, 42

Zhang e Zhang (2010) Cha Zhang e Zhengyou Zhang. *Boosting-Based Face Detection and Adaptation*. Morgan and Claypool Publishers. ISBN 160845133X, 9781608451333. Citado na pág. 1, 37

Zhang *et al.* **(2007a)** Lun Zhang, Rufeng Chu, Shiming Xiang, Shengcai Liao e StanZ. Li. Face detection based on multi-block lbp representation. Em Seong-Whan Lee e StanZ. Li, editors, *Advances in Biometrics*, volume 4642 of *Lecture Notes in Computer Science*, páginas 11–18. Springer Berlin Heidelberg. ISBN 978-3-540-74548-8. doi: 10.1007/978-3-540-74549-5_2. URL http://dx.doi.org/10.1007/978-3-540-74549-5_2. Citado na pág. 64

Zhang *et al.* **(2007b)** Lun Zhang, Rufeng Chu, Shiming Xiang, Shengcai Liao e StanZ. Li. Face detection based on multi-block lbp representation. Em Seong-Whan Lee e StanZ. Li, editors, *Advances in Biometrics*, volume 4642 of *Lecture Notes in Computer Science*, páginas 11–18. Springer Berlin Heidelberg. ISBN 978-3-540-74548-8. doi: 10.1007/978-3-540-74549-5_2. URL http://dx.doi.org/10.1007/978-3-540-74549-5_2. Citado na pág. 48

Zhang *et al.* **(2011)** Zhong Zhang, Rommel Alonzo e Vassilis Athitsos. Experiments with computer vision methods for hand detection. Em *Proceedings of the 4th International Conference on PErvasive Technologies Related to Assistive Environments*, PETRA '11, páginas 21:1–21:6, New York, NY, USA. ACM. ISBN 978-1-4503-0772-7. doi: 10.1145/2141622.2141648. URL http://doi.acm.org/10.1145/2141622.2141648. Citado na pág. 2

APÊNDICE A – IMAGEM INTEGRAL

A.1 A Imagem Integral

Crow (1984) introduziu a *tabela de áreas somadas* na área de computação gráfica, posteriormente conhecida como imagem integral. A imagem integral é um método rápido que avalia as somas dos retângulos dentro de uma tabela (imagem pré-avaliada). Viola e Jones (2001) introduziram a imagem integral na tarefa de detecção de faces com bastante sucesso. Devido a esse sucesso, os trabalhos subsequentes utilizaram a imagem integral para o pré-processamento de imagens.

Dada uma função $i(x,y)$ (imagem original) sobre o domínio discreto $\prod_{j=1}^{2}\left[m_j,M_j\right] \subset \mathbb{Z}^2$, define-se uma nova função $s(x,y)$ da seguinte forma:

$$s(x,y) = \sum_{x' \leq x \bigwedge y' \leq y} i(x',y'), \tag{A.1}$$

Esta função é conhecida como imagem integral. Os valores dos pixeis que a função $i(x',y')$ processa numa grade $[a,b] \times [c,d]$, podem ser obtidas usando apenas 4 valores dentro da grade. A Equação A.2 ilustra este procedimento.

$$\sum_{x'=a,y'=c}^{b}\sum^{d} i\left(x',y'\right) = s\left(b,d\right) + s\left(a,c\right) - s\left(a,d\right) - s\left(b,c\right), \tag{A.2}$$

Onde $m_1 \leq a$, $b \leq M_1$) e $m_2 \leq c$, $d \leq M_2$,

A Figura A.1 mostra este procedimento de forma ilustrativa. Uma imagem integral limitada por 4 vértices (A, B, C, D) pode ser calculada através da soma das imagens integrais em cada vértice: $s(D) + s(A) - s(C) - s(B)$.

Uma imagem integral pode ser obtida mediante a seguinte expressão:

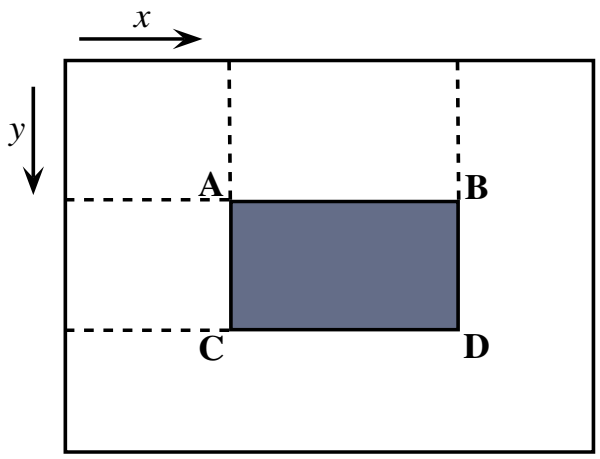
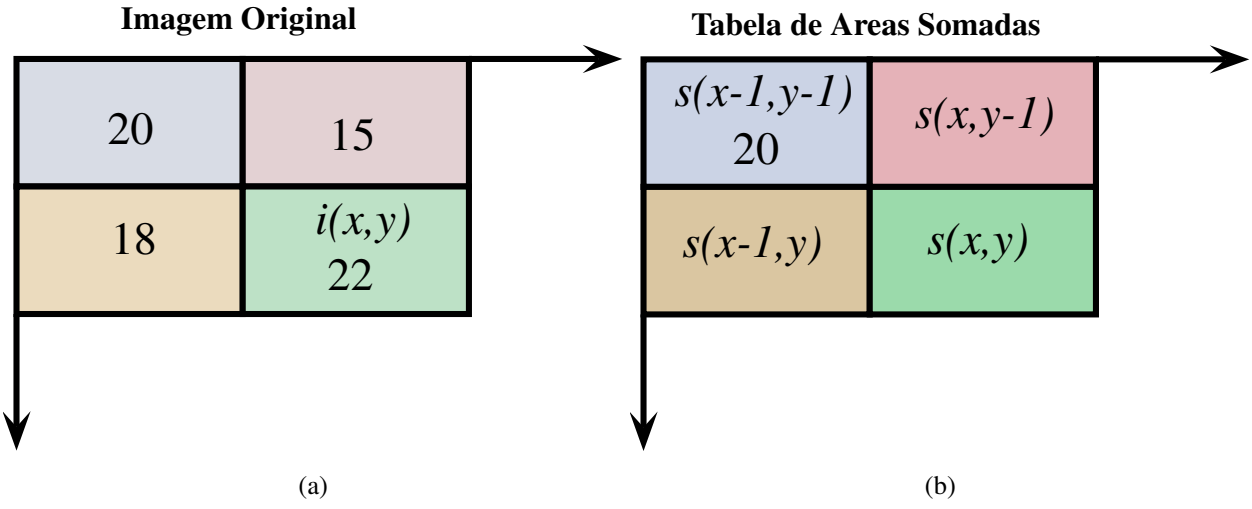


Figura A.1: Ilustração da imagem integral.

$$s(x,y) = i(x,y) + s(x-1,y) + s(x,y-1) - s(x-1,y-1) \tag{A.3}$$

A Figura A.2a apresenta um exemplo com uma imagem original, a qual possui os valores dos píxeis em cada quadrado, assumindo quatro píxeis de diferentes valores. Por outro lado, os valores para a tabela de áreas somadas são preenchidas de forma sequencial. Em primeiro lugar, o valor do píxel da imagem original $i(x-1,y-1)$ é atribuído diretamente ao valor do píxel da tabela de áreas somadas $s(x-1,y-1)$, dado que os valores para os píxeis esquerdo e de cima do píxel $i(x-1,y-1)$ não existem, portanto, estes são considerados como zero (0), tal como se apresenta na Figura A.2b.



(a) (b)

Figura A.2: Imagem original e tabela de áreas somadas.

Assim, aplicando a Equação A.3 para o píxel $i(x-1,y-1)$ se obtém o seguinte:

$$s(x-1,y-1) = i(x-1,y-1) + s(x-2,y) + s(x,y-2) - s(x-2,y-2)$$
$$s(x-1,y-1) = 20+0+0-0$$
$$s(x-1,y-1) = 20$$

Similarmente, aplicamos a Equação A.3 para obter o valor de $s(x-1,y)$:

$$s(x-1,y) = i(x-1,y) + s(x-2,y) + s(x-1,y-1) - s(x-2,y-2)$$
$$s(x-1,y) = 18+0+20-0$$
$$s(x-1,y) = 38$$

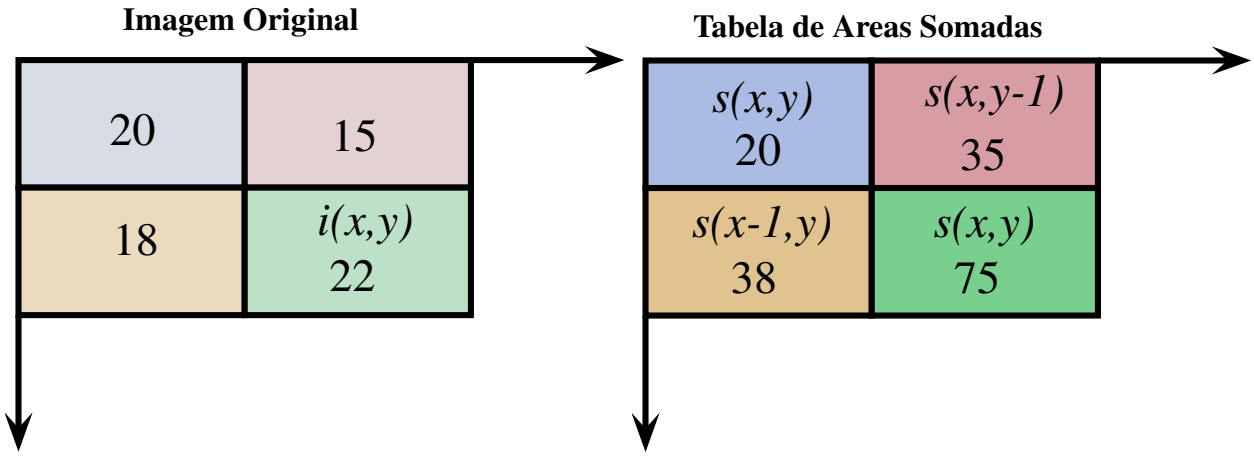
Assim como também para $s(x,y-1)$:

$$s(x,y-1) = i(x,y-1) + s(x-1,y-1) + s(x,y-2) - s(x-1,y-2)$$
$$s(x,y-1) = 15+20+0-0$$
$$s(x,y-1) = 35$$

Finalmente, o valor para a posição (x,y) da imagem integral pode ser calculado de forma rápida e eficiente, através de apenas os quatro valores obtidos anteriormente, da seguinte forma:

$$s(x,y) = i(x,y) + s(x-1,y) + s(x,y-1) - s(x-1,y-1)$$
$$s(x,y) = 22+38+35-20$$
$$s(x,y) = 75$$

A Figura A.3b apresenta a tabela de áreas somadas com os valores obtidos utilizando a Equação A.3.



(a) (b)

Figura A.3: Imagem original e tabela de áreas somadas com os valores para cada píxel da tabela.

APÊNDICE B – TREINAMENTO DO ADABOOST

Nesta seção, apresentamos o procedimento de treinamento no algoritmo *AdaBoost*. Com propósitos de ilustração, consideramos apenas 3 classificadores desempenhando uma classificação binaria. Nesta representação consideramos dois tipos de amostras, representadas por círculos e por cruzes, onde o tamanho de cada símbolo indica o peso atribuído a cada amostra. Inicialmente, todos os pesos possuem o mesmo valor, então temos símbolos (círculos e triangulos) possuem do mesmo tamanho, como se apresenta na Figura B.1.

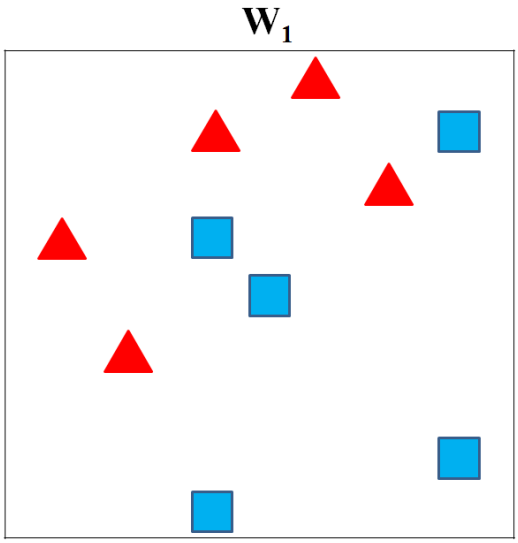


Figura B.1: Amostras de treinamento.

Após a primeira iteração, os pesos das amostras classificadas incorretamente são aumentados, visando que o classificador seguinte foque nestas amostras (Figura B.2). Para dar maior ênfase, os tamanhos dos símbolos são escalados de acordo com sua classificação. Assim, as amostras com pesos maiores, produto de uma classificação incorreta, possuem símbolos maiores, no entanto, as amostras classificadas corretamente possuem símbolos menores.

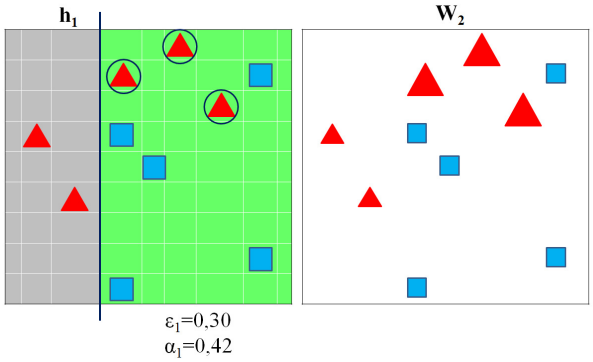
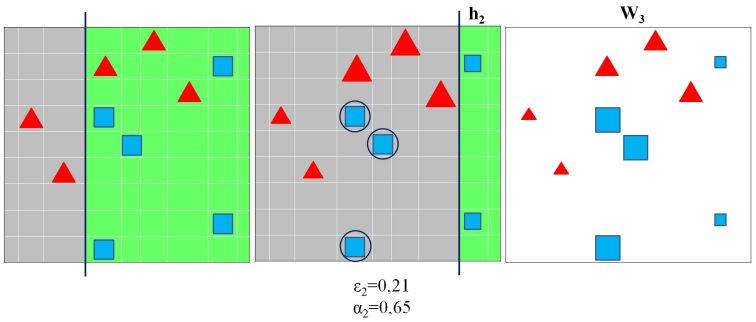
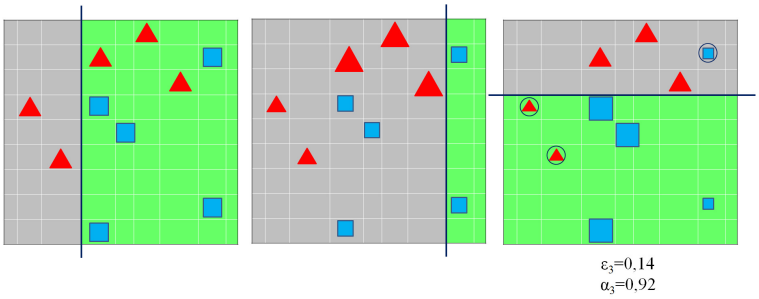


Figura B.2: Primeira curva de separação.

Com o objetivo de minimizar a taxa de erro, o ajuste dos pesos obriga ao seguinte classificador focar nas amostras com pesos maiores. Após várias iterações, o algoritmo de *Boosting* combina esses classificadores fracos num único classificador, chamado de forte, esperando maior precisão que cada um dos classificadores fracos. Neste momento, os valores dos α_t são computados, atribuindo uma importância a cada classificador, e proporcionando uma maior influência aos melhores classificadores, como se apresenta na Figura B.4.



(a) Segunda curva de separação



(b) Terceira curva de separação

Figura B.3: Vista Geral do treinamento do algoritmo *AdaBoost*.

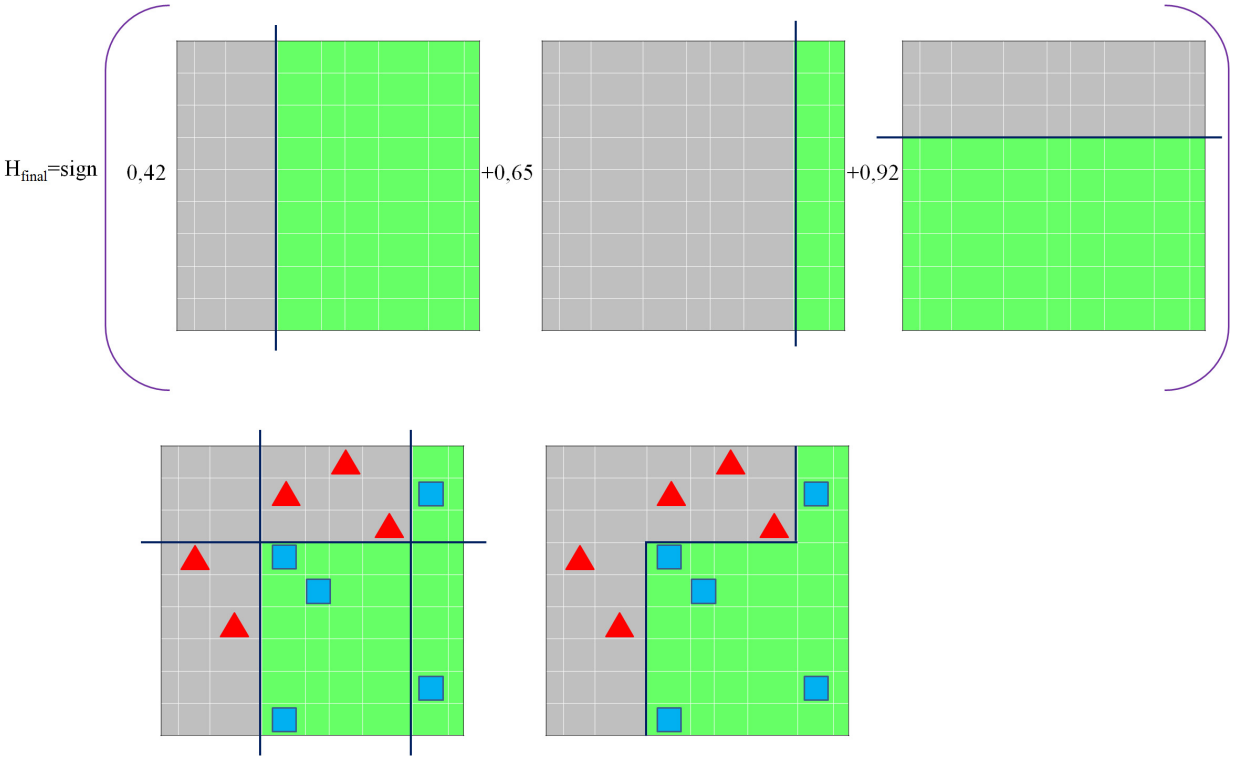


Figura B.4: Combinação linear dos classificadores fracos, no final do treinamento do AdaBoost.