Universidade Católica Dom Bosco
Curso de Bacharelado em Engenharia de Computação

Máquinas de Vetores de Suporte Aplicadas à Classificação de Defeitos em Couro Bovino

Ricardo Cezar Bonfim Rodrigues
Roberto Henrique da Rocha Viana

Prof. Orientador: Hemerson Pistori

Relatório Final submetido como um dos requisitos para a obtenção do título de Bacharel em Engenharia de Computação.

UCDB - Campo Grande - MS - Novembro/2007

às nossas famílias e amigos.

Agradecimentos

Primeiramente a Deus por todas as oportunidades concedidas.

Às nossas famílias, especialmente nossos pais, Jorge Antonio Rodrigues e Zilda Bonfim Rodrigues, José Roberto Viana dos Santos e Jussara Maciel da Rocha Viana, que sempre nos apoiaram em cada etapa de nossas vidas, nos ajudando e nos incentivando em tudo.

Ao nosso orientador Professor Dr. Hemerson Pistori pela paciência, dedicação, companheirismo e incentivo que nos ajudou a prosseguir os estudos nesta área.

A todos os professores da Universidade Católica Dom Bosco pela contribuição na nossa formação das mais diferentes maneiras (aulas, conversas nos corredores, exemplos de vida, ...).

Aos amigos e amigas do GPEC (Grupo de Pesquisa em Engenharia e Computação) pelas horas de trabalho em grupo, pela companhia, pelas reuniões semanais, pelos bate-papos e pelos almoços.

Aos colegas da graduação pela convivência e amizade durante todo o curso, em especial nosso grande amigo André Luiz Pasquali, vulgo "Dedé".

Resumo

Este trabalho propõe uma solução para a classificação de defeitos em imagens do couro bovino utilizando o algoritmo de aprendizagem supervisionada Máquinas de Vetores de Suporte (Support Vector Machines - SVM) combinado a técnicas de extração de atributos. Foram realizados experimentos para a otimização de parâmetros das SVMs e uma análise comparativa com os resultados classificação de algoritmos bem conhecidos como, Árvores de Decisão e Redes Neurais. Os experimentos utilizaram diferentes implementações de SVMs para a classificação de imagens do couro bovino em diferentes estágios da cadeia produtiva. Os resultados obtidos pelas SVMs, mostraram a sua aplicabilidade ao problema e apresentaram excelentes resultados quando comparados à outros métodos de classificação.

Conteúdo

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

A cadeia produtiva de gado é de grande importância para a economia brasileira e o Brasil é considerado um dos maiores produtores de gado do mundo [MFF04], entretanto de acordo com [dC02] apenas 8.5% do couro brasileiro é considerado de alta qualidade, o que é prejudicial para o setor. Recentemente, a Empresa Brasileira de Pesquisa Agropecuária (EMBRAPA) sugeriu uma busca por um método de padronização do sistema de classificação de couro cru[1].

Hoje, em diversos processos industriais, defeitos em madeiras, metais, couros, tecidos entre outros são classificados manualmente por especialistas [YP01]. Em geral, esta tarefa consiste em analisar visualmente a textura da superfície do produto à procura de falhas, um trabalho minucioso e cansativo que torna comum a ocorrência de erros durante a análise.

A automatização deste processo poderia reduzir os problemas que ocorrem com o sistema atual, aumentar a produtividade e permitir a remuneração diferenciada pela classificação do produto. Uma possível solução para automatizar este processo de inspeção visual de peles de couro bovino seria modelá-lo utilizando técnicas de visão computacional, como relatado em [YP01, Sob05, KP02, GKA03, KGA04, AMGA97, PPM+06], no entanto, o couro bovino é considerado uma superfície complexa, pois pode apresentar muitas variações na aparência como: cor, brilho, espessura dentre outros [GKA03].

Este trabalho está incluído no módulo de classificação do sistema DTCOURO[2], como pode ser visualizado na fase 3 da Figura 1.1. O DTCOURO propõe o desenvolvimento de um processo automatizado de classificação do couro bovino em diferentes estágios da cadeia produtiva, onde a análise é

[1]Instrução normativa número 12, 18 de Dezembro de 2002 Ministério Brasileiro de Agricultura, Pecuária e Abastecimento

[2]Projeto para a Detecção Automática de Defeitos em Peles e Couros Bovinos

realizada a partir de imagens digitais coletadas do produto alvo, utilizando técnicas de visão computacional que envolvem processamento de imagens, inteligência artificial e reconhecimento de padrões.

Dentre as várias técnicas de aprendizado supervisionado existentes, as Máquinas de Vetores de Suporte (Support Vector Machines- SVMs) têm sido amplamente utilizadas na área de classificação de padrões por demonstrarem um grande poder de generalização e capacidade de manipular grandes volumes de dados. Este estudo apresenta o uso de técnicas de extração de atributos combinadas com SVMs para o problema de classificação de defeitos do couro bovino. Em experimentos preliminares foram realizadas análises comparativas com os algoritmos de classificação *AdaBoost* e Redes Neurais mostrando resultados promissores, além da validação de um processo de seleção de parâmetros para SVMs através de métodos estocásticos.

No capítulo 2 são apresentados alguns trabalhos correlatos, em seguida, o capítulo 3 descreve o processo de classificação do couro , métodos de extração de atributos e algoritmos de classificação que serão comparados às SVMs. No capítulo 4 são fundamentados o método de máquina de vetores de suporte e duas implementações utilizadas neste trabalho, a seção 5 descreve os experimentos realizados, na seção 6 as conclusões e projeções futuras.

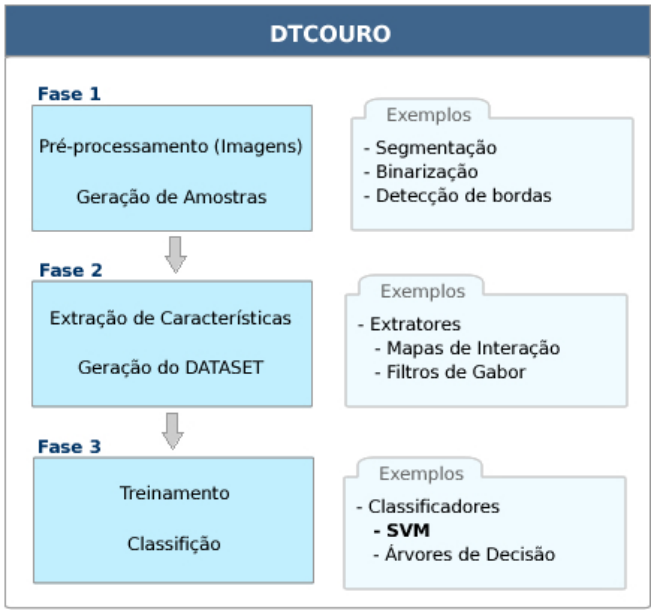


Figura 1.1: Diagrama geral de fases do sistema DTCOURO

Capítulo 2

Trabalhos Correlatos

A discussão dos trabalhos correlatos será dividida em 3 partes principais. Inicialmente, são analisados trabalhos na área de classificação de couro bovino, em seguida, uma revisão na utilização de Máquinas de Vetores de Suporte na área de reconhecimento de padrões e classificação. Por fim, serão apresentados trabalhos relacionados com a utilização de métodos estocásticos para otimização de parâmetros das Máquinas de Vetores de Suporte.

2.1 Classificação do couro

Em [YP01], Yeh and Perng propõe um método semi-automático para extração e detecção de defeitos em couro bovino nos estágios *wet blue* e couro cru. Os resultados de seus trabalhos se apresentaram confiáveis quando comparados com as classificações manuais por especialistas da área, mas os autores também apontam a desvantagem da necessidade de um especialista na fase de contagem total de defeitos.

Um método de inspeção de couro baseado em *Haar's wavelets* é apresentado por Sobral em [Sob05]. Segundo Sobral, o método apresenta resultados equivalentes ao de humanos experientes e em tempo real [Sob05], além de superar métodos propostos anteriormente, como o descrito por Kumar e Pang em [KP02]. Apesar de não esclarecido claramente por Sobral, aparentemente seu sistema foi testado apenas em couro no estágio final de produção, um problema bem mais simples comparado com outros estágios de produção como *wet blue* ou couro cru.

Uma medida de similaridade baseada em $\chi 2$ compara histogramas em tons de cinza extraídos de amostras ("janelas"de 65×65 *pixels*) de imagens em estágio *wet blue* com um histograma médio de imagens sem defeito em [GKA03]. Os resultados do $\chi 2$ e o limiar encontrado experimentalmente

são utilizados para segmentar regiões defeituosas. O método não foi usado na identificação dos tipos de defeito. Uma outra abordagem utilizando histograma e atributos extraídos por co-ocorrência é investigada em [KGA04].

2.2 Máquinas de Vetores de Suporte

Osuna utiliza SVM para detecção de faces em imagens em [OFG97] e demonstra a aplicabilidade das SVMs na detecção de faces em imagens, apresentando resultados equivalentes ou superiores a outros métodos baseados em exemplos [OFG97]. O sistema de Osuna percorre imagens em tons de cinza e é capaz de encontrar faces em diferentes escalas e grau de iluminação.

Em [Joa98] Joachims apresenta evidências teóricas e práticas do bom desempenho de classificação das SVMs em problemas de categorização de textos. O desempenho das SVMs é calculado com base em dois conjuntos de dados cada um contendo mais de 20000 casos de teste com cerca de 10000 atributos. Entre outros algoritmos testados no artigo, o K-NN foi o que apresentou melhor desempenho e foi superado pela SVM em 63 das 90 categorias do primeiro conjunto de testes e em todas as 23 categorias do segundo conjunto. Apesar do tempo de treinamento das SVM ser superior ao do K-NN, o ganho na classificação é compensador para este problema.

Uma outra aplicação das SVMs na área de Visão Computacional é apresentada em [PV98], onde Pontil e Verri realizam experimentos na classificação de objetos aplicando vários tipos de modificações como: adição de ruído, remoção de partes do objeto, alteração da localização espacial e combinações entre os mesmos. Os resultados foram comparados com *Perceptrons* e se mostraram superiores. Os autores atribuem estes resultados ao maior poder de generalização das SVMs [PV98].

2.3 Otimização de parâmetros

Um fator de muita importância na utilização de SVMs para classificação de problemas reais é a escolha dos parâmetros de configuração. Os métodos de escolha destes parâmetros variam muito e são conhecidos como *tuning*. A escolha manual destes parâmetros é indesejável pois é imprecisa e não garante qualidade nos resultados [IL04]. No método Grid-Search, descrito em [CL01], uma quantidade x de combinações para os parâmetros de Máquinas de Vetores de Suporte C e γ são testados dentro de uma margem definida, os parâmetros que apresentarem o melhor resultado de classificação em um teste de validação cruzada são escolhidos. A precisão do método ainda é

dependente de uma certa configuração manual, pois precisam ser determinados a margem de teste e o nível de variação dos parâmetros. O método pode se tornar computacionalmente caro dependendo das configurações aplicadas. Uma outra abordagem que se mostra interessante é a utilização do algoritmo Têmpera Simulada (*Simulated Annealing*- SA) [KGV83], pois este método converge para os valores ótimos independente da configuração inicial dos parâmetros e também apresenta maior resistência a mínimos locais.

Capítulo 3

Fundamentação Teórica

Os conceitos apresentados nesta seção servem de base para o entendimento da proposta de trabalho deste projeto. As primeiras seções apresentam um conceito de reconhecimento de padrões e algumas considerações ao problema da classificação do couro. As seções seguintes fundamentam métodos que serão utilizados nos experimentos.

3.1 Reconhecimento de Padrões

O processo de classificação de padrões é basicamente o ato de identificar uma categoria de padrões, baseada em um conjunto de dados. Este processo é muito utilizado em atividades diárias como: identificação de face, leitura de escrita manual, identificação de objetos pelo tato, entre outros. Dessa forma, a construção ou programação de máquinas capazes de reconhecer padrões é muito útil em várias áreas como, por exemplo: identificação de cadeias de DNA, reconhecimento de impressões digitais e reconhecimento de escrita manual [TK99].

Um dos métodos que vem sendo muito utilizado na definição de características do classificador é o treinamento [TK99], ou seja, a utilização de amostras de padrões para "ensinar"o classificador. Este método é interessante, pois na maioria dos problemas reais leva-se muito tempo para supor quais são os melhores atributos para classificação. Então a criação de um classificador envolve a definição de um modelo geral do mesmo juntamente com o treinamento de padrões para a aprendizagem dos padrões desconhecidos.

A aprendizagem pode ser feita de forma supervisionada ou não supervisionada. Na forma supervisionada, exemplos de cada padrão em um conjunto de treinamento são fornecidos, ou seja, amostras de casos previamente clas-

sificados são utilizados durante o treinamento [TK99]. No método não supervisionado o sistema forma conjuntos "naturalmente"a partir dos padrões de entrada, isto significa que baseado em medidas de similaridade, o classificador é responsável por arranjar seus pesos para representar as classes de padrões.

3.2 Classificação do Couro

A alta qualidade do couro é muito importante em diversos seguimentos da indústria como, de sapatos, bolsas, roupas etc. A boa aparência de produtos fabricados usando couro depende das suas regiões que não apresentam defeitos, ou seja, características na superfície do couro que possam prejudicar a aparência final do produto. O couro bovino, em particular, apresenta defeitos que ocorrem desde a fase produtiva do animal até o seu abate.

A fase produtiva ocorre ainda nas propriedades rurais e é nesta fase que cerca de 60% dos defeitos encontrados no couro bovino são originados [Gom00]. Dentre eles os defeitos mais comuns são: marcas deixadas por ectoparasitos (carrapatos e bernes entre outros), marcas de ferro (marcas de proprietários feitas com ferro quente para identificação do gado), infecções, riscos e esfolas causados por manejo inadequado do gado (ferimentos com arame farpado, ferrão, galhos e etc). Grande parte dos defeitos que aparecem nesta fase se devem a falta de estrutura e manejo inadequado do gado por seus criadores.

Durante o transporte e fase de abate do gado, o manejo inadequado continua sendo um dos principais causadores de defeitos no couro. Grande parte dos danos (riscos e esfolas) são causados por chifradas, quedas e má conservação dos meios de transporte tais como caminhões com pontas de parafusos aparentes [Gom02]. A Figura 3.1 mostra o manejo do gado em um frigorífico.

Não existe um padrão industrial universal para a classificação de defeitos na superfície de couros [YP01]. Fica a cargo de cada indústria analisar e definir o percentual que poderá ser utilizado de cada região com defeitos. Os defeitos no couro podem ser observados em *couro cru*, a peça de couro sem curtimento, ou em *wet blue*, fase pré manufaturada do couro, como podem ser visualizados na Figura 3.2 . A *wet blue* pode ser definida como uma fase intermediária do couro entre o couro não curtido e o couro final, pronto para ser utilizado. Em geral, a detecção e classificação dos defeitos do couro são conduzidas com o couro *wet blue*, já que o couro cru apresenta uma complexa superfície com diferentes texturas, cores, formas e espessuras.

Figura 3.1: Imagens do manejo gado em um frigorífico

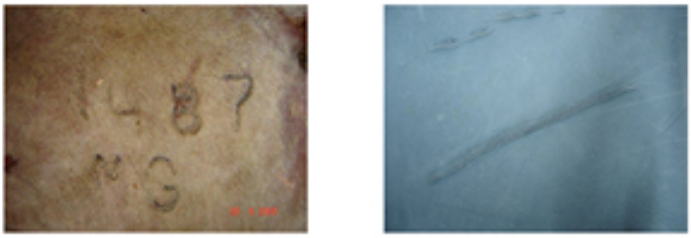


Figura 3.2: (a) Imagem de uma peça do couro bovino no estágio cru (antes de ser curtido). (b) Imagem de uma peça do couro bovino no estágio *wet blue*.

3.3 Extração de atributos

A extração de atributos sobre uma imagem tem como finalidade destacar características, similaridades ou diferenças que possam ser relevantes para uma posterior análise ou classificação. Em uma imagem do couro, por exemplo, podem ser extraídas características como, cores, ângulos, formas entre outras, a diferença destas informações ajudam na identificação dos tipos de defeito. Identificar e extrair os atributos que são pertencentes ao conjunto de interesse pode não ser uma tarefa simples, devido a esta grande variação de cores, iluminação, ângulos entre muitos outros atributos. Esta seção apresenta alguns métodos que são capazes de extrair atributos baseados em textura e em cores da imagem.

3.3.1 Matriz de Co-ocorrência

Matrizes de co-ocorrência em níveis cinza ou GLCM (Gray level co-occurrence matrix) representam um dos métodos mais conhecidos para extração de características de texturas em imagens [MPV02]. Assumindo que a imagem esteja em 8 bits (tons de cinza), ou seja, com a intensidade de cada pixel variando de 0 a 255, este método estatístico pode então ser aplicado.

As GLCMs representam as probabilidades de ocorrência de *pixels* com mesmos valores de similaridade dados dois parâmetros: distância entre *pixels* δ e orientação θ [JC04]. Em outras palavras, a partir de uma matriz de uma imagem 8bits uma nova matriz resultante armazena a co-ocorrência dos valores de similaridade de pixels da matriz original de acordo com os parâmetros δ e θ. Diferentes medidas estatísticas podem ser utilizadas para extrair características da textura da imagem. Estatísticas para estimar as similaridades como energia, entropia e correlação entre outras, são apresentadas em [JC04]

Para exemplificar, a Figura 3.3a mostra uma matriz m de 3×3 onde cada posição da matriz $m(i, j)$ representa a intensidade do pixel variando de 0 a 2. Neste exemplo são extraídas informações sobre a relação entre os valores dos pares de *pixels* com os parâmetros δ e θ em 1 e 0 grau, respectivamente. Em outras palavras, pares de *pixels* serão comparados com variação de ângulo 0 em relação aos pares anteriores e com distância de *pixels* 1. A medida que os pares são analisados sua relação é incrementada na matriz de co-ocorrência Figura 3.3b.

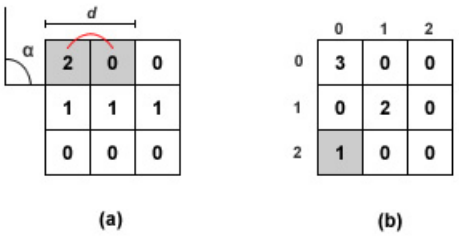


Figura 3.3: (a) Matriz de pixels (original). (b) matriz de co-ocorrência de relação de valores dos pares encontrados na matriz original

3.3.2 Mapas de Interação

Esta técnica consiste em uma análise direcional da textura que é baseada no cálculo das diferentes variações encontradas em um histograma da diferença d dos tons de cinzas GLDH (gray-level difference histogram) [Che99]. A análise sobre a imagem consiste em características de interações de pares de *pixels* no domínio espacial de um GLDH [Che96]. O mapa polar de interações $M_{pl}(i, j)$ é a entidade básica que representa a intensidade codificada das características de um GLDH e a sua representação cartesiana é utilizada para encontrar as interações mais significantes.

Como é ilustrado na Figura 3.4a, os *pixels* da imagem são escaneados por (m, n) em localizações dadas por um ângulo α e distância d. A intensidade é calculada no ponto (α_i, dj), interpolando os *pixels* vizinhos (ver Figura 3.4b) para gerar o mapa de interações polares $M_{pl}(i, j)$ diferente das matrizes de co-ocorrência onde a intensidade é o valor do *pixel*. A interação deste mapa com os valores originais resulta em similaridades e diferenças assim como nas matrizes de co-ocorrência.

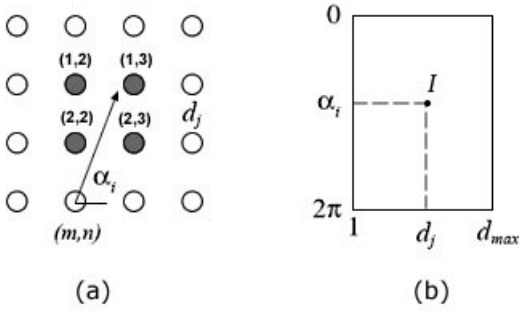


Figura 3.4: (a) Cálculo do EGLDH (extended gray-level difference histogram). (b) Mapa polar de interações, onde I é a intensidade

3.3.3 Espaço de cores HSB

A representação do espaço de cores HSB (Hue,Saturation and Brightness) pode ser utilizada para extrair características de matiz (hue), saturação (saturation) e brilho (brightness) sobre uma imagem. A matiz é uma especificação intrínseca da cor, a saturação descreve a quantidade de cinza em uma cor, o componente brilho mede a quantidade de iluminação sobre uma cor. A Figura 3.5 é a representação gráfica do espaço de cores HSB. Este método possui a vantagem de ter atributos invariantes, diferente de outros espaços de cores como o RGB [1] (Red, Green, Blue) [WS03].

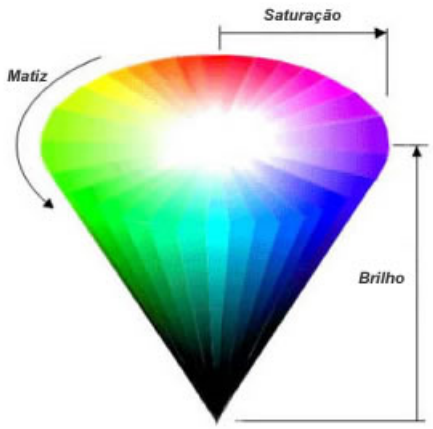


Figura 3.5: Representação do modelo de espaço de cores HSB

3.4 Adaboost e MLPs

O adaBoost é um meta-algoritmo na área de aprendizagem de máquina utilizado para melhorar a performance de um dado algoritmo de aprendizagem. A idéia básica do algoritmo é a combinação de vários classificadores "fracos" na geração de um classificador mais preciso. Com este propósito, a cada passo o adaBoost faz chamadas a um algoritmo de classificação e aumenta o peso dos exemplos classificados incorretamente, forçando o algoritmo de classificação a focar os exemplos mais difíceis. Uma descrição mais detalhada pode ser encontrada em [FS99]. Neste trabalho, após testes empíricos

[1] Método que utiliza as características sobre o nível de vermelho, verde e azul em cada *pixel* da imagem

realizados anteriormente com classificadores considerados “fracos” foram escolhidos o J48 (árvore de decisão) e o IBK (K vizinhos mais próximos), ambos implementados em Java e integrados ao Weka [WF05].

O J48 ou c4.5 separa os dados através de uma árvore de decisão que é construída com base nos atributos do conjunto de treinamento. A cada interação o algoritmo seleciona o atributo que melhor diferencia os dados com base no ganho de informação e separa o conjunto em duas partes até que restem apenas folhas, ou seja, respostas de classificação. As árvores de decisão podem ser facilmente entendidas e convertidas em regras de produção, permitindo uma avaliação rápida dos casos de teste.

O algoritmo k vizinhos mais próximos (k-nearest neighbor - KNN) é um dos algoritmos mais simples na área de aprendizagem de máquina. A classificação dos dados consiste na soma dos votos dos vizinhos onde a classe mais comum entre os k vizinhos mais próximos é atribuída ao elemento em teste. Os vizinhos são selecionados de um conjunto de dados previamente classificados representados por vetores de atributos em um espaço multidimensional. As distâncias de Manhattan e Euclideana são comumente utilizadas como medidas de similaridade.

O Perceptron Multi Camada (Multilayer Perceptron - MLP) é basicamente um conjunto de unidades de processamento agrupadas em camadas, onde o número de camadas e o número de elementos em cada camada varia de acordo com o problema, a primeira camada recebe os dados de entrada, a última camada determina o resultado, e todas as camadas entre elas são chamadas de camadas ocultas. Na camada de saída existe uma unidade para cada classe no conjunto de treinamento. As unidades em uma camada geralmente são conectadas com todas as unidades das camadas anteriores e posteriores a dela e possuem valores de peso que denotam seu comportamento, que são ajustados durante a fase de treinamento. Após a fase de treinamento, os pesos de cada unidade são ajustados na rede para todos os casos de teste, até que se obtenha um valor de saída em cada uma das unidades da última camada. É esperado que a classe correta obtenha o maior valor dentre as unidades de saída.

Capítulo 4

Máquinas de Vetores de Suporte - SVM

A SVM é uma técnica usada para o treinamento de classificadores baseada no conceito da minimização do risco estrutural [Bur98], e foi desenvolvida por Vladimir Vapnik em 1979 e está sendo amplamente utilizada desde a década de 90 em vários problemas de classificação e reconhecimento de padrões como: detecção de face em imagens [OFG97], categorização de textos [Joa98] e reconhecimento de Objetos [PV98].

Uma das grandes vantagens da SVM é seu alto poder de generalização. Isto ocorre pois a complexidade da hipótese não depende do número de atributos, mas sim da margem com que eles separam os dados [Joa98]. Este fator é muito interessante quando lidamos com problemas de classificação baseados em imagens, pois a dimensão dos seus vetores de atributos é geralmente grande.

As SVMs são classificadores lineares que separam os dados em duas classes através de um hiperplano de separação. Dado um espaço de atributos onde os dados podem assumir apenas 2 valores $(+1, -1)$, um hiperplano separa estes dados de forma que para todos os pontos de um lado são atribuídos o valor 1, e para todos os pontos do outro lado são atribuídos o valor -1.

Um hiperplano ótimo separa os dados com a máxima margem possível, que é definida pela soma das distâncias entre os pontos positivos e os pontos negativos mais próximos do hiperplano. Estes pontos são chamados de vetores de suporte e estão circulados na Figura 4.1. O hiperplano é construído com base em treinamento prévio em um conjunto finito de dados [Vap99].

Assumindo o conjunto de treinamento $\{x_i, y_i\}$, $y_i \in \{\pm\}$, $x_i \in R^d$ onde x_i é o i-ésimo elemento de entrada e y_i é o seu respectivo valor de classe para $x_i, i = 1, ..., l$. O cálculo do hiperplano com margem ótima é dado pela minimização de $||w||^2$ obedecendo as seguintes restrições:

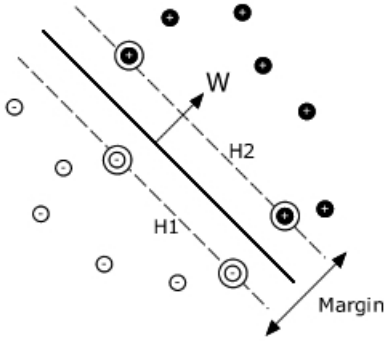


Figura 4.1: Classificação de um conjunto de dados usando uma SVM linear.

$$y_i \left(x_i \cdot w + b \right) - 1 \geq 0, \forall_i \tag{4.1}$$

Onde w é a normal ao hiperplano. Este é um problema quadrático de otimização, e pode ser transformado para forma *dual*, onde depende apenas dos Multiplicadores de Lagrange a_i:

$$u \equiv \sum_{i=1}^{N} a_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} a_i a_j y_i y_j (x_i \cdot x_j) \tag{4.2}$$

respeitando as restrições da equação linear:

$$\sum_{i}^{N} a_i y_i = 0, \tag{4.3}$$

e as restrições da inequação:

$$a_i \geq 0, \forall_i \tag{4.4}$$

Com a solução dada por:

$$w = a_i y_i x_i \tag{4.5}$$

Onde N é o número de exemplos de treinamento. Os vetores de suporte estão localizados nos planos H_1 e H_2 da Figura 4.1. Estes são os pontos mais importantes, pois são eles que definem a margem de classificação da SVM. Remover ou mover estes pontos do *dataset* mudaria todo o processo de treinamento realizado, ao mesmo passo que todos os outros pontos podem ser removidos que o resultado do treinamento seria o mesmo [Bur98].

Infelizmente para a maioria dos problemas reais o conjunto de dados não é separável através de um hiperplano linear, e o cálculo dos vetores de

suporte utilizando as formulações descritas acima não se aplicam [Pla98]. Este problema é geralmente resolvido através da introdução de variáveis de alargamento de margem ξ_i que "relaxam"as restrições da SVM linear permitindo algumas falhas na margem, mas também penaliza as falhas através da variável de controle C. Estas modificações alteram o problema de otimização (1) para:

$$\frac{1}{2}||w||^2 + C\sum_{i=1}^{N}\xi_i, \tag{4.6}$$

respeitando,

$$y_i\left(x_i \cdot w + b\right) - 1 + \xi_i \geq 0, \forall_i \tag{4.7}$$

A transformação deste problema de otimização para sua forma *dual* apenas altera a restrição (4) para:

$$0 \leq a_i \leq C, \forall_i \tag{4.8}$$

Uma outra solução para o problema da não linearidade dos dados é a aplicação de funções de mapeamento sobre o conjunto de dados de treinamento. Estas modificações podem ser aplicadas mapeando o conjunto de dados em um espaço Euclideano de alta dimensão (possivelmente infinita), estes mapeamentos são feitos utilizando funções de *kernel* (núcleo) na fase de treinamento[Bur98]. O cálculo da é definido por:

$$u \equiv \sum_{i=1}^{N} a_i y_i K\left(x_i, x\right) - b, \tag{4.9}$$

Onde a minimização dos Multiplicadores de Lagrange ainda é um problema quadrático,

$$\min_a \sum_{i=1}^{N} a_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} K\left(x_i, x\right) a_i a_j y_i y_j, \tag{4.10}$$

Aplicando as condições de Karush-Kuhn-Tucker (KKT) nas equações acima obtemos as seguintes condições para o cálculo do ponto ótimo de um problema quadrático positivo definido. Desta forma, o problema quadrático está resolvido quando para todo i:

$$\begin{aligned} a_i = 0 &\Leftrightarrow y_i u_i \geq 1, \\ 0 < a_i < C &\Leftrightarrow y_i u_i = 1, \\ a_i = C &\Leftrightarrow y_i u_i \leq 1 \end{aligned} \tag{4.11}$$

Onde u_i é a saída da SVM para o i-ésimo exemplo de entrada. Para garantir que o problema quadrático definido na equação (4.2) seja positivo definido a função de *kernel* K deve obedecer as condições de Mercer[Bur98]

4.1 Implementações de SVM

Nesta seção são descritas duas implementações de SVM em Java [1], que são utilizadas nos experimentos. Estas implementações foram selecionadas pois são amplamente utilizadas por pesquisadores de todo o mundo, apresentando portanto um alto grau de confiabilidade.

4.1.1 Mínima Sequência de Otimização - SMO

O SMO é um algoritmo para treinamento de SVM desenvolvido por John C. Platt em 1998. Em [Pla98], Platt afirma que o SMO é uma técnica simples e rápida para a solução do problema quadrático da SVM, o algoritmo divide o problema quadrático em vários subproblemas mais simples e que podem ser resolvidos sem nenhum tipo de armazenamento de dados, reduzindo o uso de memória.

A principal vantagem do SMO é que ele seleciona sempre o menor problema quadrático para otimizar a cada interação, e como este problema sempre envolve apenas dois multiplicadores de Lagrange, a cada interação o SMO busca estes dois multiplicadores, executa a otimização e ajusta seus valores na SVM [Pla98]. A escolha dos multiplicadores é baseada em uma heurística.

Como a otimização de dois multiplicadores de Lagrange pode ser resolvida analiticamente, este processo dispensa cálculos numéricos custosos, por outro lado aumenta o número de interações até a resposta final. Apesar do aumento no número de interações o SMO se mostrou mas rápido quando comparados com outros algoritmos como "chunking"[Pla98]. Platt atribui estes resultados ao ganho em velocidade que se tem resolvendo analiticamente os multiplicadores de Lagrange. Naturalmente o SMO não possui nenhuma abordagem multiclasse, porém a ferramenta Weka oferece suporte a multiclasse através da classe *MultiClassClassifier*.

4.1.2 Implementação de *C*-SVC em LibSVM

O LibSVM é uma biblioteca de implementações de SVM desenvolvida por Chin-Chung Chang [CL01] com várias finalidades: classificação, regressão e

[1] Linguagem de programação orientada objeto desenvolvida pela Sun Microsystems

estimativa de distribuição. O algoritmo de classificação implementado na biblioteca leva o nome de *C*-SVC.

Para resolução do problema quadrático o *C*-SVC utilizada a mesma abordagem do SMO, ou seja, decompõe o conjunto de Multiplicadores de Lagrange em um subconjunto menor. Porém ele não seleciona apenas dois operadores arbitrariamente como no SMO mas sim um subconjunto com tamanho variável [CL01].

Além da seleção de um subconjunto para otimização o *C*-SVC também implementa as técnicas *Shrinking* e *Caching* para redução do tempo computacional. O *Shrinking* tenta reduzir o tamanho do problema quadrático a ser resolvido eliminando Multiplicadores de Lagrange que não poderiam ser alterados com base em uma heurística demonstrada em [CL01]. A técnica de *Caching* simplesmente armazena cálculos de matrizes utilizados recentemente para utilizações futuras, reduzindo parte dos cálculos de *kernel* realizados nas interações finais.

Na resolução de problemas multiclasse o *C*-SVC utiliza o método "um-contra-um", que consiste na resolução de um problema de duas classes para cada classe, atribuindo para todos os pontos um valor de classe baseado em uma estratégia de votação. Para pontos com valores de classe idênticos, um algoritmo seleciona o de menor índice.

Capítulo 5

Experimentos

Esta seção descreve os detalhes sobre os experimentos conduzidos para este trabalho e as análises de seus resultados. Inicialmente é apresentado o experimento de otimização de parâmetros das SVMs e uma comparação com outros métodos de classificação em imagens do couro no estágio *wet blue*, em seguida utilizando os mesmos métodos de avaliação e um conjunto de imagens maior, um experimento com imagens do couro no estágio cru. Por fim, um experimento com classificações visuais é apresentado utilizando imagens do couro cru e *wet blue*.

5.1 Otimização de parâmetros da SVM

O desempenho de SVMs, como muitos outros algoritmos de aprendizagem de máquinas, é muito sensível a configurações de parâmetros, principalmente em problemas do mundo real. Este experimento apresenta duas implementações de SVMs bem conhecidas e muito utilizadas, Weka SMO e LibSVM, que foram comparadas usando *Simulated Annealing* (SA) para configurações de parâmetros que resultem em melhores classificações. Esta aproximação aumenta significantemente a precisão da classificação sobre a configuração padrão das implementações Weka SMO e LibSVM.

O experimento também apresenta uma avaliação do algoritmo AdaBoost, para resolver o problema de classificação do couro bovino. A idéia básica do algoritmo é um método geral que produz previsões muito precisas, combinando moderadamente classificadores pouco precisos (fracos) [FS99]. Os resultados são promissores apesar de não serem superiores aos alcançados pelas outras implementações.

5.1.1 Conjunto de treinamento (*Dataset*)

Para gerar a base de aprendizagem quatro imagens do couro bovino no estágio *wet blue* foram selecionadas do repositório do projeto DTCOURO. As imagens foram capturadas usando uma câmera digital de cinco *megapixels* durante visitas técnicas a curtumes localizados na região de Mato Grosso do Sul, Brasil em setembro de 2005. Para este experimento as imagens foram redimensionadas de 2592×1944 para 600×450 *pixels* com a intenção de economizar tempo de processamento. Testes empíricos mostraram que não há perda na taxa de classificação com o uso da escala adotada. Além do mais, as imagens foram capturadas sem variações de ambiente como iluminação, ângulo e distância.

Um conjunto de quatro tipo de defeitos foi escolhido, são eles, marca de carrapato, marca ferro, risco e sarna. Estes defeitos foram escolhidos porque são comuns no rebanho brasileiro e porque possuem um número de amostras mais significativo que os demais defeitos no banco de imagens do projeto DTCOURO. Os defeitos foram segmentados manualmente com ajuda da ferramenta DTCOURO. Um total de vinte e três segmentações foram realizadas sobre as imagens, incluindo exemplos dos defeitos previamente citados e regiões do fundo onde se encontravam as peças de couro. Após a segmentação manual dos defeitos, um algoritmo implementado para o projeto DTCOURO foi usado para extrair "janelas" de 20×20 *pixels* "varrendo" todas as segmentações. Desta maneira um total de 3809 amostras de 20×20 *pixels* foram criadas.

O próximo passo foi a extração de atributos para cada amostra. Um conjunto de trinta e oito atributos foi extraído usando Mapas de Interações [Che99] e Matrizes de Co-ocorrência [HBW99] para atributos de textura (35) e os valores médios do histograma para matiz, saturação e brilho no espaço de cores HSB (3). Neste trabalho, as características dos extratores são configuradas com os mesmos parâmetros usados em [AVRP06] e podem ser vistos na Tabela 5.1.

Para cada uma das 3809 amostras, um vetor de características $\bar{x}$ foi calculado e armazenado no *dataset*. Ao mesmo tempo, todas as amostras foram tituladas com uma das seguintes classes: *Carrapato, Marca Ferro, Risco, Sarna, Fundo*. A distribuição das classes é a seguinte: 998 $Carrapato$, 627 $MarcaFerro$, 509 $Risco$, 1045 $Sarna$ e 630 $Fundo$, onde o número de amostras em cada classe é proporcional a área de cada região de defeito da imagem original. A Figura 5.1 apresenta um diagrama com todas as etapas do processo de criação do *dataset*, utilizando como exemplo uma imagem do couro bovino.

Tabela 5.1: Parâmetros usados para extração de atributos no experimento de otimização. 35 características de textura foram extraídas para cada amostra

	Mapas Int.	Matrizes Co.
Ângulo inicial:	0	0
Ângulo final:	180	180
Variação do ângulo:	45	45
Distância inicial:	0	-
Distância final :	2	-
Variação da distancia:	1	1

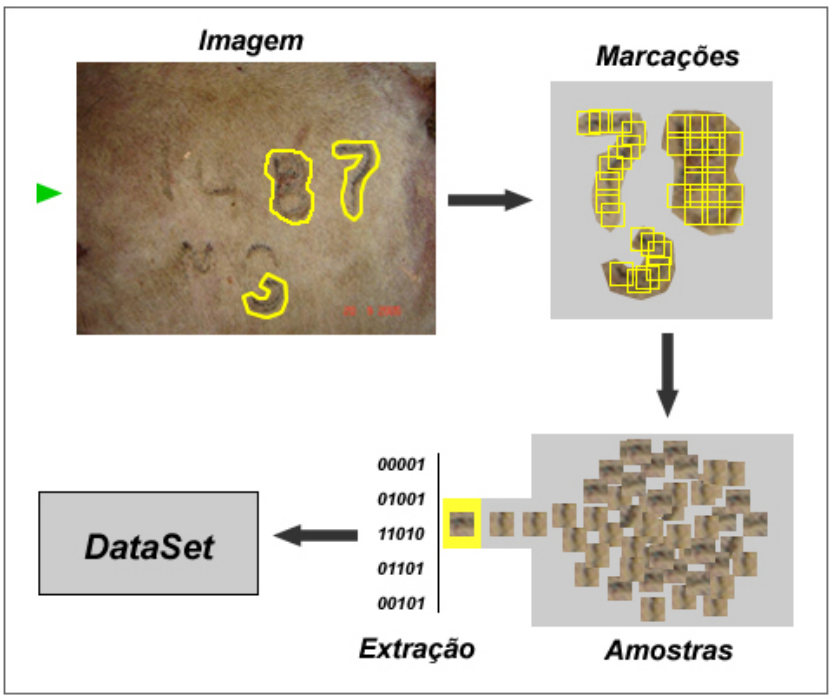


Figura 5.1: Modelo visual do processo de geração do conjunto de treinamento (*datset*).

5.1.2 Configurações do experimento

Os experimentos foram conduzidos utilizando a versão 3.5.2 do software Weka [1], a biblioteca LibSVM escrita por Chang e Lin[2] e a implementação WLSVM

[1] `http://www.cs.waikato.ac.nz/ml/weka/`
[2] `http://www.csie.ntu.edu.tw/~cjlin/libsvm/`

[3]. Duas diferentes implementações de SVMs, LibSVM e SMO foram testadas juntamente com o AdaBoost e J48. Para cada algoritmo foi utilizada uma validação cruzada com 10 dobras sobre o *dataset* para certificar uma estimação mais confiável da generalização do erro [IL04]. O processo de validação cruzada consiste na divisão do conjunto de dados em k subconjuntos (número de dobras), onde K-1 subconjuntos são utilizados na fase de treinamento do classificador e o subconjunto que restou é utilizado para teste. Esta rotação é feita até que todos os subconjuntos sejam utilizados para teste. A média dos k resultados parciais é o resultado final.

5.1.3 Avaliação dos algoritmos supervisionados

Os experimentos são basicamente exploratórios e foram conduzidos com a intenção de avaliar a eficácia e eficiência das SVMs e do AdaBoost. Os resultados são analisados usando medidas tradicionais incluindo, *precision* (captura o efeito do alto número de exemplos negativos na performance dos algoritmos), *recall* (representa taxa de positivos verdadeiros) e a área sobre a curva de ROC (Receiver Operator Characteristic) que mostra como o número de exemplo positivos classificados corretamente variam com o número de negativos classificados incorretamente [DG06].

Inicialmente, o objetivo dos experimentos era a busca de melhores parâmetros C e γ para SVMs. O algoritmo de busca *Simulated Annealing* foi aplicado sobre o conjunto de treinamento, onde para cada interação novos valores de C e γ eram calculados e configuravam novas SVMs à procura de melhores resultados. A medida de avaliação utilizada para a busca foi a minimização de erro, ou seja, quantidade de exemplos classificados corretamente. Os valores de partida para os parâmetros C e γ são os definidos como padrão de cada implementação. Na Tabela 5.2 podem ser vistas as informações e resultados da busca como: tempo de execução, melhores valores dos parâmetros C e γ e seus respectivos percentuais do total de acertos, usando os valores padrões (Pad. Ac) e os valores otimizados com o SA (Oti. Ac).

Como pode ser observado na Tabela 5.2, os resultados mostram claramente um alto desempenho atingido por ambas as implementações, apesar disso pode-se concluir que o uso de LibSVM é recomendado pela sua agilidade na fase de treinamento. Além do mais, o número total de interações para o LibSVM é 136 e o total de interações para o SMO é 142, indicando que a diferença em tempo não se deve ao comportamento dos parâmetros

[3]Yasser EL-Manzalawy and Vasant Honavar, WLSVM : Integração da LibSVM com o ambiente Weka, 2005. Software disponível em `http://www.cs.iastate.edu/~yasser/wlsvm`

Tabela 5.2: Tempo de execução, melhores C e γ, acertos com parâmetros padrões e acertos com parâmetros otimizados com o *Simulated Annealing* para estimação de parâmetros de SVMs usando as implementações LibSVM e SMO

	Tempo	Melhores C e γ	Pad. Ac.	Oti. Ac.
SMO	16.09hrs	143.957 0.932	96.928%	99.921%
LibSVM	2.06hrs	242.473 0.959	78.708%	99.894%

encontrados com o SA. Uma das possíveis razões para a diferença em tempo pode ser devido a implementação de *shrinking* e *caching* da LibSVM [CL01]. A Tabela 5.2 também mostra que a otimização do SA aumenta a acurácia da classificação em 26% sobre os parâmetros padrão de configuração da LibSVM e 3% sobre o SMO padrão.

A matriz de confusão é um vetor bidimensional $|Y| \times |Y|$, onde a posição (i, j) denota o número de exemplos da classe i classificados como exemplos da classe j. Elas podem ser usadas para comparar os classificadores pela combinação de seus elementos em fórmulas mais sofisticadas como, *precision*, *recall* e área sobre a curva ROC. A fórmula tradicional para a *precision*, ou precisão, é dada por:

$$P = \frac{tp}{tp + fp}$$

Onde tp é o número de positivos verdadeiros e fp é o número de positivos falsos. *Precision* é a relação entre os exemplos classificados corretamente de uma dada classe sobre o número total dos exemplos atuais da classe. Do outro lado, *recall* é definido como a relação entre o número de exemplos classificados corretamente de uma dada classe e o número total de exemplos classificados para a classe. *Recall* é geralmente chamado de sensibilidade e é tradicionalmente definido por:

$$R = \frac{tp}{tp + fn}$$

Onde fn é o número de falsos negativos. Na Figura 5.2 é possível observar o comportamento dos algoritmos com respeito a *precision*, *recall* e área sobre a curva de ROC. Note que SVMs obtiveram melhores resultados quando treinadas com os parâmetros otimizados.

A alta precisão, os valores de *recall* e a área sobre a curva ROC demonstram a conveniência de algoritmos de aprendizagem supervisionada para o problema de classificação de defeitos. Em adição, pode ser concluído que

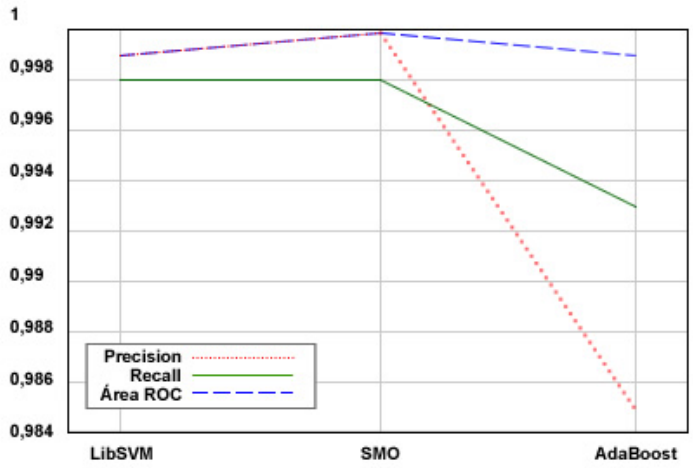


Figura 5.2: A *precision*, *recall* e área sobre a curva ROC para os três algoritmos: LibSVM, SMO e AdaBoost. O SMO apresenta uma pequena vantagem sobre os outros dois.

o conjunto de características extraídas das imagens originais favorecem a efetividade do classificador.

SVMs e AdaBoost tem mostrado excelentes e similares performances com relação a tarefas de classificação. A eficiência dos algoritmos durante a fase de testes também é interessante. A Tabela 5.3 mostra o tempo de execução para as fases de teste e treinamento, assim como a respectiva acurácia dos três classificadores. Note que a fase de treinamento do AdaBoost é melhor em termos de eficiência. Isto é justificado pelo fato do tempo para o cálculo dos exemplos de testes ser proporcional ao número de classificadores de base multiplicado pela altura de cada árvore de decisão. No caso das SVMs, ambas apresentam tempos de execução parecidos, que é proporcional ao número final de vetores de suporte. A LibSVM apresenta o melhor tempo durante o treinamento, correspondente ao tempo de treinamento com os parâmetros já selecionados pelo *Simulated Annealing*. A acurácia apenas confirma que todos os classificadores podem discriminar os defeitos com muita precisão.

Tabela 5.3: Tempo de teste e treinamento para AdaBoost e SVMs depois da seleção de parâmetros pelo SA. Note o rápido tempo de avaliação do AdaBoost quando comparado com SVMs sem perda eficácia.

	Tempo de teste	Tempo de treinamento	Acurácia
SMO	1.008s	20.067s	99.921
LibSVM	0.742s	7.682s	99.895
AdaBoost	0.014s	23.701s	99.317

5.2 Classificação de Imagens no estágio Couro Cru

Esta seção descreve o experimento de classificação de imagens do couro no estágio cru. Nesse estágio a peça de couro não possui nenhum tipo de processamento, tornando a classificação de defeitos uma tarefa complexa e não trivial, devido a grande variação de textura e cor. O experimento foi realizado utilizando as mesmas metodologias do experimento anterior. Porém foram adicionadas novas configurações, novos testes comparativos com outros algoritmos como MLP (Redes Neurais) e IBK, além de um conjunto maior de imagens.

5.2.1 Conjunto de treinamento (*Dataset*)

Na construção do *dataset* foram utilizadas quinze imagens do repositório de imagens do DTCOURO no estágio couro cru de produção. As imagens na resolução de 2592×1944 foram capturadas em visitas realizadas pela EMBRAPA a curtumes e frigoríficos no ano de 2005. Posteriormente foram redimensionadas para 640×480, pois testes empíricos mostraram um tempo de processamento muito elevado e experimentos comprovaram que o redimensionamento não causava perda de acurácia na classificação.

Como as imagens selecionadas para o experimento não apresentam regiões que caracterizam uma área do couro bovino sem defeitos, o conjunto de treinamento representa as classes dos tipos de defeitos: marcas de carrapato, marca ferro, risco e sarna. Um exemplo de cada um dos defeitos pode ser visualizado na Figura 5.3, em seguida, utilizando as imagens selecionadas, trinta marcações manuais foram realizadas representando os defeitos citados.

O próximo passo na composição do *dataset* foi a extração de atributos a partir das amostras. Nesta fase foram extraídos 139 atributos de textura e 6 atributos de cor de cada amostra. Para atributos de textura foram utilizados Mapas de Interação [Che99] e Matrizes de Co-Ocorrência [HBW99]

Figura 5.3: Amostra de (a) carrapato, (b) marca ferro, (c) risco e (d) sarna sobre o couro crú.

Tabela 5.4: Parâmetros de configuração dos extratores de atributos.

	Mapas int.	Matriz Co.
Ângulo inicial:	10	10
Ângulo final:	180	180
Variação do ângulo:	10	10
Distância (pixels):	2	-
Variação da distância:	1	1

e para atributos de cor foram extraídos os valores médios de HSB e RGB. A configuração dos extratores baseados em textura estão apresentados na Tabela 5.4

Após a realização das marcações 14722 amostras de 20×20 foram extraídas e rotuladas a partir das marcações utilizando o extrator de amostras implementado no DTCOURO. Para cada umas das 1472 amostras um vetor $\bar{x}$ de atributos foi calculado e armazenado , em seguida foram atribuídos aos vetores um valor correspondente as classes : $carrapato, marca ferro, risco, sarna$, dessa forma a distribuição das classes no *dataset* é: 2819 amostras de carrapato, 3716 amostras de marca ferro, 2804 amostras de risco e 5383 amostras de sarna.

5.2.2 Avaliação dos algoritmos supervisionados

Apesar do foco deste experimento não ter sido a otimização de parâmetros, as duas implementações de SVM foram otimizadas utilizando a mesma metodologia do experimento anterior, seus resultados antes e depois da otimização são apresentados na Tabela 5.5

Como pode ser observado o LIBSM superou o SMO tanto em tempo de otimização quanto em taxa de acerto e mais uma vez o SA se mostrou

Tabela 5.5: Tempo de execução, melhor C e γ, acurácia padrão e acurácia com o *Simulated Annealing* para as implementações LibSVM e SMO.

	Tempo	Melhor C e γ	Pad. Ac.	Oti Ac.
SMO	35655s	24.165 0.931	88.95%	93.10%
LibSVM	12786s	49.494 1.008	76.16%	99.59%

eficiente na tarefa de otimização dos parâmetros melhorando em 23% a taxa de classificação do LibSVM e em 5% a do SMO.

Os métodos de avaliação dos algoritmos de classificação foram mantidos de acordo com o experimento anterior. Os resultados refentes a *precision*, *recall*, e área sobre a curva ROC estão apresentados na Tabela 5.6.

Tabela 5.6: Resultados de execução para *precision*, *recall*, e área sobre a curva ROC

	Roc	Recall	Precision
SMO	0.9979	0.9965	0.9879
BoostIBK	0.9916	0.9876	0.9870
BoostJ48	0.9999	0.9959	0.9946
MLP	**1.0000**	0.9978	0.9978
LibSVM	0.9991	**0.9983**	**0.9997**

Novamente os altos valores atingidos por ambos os algoritmos demonstram a conveniência do uso de algoritmos de aprendizagem supervisionada neste tipo de aplicação, além da representatividade do conjunto de características. Os resultados referentes aos tempos de execução e classificação dos algoritmos são apresentados na Tabela 5.7. O LibSVM e o MLP apresentaram resultados excelentes e similares quanto a taxa de classificação. Note que o tempo de teste do AdaBoost-J48 e do SMO são de longe os melhores. Isso se justifica, pois o tempo para avaliação da árvore de decisão é baseado no número de classificadores base e o tempo de avaliação das SMV é baseado no número de vetores de suporte. O AdaBoost-IBK apresenta o melhor tempo de treinamento seguido pela LibSVM. A alta taxa de classificação dos algoritmos apenas demonstra que todos discriminam os valores precisamente.

Tabela 5.7: Tempo de treinamento e teste para AdaBoosts,SVMs (depois ta otimização de parâmetros) e MLP.

	Tempo de teste	Tempo de treinamento	Acurácia (%)
SMO	0.21s	2433.62s	93.10
BoostIBK	38.99s	**110.41s**	95.75
BoostJ48	**0.14s**	699.89s	98.74
MLP	1.93s	7322.86s	99.24
LibSVM	36.70s	158.23s	**99.59**

5.3 Avaliação Visual dos Resultados de Classificação

Esta seção apresenta resultados visuais da detecção de defeitos no couro bovino através de imagens classificadas pelo módulo de classificação automática do DTCOURO. Este experimento utilizou os modelos de classificação (*datasets*) gerados pelos experimentos anteriores, porém os testes foram aplicados a dados desconhecidos, ou seja, dados que não foram utilizados na geração doconjunto de treinamento.

Este experimento teve como objetivo proporcionar uma análise visual sobre os resultados dos experimentos anteriores.

5.3.1 Módulo de Classificação Automática

O módulo de classificação automática está integrado ao projeto DTCOURO, sua finalidade é proporcionar uma avaliação visual dos resultados de classificação sobre uma imagem. O módulo é baseado em técnicas de aprendizagem de máquina e visão computacional e é integrado com ferramentas livres como Weka e ImageJ [4], que foram utilizadas nos experimentos anteriores para seleção de algoritmos de classificação e processamento de imagem na fase de extração de características.

A configuração dos parâmetros para um experimento de classificação visual com o módulo é dada pela seleção do conjunto de treinamento *(dataset)*, seleção dos extratores de características, seleção de cores para a legenda dos defeitos e dimensões das amostras e marcações que serão realizadas sobre uma imagem alvo.

Após a configuração do experimento, o processo automático acontece da seguinte forma; amostras são retiradas da matriz de *pixels* da imagem, de

[4]Processador de imagens em Java desenvolvido no United States Department of Health and Human Services

acordo com um determinado intervalo de linha e coluna. Os extratores selecionados extraem informações dessas amostras e são passadas para o classificador que retorna a classe (defeito) que a amostra discrimina, uma marcação de tamanho $m \times n$ é feita no centro da amostra com a cor (legenda) correspondente a classe encontrada. Essa varredura é feita na matriz da esquerda para a direita e de cima para baixo até a última amostra possível na imagem, o resultado é uma cópia da imagem original classificada visualmente através das marcações.

5.3.2 Configurações do experimento

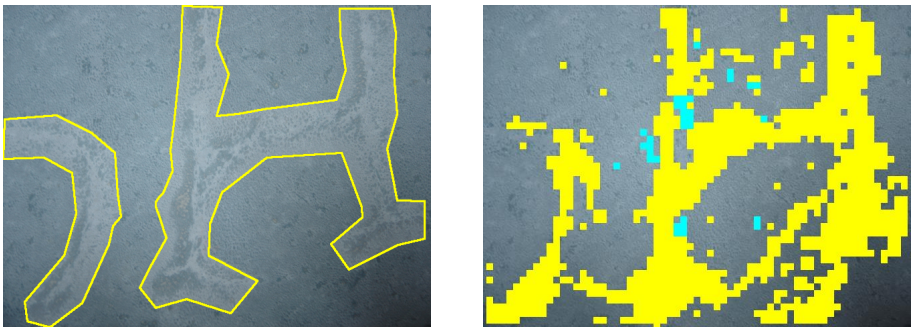
Duas imagens do couro cru e outras duas do couro *wetblue* foram classificadas utilizando o módulo de classificação. Foram utilizados, a implementação de Máquina de Vetores de Suporte SMO e os conjuntos de treinamento construídos nos experimentos apresentados anteriormente. Os *datasets* foram construídos com imagens do couro bovino em estágios couro cru e *wet blue*, a dimensão das amostras utilizadas na classificação é a mesma utilizada na criação dos conjuntos de treinamentos, 20×20 *pixels*. As amostras foram extraídas da imagem alvo utilizando um intervalo de linha por coluna de 10×10 *pixels* e as marcações também são de 10×10 *pixels*.

5.3.3 Resultados de Classificação

O módulo de classificação automática tem como resultado uma cópia da imagem original alvo com marcações nas regiões que contém um determinado defeito. As Figuras 5.4, 5.5, 5.6 e 5.7 mostram oito imagens, nos estágios cru e *wet blue*, marcadas manualmente e automaticamente. Para exemplificar, a Figura 5.4(a) mostra uma imagem no estágio *wet blue*, com uma marcação manual ao redor das regiões que apresentam o defeito " marca fogo ", a Figura 5.4(b) apresenta o resultado de classificação automática sobre a imagem da Figura 5.4(a). A marcação do defeito na imagem classificada automaticamente é representada pela cor amarela, isto significa que está região foi classificada como região contendo o defeito "marca fogo".

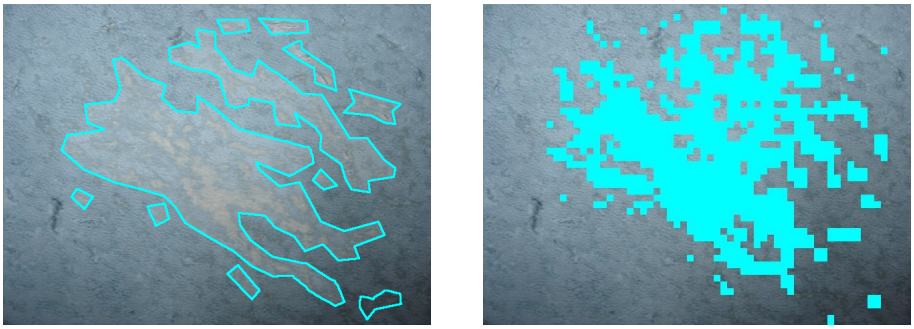
Como pode ser observado, as imagens marcadas automaticamente apresentam regiões que não deveriam ser marcadas como defeito quando comparadas as imagens marcadas manualmente. Já que os valores de classificação dos experimentos 5.1 e 5.2. foram altos, significa que a classificação neste experimento utilizando um novo conjunto de dados para testes apresenta "falsos positivos", ou seja, o classificador "acredita"que as regiões marcadas foram classificadas corretamente. Isto ocorre pois quantidade de amostras capturadas para a construção do conjunto de treinamento não foi suficiente

para discriminar completamente todos os defeitos. As imagens do couro, principalmente em estágio cru, são complexas, pois possuem muita variação de textura e cor, dificultando a discriminação entre as regiões interesse.



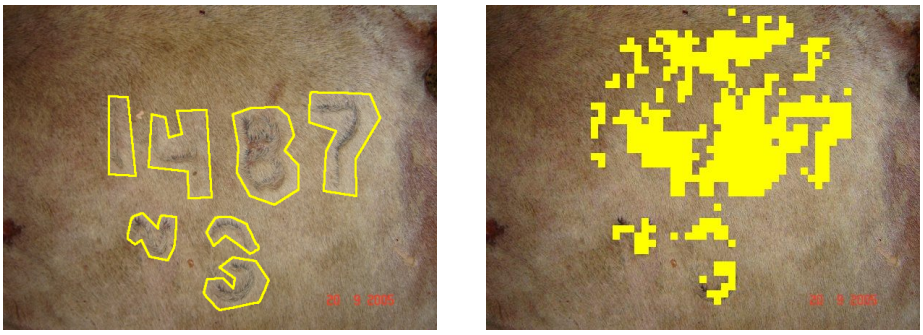
(a) Classificação manual. (b) Classificação automática.

Figura 5.4: Exemplo de classificação visual em uma imagem do couro bovino em estágio *wet blue* que apresenta regiões com o defeito "marca fogo".



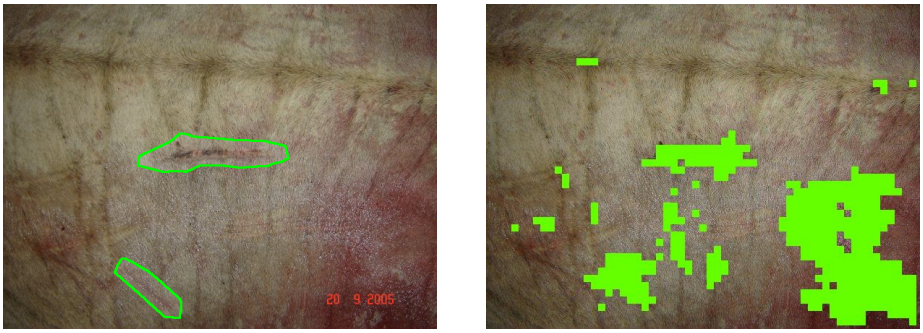
(a) Classificação manual. (b) Classificação automática.

Figura 5.5: Exemplo de classificação visual em uma imagem do couro bovino em estágio *wet blue* que apresenta regiões com o defeito "sarna".



(a) Classificação manual. (b) Classificação automática.

Figura 5.6: Exemplo de classificação visual em uma imagem do couro bovino em estágio cru que apresenta regiões com o defeito “marca fogo”.



(a) Classificação manual. (b) Classificação automática.

Figura 5.7: Exemplo de classificação visual em uma imagem do couro bovino em estágio cru que apresenta regiões com o defeito “risco”.

Capítulo 6

Conclusão e Trabalhos Futuros

A escolha de parâmetros para o treinamento de SVMs é de suma importância para obtenção de resultados satisfatórios na fase de testes, porém o processo de otimização dos parâmetros não é uma tarefa trivial e quando realizada manualmente, mesmo com conhecimento prévio do problema não garante resultados ótimos. Os experimentos realizados neste trabalho demonstram a aplicabilidade do algoritmo *Simulated Annealing* para o problema da otimização dos parâmetros C e γ da SVM.

Quando comparadas, as duas diferentes implementações de SVMs, o algoritmo que apresentou melhores resultados tanto na classificação quanto no tempo de treinamento foi o LibSVM. Note que a diferença na taxa de classificação entre o SMO e o LibSVM no primeiro experimento pode ser desprezada, uma vez que ambos os resultados são satisfatórios, porém quando aumentado o conjunto de imagens e atributos no segundo experimento os resultados obtidos pelo SMO não se mostraram tão satisfatórios, principalmente em relação aos tempos de otimização e treinamento. Acredita-se que a demora nas fases de treinamento e otimização do SMO deve-se ao modelo de implementação do algoritmo que não possui estratégias de *shirinking* nem *caching*.

Outra observação interessante é que as soluções do AdaBoost tendem a ser mais rápidas que SVMs na fase de teste e ainda apresentam pouca perda de acurácia em relação as mesmas no primeiro experimento. No entanto no segundo experimento com um conjunto maior de imagens e atributos, os resultados obtidos pelo AdaBoost (com IBK e J48) pioraram, tanto na taxa de classificação quanto no tempo de treinamento e teste. Estes desempenhos se devem ao fraco poder de generalização destes algoritmos, que geralmente são utilizados para problemas de baixa complexidade.

Nos testes realizados utilizando o módulo de marcação visual foi possível comprovar uma maior dificuldade na classificação de imagens no estágio couro

cru, quando comparado com a classificação de imagens *wet blue*. Testes empíricos sobre as imagens resultantes revelaram a presença de resultados de classificação considerados "falso positivos", isto ocorre devido a pouca representatividade do *dataset*. A complexidade do conjunto de características das imagens no estágio cru favoreceu a "confusão"do classificador em determinadas regiões das imagens do couro que apresentam poucas dissimilaridades entre as classes.

Para obter tempos de cálculos menores, um passo natural é a redução do número de características usando a seleção de atributos dos algoritmos de extração. Uma outra direção de pesquisa é a aplicação de métodos de pré-processamento de imagens afim de aumentar a representatividade dos defeitos. Dessa forma, serão realizados testes para verificação do ganho de informação de métodos de extração de atributos utilizando filtros de Gabor e segmentação de imagens relacionados a cada tipo de defeito.

Um passo importante para a construção de futuros conjuntos de treinamento é garantir que os dados que não estão segmentados como "defeitos"sejamautomaticamente incluídos como amostras dos conjunto de dados "sem defeito", para isso uma nova etapa de testes utilizando um conjunto maior de imagens em diferentes estágios do couro está prevista.

Referências Bibliográficas

[AMGA97] Branca A., Tafuri M., Attolico G., and Distante A. Automated system for detection and classification of leather defects. *NDT and E International*, 30:321–321(1), October 1997.

[AVRP06] W. P. Amorim, R. Viana, R. Rodrigues, and H Pistori. Desenvolvimento de um software de processamento e geracao de imagens para classificacao de couro bovino. *SIBGRAPI- Workshop of Undergraduate Works*, 2006.

[Bur98] Christopher J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

[Che96] D. Chetverikov. Structural filtering with texture feature based interaction maps: Fast algorithm and applications. In *In Proceedings of International Conf. on Pattern Recognition*, volume 2, pages 795–799, 1996.

[Che99] Dmitry Chetverikov. Texture analysis using feature-based pairwise interaction maps. *Pattern Recognition*, 32(3):487–502, 1999.

[CL01] Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[dC02] Achyles Barcelos da Costa. Estudo da competitividade de cadeias integradas no brasil: Impactos das zonas de livre comercio. Technical report, Instituto de Economia da Universidade Estadual de Campinas, Dezembro 2002.

[DG06] Jesse Davis and Mark Goadrich. The relationship between precision-recall and roc curves. In *ICML '06: Proceedings of the*

23rd international conference on Machine learning, pages 233–240, New York, NY, USA, 2006. ACM Press.

[FS99] Yoav Freund and Robert E. Schapire. A short introduction to boosting. *Journal of Japanese Society for Artificial Intelligence*, 14(5):771–780, 1999.

[GKA03] Lidiya Georgieva, Kaloyan Krastev, and Nikola Angelov. Identification of surface leather defects. In *CompSysTech '03: Proceedings of the 4th international conference conference on Computer systems and technologies*, pages 303–307, New York, NY, USA, 2003. ACM Press.

[Gom00] A. Gomes. Beneficiamento do couro - 60% dos defeitos no couro do boi ocorrem na fazenda. *Pecuária de Corte*, 11:38, Setembro 2000.

[Gom02] A. Gomes. Aspectos da cadeia produtiva do couro bovino no Brasil e em Mato Grosso do Sul. In *Palestras e proposicoes: Reunioes Tecnicas sobre Couros e Peles, 25 a 27 de setembro e 29 de outubro a 1º de novembro de 2001*, pages 61–72. Embrapa Gado de Corte, 2002.

[HBW99] Hsing-Wen Roger Hseu, Abhir Bhalerao, and R. G. Wilson. Image matching based on the co-occurrence matrix. Technical Report CS-RR-358, Coventry, UK, 1999.

[IL04] F. Imbault and K. Lebart. A stochastic optimization approach for parameter tuning of support vector machines. In *Proceedings of the Pattern Recognition, 17th International Conference on (ICPR'04)*, pages 597–600, Washington, DC, USA, 2004. IEEE Computer Society.

[JC04] R. Jobanputra and D.A. Clausi. Texture analysis using gaussian weighted grey level co-occurrence probabilities. In *Proceedings of the Canadian Conference on Computer and Robot Vision - CRV*, pages 51–57, 2004.

[Joa98] Thorsten Joachims. Text categorization with support vector machines: learning with many relevant features. In Claire Nédellec and Céline Rouveirol, editors, *Proceedings of ECML-98, 10th European Conference on Machine Learning*, number 1398, pages 137–142, Chemnitz, DE, 1998. Springer Verlag, Heidelberg, DE.

[KGA04] Kaloyan Krastev, Lidia Georgieva, and Nikola Angelov. Leather features selection for defects' recognition using fuzzy logic. In *CompSysTech '04: Proceedings of the 5th international conference on Computer systems and technologies*, pages 1–6, New York, NY, USA, 2004. ACM Press.

[KGV83] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science, Number 4598, 13 May 1983*, 220, 4598:671–680, 1983.

[KP02] A. Kumar and G. Pang. Defect detection in textured materials using gabor filters. *IEEE Transactions on Industry Applications*, 38(2), March 2002.

[MFF04] Holger Matthey, Jacinto F. Fabiosa, and Frank H. Fuller. Brazil: The future of modern agriculture. *MATRIC*, May 2004.

[MPV02] Moncef Gabbouj Mari Partio, Bogdan Cramariuc and Ari Visa. Rock texture retrieval using gray level co-occurrence matrix. In *In Proceedings of 5th Nordic Signal Processing Symposium*, pages 795–799, 2002.

[OFG97] E. Osuna, R. Freund, and F. Girosi. Training support vector machines:an application to face detection. *CVPR'97, Puerto Rico*, pages 130–136, 1997.

[Pla98] J. Platt. Sequential minimal optimization: A fast algorithm for training support vector machines, 1998.

[PPM+06] Hemerson Pistori, William A. Paraguassu, Priscila S. Martins, Mauro P. Conti, Mariana A. Pereira, and Manuel A. Jacinto. Defect detection in raw hide and wet blue leather. In *CompImage*, 2006.

[PV98] Massimiliano Pontil and Alessandro Verri. Support vector machines for 3d object recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20(6):637–646, 1998.

[Sob05] Joao Luis Sobral. Optimised filters for texture defect detection. In *Proc. of the IEEE International Conference on Image Processing*, volume 3, pages 565–573, September 2005.

[TK99] Sergios Theodoridis and Konstantinos Koutroumbas. *Pattern recognition*. San Diego : Academic press, 1999., 1999.

[Vap99] Vladimir N. Vapnik. An overview of statistical learning theory. *IEEE Transactions on Neural Networks*, 10(5):988–999, 1999.

[WF05] Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, San Francisco, CA, USA, second edition, 2005.

[WS03] H. Wang and D. Suter. Color image segmentation using global information and local homogeneity. *7th International Conference on Digital Image Computing: Techniques and Applications (DICTA'03)*, pages 89–98, 2003.

[YP01] C. Yeh and D. B. Perng. Establishing a demerit count reference standard for the classification and grading of leather hides. *International Journal of Advanced Manufacturing*, 18:731–738, 2001.