

**UNIVERSIDADE ESTADUAL PAULISTA “JÚLIO DE MESQUITA FILHO”  
DEPARTAMENTO DE COMPUTAÇÃO DE BAURU**

**INTERFACE PARA CLASSIFICAÇÃO DE DADOS POR  
MÁQUINA DE VETORES DE SUPORTE**

Luis Antonio de Souza Júnior

Orientador: Prof. Dr. João Paulo Papa

UNESP-Bauru  
2014

**UNIVERSIDADE ESTADUAL PAULISTA “JÚLIO DE MESQUITA FILHO”  
FACULDADE DE CIÊNCIAS  
DEPARTAMENTO DE COMPUTAÇÃO**

## **Interface para Classificação de Dados por Máquinas de Vetores de Suporte**

Luis Antonio de Souza Júnior

Trabalho de Conclusão de Curso apresentado ao Departamento de Computação, Faculdade de Ciências da UNESP de Bauru como exigência para obtenção do título de Bacharel em Ciência da Computação, área de concentração: Ciência de Computação.

Orientador: Prof. Dr. João Paulo Papa

UNESP-Bauru  
2014

---

## Resumo

Neste projeto o problema do Reconhecimento de Padrões é abordado com a técnica de Máquinas de Vetores de Suporte (*Support Vector Machines - SVM*), um método de classificação binário que fornece a melhor solução separando os dados de maneira ótima a partir da obtenção de um hiperplano e de uma extensão da dimensão do espaço de entrada do problema, sendo uma técnica de Aprendizado de Machina. O sistema objetiva classificar duas classes de *pixels* escolhidos pelo usuário na interface nas fases de seleção de interesse e seleção de fundo, gerando todos os dados a serem utilizados na biblioteca LibSVM, uma biblioteca que implementa a técnica de Máquinas de Vetores de Suporte, ilustrando o funcionamento da biblioteca de uma maneira casual. Os dados fornecidos pela interface são organizados de três maneiras: RGB (Sistema de Cores baseado na combinação linear de Vermelho, Verde e Azul), textura (calculado) e RGB + textura. Por fim, o projeto apresentou resultados satisfatórios, onde a classificação dos *pixels* da imagem foram mostrados como sendo de uma das duas classes, da classe da área de interesse ou da classe da área de fundo. Os resultados mais facilmente identificáveis de maneira visual pelos usuários são os que usam somente os dados de RGB, já que esta é a forma mais concreta de aquisição de dados.

**Palavras-chave:** Máquinas de Vetores de Suporte, Inteligência Artificial, Aprendizado de Máquina, Reconhecimento de Padrões.

## Abstract

In this project the Pattern Recognition Problem is approached with the Support Vector Machines (SVM) technique, a binary method of classification that provides the best solution separating the data in the better way with a hiperplan and an extension of the input space dimension, as a Machine Learning solution. The system aims to classify two classes of pixels chosen by the user in the interface in the interest selection phase and in the background selection phase, generating all the data to be used in the LibSVM library, a library that implements the SVM, illustrating the library operation in a casual way. The data provided by the interface is organized in three types, RGB (Red, Green and Blue color system), texture (calculated) or RGB + texture. At last the project showed successful results, where the classification of the image pixels was showed as been from one of the two classes, from the interest selection area or from the background selection area. The simplest user view of results classification is the RGB type of data arrange, because it's the most concrete way of data acquisition.

**Keywords:** Machine Learning, Support Vector Machines, Artificial Inteligence, Pattern Recognition.

# Sumário

<b>Lista de Figuras</b>	<b>vii</b>
<b>Lista de Tabelas</b>	<b>viii</b>
<b>1 Introdução</b>	<b>1</b>
1.1 Problema e Justificativa . . . . .	1
1.2 Objetivos . . . . .	2
1.3 Estrutura do Trabalho . . . . .	2
<b>2 Introdução ao Aprendizado de Máquina e ao Aprendizado Estatístico para escolha de um Classificador</b>	<b>3</b>
<b>3 Técnicas de Aprendizado de Máquina para Classificação de Padrões</b>	<b>7</b>
3.1 Redes Neurais Artificiais . . . . .	7
3.1.1 Mapas de Auto-Organização . . . . .	8
3.1.2 Redes Neurais Perceptron Multicamadas . . . . .	10
3.1.2.1 Perceptron . . . . .	10
3.1.2.2 Perceptron Multicamadas . . . . .	12
3.1.2.3 Treinamento por Retropropagação . . . . .	14
3.2 Máquinas de Vetores de Suporte . . . . .	16
3.2.1 Classificadores por Hiperplano . . . . .	16
3.2.2 Nuclearização e Hiperplanos de Margem Suave . . . . .	20
3.2.3 Considerações Adicionais . . . . .	23
3.3 $k$ -Vizinhos Mais Próximos . . . . .	23
3.4 Floresta de Caminhos Ótimos . . . . .	25
3.4.1 Classificador por OPF . . . . .	26
3.4.2 Definição Teórica . . . . .	26
3.4.3 Treinamento . . . . .	28
3.4.4 Classificação . . . . .	30
<b>4 Técnicas utilizadas para a Extração de Características</b>	<b>31</b>
4.1 Modelo de cor RGB . . . . .	31
4.2 Matrizes de Co-ocorrência e Textura . . . . .	32
4.3 Organização dos dados de saída da Interface para treinamento e classificação . . . . .	35

---

4.3.1	Arquivos para treinamento RGB . . . . .	35
4.3.2	Arquivos para teste RGB . . . . .	35
4.3.3	Arquivos para treinamento de textura . . . . .	36
4.3.4	Arquivos para teste de textura . . . . .	36
4.3.5	Arquivos para treinamento de RGB+textura . . . . .	37
4.3.6	Arquivos para teste de RGB+textura . . . . .	38
<b>5</b>	<b>A Biblioteca LIBSVM</b>	<b>39</b>
5.1	Descrição da LibSVM . . . . .	39
5.2	Funcionalidades utilizadas da LibSVM . . . . .	40
5.2.1	Ilustração dos comandos utilizados no Terminal . . . . .	41
<b>6</b>	<b>Projeto de Software</b>	<b>43</b>
6.1	Ferramentas de Desenvolvimento . . . . .	43
6.2	Análise de Requisitos do Sistema . . . . .	43
<b>7</b>	<b>Resultados Obtidos</b>	<b>45</b>
7.1	Testes . . . . .	45
<b>8</b>	<b>Conclusão</b>	<b>70</b>
	<b>Referências Bibliográficas</b>	<b>72</b>

# Lista de Figuras

1	Indução de classificadores em aprendizado supervisionado [A. C. Lorena, 2007]. . .	4
2	Conjunto de treinamento binário com três possíveis hipóteses [Schölkopf & Smola, 2002]. . . . .	5
3	Arquitetura bidimensional típica de uma rede de Kohonen adaptada de [Haykin, 1999].	11
4	Funções de Ativação Linear (a), Sigmoidal (b) e Identidade (c). . . . .	12
5	Modelo simplificado do Perceptron adaptado de [Haykin, 1999]. . . . .	13
6	Representação básica de uma rede <i>feedforward</i> (multicamadas) adaptada de [Haykin, 1999]. . . . .	14
7	Exemplo de hiperplano ótimo. . . . .	18
8	Mapeamento dos dados para um espaço de maior dimensionalidade. . . . .	21
9	Processo de classificação do <i>k</i> -NN Adaptado de [Martins <i>et al.</i> , 2005]. . . . .	25
10	Exemplo de Funcionamento do OPF . . . . .	27
11	Plano tridimensional de demonstraçom do sistema RGB [A. C. Lorena, 2007]. . . .	32
12	Ilustração do algoritmo de cálculo das matrizes de co-ocorrência. [Haralick, 1979]. .	33
13	Ilustração da utilização do comando <i>svm-train</i> . . . . .	41
14	Ilustração da utilização do comando <i>svm-predict</i> . . . . .	42
15	Imagem (1) utilizada para a seleções (interesse e fundo) e classificações do Teste 1. .	46
16	Classificação da Imagem (1) pelos três métodos de classificação. . . . .	47
17	Imagem (2) utilizada para a seleções (interesse e fundo) e classificações do Teste 2. .	50
18	Classificação da Imagem (2) pelos três métodos de classificação. . . . .	51
19	Imagem (3) utilizada para a seleções (interesse e fundo) e classificações do Teste 3. .	54
20	Classificação da Imagem (3) pelos três métodos de classificação. . . . .	55
21	Imagem (4) utilizada para a seleções (interesse e fundo) e classificações do Teste 4. .	58
22	Classificação da Imagem (4) pelos três métodos de classificação. . . . .	59
23	Imagem (5) utilizada para a seleções (interesse e fundo) e classificações do Teste 5. .	63
24	Classificação da Imagem (5) pelos três métodos de classificação. . . . .	64
25	Imagem (6) utilizada para a seleções (interesse e fundo) e classificações do Teste 6. .	66
26	Classificação da Imagem (6) pelos três métodos de classificação. . . . .	67

# Lista de Tabelas

1	Coordenadas e RGB da área de interesse do teste 1 . . . . .	47
2	Coordenadas e RGB da área de fundo do teste 1 . . . . .	48
3	Normalizações do teste 1 . . . . .	49
4	Coordenadas e RGB da área de interesse do teste 2 . . . . .	51
5	Coordenadas e RGB da área de fundo do teste 2 . . . . .	52
6	Normalizações do teste 2 . . . . .	53
7	Coordenadas e RGB da área de interesse do teste 3 . . . . .	55
8	Coordenadas e RGB da área de fundo do teste 3 . . . . .	56
9	Normalizações do teste 3 . . . . .	57
10	Coordenadas e RGB da área de interesse do teste 4 . . . . .	59
11	Coordenadas e RGB da área de fundo do teste 4 . . . . .	60
12	Normalizações do teste 4 . . . . .	62
13	Coordenadas e RGB da área de interesse do teste 5 . . . . .	64
14	Coordenadas e RGB da área de fundo do teste 5 . . . . .	65
15	Normalizações do teste 5 . . . . .	66
16	Coordenadas e RGB da área de interesse do teste 6 . . . . .	67
17	Coordenadas e RGB da área de fundo do teste 6 . . . . .	68
18	Normalizações do teste 6 . . . . .	69



# Capítulo 1

## Introdução

Neste capítulo está listada toda a informação indispensável para o entendimento e realização deste projeto, enquadrando ainda informações que garantem a pertinência do desenvolvimento deste, demonstrando sua relevância. Inicialmente, o problema abordado pelo projeto está descrito, juntamente com a justificativa de seu desenvolvimento. Em seguida, os objetivos estão descritos com o método de pesquisa aplicado para sua obtenção.

### 1.1 Problema e Justificativa

O problema central deste projeto é a utilização da técnica de reconhecimento de padrões Máquinas de Vetores de Suporte (*Support Vector Machines - SVM*) para classificar imagens de acordo com duas seleções realizadas por usuários. Desta maneira, é indispensável o desenvolvimento de uma interface amigável que garanta a possibilidade de usuários realizarem seleções em imagens, permitindo então a extração de dados (características) para a aplicação assim da técnica de inteligência artificial citada. Esta técnica já possui implementação por uma biblioteca chamada LibSVM [Chang & Lin, 2011], e é de alta valia para os projetos de pesquisa destinados a reconhecimento de padrões. Visto a alta aplicabilidade desta técnica na literatura e a escassez de recursos que apresentem interfaces amigáveis para a utilização da LibSVM, torna-se pertinente o desenvolvimento de uma interface que realize a tarefa de classificar imagens de acordo com características selecionadas por usuários (aplicação esta inovadora para a biblioteca). Além disso, é importante ressaltar que uma saída visual deve compor a interface, analisando os dados de saída da LibSVM e tratando-os para que possam ser exibidos de maneira visual para o usuário que esteja utilizando a interface.

## 1.2 Objetivos

Este projeto tem, portanto, como objetivo geral o desenvolvimento de uma interface amigável que utiliza da biblioteca LibSVM [Chang & Lin, 2011] para a classificação de imagens quanto a características de cor e textura, obtidas a partir de duas seleções realizadas pelo usuário. A partir disto, pode-se listar os seguintes objetivos específicos deste projeto:

1. Estudo literário sobre aprendizado de máquina com foco na compreensão de problemas que utilizam resoluções por Máquinas de Vetores de Suporte;
2. Desenvolvimento da interface que utiliza a biblioteca LibSVM, definindo as maneiras de extração das características desejadas e disposição destas características, para sua utilização na biblioteca;
3. Comunicação entre a interface implementada e a biblioteca LibSVM e;
4. Realização de testes para validação.

## 1.3 Estrutura do Trabalho

Este trabalho está dividido em 7 capítulos, onde após este primeiro de introdução, têm-se o segundo que é uma introdução a área de Aprendizado de Máquina, área de suma importância que comporta as técnicas de inteligência artificial responsáveis pela implementação computacional de aprendizado, que é a base deste projeto. O terceiro capítulo trata das técnicas específicas do aprendizado de máquina, como Redes Neurais Artificiais e a própria Máquinas de Vetores de Suporte. No quarto capítulo estão definidas as técnicas utilizadas para extração de características da interface. O quinto capítulo está incumbido de descrever as funcionalidades da biblioteca LibSVM, utilizada no sistema desenvolvido neste projeto. O projeto de software é listado no sexto capítulo e por fim, no sétimo e último capítulo os resultados obtidos pelo sistema são evidenciados. A conclusão do projeto e as referências bibliográficas vem a seguir.

## Capítulo 2

# Introdução ao Aprendizado de Máquina e ao Aprendizado Estatístico para escolha de um Classificador

As técnicas de Aprendizado de Máquina (AM) apresentam o princípio de inferência por indução, onde se obtém soluções genéricas a partir de um conjunto particular de exemplos, sendo ainda, um aprendizado dividido em: supervisionado e não-supervisionado.

No aprendizado supervisionado, tem-se a apresentação de exemplos que representam o conhecimento do ambiente na forma de entrada e saída desejada [Haykin, 1999]. O algoritmo AM extrai a representação de conhecimento, a partir destes exemplos, objetivando que a representação gerada seja capaz de produzir novas saídas corretamente para novos conjuntos de entradas não apresentadas anteriormente. Já no aprendizado não-supervisionado não existem exemplos rotulados. Desta maneira, o algoritmo AM aprende a representar as entradas nele inseridas segundo uma medida de qualidade. O aprendizado de máquina não-supervisionado é geralmente utilizado na busca de padrões ou tendências que auxiliem no entendimento dos dados [M. C. P. Souto, 2003].

Abordando o aprendizado supervisionado, tem-se o conjunto de exemplos rotulados da forma  $(\mathbf{x}_i, y_i)$ , em que  $\mathbf{x}_i$  representa um exemplo e  $y_i$  o seu rótulo (fenômeno) sobre o qual se deseja fazer previsões, um classificador (ou modelo, preditor ou ainda hipótese) deve ser produzido capaz de prever com precisão o rótulo de novos dados de entrada. Esta forma de indução a partir de uma amostra de dados de um classificador, que pode ser visto como uma função  $f$ , que recebe um dado  $\mathbf{x}$  e fornece uma predição  $y$ , é denominada treinamento.

Muitos conjuntos de dados apresentam imperfeições em seu conteúdo, como atributos ou rótulos incorretos. Desta maneira, as técnicas de AM devem ser capazes de lidar com dados imperfeitos, denominados ruídos. As técnicas de AM devem ter robusteza quanto a dados que apresentem ruídos,

não focando a obtenção dos classificadores nestes casos específicos. A obtenção dos classificadores também não deve ser muito influenciada por exemplos muito distintos dos demais presentes nos conjuntos de dados, os chamados *outliers*, dados que podem ser muito particulares, ruídos, ou ainda estarem fora do domínio do classificador.

A geração de um classificador a partir do aprendizado supervisionado é representada pela Figura 1. Tem-se nessa figura um conjunto  $n$  de dados, onde cada  $\mathbf{x}_i$  possui  $m$  atributos. Ou seja,  $\mathbf{x}_i = (x_{i1}, \dots, x_{im})$ . As variáveis  $y_i$  representam as classes. A partir dos exemplos e suas respectivas classes, o algoritmo AM extrai o classificador.

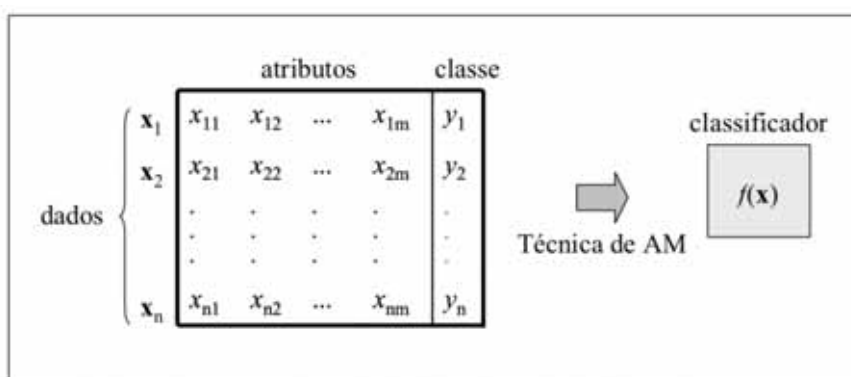


Figura 1: Indução de classificadores em aprendizado supervisionado [A. C. Lorena, 2007].

O problema de obtenção de um classificador por um algoritmo de AM a partir de uma amostra de dados pode ser considerado um processo de busca [Mitchell, 1997], onde procura-se entre todas as hipóteses geradas pelo algoritmo a com melhor capacidade de descrever o domínio em que ocorre o aprendizado. O classificador gerado deve ser o mais generalizado quanto possível, para que sua capacidade de predição de novos dados não seja comprometida. Classificadores podem sofrer de superajustamento (*overfitting*), quando a taxa de acerto quando este é confrontado com novos dados é baixa devido a uma especialização do classificador aos dados de entrada, ou de subajustamento (*underfitting*), quando os exemplos de treinamento disponíveis são pouco representativos, resultando na dificuldade de predição de novos dados de entrada. Estas duas ocorrências devem ser evitadas para garantir o bom desempenho de um classificador.

Para estimar a taxa de predições de um classificador como correta ou incorreta (taxa de acerto ou erro, respectivamente) obtidas por um classificador quando confrontado com novos dados, o conjunto de exemplos geralmente é dividido em dois subconjuntos disjuntos: de treinamento, utilizado no aprendizado do conceito, e de teste, utilizado para medir a efetividade do conceito aprendido na predição da classe de novos dados. Portanto, seja  $f$  um classificador pertencente ao conjunto de todos os classificadores possíveis a serem determinados pelo algoritmo de AM  $F$ , e  $T$  o conjunto de treinamento composto por  $n$  pares  $(\mathbf{x}_i, y_i)$ , utilizado para gerar um classificador particular  $\hat{f} \in F$ .

Considerando a Figura 2, tem-se por objetivo a obtenção de um classificador que separe os dados de duas classes, "círculo" e "triângulo". As funções consideradas são representadas na figura por meio de bordas entre as classes.

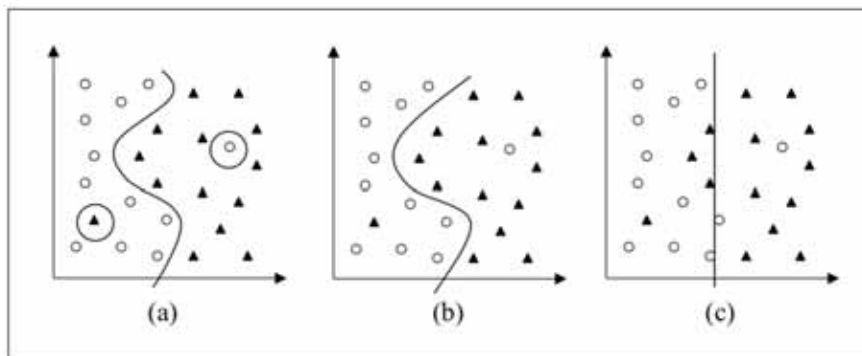


Figura 2: Conjunto de treinamento binário com três possíveis hipóteses [Schölkopf & Smola, 2002].

Na imagem da Figura 2a, tem-se um exemplo de super-ajustamento do modelo de dados de treinamento, onde todos os exemplos são classificados corretamente, incluindo ruídos. Neste caso, a função tem alta suscetibilidade a cometer erros quando confrontada com novos dados. Já na imagem da Figura 2c representa a alternativa de desconsiderar pontos pertencentes a classes opostas que estão muito próximos entre si. Por cometer muitos erros de treinamento, tem-se um exemplo de sub-ajustamento do classificador, que não consegue se ajustar nem mesmo aos exemplos de treinamento, portanto, não se ajustará a novos exemplos. Por fim, na imagem da Figura 2b tem-se o intermediário entre os dois classificadores anteriores, onde o preditor possui complexidade intermediária e classifica satisfatoriamente grande parte dos dados sem se fixar em pontos individuais.

A Teoria de Aprendizado Estatístico (TAE), portanto, estabelece condições matemáticas para que um classificador particular  $\hat{f}$  seja escolhido a partir de um conjunto de dados de treinamento, visando um classificador com bom desempenho no treinamento e na classificação de novos dados do mesmo domínio. Assumindo-se que os dados do domínio do aprendizado são inicialmente gerados de maneira independente e identicamente distribuídas, tem-se uma distribuição de probabilidade  $P(\mathbf{x}, y)$  que descreve a relação entre os dados e seus respectivos rótulos [Burges, 1998] [Chang & Lin, 2011]. O erro, ou risco, esperado por um classificador  $f$  pode ser quantificado pela Equação 1, e mede a capacidade de generalização do classificador [K. R. Müller & Schölkopf, 2001].

Na Equação 1,  $c(f(\mathbf{x}), y)$  é uma função de custo que relaciona a previsão  $f(\mathbf{x})$  com a saída  $y$  esperada. A função de custo 0/1 pode ser empregada ( $c(f(\mathbf{x}), y) = \frac{1}{2}|y - f(\mathbf{x})|$ ) [Schölkopf & Smola, 2002] retornando 0 se  $\mathbf{x}$  é classificado corretamente, e 1 caso contrário.

$$R(f) = \int c(f(\mathbf{x}, y)dP(\mathbf{x}, y) \quad (2.1)$$

---

O Risco Empírico também descreve algo importante na definição de um classificador particular  $\hat{f}$ . Este risco mede o desempenho do classificador nos dados de treinamento, por meio de classificações incorretas obtidas. O Risco Empírico é definido pela Equação 2:

$$R_{emp}(f) = \frac{1}{n} \sum_{i=1}^n c(f(\mathbf{x}_i, y_i)) \quad (2.2)$$

Com a Equação 2 é possível constituir o princípio de minimização do risco empírico [Schölkopf & Smola, 2002]. Assintoticamente, com  $n \rightarrow \infty$  é possível estabelecer condições para o algoritmo de aprendizado que garantam a obtenção de conjuntos de dados menores, mas essa garantia não pode ser assegurada para dados maiores.

## Capítulo 3

# Técnicas de Aprendizado de Máquina para Classificação de Padrões

Neste capítulo são exploradas as técnicas de Inteligência Artificial (IA), inseridas na área de Aprendizado de Máquina para classificação de padrões. Será feita uma revisão das principais técnicas e será detalhado o funcionamento de cada uma. Inicializará com Redes Neurais Artificiais, uma técnica que apresenta um modelo matemático baseado na estrutura neural de organismos inteligentes. Em seguida se discutirá as Máquinas de Vetores de Suporte, uma técnica que trata a separabilidade das classes com um modelo estatístico. Logo após, será apresentada a técnica denominada K-Vizinhos mais próximos. Finalmente, será destacada a técnica de Floresta de Caminhos Ótimos, a qual é baseada na teoria de grafos, e apresenta ótimos resultados quando comparada com outras técnicas.

### 3.1 Redes Neurais Artificiais

Redes Neurais Artificiais (*Artificial Neural Networks* - ANN) são técnicas computacionais que apresentam um modelo matemático inspirado na estrutura neural de organismos inteligentes e que adquirem conhecimento através da experiência [Haykin, 1994a]. Uma ANN é caracterizada pelo padrão de conexão (modelo conexionista) entre os neurônios (topologia), pelo método de determinação dos pesos e conexões (treinamento ou aprendizagem) e pela função de ativação responsável pela saída da rede. Esses processos comportam-se de maneira similar aos grupos de neurônios do cérebro humano que recebem e transmitem informações através dos dendritos e axônios, respectivamente [Kovacs, 1996]. Quando é apresentado à rede um conjunto de entradas e suas respectivas saídas, as quais estão representando o comportamento de um processo específico, a mesma, através do treinamento, é capaz de se auto-ajustar com o objetivo de mapear o relacionamento funcional entre as entradas e saídas. Por conseguinte, após a execução desse algoritmo de treinamento, a rede

deve ser capaz de generalizar o comportamento do processo quando outras entradas, diferentes daquelas utilizadas no treinamento, são apresentadas. Tal característica é particularmente útil quando o relacionamento entre as entradas e saídas do processo analisado é não-linear, ou então, quando o relacionamento não é claramente definido de forma que a sua modelagem, por técnicas de identificação convencionais, torna-se muito difícil.

As redes neurais representam o desenvolvimento de sistemas computacionais capazes de reconhecer e classificar padrões, resolver problemas complexos, realizar processos indutivos e dedutivos, entre outros.

Em geral, a utilização de redes neurais, comparando-se com outras técnicas, possui as seguintes vantagens [Raghu *et al.*, 1995]:

- os relacionamentos funcionais entre os padrões de entrada e saída podem ser capturados por uma rede neural, sendo que tais relacionamentos não precisam ser conhecidos ou descritos explicitamente;
- nenhuma hipótese precisa ser feita sobre as distribuições estatísticas dos padrões de entrada;
- as redes neurais são tolerantes à falhas no sentido que, mesmo quando alguns elementos ou conexões do processamento apresentam-se degradados, a performance da rede pode ser pouco afetada.

As arquiteturas de redes neurais artificiais utilizadas para modelar os sistemas neurais biológicos podem ser divididas em três categorias distintas. A primeira categoria de redes neurais, "*feedforward*" transforma conjuntos de sinais de entrada em conjuntos de sinais de saída, sendo que os parâmetros dessa transformação são ajustados de forma supervisionada, de acordo com os resultados desejados. Na segunda categoria de redes neurais, "*feedback*", as informações de entrada definem o estado inicial de atividade do sistema de realimentação. Depois de algumas transições de estado, atinge-se um estado que é considerado o resultado final da computação. Na terceira categoria, as células vizinhas da rede neural interagem mutuamente, a fim de transformarem-se adaptativamente em detectores especializados de diferentes padrões. Nessa última categoria, o aprendizado da rede neural é realizado de forma não supervisionada, através de mapas de auto-organização (*Self-Organizing Maps - SOM*) [Kohonen, 1990].

### 3.1.1 Mapas de Auto-Organização

No modelo de redes neurais baseado em mapas de auto-organização SOM proposto por Kohonen [Kohonen, 1989, Kohonen, 1990, Kohonen, 1993], a segregação espacial de diferentes respostas



e suas organizações em sub-conjuntos topologicamente relacionados resultam em um alto grau de eficiência nas operações típicas de uma rede neural.

O modelo SOM implementa uma projeção não-linear de um espaço multidimensional  $X$  em um espaço bidimensional  $M$ , denominado mapa de auto-organização. Essa projeção, de modo análogo à maioria das projeções encontradas no cérebro, implementam mapeamentos topológicos, ou seja, elementos vizinhos em  $X$  são geralmente mapeados em elementos vizinhos em  $M$ .

O mapeamento topológico constitui uma importante característica do modelo de redes neurais proposta por Kohonen, pois permite encontrar características e outras abstrações do espaço multidimensional através de identificação de agrupamentos ("clusters") no mapa bidimensional.

O mapeamento do espaço multidimensional no bidimensional é realizado por uma função  $f : X \rightarrow M$ , onde  $X \subset R^n$  e  $M \subset R^2$ , que associa cada elemento  $x \in X$  a um par  $(i, j) \in M$ . Os elementos  $m_{ij}$  do mapa  $M$ , assim como os dados de entrada, são vetores de tamanho  $n$ , que mantém os pesos das ligações sinápticas da rede neural.

Apresentado um estímulo específico  $x \in X$  à rede neural, o "nó vencedor" do mapa  $M$  para esse estímulo é determinado verificando-se qual dos elementos do mapa possui a menor distância ao estímulo, ou seja, o nó vencedor tem as coordenadas  $(i, j) \in M$ , tal que:

$$\|m_{ij} - x\| = \min\{\|m_{kl} - x\|, \forall i, j, k, l = 0 \dots D\} \quad (3.1)$$

onde  $D$  é o tamanho de  $M$ .

Portanto, um estímulo  $x \in X$  apresentado à camada de entrada da rede neural é mapeado no elemento  $(i, j)$  da camada de saída da rede neural, ou seja do mapa de auto-organização  $M$ , para o qual a distância entre esse estímulo e o vetor dos pesos das ligações sinápticas  $m_{ij}$  da rede neural seja mínima, comparando-se com os demais vetores de pesos das ligações sinápticas  $m_{kl}$ .

Nesse modelo de rede neural, os neurônios da camada de saída interagem lateralmente com seus vizinhos. Essas interações se dão na fase de aprendizado, quando os pesos da célula "vencedora" e dos seus vizinhos são ajustados, de tal modo que a célula vencedora tenha pesos idênticos aos valores do estímulo sendo apresentado à rede num determinado instante do treinamento, e as células vizinhas tenham pesos semelhantes aos da célula vencedora, sendo que o grau de semelhança é ponderado de acordo com a distância entre a célula vizinha e a célula vencedora.

A interação da célula vencedora  $(i, j)$  com uma célula vizinha  $(k, l)$ , num determinado instante  $t$  do treinamento, é definida por uma função  $h_{kl}(t)$  da distância entre elas. Normalmente, a função  $h_{kl}(t)$  é uma Gaussiana.

O aprendizado nesse modelo é dito não supervisionado porque o mapeamento do espaço multidimensional para o bidimensional é feito a despeito dos resultados esperados na saída do processamento, ao contrário dos outros modelos, onde os valores esperados e os valores obtidos pela rede,

para um dado estímulo de entrada, influenciam no reajuste dos pesos das ligações sinápticas.

No modelo de redes neurais baseadas em mapas de auto-organização, os pesos das ligações sinápticas são ajustadas da seguinte forma:

$$m_{kl}(t+1) = m_{kl}(t) + (x - m_{kl}(t))h_{kl}(t) \quad (3.2)$$

onde  $h_{kl}(t) = \alpha(t) \exp\left[-\frac{\|r_{kl} - r_{ij}\|^2}{\delta(t)^2}\right]$ ,  $(i, j)$  são as coordenadas do nó vencedor para o estímulo  $x$ ,  $\|r_{kl} - r_{ij}\|$  é a distância espacial entre o nó vencedor e o nó  $(k, l)$ , e  $\delta(t)$  é uma função do tempo de treinamento  $t$ , que determina a variância da Gaussiana  $h_{kl}$ .

O tamanho da vizinhança  $N_{ij}(t)$  decresce de acordo com o número de iterações  $T$  da fase de treinamento. No início do treinamento, isto é, quando  $t = 0$ ,  $N_{ij}(t)$  inclui toda a rede, e no final, quando  $t = T$ ,  $N_{ij}(t)$  contém somente o nó vencedor  $(i, j)$ . A seguinte função linear pode ser adotada para determinar os ajustes nas extensões da vizinhança:  $N_{ij}(t) = D \left[1 - \frac{t}{T}\right]$ , para  $t = 0, \dots, T$ .

De forma análoga, as funções  $\delta(t)$  e  $\alpha(t)$ , podem ser reajustadas em cada iteração através das funções lineares  $\delta(t) = 2 - \frac{t}{T}$  e  $\alpha(t) = 1 - \frac{t}{T}$ , para  $t = 1, \dots, T$ , respectivamente.

Após a fase de treinamento, os nós da rede são rotulados de acordo com a classe de estímulos para os quais eles apresentam respostas mais intensas. Se essa fase for supervisionada, então deve-se apresentar para a rede conjuntos de estímulos para os quais se conhece a classificação. Para cada classe de estímulos conhecidos verifica-se quais são os nós vencedores no mapa de auto-organização. Esses nós são rotulados com o rótulo associado àquela classe de estímulos. Se essa fase não for supervisionada, é necessário fazer uma análise de agrupamentos no mapa de auto-organização.

Um modelo da arquitetura bidimensional típica da rede de Kohonen é apresentado na Figura 26.

### 3.1.2 Redes Neurais Perceptron Multicamadas

#### 3.1.2.1 Perceptron

Durante as décadas de 50 e 60, uma grande revolução ocorreu no campo de pesquisa relacionado à teoria de reconhecimento de padrões com o aparecimento das chamadas máquinas que aprendem, os perceptrons [Haykin, 1994b].

Em sua metodologia mais básica, o perceptron aprende uma função de decisão linear que dicotomiza dois conjuntos de treinamento, os quais são linearmente separáveis, em um número finito de passos iterativos. A resposta desse dispositivo básico baseia-se na soma ponderada de sua entrada, ou seja,

$$d(x) = \phi(\chi); \quad (3.3)$$

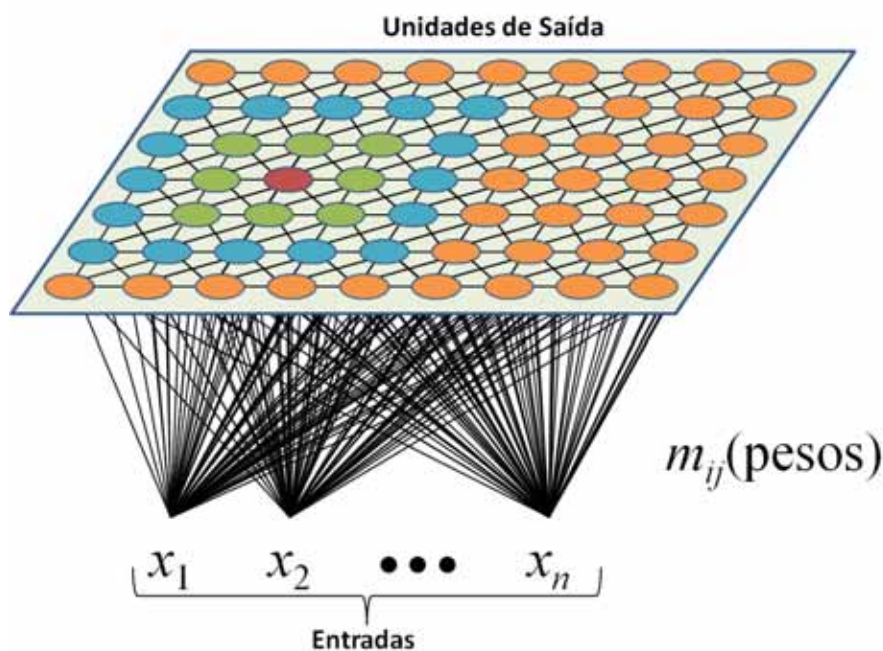


Figura 3: Arquitetura bidimensional típica de uma rede de Kohonen adaptada de [Haykin, 1999].

$$\chi = \sum_{i=1}^n w_i x_i + w_{n+1} \quad (3.4)$$

que é uma função de decisão linear em relação aos elementos do vetor de características. Os coeficientes  $w_j$ ,  $j = 1 \dots n + 1$ , são os chamados pesos das conexões, sendo análogos às sinapses no sistema neural humano. Tem-se ainda que  $\phi$  é a chamada função de ativação e  $\chi$  é o somatório da rede [Haykin, 1994a].

Dentre as várias escolhas para  $\phi$ , tem-se:

- Função limiar (threshold):

$$\phi(\chi) = \begin{cases} 1 & \text{se } \chi \geq 1, \\ 0 & \text{caso contrário} \end{cases} \quad (3.5)$$

- Função sigmoideal:

$$\phi(\chi) = \frac{1}{1 + \exp(-\chi)} \quad (3.6)$$

- Função identidade:

$$\phi(\chi) = \chi \quad (3.7)$$

A Figura 4 ilustra essas funções.

O algoritmo de treinamento do perceptron utiliza uma função de ativação por limiarização muito

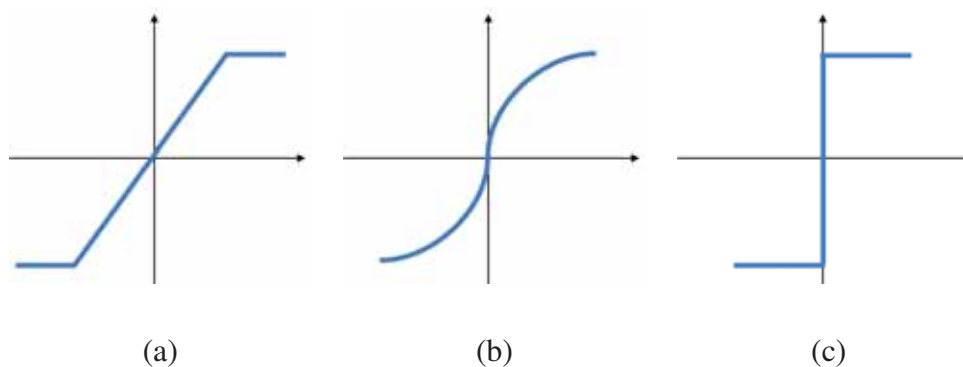


Figura 4: Funções de Ativação Linear (a), Sigmoidal (b) e Identidade (c).

semelhante à Equação (3.5), descrita por

$$\phi(\chi) = \begin{cases} 1 & \text{se } \chi \geq 1, \\ -1 & \text{caso contrário.} \end{cases} \quad (3.8)$$

Geometricamente, a equação

$$\chi = \sum_{i=1}^n w_i x_i + w_{n+1} = 0 \quad (3.9)$$

ou

$$\chi = w_1 x_1 + w_2 x_2 + \dots + w_n x_n + w_{n+1} = 0 \quad (3.10)$$

define um hiperplano no espaço  $n$ -dimensional de padrões. Os  $n$  primeiros coeficientes estabelecem a orientação do hiperplano, enquanto o último coeficiente,  $w_{n+1}$ , também chamado de *bias*, é proporcional à distância perpendicular à origem do hiperplano.

Caso as duas classes apresentadas ao algoritmo de perceptron sejam linearmente separáveis em  $\mathbb{R}^n$ , o mesmo converge em um número finito de passos. Entretanto, caso ambas classes não obedeçam a tal critério, o algoritmo executará indefinidamente, não convergindo para a solução ótima.

O modelo simplificado do perceptron é apresentado na Figura 5.

### 3.1.2.2 Perceptron Multicamadas

Para o tratamento de problemas que possuem várias classes, independentemente de as mesmas serem separáveis ou não, o algoritmo de Perceptron visto anteriormente é ineficiente, visto que o mesmo funciona adequadamente quando a tarefa de classificação consiste em duas classes linearmente separáveis. Conectando vários perceptrons, pode-se projetar uma estrutura chamada Perceptron Multicamadas, também conhecida como MLP [Haykin, 1994a], a qual consiste de várias cama-

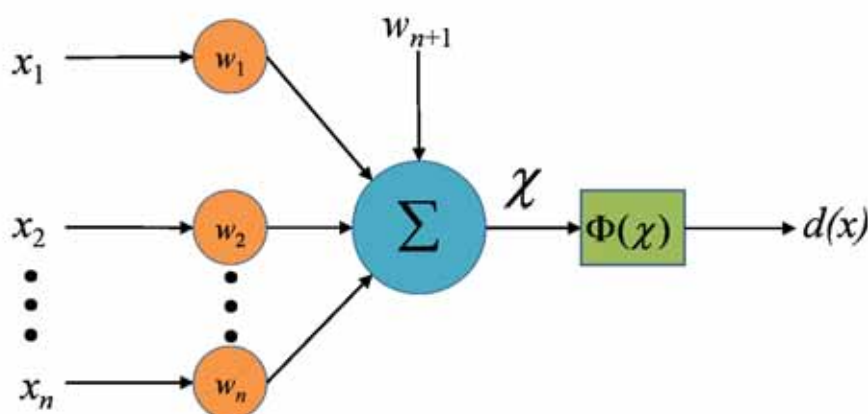


Figura 5: Modelo simplificado do Perceptron adaptado de [Haykin, 1999].

das de elementos computacionais idênticos (neurônios) dispostos de tal maneira que a saída de cada um deles alimente a entrada de cada neurônio da camada seguinte. O número de neurônios da primeira camada, denominada  $A$ , corresponde a  $N_A$ ; sendo, frequentemente denotado  $N_A = n$ , ou seja, o número de elementos da primeira camada é igual à dimensionalidade do vetor de características. O número de neurônios da camada de saída, chamada  $Q$ , corresponde a  $N_Q = M$ , sendo  $M$  igual ao número de classes do problema de classificação. A rede neural reconhece um vetor de padrões  $\mathbf{x}$  como pertencente à classe  $\omega_m$  caso a  $m$ -ésima saída da rede possuir o maior valor dentre as  $m - 1$  saídas restantes.

No caso da arquitetura citada anteriormente, a entrada de cada elemento em qualquer camada corresponde à soma ponderada das saídas da camada anterior. Denotando  $K$  a camada anterior a  $J$ , tem-se que a entrada de cada neurônio na camada  $J$ , denotada por  $I_j$  é dada por

$$I_j = \sum_{k=1}^{N_K} w_{jk} O_k \quad (3.11)$$

sendo que

$$O_k = \phi(I_k), \quad (3.12)$$

onde  $j = 1, 2, \dots, N_J$ , sendo  $N_J$  e  $N_K$  o número de elementos da camada  $J$  e  $K$ , respectivamente, e  $w_{jk}$  são os pesos que modificam as saídas  $O_k$  dos elementos da camada  $K$ .

Algumas convenções geralmente adotadas para tais estruturas de redes neurais são:

- a função de ativação utilizada na camada de entrada é a função identidade, dada pela Equação (3.7);
- camadas não adjacentes não são conectadas diretamente e, por conseguinte,

- todos os neurônios das camadas escondidas possuem a mesma função de ativação  $\phi$ .

Sendo assim, este modelo de rede neural é chamado de *feedforward*, devido ao fato de tanto a camada de entrada quanto as intermediárias, ou escondidas, serem submetidas somente à camada mais alta, ou seja, a camada  $K$  é subalterna à camada  $K - 1$ , assim por diante.

Uma representação de um modelo básico do modelo *feedforward* pode se visto na Figura 6.

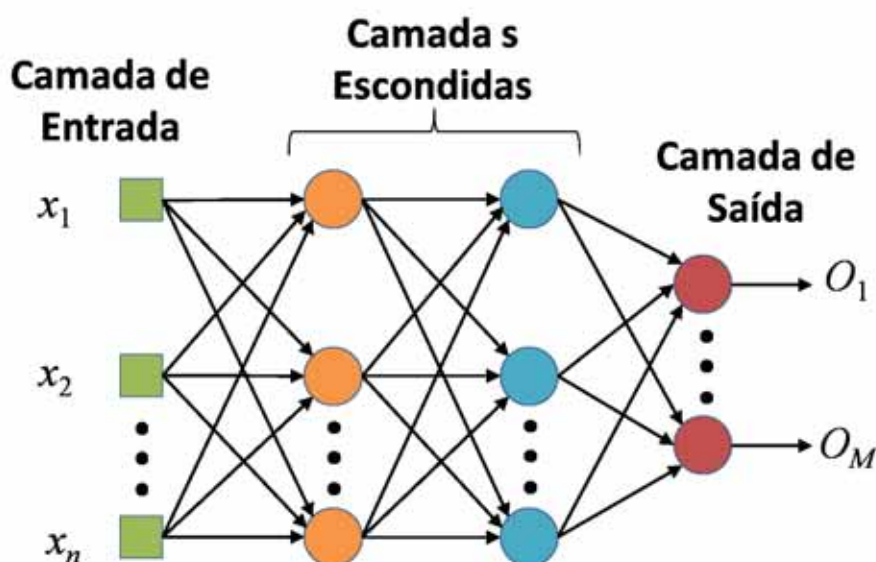


Figura 6: Representação básica de uma rede *feedforward* (multicamadas) adaptada de [Haykin, 1999].

### 3.1.2.3 Treinamento por Retropropagação

O principal objetivo deste algoritmo é desenvolver uma regra de treinamento com o intuito de minimizar o erro quadrático total entre as saídas desejadas  $r_q$  e as saídas reais  $\phi_q$  dos nós em uma camada de saída  $Q$ , ou seja, minimizar a equação abaixo:

$$E_Q = \frac{1}{2} \sum_{q=1}^{N_Q} (r_q - O_q)^2, \quad (3.13)$$

onde  $N_Q$  é o número de nós da camada de saída  $Q$ .

Durante o treinamento com o algoritmo de retropropagação, primeiramente, um padrão é apresentado à camada de entrada da rede. A saída obtida pela última camada é comparada com a saída desejada. Caso a mesma não estiver correta, o erro é calculado através da Equação (3.13) e propagado a partir da camada de saída até a camada de entrada, sendo que os pesos das conexões são modificados através da regra delta generalizada conforme o erro é retropropagado.

Desta forma, a regra delta generalizada  $\Delta w_{qp}$  permite o ajuste dos pesos em cada uma das camadas de maneira a tentar minimizar a função de erro mostrada acima, ou seja, calcular o gradiente negativo de  $E_Q$ , dado pela equação

$$\Delta w_{qp} = -\alpha \frac{\partial E_Q}{w_{qp}}, \quad (3.14)$$

em que  $P$  precede a camada  $Q$  e  $\alpha$  é um incremento positivo de correção.

Pode-se resumir e generalizar o procedimento de treinamento da seguinte maneira: para quaisquer duas camadas  $K$  e  $J$ , em que  $K$  precede imediatamente  $J$ , a Equação (3.14), através de algumas manipulações algébricas, pode ser re-escrita como

$$\Delta w_{jk} = \alpha \delta_j O_k. \quad (3.15)$$

Caso  $J$  seja uma camada de saída,  $\delta_j$  é dado por:

$$\delta_j = (r_j - O_j) O'_j, \quad (3.16)$$

onde  $O'_j$  de nota a derivada de  $O_j$ .

Se  $J$  for uma camada interna e  $P$  for a próxima camada, então  $\delta_j$  é dado por

$$\delta_j = O'_j \sum_{p=1}^{N_P} \delta_p w_{jp}, \quad (3.17)$$

onde  $j = 1, 2, \dots, N_J$ .

Desta forma, todo o processo de treinamento começa com um conjunto arbitrário de pesos da rede. Em seguida, a aplicação da regra delta generalizada em qualquer passo iterativo envolve duas etapas. Na primeira fase, um vetor de treinamento é apresentado à rede e propagado através das camadas da rede para o cálculo de  $O_j$  para cada nó. As saídas  $O_q$  dos nós da camada de saída são então comparadas com as respostas desejadas  $r_q$  para que os termos de erro  $\delta_q$  sejam gerados. A segunda fase envolve uma passagem para trás na rede, retropropagação, durante a qual o sinal de erro apropriado é passado por cada nó e as mudanças correspondentes nos pesos são realizadas. Uma vez que o sistema tenha sido treinado, o mesmo passa a classificar os padrões utilizando os parâmetros estabelecidos durante a fase de treinamento descrita acima.

## 3.2 Máquinas de Vetores de Suporte

As Máquinas de Vetores de Suporte, do inglês "Support Vector Machines" (SVM), são normalmente consideradas a primeira aplicação prática da teoria do aprendizado estatístico [Schölkopf *et al.*, 2000, Schölkopf & Smola, 2002]. Trata-se de uma área de pesquisa que oferece muitas opções para se trabalhar, sendo grande parte delas mais conceituais que meramente técnicas. Nos últimos anos seu escopo tem crescido significativamente tanto em termos de novos algoritmos quanto de um entendimento teórico mais aprofundado. Parte destes novos algoritmos se deve aos chamados métodos de nuclearização, uma proposta para solução de problemas de *aprendizado de máquina* cuja arquitetura tem se demonstrado capaz de lidar com questões relativas às bases desta teoria. Além disso, aplicações bem sucedidas de SVM demonstram que esta técnica não só possui uma fundamentação mais sólida do que as Redes Neurais Artificiais como também são capazes de substituí-las com desempenho melhor ou semelhante [Schölkopf & Smola, 2002].

As SVM propõem resolver o problema de classificação de padrões assumindo ser possível separar as classes em um espaço de mais alta dimensão. Suponha uma situação na qual os dados não são linearmente separáveis. Tais amostras podem ser separadas em grupos usando curvas ou círculos como superfícies de decisão, porém encontrar tais limiares é uma tarefa custosa. A principal ideia de uma SVM é pré-processar os dados de tal forma que o problema de encontrar uma função discriminante não-linear seja transformado em um problema de encontrar um hiperplano, ou seja, mapear os dados que estão em uma dimensão qualquer para outra maior, tornando os mesmos linearmente separáveis. Isto é feito definindo um mapeamento que transforma o vetor de entrada em outro (usualmente maior) vetor. Espera-se que, escolhendo um mapeamento adequado, o novo conjunto de treinamento seja linearmente separável.

### 3.2.1 Classificadores por Hiperplano

De acordo com as questões destacadas para o controle da efetividade dos algoritmos de aprendizado, se faz necessário que a capacidade da classe de funções possa ser calculada. Nos primórdios de seu estudo, Vapnik [Vapnik, 1999] considerou uma classe de hiperplanos em um espaço  $\mathcal{H}$  com produto interno,

$$\langle \mathbf{w}, \mathbf{x} \rangle - b = 0 \quad (3.18)$$

onde  $\mathbf{w} \in \mathcal{H}$ ,  $b \in \mathbb{R}$ , correspondendo com funções de decisão do tipo

$$f(x) = \text{sgn}(\langle \mathbf{w}, \mathbf{x} \rangle - b) \quad (3.19)$$



Baseado em dois argumentos, ele então propôs o algoritmo de aprendizado denominado Retrato Generalizado (do inglês "Generalized Portrait") para problemas separáveis por hiperplanos.

1. Dentre todos os hiperplanos que separam os dados, existe apenas um hiperplano ótimo distinguido pela margem de máxima separação entre qualquer ponto de treinamento e este hiperplano. Esta é a solução de

$$\underset{\mathbf{w} \in \mathcal{H}, b \in \mathbb{R}}{\text{maximize}} \quad \min\{\|\mathbf{x} - \mathbf{x}_i\| \mid \mathbf{x} \in \mathcal{H}, \langle \mathbf{w}, \mathbf{x} \rangle - b = 0\} \quad (3.20)$$

onde  $i = 1, \dots, m$ .

2. A capacidade da classe de hiperplanos de separação decresce com o crescimento da margem.

Para construir tal hiperplano ótimo, é necessário resolver

$$\underset{\mathbf{w} \in \mathcal{H}, b \in \mathbb{R}}{\text{minimize}} \quad \tau(\mathbf{w}) = \frac{1}{2} \|\mathbf{w}\|^2 \quad (3.21)$$

sujeito à

$$(\langle \mathbf{w}, \mathbf{x}_i \rangle - b) \geq 1 \quad \text{para todo } i = 1, \dots, m \quad (3.22)$$

ou

$$(\langle \mathbf{w}, \mathbf{x}_i \rangle - b) \leq -1 \quad \text{para todo } i = 1, \dots, m. \quad (3.23)$$

Temos que as Equações 3.22 e 3.23 pode ser escritas como:

$$y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle - b) \geq 1 \quad \text{para todo } i = 1, \dots, m, \quad (3.24)$$

onde  $y_i \in \{+1, -1\}$  denota o rótulo da amostra  $\mathbf{x}_i$ .

A razão por detrás da minimização de  $\mathbf{w}$  (Equação (3.21)) pode ser interpretada da seguinte maneira: se  $\|\mathbf{w}\|$  fosse 1, então o termo da esquerda na Equação (3.24) seria igual à distância de  $x_i$  ao hiperplano. Em geral, é preciso dividir  $y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle - b)$  por  $\|\mathbf{w}\|$  para transformá-lo nesta distância. Sendo assim, se a Equação (3.24) for satisfeita para todos  $i = 1, \dots, m$  com um  $\mathbf{w}$  de tamanho mínimo, a margem será maximizada como um todo. Um resumo destes argumentos é dado na Figura 7.

A função  $\tau$  na Equação (3.21) é chamada de *função objetivo*, enquanto a Equação (3.24) representa as *restrições de desigualdade*. Juntas elas dão origem ao que se conhece como *problema de otimização restrita*, que por sua vez é resolvido através da introdução do *Lagrangiano*

$$L(\mathbf{w}, b, \alpha) = \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{i=1}^m \alpha_i (y_i(\langle \mathbf{x}_i, \mathbf{w} \rangle + b) - 1) \quad (3.25)$$

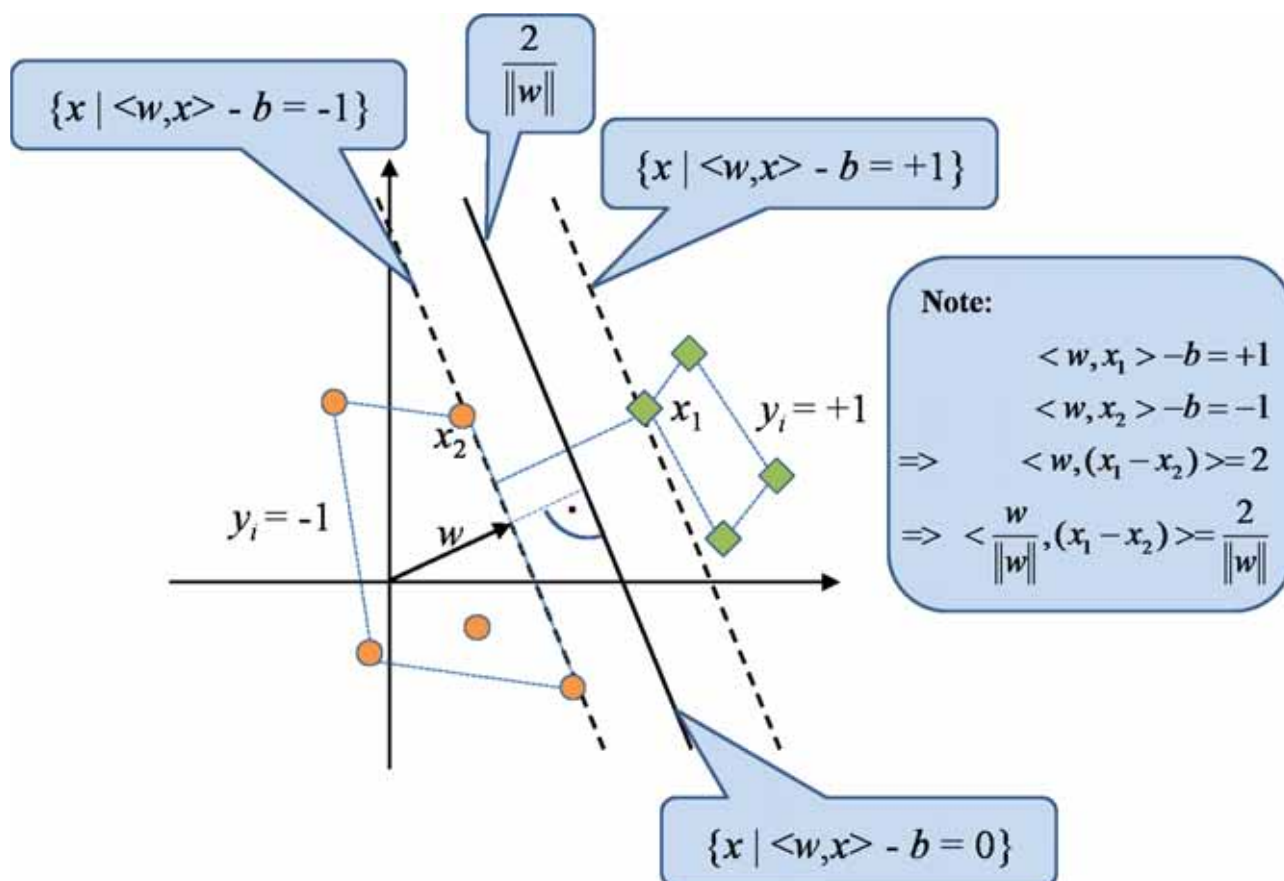


Figura 7: Um problema de classificação binária extraído de [Schölkopf & Smola, 2002] cujo propósito é separar as bolas dos diamantes. O hiperplano ótimo (Equação (3.21)) é mostrado com uma linha sólida. No problema sendo separado há apenas um vetor ponderado  $w$  e um limiar  $b$  tal que  $y_i(\langle w, x_i \rangle + b) > 0$  ( $i = 1, \dots, m$ ). Ao reescalar  $w$  e  $b$  com o objetivo de que o(s) ponto(s) mais próximos ao hiperplano satisfaçam  $|\langle w, x_i \rangle + b| = 1$  se obtém a forma *canônica*  $(w, b)$  do hiperplano.

onde  $\alpha_i \geq 0$  são os *multiplicadores de Lagrange* e  $L$  tem que ser minimizada com relação às *variáveis primais*  $w$  e  $b$  e maximizada com relação às *variáveis duais*  $\alpha_i$ . Em outras palavras, um ponto de sela deve ser encontrado. Como se pode notar, as restrições do problema original foram incorporadas no segundo termo no Lagrangiano.

Olhando melhor para o que o problema de otimização restrita nos mostra, é possível observar que se a restrição da Equação (3.25) no segundo termo do Lagrangiano for violada com  $\sum_{i=1}^m \alpha_i (y_i (\langle x_i, w \rangle + b) - 1) < 0$ ,  $L$  pode ser incrementado através do decréscimo do  $\alpha_i$  correspondente. Ao mesmo tempo,  $w$  e  $b$  terão que mudar (reduzindo a margem) de tal forma que  $L$  decresça. Dado que o problema é separável, num dado momento as restrições serão finalmente satisfeitas. De forma análoga, podemos entender que para todas as restrições não atendidas precisamente

como igualdades, seu  $\alpha_i$  correspondente deve ser 0: este é o valor de  $\alpha_i$  que maximiza  $L$ . Desta última observação extrai-se as chamadas condições de complementaridade de Karush-Kuhn-Tucker (KKT) presente na teoria da otimização.

Considerando que num ponto de sela as derivadas de  $L$  com relação às variáveis primais deve desaparecer,

$$\frac{\partial}{\partial b}L(\mathbf{w}, b, \alpha) = 0 \quad \text{e} \quad \frac{\partial}{\partial \mathbf{w}}L(\mathbf{w}, b, \alpha) = 0 \quad (3.26)$$

nos leva à

$$\sum_{i=1}^m \alpha_i y_i = 0 \quad (3.27)$$

e

$$\mathbf{w} = \sum_{i=1}^m \alpha_i y_i \mathbf{x}_i \quad (3.28)$$

Consequentemente, o vetor-solução nada mais é que uma expansão (Equação (3.28)) de um subconjunto dos padrões de treinamento, especificamente aqueles com  $\alpha_i$  diferente de zero, que são os chamados vetores de suporte (*Support Vectors* - SVs).

Ao substituir as Equações (3.27) e (3.28) no Lagrangiano (Equação (3.25)), elimina-se as variáveis primais  $\mathbf{w}$  e  $b$  resultando no chamado *problema de otimização dual*, que é o tipo de problema que as SVM normalmente resolvem na prática:

$$\underset{\alpha \in \mathcal{R}^m}{\text{maximize}} \quad W(\alpha) = \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i,j=1}^m \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle \quad (3.29)$$

sujeito à

$$\alpha_i \geq 0 \quad \text{para todo} \quad i = 1, \dots, m$$

e

$$\sum_{i=1}^m \alpha_i y_i = 0$$

Ainda baseado na Equação (3.28), a função-hiperplano de decisão (Equação (3.19)) pode ser escrita como

$$f(x) = \text{sgn} \left( \sum_{i=1}^m y_i \alpha_i \langle \mathbf{x}, \mathbf{x}_i \rangle - b \right) \quad (3.30)$$

Para fechar esta parte, nos falta estabelecer como  $b$  pode ser calculado. De acordo com as condições KKT, todos os pontos com

$$\alpha_i [y_i (\langle \mathbf{x}_i, \mathbf{w} \rangle - b) - 1] = 0 \quad \text{para todo } i = 1, \dots, m \quad (3.31)$$

são SVs situados na margem e, sendo assim,  $\alpha_i > 0$ . Em tais casos, é possível demonstrar que

$$\langle \mathbf{x}_i, \mathbf{w} \rangle - b = y_i \quad (3.32)$$

Dessa forma, o limiar pode ser obtido, por exemplo, por

$$b = y_i - \langle \mathbf{x}_i, \mathbf{w} \rangle \quad (3.33)$$

Alternativamente,  $b$  também pode ser calculado a partir dos valores das variáveis duais  $\alpha_i$  e  $\alpha_j$  correspondentes.

### 3.2.2 Nuclearização e Hiperplanos de Margem Suave

Esta seção está relacionada ao mapeamento das entradas para um espaço de maior dimensionalidade e ao acréscimo de artifícios de tolerância para que as SVM possam lidar com problemas não-separáveis.

Para expressar as fórmulas a partir dos padrões de entrada  $\mathcal{X}$ , emprega-se o produto interno dos vetores  $\mathbf{x}, \mathbf{x}'$  em termos do núcleo  $k$  estimado pelos elementos de entrada  $x, x'$

$$k(x, x') = \langle \mathbf{x}, \mathbf{x}' \rangle \quad (3.34)$$

Esta substituição, também conhecida como truque do núcleo (do inglês "kernel trick"), é usada para estender o conceito de classificação por hiperplanos para máquinas de vetores de suporte não-lineares. O truque do núcleo pode ser aplicado desde que todos os vetores de característica ocorram apenas em produto internos, o que pode ser observado nas Equações (3.29) e (3.30). O vetor ponderado (Equação (3.28)) torna-se então uma expansão no espaço de características e, sendo assim, não mais corresponderá ao seu respectivo vetor no espaço de entradas. A partir da Equação (3.30), a função de decisão se tornará

$$\begin{aligned} f(x) &= \operatorname{sgn} \left( \sum_{i=1}^m y_i \alpha_i \langle \Phi(x), \Phi(x_i) \rangle - b \right) \\ &= \operatorname{sgn} \left( \sum_{i=1}^m y_i \alpha_i k(x, x_i) - b \right) \end{aligned} \quad (3.35)$$

e o problema de otimização (Equação (3.29)) assumirá a forma

$$\underset{\alpha \in \mathcal{R}^m}{\text{maximize}} \quad W(\alpha) = \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i,j=1}^m \alpha_i \alpha_j y_i y_j k(x_i, x_j) \quad (3.36)$$

sujeito à  $\alpha_i \geq 0$  para todo  $i = 1, \dots, m$ , e  $\sum_{i=1}^m \alpha_i y_i = 0$ . A Figura 8 captura a ideia de mapear os dados do espaço de entradas para o espaço de características.

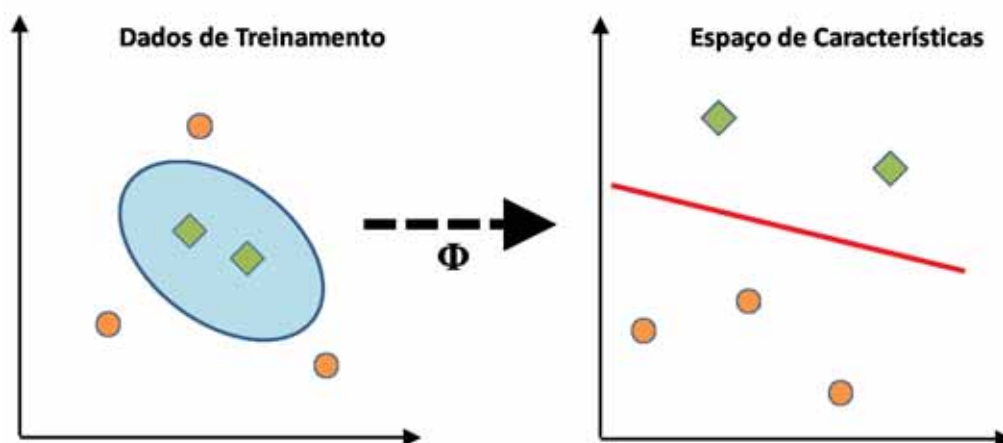


Figura 8: A ideia de se aplicar o truque do núcleo nas SVM: mapear os dados de treinamento (*input space*) em um espaço de características (*feature space*) de maior dimensão através de  $\Phi$  e construir lá um hiperplano de máxima margem de separação. Apesar da fronteira de decisão ser linear no espaço de características, no espaço de entradas original ela ganha formas não-lineares. Adaptado de [Haykin, 1999].

Mesmo com as vantagens da nuclearização do problema, na prática, o hiperplano de separação ainda assim pode não existir. Isto pode acontecer, por exemplo, em um conjunto de dados com muito ruído onde haja sobreposição das classes. Para permitir que os exemplos que violam a Equação (3.34) possam ser considerados, as variáveis de afrouxamento  $\xi_i$  são introduzidas [Schölkopf *et al.*, 2000, Schölkopf & Smola, 2002], onde

$$\xi_i \geq 0 \quad \text{para todo } i = 1, \dots, m \quad (3.37)$$

fazendo com que as restrições assumam a forma

$$y_i (\langle \mathbf{w}, \mathbf{x}_i \rangle - b) \geq 1 - \xi_i \quad \text{para todo } i = 1, \dots, m \quad (3.38)$$

Desta maneira, um classificador de boa generalização pode ser obtido pelo controle tanto da capacidade (através de  $\|\mathbf{w}\|$ ) quando da soma das variáveis de afrouxamento  $\sum_i \xi_i$ . É possí-

vel demonstrar que este somatório provê um limite superior no número de erros de treinamento [Schölkopf & Smola, 2002].

Neste contexto, uma possível elaboração de tal classificador de margem suave é obtida pela minimização da função-objetivo

$$\tau(\mathbf{w}, \xi) = \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^m \xi_i \quad (3.39)$$

sujeita às restrições da Equações (3.37) e (3.38), onde a constante  $C > 0$  determina o equilíbrio entre a maximização da margem e a minimização dos erros de treinamento. Quanto maior for  $C$ , menor será a margem, menor o número de erros de treinamento e menor também a capacidade de generalização da máquina de aprendizado.

Incorporando o núcleo e reescrevendo o problema em termos dos multiplicadores de Lagrange novamente nos leva a maximização da Equação (3.36), sujeita às restrições  $0 \leq \alpha_i \leq C$  para todo  $i = 1, \dots, m$  e  $\sum_{i=1}^m \alpha_i y_i = 0$ . A única diferença deste caso para o caso separável está no limite superior  $C$  dos multiplicadores de Lagrange  $\alpha_i$ . Sendo assim, a influência dos padrões individuais (que podem ser valores discrepantes) se torna limitada. A solução na Equação (3.35) não muda e o limiar  $b$  pode ser calculado pela exploração do fato de que para todos os SVs  $x_i$  com  $\alpha_i < C$ , a variável de afrouxamento  $\xi$  é zero e, assim sendo

$$\sum_{j=1}^m \alpha_j y_j k(x_i, x_j) - b = y_i \quad (3.40)$$

Analisando geometricamente, escolher  $b$  significa deslocar o hiperplano, ao passo que a Equação (3.40) determina ser necessário situá-lo de tal forma que todo SVs com variável de afrouxamento iguais à zero estejam nas linhas  $\pm 1$  (conforme Figura 7).

A função de núcleo apresentada como produto interno na Equação (3.34) é normalmente chamada de função de Núcleo Linear e, dadas algumas restrições [Schölkopf & Smola, 2002], pode ser substituída por outras funções. Outra função de mapeamento bastante conhecida é a gaussiana chamada Função de Base Radial (do inglês "*Radial Basis Function*" - RBF),

$$k(x, x') = \exp\left(-\frac{\|x - x'\|^2}{2\sigma^2}\right) \quad (3.41)$$

onde  $\sigma > 0$  é um parâmetro da função de núcleo. Em função da variável de ajuste  $C$ , este tipo de SVM é comumente chamado de C-SVC e representa a classificação por SVM em sua forma original [Cortes & Vapnik, 1995].

### 3.2.3 Considerações Adicionais

Como foi possível observar ao longo desta seção, as SVM são fundamentalmente máquinas de classificação binária. No entanto, a maioria dos problemas reais a serem otimizados requerem a classificação de mais de duas classes. Uma questão que emerge desta constatação é sobre como lidar com a propriedade multiclases inerente a muitos problemas em um classificador binário por definição. Três estratégias são comumente adotadas para emprego de SVM em problemas de  $k$  classes:

**Uma Contra o Resto:** Na abordagem uma-contra-o-resto,  $k$  SVM são treinadas, onde cada uma delas separa uma determinada classe de todas as demais.

**Classificação aos Pares:** Na abordagem de classificação aos pares,  $k(k - 1)/2$  máquinas são treinadas. Cada uma destas SVM separa um par de classes. Os classificadores são então arranjadas como nós de uma árvore, onde cada nó representa uma SVM. Tanto a busca de cima para baixo quanto de baixo para cima podem ser adotadas, sendo esta última análoga ao processo de eliminação de equipes em um campeonato esportivo.

Uma biblioteca de código aberto bastante conhecida que implementa SVM nas mais variadas formas é a chamada LibSVM [Chang & Lin, 2011]. Trata-se de um repositório de código aberto cujo objetivo é ajudar as pessoas a usarem facilmente SVM. Além de prover 4 diferentes tipos de função de núcleo, ela oferece SVM não só para classificação, mas também para "clusterização" e regressão. Através de uma interface por linhas de comando, é possível manipular os conjuntos de treinamento e teste, treinar SVM de diferentes maneiras, buscar por parâmetros ótimos das funções de núcleo e executar as classificações.

## 3.3 *k*-Vizinhos Mais Próximos

O classificador *k*-Vizinhos Mais Próximos (*k*-NN, do inglês *k-Nearest Neighbor*), que é um método bastante utilizado em aplicações que envolvem tarefas de classificação, pois é de fácil entendimento e implementação. O *k*-NN é um classificador onde o aprendizado é baseado na analogia [Mitchell, 1997].

A regra de classificação do *k*-NN [Pan *et al.*, 2004] é um método de classificação supervisionado e não-paramétrico, onde um padrão é dito pertencer a uma classe de acordo com a quantidade de vizinhos que pertençam a essa classe, conforme um critério de distância, normalmente usa-se a distância Euclidiana, porém existem outras distâncias como a Manhattan e a Minkowski.

Seja  $p = (p_1, p_2, \dots, p_n)$  e  $q = (q_1, q_2, \dots, q_n)$  dois pontos de  $\mathbb{R}^n$ :

- A distância Euclidiana entre  $p$  e  $q$  é dada por:

$$d(p, q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \dots + (p_n - q_n)^2}, \quad (3.42)$$

- A distância Manhattan entre  $p$  e  $q$  é dada por:

$$d(p, q) = |p_1 - q_1| + |p_2 - q_2| + \dots + |p_n - q_n|, \quad (3.43)$$

- A distância Minkowski entre  $p$  e  $q$  é dada por:

$$d(p, q) = (|p_1 - q_1|^j + |p_2 - q_2|^j + \dots + |p_n - q_n|^j)^{1/j}, \quad (3.44)$$

onde  $j \in \mathbb{N}$ .

A distância Minkowski é a generalização das duas distâncias anteriores. Quando  $j = 1$ , esta distância representa a distância de Manhattan e quando  $j = 2$ , a distância Euclidiana. Os classificadores representam diferentes paradigmas, sendo que o  $k$ -NN faz parte do chamado *lazy learning*, pois não há um treinamento explícito, considerando que cada padrão de treinamento é adicionado nos vetores de treinamento, diferente do SVM e da ANN-MLP com *backpropagation* que estão, respectivamente, no contexto de aprendizado estatístico e conexionista. Logo, uma de suas principais vantagens do  $k$ -NN é o pequeno tempo de treinamento, pois se resume apenas a adicionar padrões nos vetores. O conjunto de treinamento é formado por vetores  $n$ -dimensionais e cada elemento deste conjunto representa um ponto no espaço  $n$ -dimensional.

O método  $k$ -NN é um método de classificação que não possui processamento na fase de treinamento. Para cada padrão de teste é medida a distância entre ele e todos os padrões de treinamento. Verifica-se a quais classes pertencem os  $k$  padrões mais próximos e a classificação é feita, associando-se o padrão de teste à classe que for predominante [Mitchell, 1997]. O Algoritmo 1 implementa esta ideia.

#### **Algoritmo 1** – CLASSIFICADOR $k$ -NN

ESTRUTURAS AUXILIARES: Variáveis  $p$ , e  $k$ .

1. **Para** cada padrão de teste  $p$ , **faça**
2.     | Selecionar os  $k$  padrões de treinamento mais próximos de  $p$ .
3.     | Determinar a classe de  $p$  como sendo a classe de maior ocorrência entre  $k$  padrões.
4.     | Em case de empate, selecione uma classe aleatoriamente.



A classificação pode ser para o caso em que o parâmetro  $k$  é igual a 1 (NN), e se o  $k$  for maior que um, por exemplo,  $k = 3$ , na qual são considerados três vizinhos do novo padrão, ou seja, são analisadas as três menores distâncias do novo padrão para os padrões de treinamento. A classe que tiver o maior número de padrões entre essas distâncias é que irá determinar a classe do novo padrão.

A Figura 9 ilustra esse processo de classificação. Para classificar um ponto, primeiro tomam-se os  $k$ -vizinhos mais próximos dele e, dentro desse conjunto, encontra-se a classe mais representativa. Na Figura 9, o ponto Desconhecido 1 será classificado como classe B e o ponto Desconhecido 2 será classificado como classe A.

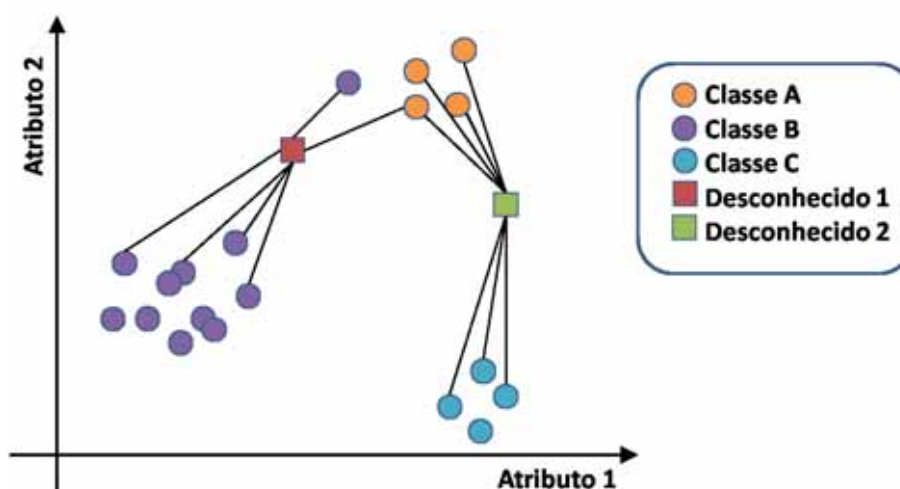


Figura 9: Processo de classificação do  $k$ -NN Adaptado de [Martins *et al.*, 2005].

Deve-se ter cuidado na escolha do parâmetro  $k$  para ser um número ímpar, pois um número par poderia causar conflito, quando um ponto tivesse o mesmo número de vizinhos de cada classe, por exemplo. A classificação  $k$ -NN é bem simples, porém tem a desvantagem de ter que armazenar todos os dados na memória, ou seja, gerar classificadores complexos, por causa do armazenamento, além da necessidade de se realizar muitos cálculos de distância.

O  $k$ -NN é um classificador que possui apenas um parâmetro livre (o número de  $k$ -vizinhos) que é controlado pelo usuário com o objetivo de obter uma melhor classificação. Este processo de classificação pode ser computacionalmente exaustivo se considerado um conjunto com muitos dados. Para determinadas aplicações, no entanto, o processo é bem aceitável.

### 3.4 Floresta de Caminhos Ótimos

É apresentada nesta seção uma técnica de classificação supervisionada de padrões baseadas em grafos, chamada Floresta de Caminhos Ótimos (*Optimum-Path Forest - OPF*) [Papa *et al.*, 2009].

Abordagens baseadas em OPF tratam as amostras como sendo os nós de um grafo, sendo os arcos definidos por uma relação de adjacência e ponderados por alguma métrica de distância aplicada em seus vetores de atributos, e diferem dos métodos tradicionais por não utilizar a ideia de geometria do espaço de atributos conseguindo, assim, melhores resultados em bases com *outliers* (amostras de uma determinada classe que estão presentes em uma região de outra classe no espaço de atributos) e sobreposição de classes. Duas abordagens supervisionadas já foram propostas atualmente, as quais diferem tanto na relação de adjacência e na função de valor de caminho utilizadas, quanto na maneira de encontrar os protótipos. A primeira delas utiliza como relação de adjacência o grafo completo e busca como protótipos as amostras que pertencem à intersecção entre as classes no conjunto de treinamento. A outra metodologia desenvolvida utiliza um grafo  $k$ -NN (*k-Nearest Neighbor*, ou seja,  $k$ -vizinhos mais próximos) e encontra os protótipos como sendo os máximos regionais ou amostras de cada classe na junção entre as densidades [Papa & Falcão, 2009]. Será abordada aqui a primeira técnica desenvolvida, ou seja, aquela que faz uso do grafo completo.

### 3.4.1 Classificador por OPF

A técnica de classificação supervisionada baseada em florestas de caminhos ótimos apresentada neste capítulo modela as amostras como sendo os nós de um grafo completo. Os elementos mais representativos de cada classe do conjunto de treinamento, isto é, os protótipos, são escolhidos como sendo os elementos pertencentes às regiões de fronteira entre as classes. Os protótipos participam de um processo de competição disputando as outras amostras oferecendo-lhes caminhos de menor custo e seus respectivos rótulos. Ao final deste processo, obtemos um conjunto de treinamento particionado em árvores de caminhos ótimos, sendo que a união das mesmas nos remete a uma floresta de caminhos ótimos. Esta abordagem apresenta vários benefícios com relação a outros métodos de classificação de padrões supervisionados: (i) é livre de parâmetros, (ii) possui tratamento nativo de problemas multiclases e (iii) não faz alusão sobre forma e/ou separabilidade das classes.

### 3.4.2 Definição Teórica

Seja  $Z$  uma base de dados  $\lambda$ -rotulada e  $Z_1$  e  $Z_2$  os conjuntos de treinamento e teste, respectivamente, com  $|Z_1|$  e  $|Z_2|$  amostras, tal que  $Z = Z_1 \cup Z_2$ . Seja  $\lambda(s)$  uma função que associa o rótulo correto  $i$ ,  $i = 1, 2, \dots, c$  da classe  $i$  a qualquer amostra  $s \in Z_1 \cup Z_2$ . Seja  $S \in Z_1$  um conjunto de protótipos de todas as classes (isto é, amostras que melhor representam as classes). Seja  $\vec{o}$  um algoritmo que extrai  $n$  atributos de qualquer amostra  $s \in Z_1 \cup Z_2$ , e retorna um vetor de atributos  $\vec{o}(s) \in \mathbb{R}^m$ . A distância  $d(s, t)$  entre duas amostras,  $s$  e  $t$ , é dada pela distância entre seus vetores de atributos  $\vec{o}(s)$  e  $\vec{o}(t)$ . Nosso problema consiste em usar  $S$ ,  $(\vec{o}, d)$  e  $Z_1$  para projetar um classificador

ótimo, o qual pode prever o rótulo correto  $\lambda(s)$  de qualquer amostra  $s \in Z_2$ . Assim sendo, propomos um classificador que cria uma partição discreta ótima, a qual é uma floresta de caminhos ótimos computada em  $\mathfrak{R}^m$  pelo algoritmo da transformada imagem floresta [Falcão *et al.*, 2004].

Seja  $(Z_1, A)$  um grafo completo cujos nós são as amostras em  $Z_1$ , onde qualquer par de amostras define um arco em  $A$  (isto é,  $A = Z_1 \times Z_1$ ) (Figura 10a). Note que os arcos não precisam ser armazenados e o grafo não precisa ser explicitamente representado.

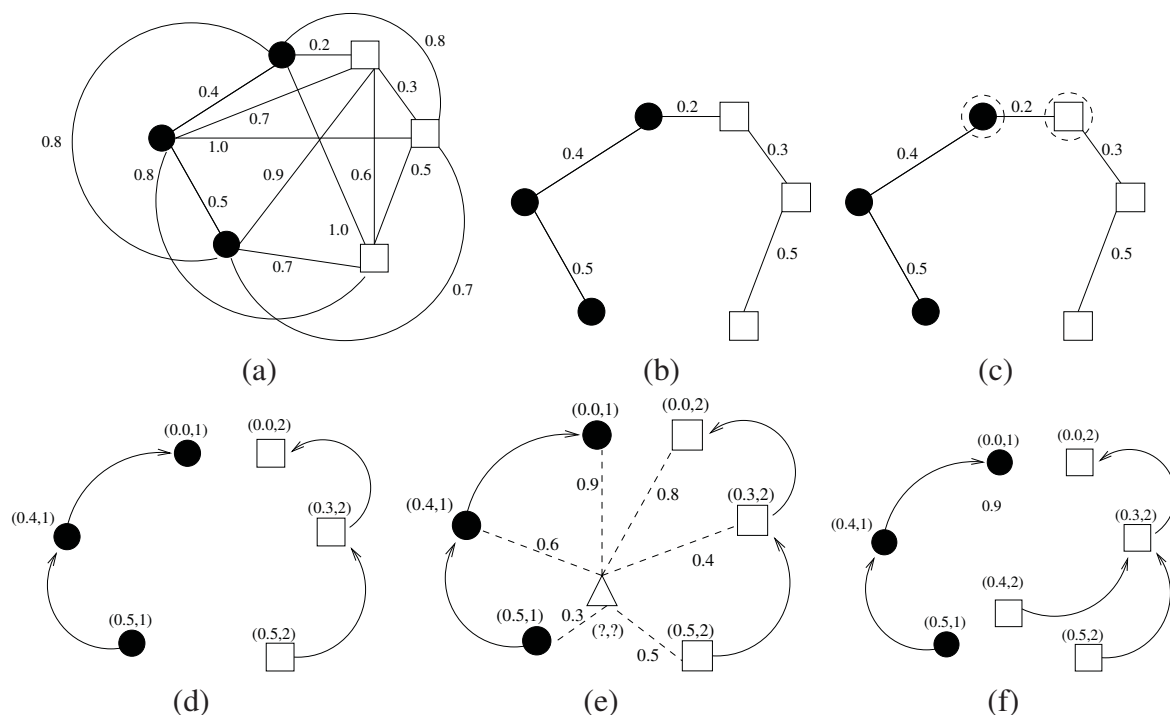


Figura 10: (a) Grafo completo ponderado nas arestas para um determinado conjunto de treinamento. (b) MST do grafo completo. (c) Protótipos escolhidos como sendo os elementos adjacentes de classes diferentes na MST (nós circulados). (d) Floresta de caminhos ótimos resultante para a função de valor de caminho  $f_{max}$  e dois protótipos. Os identificadores  $(x, y)$  acima dos nós são, respectivamente, o custo e o rótulo dos mesmos. A seta indica o nó predecessor no caminho ótimo. (e) Uma amostra de teste (triângulo) da classe 2 e suas conexões (linhas pontilhadas) com os nós do conjunto de treinamento. (f) O caminho ótimo do protótipo mais fortemente conexo, seu rótulo 2 e o custo de classificação 0.4 são associados a amostra de teste. Note que, mesmo a mostra de teste estando mais próxima de um nó da classe 1, ela foi classificada como sendo da classe 2. Extraído de [Papa *et al.*, 2009].

Tem-se, também, que um caminho em um grafo é uma seqüência de amostras  $\pi_{s_k} = \langle s_1, s_2, \dots, s_k \rangle$ , onde  $(s_i, s_{i+1}) \in A$  para  $1 \leq i \leq k - 1$ . Um caminho é dito ser trivial se  $\pi_s = \langle s \rangle$ . A cada caminho  $\pi_s$  é associado um valor dado por uma função de valor de caminho  $f$ , denotada  $f(\pi_s)$ . Diz-se que um caminho  $\pi_s$  é ótimo se  $f(\pi_s) \leq f(\tau_s)$  para qualquer caminho  $\tau_s$ , onde  $\pi_s$  e  $\tau_s$  terminam na mesma amostra  $s$ , independente de sua origem. Também denotamos  $\pi_s \cdot \langle s, t \rangle$  a conca-

tenação do caminho  $\pi_s$  com término em  $s$  e o arco  $(s, t)$ . O algoritmo da OPF pode ser utilizado com qualquer função de valor de caminho suave [Falcão *et al.*, 2004]. Uma função de valor de caminho  $f$  é suave quando, para qualquer amostra  $t$ , existe um caminho ótimo  $\pi_t$  o qual é trivial ou possui a forma  $\pi_s \cdot \langle s, t \rangle$ , onde

- $f(\pi_s) \leq f(\pi_t)$ ;
- $\pi_s$  é ótimo, e
- para qualquer caminho ótimo  $\tau_s$ ,  $f(\tau_s \cdot \langle s, t \rangle) = f(\pi_t)$ .

Na versão OPF com grafo completo a função de custo abordada foi a  $f_{max}$ , a qual é definida como:

$$\begin{aligned} f_{max}(\langle s \rangle) &= \begin{cases} 0 & \text{se } s \in S, \\ +\infty & \text{caso contrário} \end{cases} \\ f_{max}(\pi_s \cdot \langle s, t \rangle) &= \max\{f_{max}(\pi_s), d(s, t)\}, \end{aligned} \quad (3.45)$$

onde  $f_{max}(\pi_s)$  computa a distância máxima entre amostras adjacentes em  $\pi_s$ , quando  $\pi_s$  não é um caminho trivial.

O algoritmo baseado em OPF associa um caminho ótimo  $P^*(s)$  de  $S$  a toda amostra  $s \in Z_1$ , formando uma floresta de caminhos ótimos  $P$  (uma função sem ciclos, a qual associa a todo  $s \in Z_1$  seu predecessor  $P(s)$  em  $P^*(s)$ , ou uma marca *nil* quando  $s \in S$ , como mostrado na Figura 10d). Seja  $R(s) \in S$  a raiz de  $P^*(s)$  a qual pode ser alcançada por  $P(s)$ . O algoritmo computa, para cada  $s \in Z_1$ , o custo  $V(s)$  de  $P^*(s)$ , o rótulo  $L(s) = \lambda(R(s))$  e o seu predecessor  $P(s)$ , como segue.

As Linhas 1–4 inicializam os mapas e inserem protótipos em  $Q$ . O laço principal calcula um caminho ótimo de  $S$  para cada amostra  $s \in Z_1$  em uma ordem não decrescente de custos (Linhas 5–12). A cada iteração um caminho de custo ótimo  $V(s)$  é obtido em  $P$  (Linha 6). Empates são resolvidos em  $Q$  utilizando a política FIFO (first-in-first-out), ou seja, quando dois caminhos atingem uma determinada amostra  $s$  com o mesmo custo mínimo,  $s$  é associado ao primeiro caminho que o atingiu. O restante das linhas avalia se o caminho que atinge uma amostra adjacente  $t$  através de  $s$  é mais barato que o caminho que termina em  $t$ . Caso positivo, atualiza  $Q$ ,  $P(t)$ ,  $L(t)$  e  $V(t)$ . No final do algoritmo,  $V$  armazena o valor do custo do caminho ótimo de  $S$  a cada amostra  $s \in Z_1$  de acordo com  $f_{max}$ .

### 3.4.3 Treinamento

A fase de treinamento do classificador baseado em floresta de caminhos ótimos usando o grafo completo consiste, basicamente, em encontrar o conjunto  $S$  de protótipos, ou seja, os elementos

**Algoritmo 2** – CLASSIFICADOR SUPERVISIONADO BASEADO EM FLORESTA DE CAMINHOS ÓTIMOS USANDO GRAFO COMPLETO

ENTRADA: Um conjunto de treinamento  $Z_1$   $\lambda$ -rotulado, protótipos  $S \subset Z_1$  e o par  $(v, d)$  para vetor de atributos e cálculo das distâncias.

SAÍDA: Floresta de caminhos ótimos  $P$ , mapa de valores de custo de caminhos  $V$  e mapa de rótulos  $L$

ESTRUTURAS AUXILIARES: Fila de prioridades  $Q$ , e variável  $cst$ .

1. **Para todo**  $s \in Z_1$ , **Faça**  $P(s) \leftarrow nil$  e  $V(s) \leftarrow +\infty$ .
2. **Para todo**  $s \in S$ , **Faça**
3.      $V(s) \leftarrow 0$ ,  $P(s) \leftarrow nil$ ,  $L(s) = \lambda(s)$
4.      $Insira\ s\ em\ Q$ .
5. **Enquanto**  $Q$  *é não vazia*, **Faça**
6.      $Remova\ de\ Q\ uma\ amostra\ s\ tal\ que\ V(s)$  *é mínimo*.
7.     **Para cada**  $t \in Z_1$  *tal que*  $s \neq t$  e  $V(t) > V(s)$ , **Faça**
8.          $Calcule\ cst \leftarrow \max\{V(s), d(s, t)\}$ .
9.         **Se**  $cst < V(t)$ , **Então**
10.             **Se**  $V(t) \neq +\infty$ , **Então**  $remova\ t\ de\ Q$ .
11.              $P(t) \leftarrow s$ ,  $L(t) \leftarrow L(s)$  e  $V(t) \leftarrow cst$ .
12.              $Insira\ t\ em\ Q$ .

mais representativos de cada classe e iniciar o processo de competição entre eles no conjunto de treinamento. Várias heurísticas poderiam ser adotadas como, por exemplo, uma escolha aleatória de protótipos. Entretanto, tal escolha pode prejudicar o desempenho do classificador, tornando-o instável e com um alto grau de sensibilidade com relação aos protótipos escolhidos. Deseja-se, assim, estimar protótipos nas regiões de sobreposição de amostras e nas fronteiras entre as classes, visto que são regiões muito susceptíveis a erros de classificação.

Computando uma Árvore de Espalhamento Mínima (*Minimum Spanning Tree - MST*) no grafo completo  $(Z_1, A)$ , obtém-se um grafo conexo acíclico cujos nós são todas as amostras em  $Z_1$ , e os arcos são não direcionados e ponderados (Figura 10b). Seus pesos são dados pela distância  $d$  entre os vetores de atributos de amostras adjacentes. Esta árvore de espalhamento é ótima no sentido em que a soma dos pesos de seus arcos é mínima se comparada a outras árvores de espalhamento no grafo completo. Os protótipos a serem escolhidos são os elementos conectados na MST com diferentes rótulos em  $Z_1$ , isto é, elementos mais próximos de classes diferentes (Figura 10c). Removendo-se os arcos entre classes diferentes, tais amostras adjacentes tornam-se protótipos em  $S$  e o Algoritmo 2 pode computar uma floresta de caminhos ótimos em  $Z_1$  (Figura 10d). Note que uma dada classe pode ser representada por múltiplos protótipos (isto é, árvores de caminhos ótimos) e deve existir pelo menos um protótipo por classe.

### 3.4.4 Classificação

Para qualquer amostra  $t \in Z_2$ , consideramos todos os arcos conectando  $t$  com amostras  $s \in Z_1$ , tornando  $t$  como se fosse parte do grafo original (ver Figura 10e, onde a amostra  $t$  é representada pelo triângulo no grafo). Considerando todos os possíveis caminhos entre  $S$  e  $t$ , é desejado encontrar o caminho ótimo  $P^*(t)$  de  $S$  até  $t$  com a classe  $\lambda(R(t))$  de seu protótipo  $R(t) \in S$  mais fortemente conexo. Este caminho pode ser identificado incrementalmente, avaliando o valor do custo ótimo  $V(t)$  como segue:

$$V(t) = \min\{\max\{V(s), d(s, t)\}\}. \forall s \in Z_1. \quad (3.46)$$

Seja  $s^* \in Z_1$  o nó que satisfaz a equação acima (isto é, o predecessor  $P(t)$  no caminho ótimo  $P^*(t)$ ). Dado que  $L(s^*) = \lambda(R(t))$ , a classificação simplesmente associa  $L(s^*)$  como a classe de  $t$  (Figura 10f). Um erro ocorre quando  $L(s^*) \neq \lambda(t)$ .

## Capítulo 4

# Técnicas utilizadas para a Extração de Características

Neste capítulo são exploradas as técnicas de extração de características utilizadas na interface deste projeto. O processo de extração de características é resultado de duas seleções realizadas pelo usuário do sistema, onde com a primeira, a área de seleção será selecionada com a extração de suas características, e a com a segunda, a extração de características da área de fundo será realizada. Esta extração de características, portanto, é responsável pela aquisição de dados para a utilização da biblioteca que implementa as máquinas de vetores de suporte, a LibSVM.

### 4.1 Modelo de cor RGB

O modelo RGB possui as cores aditivas vermelho (R - RED), verde (G - GREEN) e azul (B - BLUE) representadas por um sistema cartesiano tridimensional R,G,B como primárias baseadas na sensibilidade dos olhos. O subspaço de interesse é o cubo unitário mostrado na figura a seguir:

A escala de cinza neste sistema é representada pela diagonal principal do cubo, onde o nível de cinza é representado por uma combinação linear das coordenadas de R,G e B inscritas nesta diagonal principal. As cores são, desta maneira, representadas por pontos que estejam dentro dos limites do cubo, variando de 0 a 1. Neste caso de uso, as variações foram de 0 a 255 (256 possibilidades para cada banda de cor). Este sistema de representação de cores apresenta certa deficiência de representação, visto que a resposta do olho aos estímulos espectrais não é linear. Por isto, algumas cores não podem ser representadas por este sistema, que é em suma, uma combinação linear de vermelho, verde e azul. Todavia, este sistema de representação de cores é mais simples de ser utilizado quando comparado com outros como o CMYK.

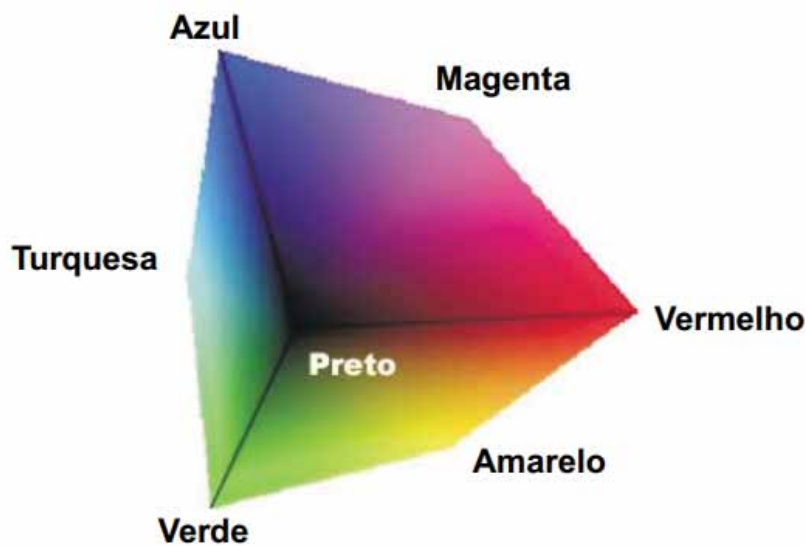


Figura 11: Plano tridimensional de demonstração do sistema RGB [A. C. Lorena, 2007].

## 4.2 Matrizes de Co-ocorrência e Textura

A extração de características para representação de texturas é realizada utilizando um método estatístico que se baseia na análise da imagem e extração de suas características por meio da relação entre os níveis de cinza da imagem. As matrizes de co-ocorrência, matrizes que possuem o nível de cinza (combinação linear de R,G,B) de um pixel, indicam a probabilidade de um par de níveis cinza  $[i,j]$  a uma determinada distância  $\delta$  em uma orientação  $\theta$ .

O cálculo das matrizes de co-ocorrência é realizado considerando a distância  $\delta$  e a orientação  $\theta$  dos pixels e de suas respectivas vizinhanças. As orientações aqui consideradas foram  $0^\circ$ ,  $45^\circ$ ,  $90^\circ$  e  $135^\circ$  e a distância foi 1. Segue o algoritmo de cálculo das matrizes de co-ocorrência:

1. Acesso a um pixel  $i$ ;
2. Análise do valor do pixel  $j$  de sua vizinhança considerando a orientação  $0^\circ$ ;
3. Incremento de 1 o valor correspondente a coordenada  $[i,j]$  da matriz de co-ocorrência de orientação  $0^\circ$ ;
4. Realização as etapas 2 e 3 até que todos os vizinhos de orientações  $45^\circ$ ,  $90^\circ$  e  $135^\circ$  do pixel  $i$  sejam analisados;
5. Após analisar todas as orientações, o processo retorna a etapa 1 considerando o próximo pixel da imagem para  $i$ .



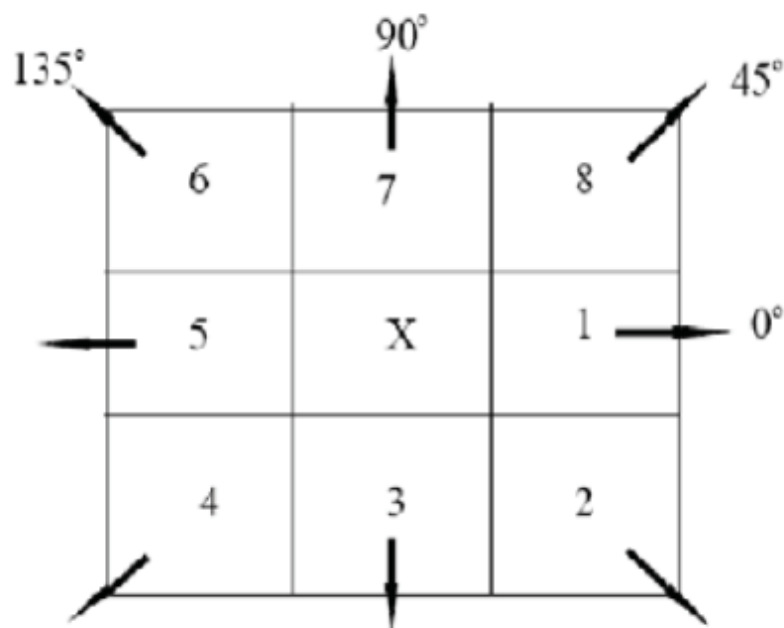


Figura 12: Ilustração do algoritmo de cálculo das matrizes de co-ocorrência. [Haralick, 1979].

Para cada orientação é calculada uma matriz, ou seja, quatro matrizes serão calculadas. Entretanto, para o cálculo de cada uma das matrizes, inicialmente, a imagem colorida deve ser convertida para imagem em tons de cinza já que as matrizes de co-ocorrência dependem dos níveis de cinza de uma imagem. Se imagens RGB fossem consideradas sem a conversão para tons de cinza, seriam calculadas 12 matrizes de co-ocorrência, 4 para cada banda, o que tornaria o processo mais trabalhoso e demorado. Portanto, a partir de imagens convertidas para tons de cinza, as matrizes são calculadas a partir do algoritmo apresentado, e a matriz de co-ocorrência final pode ser obtida a partir da normalização das matrizes de co-ocorrência orientadas:

$$N|\delta, 0 = 2 * N_c * (N_c - 1) \quad (4.1)$$

$$N|\delta, 45 = 2(N_c - 1)^2 \quad (4.2)$$

$$N|\delta, 90 = 2 * N_c * (N_c - 1) \quad (4.3)$$

$$N|\delta, 135 = 2(N_c - 1)^2 \quad (4.4)$$

onde  $N_c$  indica a quantidade de níveis de cinza da imagem.

Desta forma, a matriz de probabilidades final será dada por:

$$P(i, j, \delta, \theta) = M(i, j, \delta, \theta) / N | \delta, \theta \quad (4.5)$$

De acordo com [Haralick, 1979] é possível obter quatorze diferentes descritores de textura a partir das matrizes de co-ocorrência, entretanto, três descritores de textura foram utilizados neste trabalho, que foram: energia, entropia e contraste, descritos pelas fórmulas:

$$Energia = \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} [P(i, j, \delta, \theta)]^2 \quad (4.6)$$

$$Entropia = - \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} P(i, j, \delta, \theta) \log[P(i, j, \delta, \theta)] \quad (4.7)$$

$$Contraste = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} (i - j)^2 P(i, j, \delta, \theta) \quad (4.8)$$

onde  $P(i, j, \delta, \theta)$  significa a probabilidade de ocorrência do tom de *pixel*  $i$  e do tom de *pixel*  $j$  a uma distância  $\delta$  e orientação  $\theta$  e  $m$  por  $n$  pixels da imagem. Os valores de textura, assim como os valores de cor de cada imagem, são utilizados para a realização de treinamento e testes com a biblioteca LibSVM.

### 4.3 Organização dos dados de saída da Interface para treinamento e classificação

O intuito da interface é, desde sua concepção, a geração de dados para a classificação dos *pixels* de uma imagem por suas bandas de cores (RGB) e sua textura, descritos nos capítulos acima. Desta maneira, a partir de duas seleções, uma inicial, sendo a área de interesse do usuário na imagem e uma segunda, área de fundo definida pelo usuário na imagem, saídas são geradas para que o treinamento e classificação a partir da LibSVM sejam gerados. Para todos os casos, RGB, textura e um híbrido de RGB e textura, aqui designado por RGB+textura, um arquivos de dados básico, no formato .txt, é gerado no diretório do fonte do programa, contendo as coordenadas selecionadas para a área de interesse, os valores de normalizações ( $N | \delta, \theta$ ) para as matrizes de probabilidade finais ( $P(i, j, \delta, \theta)$ ) e as coordenadas selecionadas na área de fundo da imagem, juntamente com a geração dos arquivos de efetivo treinamento/classificação, no formato .txt.

### 4.3.1 Arquivos para treinamento RGB

Os arquivos para treinamento RGB seguem a ordem descrita acima, onde inicialmente o usuário seleciona (pinta) a área de interesse na imagem, e após, seleciona a área de fundo da imagem.

$\langle label \rangle \langle feature \rangle : \langle R \rangle \langle feature \rangle : \langle G \rangle \langle feature \rangle : \langle B \rangle$

onde:  $\langle label \rangle$  representa a classe do *pixel* selecionado, sendo -1 para área de interesse e 1 para área de fundo,  $\langle feature \rangle$  o índice do *pixel* na linha, variando de 0 a 2,  $R$  a banda de vermelho (RED) obtida na seleção do *pixel*,  $G$  a banda de verde (GREEN) obtida na seleção do *pixel*, e finalmente  $B$  a banda de azul (BLUE) obtida na seleção do *pixel*.

Exemplo:

-1 0:145 1:85 2:12

-1 0:146 1:81 2:22

1 0:0 1:0 2:0

1 0:0 1:0 2:0

### 4.3.2 Arquivos para teste RGB

Os arquivos para teste RGB seguem a mesma estrutura dos arquivos de treinamento, entretanto, comportam dados das bandas de todos os *pixels* da imagem, tendo todavia uma única label (1 ou -1).

$\langle label \rangle \langle feature \rangle : \langle R \rangle \langle feature \rangle : \langle G \rangle \langle feature \rangle : \langle B \rangle$

onde:  $\langle label \rangle$  representa a classe do *pixel* selecionado, sendo -1 para área de interesse e 1 para área de fundo,  $\langle feature \rangle$  o índice do *pixel* na linha, variando de 0 a 2,  $R$  a banda de vermelho (RED) obtida na seleção do *pixel*,  $G$  a banda de verde (GREEN) obtida na seleção do *pixel*, e finalmente  $B$  a banda de azul (BLUE) obtida na seleção do *pixel*.

### 4.3.3 Arquivos para treinamento de textura

Os arquivos para treinamento de textura seguem uma ordem similar a do treinamento RGB, onde inicialmente o usuário seleciona (pinta) a área de interesse na imagem, e após, seleciona a área de fundo da imagem. Os dados, entretanto, são alocados no arquivo texto de acordo com o cálculo da matriz de co-ocorrência para obtenção dos descritores de textura.

$\langle label \rangle \langle feature \rangle : \langle energia, 0 \rangle \langle feature \rangle : \langle energia, 45 \rangle$

$\langle feature \rangle : \langle energia, 90 \rangle \langle feature \rangle : \langle energia, 135 \rangle$

$\langle label \rangle \langle feature \rangle : \langle entropia, 0 \rangle \langle feature \rangle : \langle entropia, 45 \rangle$

$\langle feature \rangle : \langle entropia, 90 \rangle \langle feature \rangle : \langle entropia, 135 \rangle$

$\langle feature \rangle : \langle contraste, 0 \rangle \langle feature \rangle : \langle contraste, 45 \rangle$

$\langle feature \rangle : \langle contraste, 90 \rangle \langle feature \rangle : \langle contraste, 135 \rangle$

onde:  $\langle label \rangle$  representa a classe do *pixel* selecionado, sendo -1 para área de interesse e 1 para área de fundo,  $\langle feature \rangle$  o índice do *pixel* na linha, variando de 0 a 11, energia, $\theta$  a descrição de energia para cada orientação  $\theta$  (0°, 45°, 90°, 135°) obtidas na seleção do *pixel*, entropia, $\theta$  a descrição de entropia para cada orientação  $\theta$  (0°, 45°, 90°, 135°) obtidas na seleção do *pixel*, e finalmente contraste, $\theta$  a descrição de contraste para cada orientação  $\theta$  (0°, 45°, 90°, 135°) obtidas na seleção do *pixel*.

Exemplo:

```
-1 0:0 1:0 2:0 3:0 4:0 5:0 6:0 7:0
-1 0:95.3481 1:216.226 2:482.7 3:294.307 4:4.94056e+07 5:7.44001e+07 6:1.11163e+08
7:8.68002e+07
1 0:0 1:0 2:0 3:6.00627 4:0 5:0 6:0 7:2.2599e+09
1 0:0 1:0 2:0 3:0 4:0 5:0 6:0 7:0
```

#### 4.3.4 Arquivos para teste de textura

Os arquivos para teste de textura seguem a mesma estrutura dos arquivos de treinamento, entretanto, comportam dados de textura de todos os *pixels* da imagem, tendo todavia uma única label (1 ou -1).

```
 $\langle label \rangle \langle feature \rangle : \langle energia, 0 \rangle \langle feature \rangle : \langle energia, 45 \rangle$ 
 $\langle feature \rangle : \langle energia, 90 \rangle \langle feature \rangle : \langle energia, 135 \rangle$ 
 $\langle label \rangle \langle feature \rangle : \langle entropia, 0 \rangle \langle feature \rangle : \langle entropia, 45 \rangle$ 
 $\langle feature \rangle : \langle entropia, 90 \rangle \langle feature \rangle : \langle entropia, 135 \rangle$ 
 $\langle feature \rangle : \langle contraste, 0 \rangle \langle feature \rangle : \langle contraste, 45 \rangle$ 
 $\langle feature \rangle : \langle contraste, 90 \rangle \langle feature \rangle : \langle contraste, 135 \rangle$ 
```

onde:  $\langle label \rangle$  representa a classe do *pixel* selecionado, sendo -1 para área de interesse e 1 para área de fundo,  $\langle feature \rangle$  o índice do *pixel* na linha, variando de 0 a 11, energia, $\theta$  a descrição de energia para cada orientação  $\theta$  (0°, 45°, 90°, 135°) obtidas na seleção do *pixel*, entropia, $\theta$  a descrição de entropia para cada orientação  $\theta$  (0°, 45°, 90°, 135°) obtidas na seleção do *pixel*, e finalmente contraste, $\theta$  a descrição de contraste para cada orientação  $\theta$  (0°, 45°, 90°, 135°) obtidas na seleção do *pixel*.

#### 4.3.5 Arquivos para treinamento de RGB+textura

Os arquivos para treinamento de RGB+textura seguem uma ordem similar a dos demais arquivos de treinamento, porém comportam dados de RGB e textura juntos dos *pixels* selecionados tanto na etapa de área de seleção quanto fundo da imagem.

$\langle label \rangle \langle feature \rangle: R \langle feature \rangle: \langle G \rangle \langle feature \rangle: \langle B \rangle$   
 $\langle feature \rangle: \langle energia, \theta \rangle$   
 $\langle feature \rangle: \langle entropia, \theta \rangle$   
 $\langle feature \rangle: \langle contraste, \theta \rangle$

onde:  $\langle label \rangle$  representa a classe do *pixel* selecionado, sendo -1 para área de interesse e 1 para área de fundo,  $\langle feature \rangle$  o índice do *pixel* na linha, variando de 0 a 11,  $R$  a banda de vermelho (RED) obtida na seleção do *pixel*,  $G$  a banda de verde (GREEN) obtida na seleção do *pixel*,  $B$  a banda de azul (BLUE) obtida na seleção do *pixel*,  $energia, \theta$  a descrição de energia para cada orientação  $\theta$  (0°, 45°, 90°, 135°) obtidas na seleção do *pixel*,  $entropia, \theta$  a descrição de entropia para cada orientação  $\theta$  (0°, 45°, 90°, 135°) obtidas na seleção do *pixel*, e finalmente  $contraste, \theta$  a descrição de contraste para cada orientação  $\theta$  (0°, 45°, 90°, 135°) obtidas na seleção do *pixel*.

Exemplo:

```

-1 0:145 1:85 2:12 3:0 4:0 5:0 6:0 7:0 8:0 9:0 10:0
-1 0:146 1:81 2:22 3:95.3481 4:216.226 5:482.7 6:294.307 7:4.94056e+07 8:7.44001e+07
9:1.11163e+08 10:8.68002e+07
1 0:0 1:0 2:0 3:0 4:0 5:0 6:6.00627 7:0 8:0 9:0 10:2.2599e+09
1 0:0 1:0 2:0 3:0 4:0 5:0 6:0 7:0 8:0 9:0 10:0

```

### 4.3.6 Arquivos para teste de RGB+textura

Os arquivos para teste de RGB+textura seguem a mesma estrutura dos arquivos de treinamento, entretanto, comportam dados de RGB e textura de todos os *pixels* da imagem, tendo todavia uma única label (1 ou -1).

$\langle label \rangle \langle feature \rangle: R \langle feature \rangle: \langle G \rangle \langle feature \rangle: \langle B \rangle$   
 $\langle feature \rangle: \langle energia, \theta \rangle$   
 $\langle feature \rangle: \langle entropia, \theta \rangle$   
 $\langle feature \rangle: \langle contraste, \theta \rangle$

onde:  $\langle label \rangle$  representa a classe do *pixel* selecionado, sendo -1 para área de interesse e 1 para área de fundo,  $\langle feature \rangle$  o índice do *pixel* na linha, variando de 0 a 11,  $R$  a banda de vermelho (RED) obtida na seleção do *pixel*,  $G$  a banda de verde (GREEN) obtida na seleção do *pixel*,  $B$  a banda de azul (BLUE) obtida na seleção do *pixel*,  $energia, \theta$  a descrição de energia para cada orientação  $\theta$  (0°, 45°, 90°, 135°) obtidas na seleção do *pixel*,  $entropia, \theta$  a descrição de entropia para cada orientação  $\theta$  (0°, 45°, 90°, 135°) obtidas na seleção do *pixel*, e finalmente  $contraste, \theta$  a descrição de contraste para cada orientação  $\theta$  (0°, 45°, 90°, 135°) obtidas na seleção do *pixel*.

# Capítulo 5

## A Biblioteca LIBSVM

A biblioteca LibSVM, desenvolvida na linguagem C na plataforma Linux, apresenta a ferramenta mais importante para a realização deste projeto, a implementação da técnica de reconhecimento de padrões por Máquinas de Vetores de Suporte (Support Vector Machines - SVM). Desta maneira, neste capítulo será descrita a maneira de utilizá-la, dando destaque para a utilização juntamente com a interface de extração de características desenvolvida neste projeto.

### 5.1 Descrição da LibSVM

A LibSVM, como citado, é uma biblioteca desenvolvida para a aplicação das máquinas de vetores de suporte, mas possui ainda outras funcionalidades implementadas, como regressão e suporte para classificação multi-classes (ainda tratando de problemas de classificação de padrões SVM com mais de duas classes). Sendo assim, a LibSVM, que é atualizada constantemente, apresenta outras utilidades além do treinamento e classificação de problemas binários. O intuito da biblioteca é facilitar a utilização das SVMs, que apresentam grau matemático complexo de entendimento e implementação. Com isso, as principais funções oferecidas pela biblioteca podem ser resumidas em:

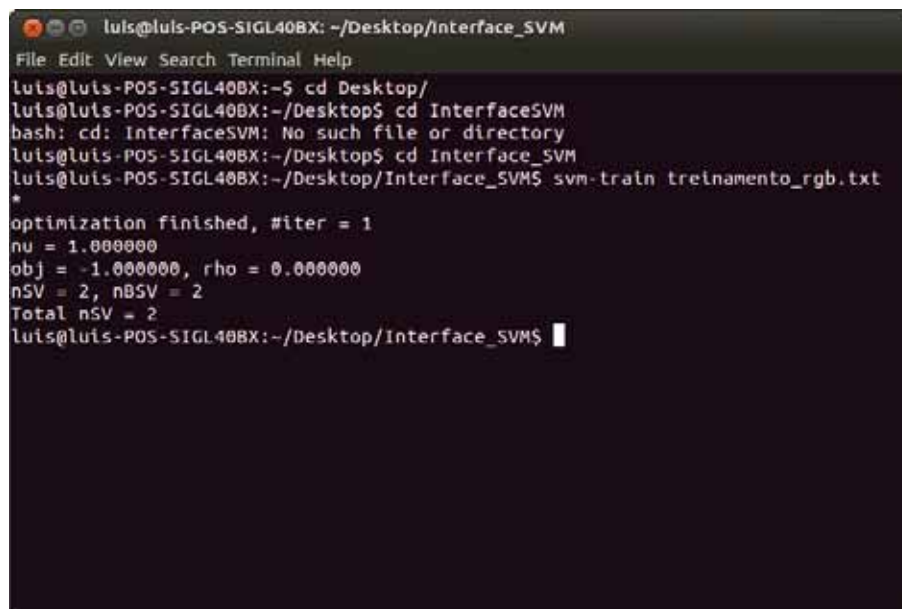
- diferentes formulações para problemas de classificação de padrões com a LibSVM;
- classificação de problemas multi-classe de maneira eficiente;
- classificação por probabilidades estatísticas;
- implementação de vários kernels e;
- implementação da biblioteca em C++ e Java.

## 5.2 Funcionalidades utilizadas da LibSVM

A LibSVM proporciona diversas funcionalidades que podem ser utilizadas para diversos tipos de conjuntos de dados, desde que estes estejam devidamente organizados e padronizados para serem usados. Desta maneira, aqui serão descritas as funcionalidades utilizadas para o treinamento e classificação de dados. Os comandos da LibSVM devem ser realizados a partir do terminal do Linux, onde o diretório da biblioteca deve ser acessado:

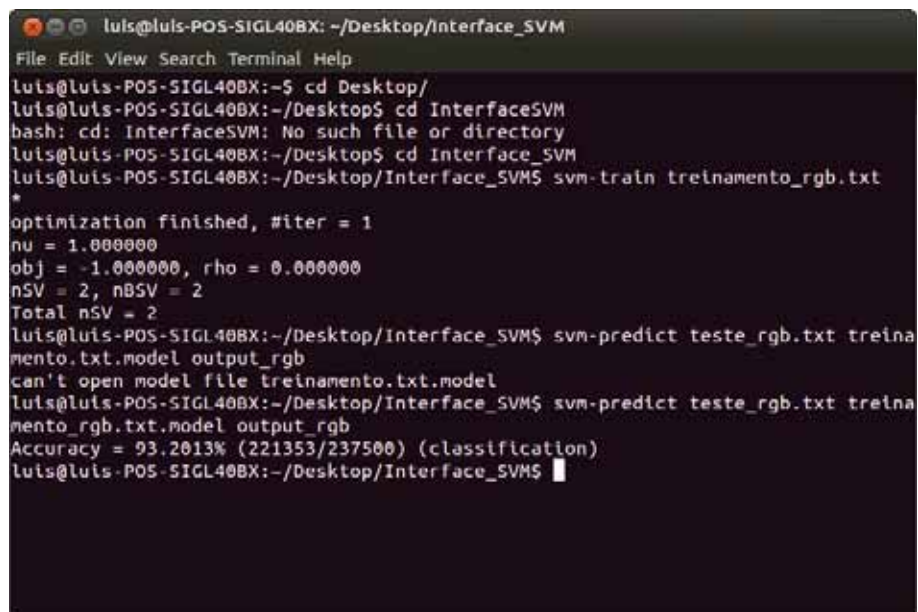
- *svm-train*: comando destinado ao treinamento a partir de um conjunto de dados de treinamento. Como entrada o comando tem o conjunto de dados para o treinamento, e como saída é gerado um arquivo `.model` que contém os dados de saída gerados após o treinamento pela biblioteca. Os arquivos destinados a treinamento, portanto, devem ser utilizados com este comando no terminal;
- *svm-predict*: após o treinamento, o conjunto de dados a ser classificado deve ser utilizado com este comando. Como entrada para este comando de classificação, tem-se os dados a serem classificados, o arquivo `.model` gerado no treinamento e a saída da classificação dos dados, que será gerada no diretório da biblioteca. Os arquivos destinados a teste, portanto, devem ser utilizados com este comando no terminal.

### 5.2.1 Ilustração dos comandos utilizados no Terminal



```
luis@luis-POS-SIGL40BX: ~/Desktop/Interface_SVM
File Edit View Search Terminal Help
luis@luis-POS-SIGL40BX:~$ cd Desktop/
luis@luis-POS-SIGL40BX:~/Desktop$ cd InterfaceSVM
bash: cd: InterfaceSVM: No such file or directory
luis@luis-POS-SIGL40BX:~/Desktop$ cd Interface_SVM
luis@luis-POS-SIGL40BX:~/Desktop/Interface_SVM$ svm-train treinamento_rgb.txt
*
optimization finished, #iter = 1
nu = 1.000000
obj = -1.000000, rho = 0.000000
nSV = 2, nBSV = 2
Total nSV = 2
luis@luis-POS-SIGL40BX:~/Desktop/Interface_SVM$
```

Figura 13: Ilustração da utilização do comando *svm-train*.

A terminal window with a dark background and light text. The window title is 'luis@luis-POS-SIGL408X: ~/Desktop/Interface\_SVM'. The terminal shows a sequence of commands and their outputs. The user navigates to the Desktop directory, then to a subdirectory named 'Interface\_SVM'. They run 'svm-train treinamento\_rgb.txt', which outputs training statistics: 'optimization finished, #iter = 1', 'nu = 1.000000', 'obj = -1.000000, rho = 0.000000', 'nSV = 2, nBSV = 2', and 'Total nSV = 2'. Next, they run 'svm-predict teste\_rgb.txt treinamento\_rgb.txt.model output\_rgb', which outputs 'can't open model file treinamento\_rgb.txt.model'. Finally, they run 'svm-predict teste\_rgb.txt treinamento\_rgb.txt.model output\_rgb', which outputs 'Accuracy = 93.2013% (221353/237500) (classification)'.

```
luis@luis-POS-SIGL408X: ~/Desktop/Interface_SVM
File Edit View Search Terminal Help
luis@luis-POS-SIGL408X:~$ cd Desktop/
luis@luis-POS-SIGL408X:~/Desktop$ cd InterfaceSVM
bash: cd: InterfaceSVM: No such file or directory
luis@luis-POS-SIGL408X:~/Desktop$ cd Interface_SVM
luis@luis-POS-SIGL408X:~/Desktop/Interface_SVM$ svm-train treinamento_rgb.txt
*
optimization finished, #iter = 1
nu = 1.000000
obj = -1.000000, rho = 0.000000
nSV = 2, nBSV = 2
Total nSV = 2
luis@luis-POS-SIGL408X:~/Desktop/Interface_SVM$ svm-predict teste_rgb.txt treina
mento_rgb.txt.model output_rgb
can't open model file treinamento_rgb.txt.model
luis@luis-POS-SIGL408X:~/Desktop/Interface_SVM$ svm-predict teste_rgb.txt treina
mento_rgb.txt.model output_rgb
Accuracy = 93.2013% (221353/237500) (classification)
luis@luis-POS-SIGL408X:~/Desktop/Interface_SVM$
```

Figura 14: Ilustração da utilização do comando *svm-predict*.



# Capítulo 6

## Projeto de Software

Neste capítulo está registrado o projeto do software desenvolvido, com seus respectivos requisitos e ferramentas utilizadas para o desenvolvimento.

### 6.1 Ferramentas de Desenvolvimento

A interface desenvolvida dependeu de diversas ferramentas de desenvolvimento que estão listadas.

- C/C++ (plataforma Linux);
- GTK+ (biblioteca gráfica em C++ para Linux);
- QT Creator (plataforma de desenvolvimento em C/C++ para Linux que implementa funções gráficas);
- Biblioteca LibSVM v.3.14 (desenvolvida em C++ para Linux).

### 6.2 Análise de Requisitos do Sistema

O software desenvolvido deveria apresentar inicialmente as seguintes funcionalidades básicas:

1. Carregamento de imagem para início do processo;
2. Salvamento de imagem com seleção ou fundo selecionados;
3. Alteração do tamanho do cursor para "pintar" uma área;
4. Alteração da cor do cursor para seleção de uma área;

5. Aquisição de características para geração de arquivos de treinamento e classificação (RGB, textura e RGB+textura).

Para obtenção de tais funcionalidades, os seguintes passos foram organizados e executados:

1. Introdução a plataforma Linux;
2. Estudo das ferramentas disponíveis para desenvolvimento em Linux;
3. Estudo da linguagem Python;
4. Estudo das bibliotecas gráficas GTK+ e Qt (C/C++);
5. Definição da linguagem de desenvolvimento do sistema;
6. Implementação do sistema; Estudo da biblioteca LibSVM;
7. Integração dos dados gerados pela interface com a LibSVM;
8. Geração de testes e validação.

# Capítulo 7

## Resultados Obtidos

Neste capítulo serão listados todos os testes realizados visando ilustrar o funcionamento da interface e confrontar os resultados obtidos com os três métodos de organização dos dados obtidos para treinamento e classificação pela LibSVM. Para cada teste realizado, estão dispostos: a imagem original utilizada na seleção da área de interesse e área de fundo, a imagem com as seleções delineadas (azul - área de interesse, vermelho - área de fundo), as imagens classificadas utilizando RGB, textura e RGB + textura como dados para a realização do treinamento e classificação dos testes, e por fim as coordenadas dos *pixels* da área de seleção e de fundo com seus respectivos valores de RGB, incluindo os valores de normalizações obtidos. Nas classificações, os *pixels* pintados de preto correspondem a *pixels* classificados como da área de interesse, enquanto que os pintados de branco são classificados como da área de fundo. Após as seleções, que servem para o treinamento do sistema, todas as características dos *pixels* da imagem são extraídas e classificadas pela LibSVM na etapa de classificação, gerando os resultados a seguir.

### 7.1 Testes

O primeiro teste foi realizado utilizando uma imagem preta e branca para as seleções do usuário. Como pode ser observado, a área de interesse selecionada foi a área branca da imagem, enquanto o fundo selecionado foi a área de cor preta da imagem. Como resultado, a classificação por RGB demonstrou a inversão das cores, já que os *pixels* classificados como interesse são brancos e os de fundo são pretos. Por se tratar de uma imagem de duas cores somente, viu-se pouca eficiência na classificação por textura na figura (b), enquanto em (c) vê-se que a textura tem mais influência na classificação do que as características de RGB.

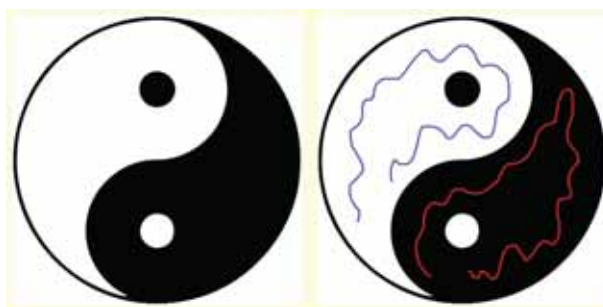


Figura 15: Imagem (1) utilizada para a seleções (interesse e fundo) e classificações do Teste 1.

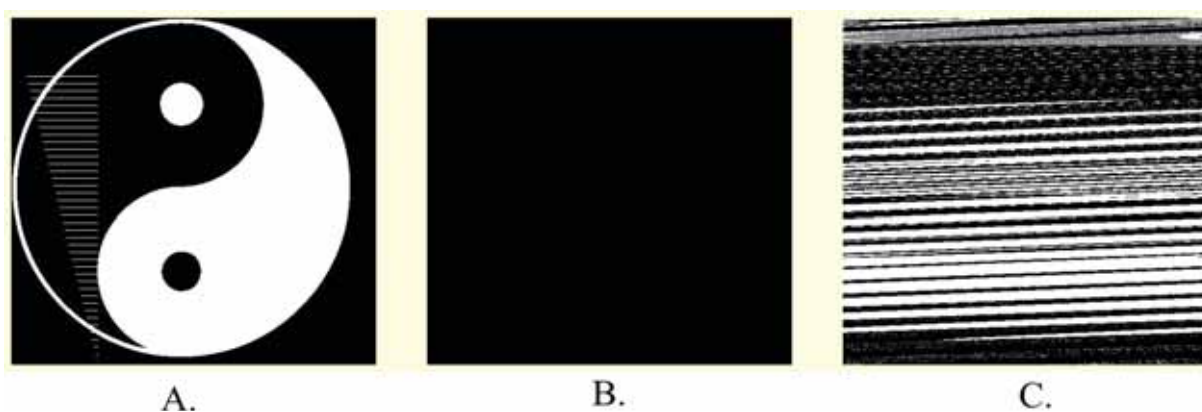


Figura 16: Classificação da Imagem (1) pelos três métodos de classificação.

Tabela 1: Coordenadas e RGB da área de interesse do teste 1

<i>Coordenadas</i>	<i>RGB</i>
48,273	254,255,252
52,267	254,255,252
53,263	254,255,252
57,259	254,255,252
60,254	254,255,252
68,245	254,255,252
73,241	254,255,252
77,237	254,255,252
80,235	254,255,252
84,231	254,255,252
87,229	254,255,252
90,228	254,255,252

93,227	254,255,252
100,225	254,255,252
104,222	254,255,252
107,221	254,255,252
110,219	254,255,252
115,219	254,255,252
126,215	254,255,252
132,213	254,255,252
137,212	254,255,252
141,212	254,255,252
146,212	254,255,252
148,212	254,255,252
151,212	254,255,252
154,212	254,255,252
158,211	254,255,252
161,211	254,255,252
163,210	254,255,252
170,209	254,255,252
173,209	254,255,252
175,209	254,255,252
178,208	254,255,252

Tabela 2: Coordenadas e RGB da área do fundo do teste 1

<i>Coordenadas</i>	<i>RGB</i>
385,214	0,2,0
385,217	0,2,0
385,218	0,2,0
386,220	0,2,0
386,223	0,2,0
387,226	0,2,0
388,228	0,2,0
390,229	0,2,0
390,231	0,2,0
392,234	0,2,0
392,235	0,2,0

392,237	0,2,0
392,239	0,2,0
392,242	0,2,0
393,243	0,2,0
393,244	0,2,0
393,245	0,2,0
393,246	0,2,0
393,248	0,2,0
393,249	0,2,0
394,250	0,2,0
394,251	0,2,0
394,253	0,2,0
394,254	0,2,0
394,256	0,2,0
394,258	0,2,0
394,259	0,2,0
394,260	0,2,0
394,261	0,2,0
394,264	0,2,0
394,265	0,2,0
394,266	0,2,0

Tabela 3: Normalizações do teste 1

0	45	90	135
107184	106722	107184	106722

No segundo teste uma imagem colorida foi utilizada. A partir da classificação por RGB (a), nota-se a influência dos dados selecionados na área de interesse sobre a classificação. Neste teste, vê-se a influência da textura nas classificações (b) e (c), onde (b) demonstrou a classificação por textura com alta taxa de acerto na classificação dos *pixels*, e a pouca influência de RGB na classificação de (c).

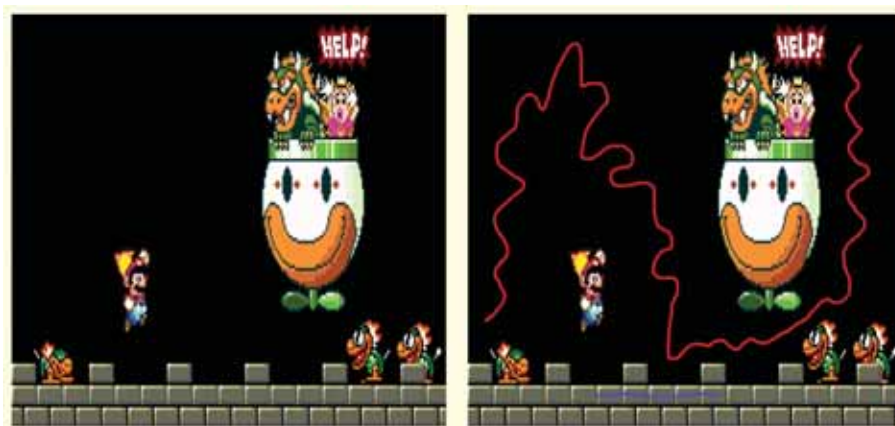


Figura 17: Imagem (2) utilizada para a seleções (interesse e fundo) e classificações do Teste 2.

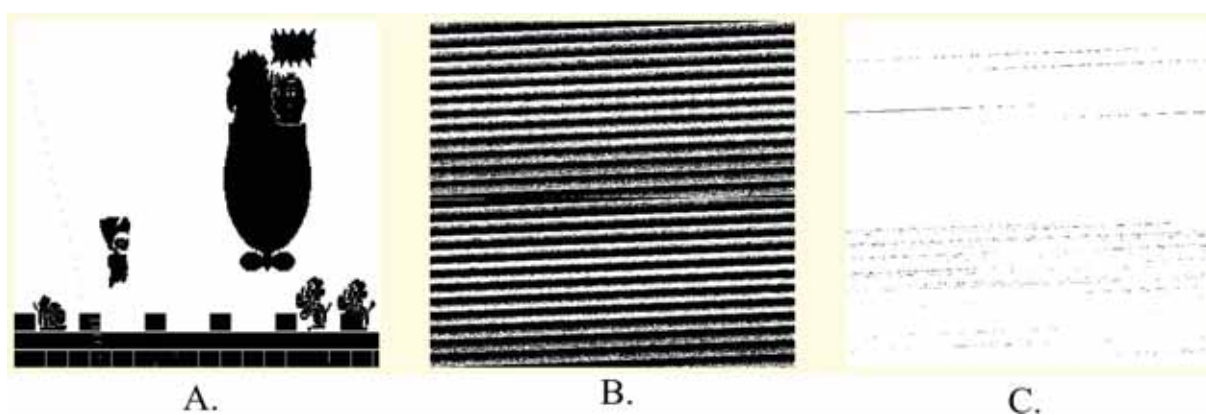


Figura 18: Classificação da Imagem (2) pelos três métodos de classificação.

Tabela 4: Coordenadas e RGB da área de interesse do teste 2

<i>Coordenadas</i>	<i>RGB</i>
25,440	104,104,88
27,440	102,103,88
28,440	70,77,87
31,440	171,170,112
32,439	100,100,86
34,439	104,104,88
35,439	104,104,88
37,439	104,104,88
41,439	104,104,88

43,439	104,104,88
45,439	104,104,88
49,439	104,104,88
51,440	104,104,88
52,440	104,104,88
56,440	105,105,88
57,440	99,100,88
59,440	104,110,97
61,440	157,157,107
62,440	97,97,85
63,440	104,104,88
66,440	104,104,88
67,440	104,104,88
70,440	104,104,88
72,440	104,104,88
77,440	104,104,88
80,440	104,104,88
83,440	104,104,88
85,440	104,104,88
86,440	106,106,88
90,440	209,208,125
91,440	144,144,102

Tabela 5: Coordenadas e RGB da área de fundo do teste 2

<i>Coordenadas</i>	<i>RGB</i>
56,185	0,0,0
55,183	0,0,0
53,181	0,0,0
49,177	0,0,0
47,173	0,0,0
46,169	0,0,0
42,166	0,0,0
37,156	0,0,0
35,151	0,0,0
32,145	0,0,0



32,140	0,0,0
32,135	0,0,0
32,124	0,0,0
32,120	0,0,0
33,118	0,0,0
34,118	0,0,0
37,118	0,0,0
38,118	0,0,0
40,118	0,0,0
41,118	0,0,0
46,118	0,0,0
48,118	0,0,0
50,118	0,0,0
56,117	0,0,0
67,114	0,0,0
72,113	0,0,0
74,113	0,0,0
76,113	0,0,0
78,112	0,0,0
79,111	0,0,0
82,110	0,0,0
82,109	0,0,0

Tabela 6: Normalizações do teste 2

0	45	90	135
129540	129032	129540	129032

A imagem utilizada no teste 3 é a mesma utilizada no teste 2. Neste teste, pode-se verificar a influência da seleção de interesse nas classificações, onde tanto a classificação por RGB (a) quanto as por textura (b) e (c) são diferentes das classificações do teste 2.



Figura 19: Imagem (3) utilizada para a seleções (interesse e fundo) e classificações do Teste 3.

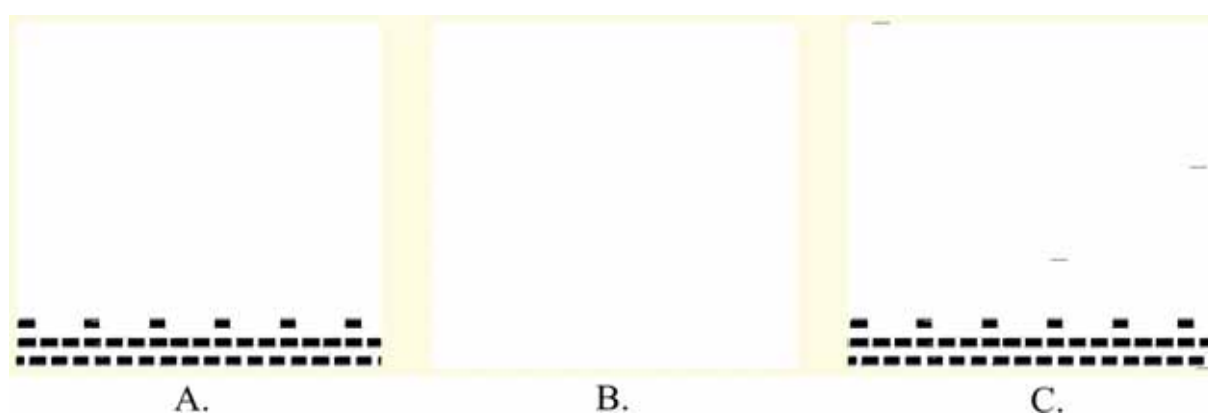


Figura 20: Classificação da Imagem (3) pelos três métodos de classificação.

Tabela 7: Coordenadas e RGB da área de interesse do teste 3

<i>Coordenadas</i>	<i>RGB</i>
31,440	171,170,112
32,439	100,100,86
34,439	104,104,88
35,439	104,104,88
37,439	104,104,88
41,439	104,104,88
43,439	104,104,88
45,439	104,104,88
49,439	104,104,88

51,440	104,104,88
52,440	104,104,88
56,440	105,105,88
57,440	99,100,88

Tabela 8: Coordenadas e RGB da área de fundo do teste 3

<i>Coordenadas</i>	<i>RGB</i>
105,85	0,0,0
108,84	0,0,0
111,83	0,0,0
113,83	0,0,0
119,82	0,0,0
121,82	0,0,0
124,82	0,0,0
127,84	0,0,0
134,86	0,0,0
137,86	0,0,0
141,87	0,0,0
144,87	0,0,0
151,87	0,0,0
155,87	0,0,0
159,86	0,0,0
161,85	0,0,0
166,82	0,0,0
168,80	0,0,0
172,78	0,0,0
175,76	0,0,0
180,73	0,0,0
182,73	0,0,0
184,72	0,0,0
187,71	0,0,0
194,71	0,0,0
196,71	0,0,0
197,73	0,0,0
199,74	0,0,0

200,76	0,0,0
202,79	0,0,0
203,82	0,0,0

Tabela 9: Normalizações do teste 3

0	45	90	135
129540	129032	129540	129032

Para o teste 4 uma nova imagem foi selecionada. Os resultados obtidos aqui são análogos aos obtidos no teste 2.



Figura 21: Imagem (4) utilizada para a seleções (interesse e fundo) e classificações do Teste 4.

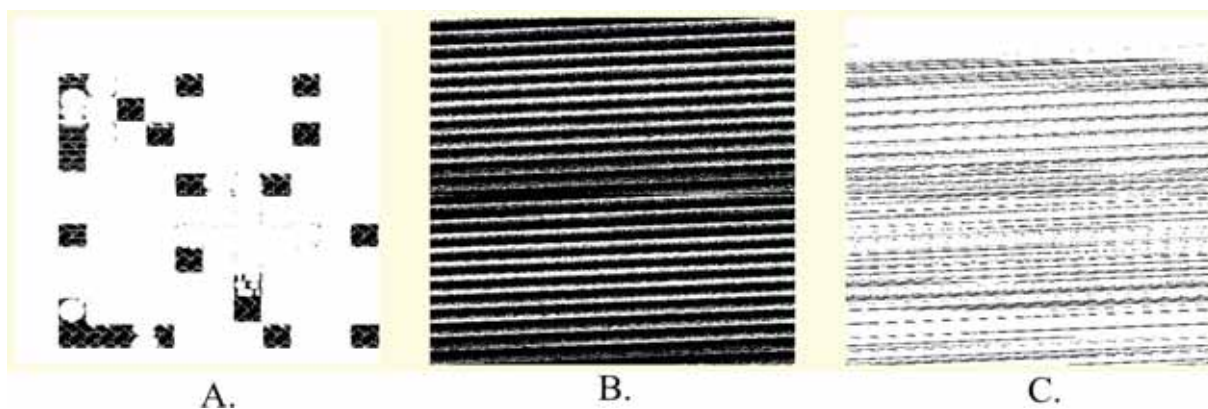


Figura 22: Classificação da Imagem (4) pelos três métodos de classificação.

Tabela 10: Coordenadas e RGB da área de interesse do teste

4

<i>Coordenadas</i>	<i>RGB</i>
76,437	44,120,0
76,436	49,131,0
76,435	49,130,0
76,434	49,130,0
77,434	49,130,0
78,433	48,128,0
82,433	44,120,0
83,433	48,127,0
87,433	49,130,0
89,433	49,130,0
92,433	49,130,0
94,433	49,130,0
96,433	49,130,0
98,434	49,130,0
99,434	48,129,0
101,434	41,112,0
103,435	49,130,0
104,435	49,130,0
105,435	49,130,0
108,435	49,130,0
109,435	49,130,0
110,435	49,130,0
111,435	49,130,0
113,435	49,130,0
114,435	49,130,0
115,436	49,130,0
118,436	41,112,0
119,436	42,113,0
121,436	49,132,0
122,436	49,130,0
125,437	49,130,0
126,437	49,130,0

129,437	49,130,0
---------	----------

Tabela 11: Coordenadas e RGB da área de fundo do teste 4

<i>Coordenadas</i>	<i>RGB</i>
72,229	140,138,148
73,229	140,138,148
74,228	140,138,148
74,227	140,138,148
75,227	140,138,148
76,226	140,138,149
78,225	91,89,91
81,224	253,250,254
83,223	187,185,189
84,223	186,182,187
84,222	72,71,79
86,222	72,71,79
87,221	62,61,70
90,220	124,122,132
92,220	124,122,132
96,219	143,141,151
97,219	143,141,151
99,219	139,137,146
100,219	106,104,108
103,221	61,60,70
105,221	62,61,70
107,222	72,71,79
108,222	72,71,79
112,224	207,203,208
113,224	208,205,209
115,224	185,181,185
115,226	138,136,147
118,227	92,90,92
119,228	94,92,95
121,229	142,140,150
123,230	144,142,152

124,231	109,107,117
127,231	109,107,117
129,231	109,107,117
132,232	54,53,62
133,232	51,50,59
135,232	51,50,59
137,232	51,50,59
138,232	51,50,59
141,232	56,55,65
143,231	109,107,117
144,231	109,107,117
146,231	109,107,117
146,229	140,138,148
149,228	140,138,148
150,228	140,138,148
151,227	140,138,148
153,227	140,138,148
155,227	140,138,148
156,226	96,94,97
159,225	209,205,209

Tabela 12: Normalizações do teste 4

0	45	90	135
129540	129032	129540	129032

Para o quinto teste, a imagem do teste 4 foi utilizada. Os resultados obtidos são análogos ao caso dos testes 2 e 3, onde pode ser vista a influência das características extraídas nas seleções para a etapa de treinamento em todas as classificações realizadas.



Figura 23: Imagem (5) utilizada para a seleções (interesse e fundo) e classificações do Teste 5.

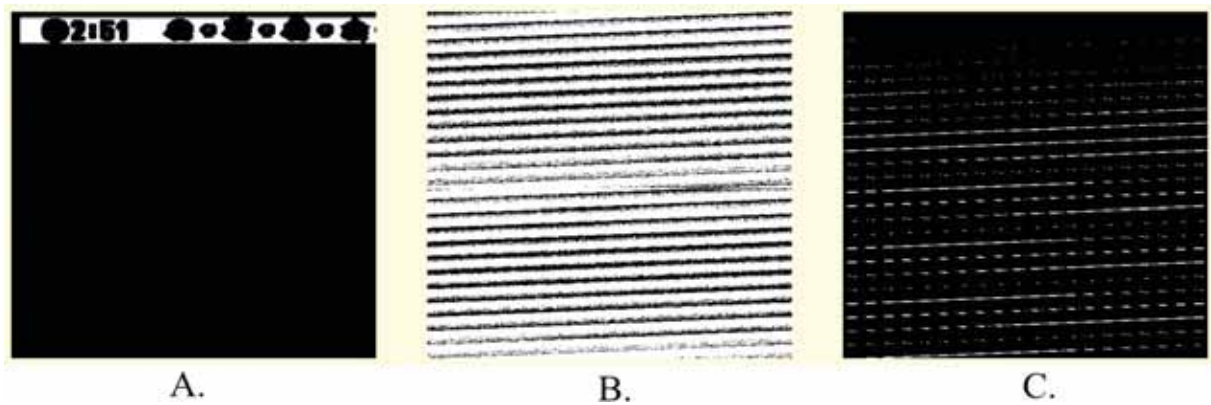


Figura 24: Classificação da Imagem (5) pelos três métodos de classificação.

Tabela 13: Coordenadas e RGB da área de interesse do teste

5

<i>Coordenadas</i>	<i>RGB</i>
266,298	230,227,198
268,299	230,227,198
268,300	230,227,200
273,301	230,227,198
274,301	230,227,192
276,302	230,227,96
277,303	230,227,136
280,304	230,227,126
281,304	230,228,97



284,304	230,211,50
284,305	230,200,4
289,305	230,227,96
292,305	230,216,63
294,305	230,195,5
297,305	230,229,103
301,305	230,229,104
303,305	230,227,99
305,305	230,228,102
306,305	230,229,101
308,305	230,227,204
312,305	230,227,97
313,305	230,227,97
314,305	230,227,97
316,305	230,227,107
319,304	230,227,201

Tabela 14: Coordenadas e RGB da área de fundo do teste 5

<i>Coordenadas</i>	<i>RGB</i>
156,41	0,122,0
157,41	0,122,0
159,41	0,122,0
160,40	0,122,0
161,39	0,122,0
164,39	0,122,0
164,38	0,122,0
165,38	0,122,0
167,37	0,122,0
167,36	0,122,0
167,35	0,122,0
168,35	0,122,0
169,34	0,122,0

Tabela 15: Normalizações do teste 5

0	45	90	135
129540	129032	129540	129032

Por fim, um último teste foi realizado com uma imagem inédita para garantir o funcionamento do sistema. Como pode ser visto aqui e nos demais testes, a classificação pura por RGB (a) é visualmente mais fácil de ser entendida. Já as classificações dependentes de textura (b) e (c) possuem entendimento mais abstrato, entretanto, ainda são pertinentes por apresentarem alta taxa de acerto na classificação dos *pixels* da imagem.



Figura 25: Imagem (6) utilizada para a seleções (interesse e fundo) e classificações do Teste 6.

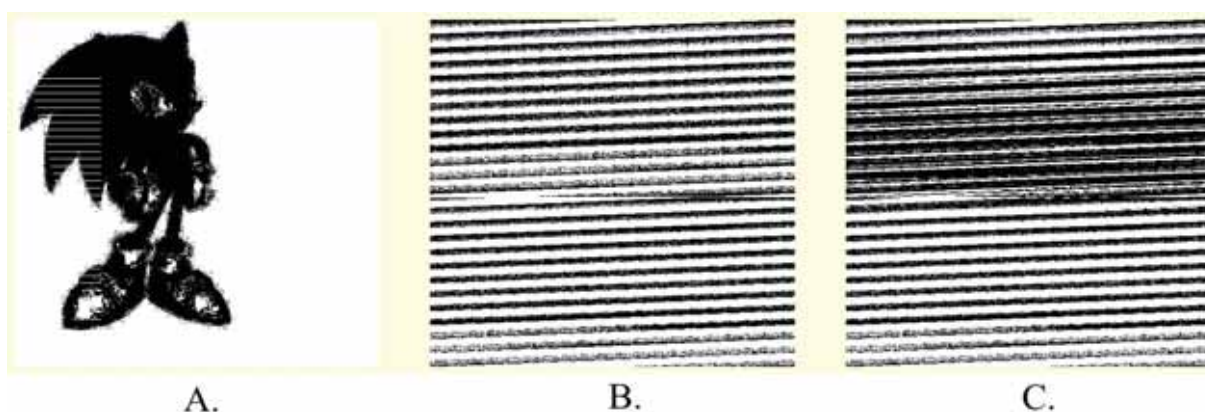


Figura 26: Classificação da Imagem (6) pelos três métodos de classificação.

Tabela 16: Coordenadas e RGB da área de interesse do teste

6

<i>Coordenadas</i>	<i>RGB</i>
64,78	6,30,180
65,78	8,29,180
68,76	6,30,180
69,76	6,30,180
70,76	6,30,182
71,76	5,30,182
73,76	8,29,182
74,75	5,31,176
76,75	2,33,175
77,75	1,33,176
81,73	5,31,180
82,72	6,30,180
84,72	6,30,180
85,71	6,30,180
88,71	6,30,180
89,71	6,30,180
90,70	6,30,180
93,70	6,30,180
95,70	6,30,180
97,70	6,30,180
100,70	6,30,180
103,70	6,30,180
104,70	8,30,176
106,70	5,29,187
106,71	5,30,185
106,72	5,30,185
106,73	5,30,185
107,73	5,30,185
107,75	5,30,182
108,75	5,31,180
108,77	6,30,180
109,77	6,30,180

109,79	6,30,180
109,80	6,30,180
110,81	6,30,180
110,82	6,30,180

Tabela 17: Coordenadas e RGB da área de fundo do teste 6

<i>Coordenadas</i>	<i>RGB</i>
29,320	255,255,255
29,319	255,255,255
29,318	255,255,255
29,316	255,255,255
29,315	255,255,255
29,314	255,255,255
29,313	255,255,255
30,311	0,0,1
30,310	255,255,255
31,310	255,255,255
33,308	255,255,255
33,307	255,255,255
,306	0,0,1
34,305	255,255,255
36,304	255,255,255
37,303	255,255,255
37,302	255,255,255
38,301	255,255,255
38,300	255,255,255

Tabela 18: Normalizações do teste 6

0	45	90	135
129540	129032	129540	129032

# Capítulo 8

## Conclusão

A LibSVM é uma poderosa ferramenta de classificação e reconhecimento de padrões usada em larga escala nos mais diversos problemas de classificações da área de pesquisa acadêmica. Todavia, seu entendimento e aplicação tornam-se complexos por se tratar de uma ferramenta especificamente acadêmica. Portanto, esta aplicação que demonstra o funcionamento das máquinas de vetores de suporte de maneira mais simplista para usuários comuns teve pertinência de desenvolvimento, apresentando resultados satisfatórios e auto-explicativos.

Seis testes foram realizados, onde imagens foram carregadas com a interface, as áreas de seleção e fundo foram selecionadas pelo usuário e os arquivos de texto foram gerados, comportando RGB, textura e RGB+textura. Com estes arquivos gerados, a LibSVM foi utilizada para todos, gerando os resultados apresentados no Capítulo 6. Visualmente, os resultados obtidos apresentam maior facilidade de entendimento com os experimentos somente com características de RGB. Isso se deve ao fato de os dados de RGB serem concretos e brutos, não havendo qualquer tipo de cálculo ou variação da imagem original, apresentando taxa média de acerto de classificação de 15,436%. Já com os experimentos de textura, a taxa de acerto média dos testes passa a ser 76,326%, entretanto, visualmente o resultado apresenta maior dificuldade de entendimento, visto que a imagem inicial passa por uma transformação para tons de cinza, e em seguida são calculadas matrizes de co-ocorrência, probabilidades para finalmente se obter os descritores de textura. Desta forma, os valores utilizados para treinamento e classificação destes experimentos são resultado de uma função, não sendo os especificamente obtidos na aquisição dos dados da imagem, mas sim calculados. Por fim, os experimentos que mesclam dados de RGB e textura descreveram aproximadamente 39,787% de taxa média de acerto na classificação, visto que os dados que compõem a parte de textura dos arquivos de texto de treinamento e classificação representam 80% destes arquivos, acarretando em uma maior influência dos valores de textura na classificação.

Sendo assim, a interface demonstra eficientemente a classificação das imagens que acontece de

acordo com as áreas de interesse e fundo selecionadas, afetando as imagens classificadas resultantes, como pode ser verificado no Capítulo 6. Melhorias ainda podem ser agregadas a este projeto, como a implantação de uma classificação multi-classe, onde várias partes da imagem seriam selecionadas, não somente 'seleção' e 'fundo' ou ainda a implementação de outros descritores de textura para os experimentos de textura e RGB+textura.

# Referências Bibliográficas

- [A. C. Lorena, 2007] A. C. Lorena, A. C. P. L. F. de Carvalho. 2007. Uma Introdução as Support Vector Machines. *RITA - Revista de Informática Teórica e Aplicada*, **XIV**(2), 43–67.
- [Burges, 1998] Burges, C. J. C. 1998. A Tutorial on support vector machines for pattern recognition. *Knowledge Discovery and Data Mining*, **2**(2), 1–43.
- [Chang & Lin, 2011] Chang, C.-C., & Lin, C.-J. 2011. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, **2**, 27:1–27:27.
- [Cortes & Vapnik, 1995] Cortes, C., & Vapnik, V. 1995. Support-Vector Networks. *Machine Learning*, 273–297.
- [Falcão *et al.*, 2004] Falcão, A.X., Stolfi, J., & Lotufo, R.A. 2004. The Image Foresting Transform: Theory, Algorithms, and Applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **26**(1), 19–29.
- [Haralick, 1979] Haralick, R.M. 1979. Statistical and Approaches to Texture. *Proceedings of the IEEE*, **67**(5), 786–804.
- [Haykin, 1994a] Haykin, S. 1994a. *Neural networks: a comprehensive foundation*. Prentice Hall.
- [Haykin, 1994b] Haykin, S. 1994b. *Neural networks: a comprehensive foundation*. Prentice Hall.
- [Haykin, 1999] Haykin, S. 1999. *Neural networks: a comprehensive foundation*. Pearson, Prentice Hall.
- [K. R. Müller & Schölkopf, 2001] K. R. Müller, S. Mika, G. Ratsch K. Tsuda, & Schölkopf, B. 2001. An introduction to kernel-based learning algorithms. *IEEE Transactions on Neural Networks*, **2**(12), 181–201.
- [Kohonen, 1989] Kohonen, T. 1989. *Self-Organizing and Associative Memory*. Springer-Verlag.
- [Kohonen, 1990] Kohonen, T. 1990. Self-Organizing Map. *Proceedings of the IEEE*, **78**, 1464–1480.

- [Kohonen, 1993] Kohonen, T. 1993. Things You Haven't Heard About Self-Organizing Map. *IEEE International Conference on Neural Networks*, **III**, 1147–1156.
- [Kovacs, 1996] Kovacs, Z. L. 1996. *Redes Neurais Artificiais: Fundamentos e Aplicações*. Escola Politécnica da Universidade de São Paulo.
- [M. C. P. Souto, 2003] M. C. P. Souto, A. C. Lorena, A.C.B. Delbem A. C. P. L. F. Carvalho. 2003. Técnicas de Aprendizado de Máquina para problemas de Biologia Molecular. *XXIII Congresso da Sociedade Brasileira de Computação*, 103–152.
- [Martins *et al.*, 2005] Martins, E. R. S., Marques, P. M., Oliveira, L. F., Pereira-Jr, R. R., & Trad, C. S. 2005. Caracterização de lesões intersticiais de pulmão em radiograma de tórax utilizando análise local de textura. *Radiologia Brasileira*, **38**(6), 421–426.
- [Mitchell, 1997] Mitchell, T. M. 1997. *Machine Learning*. WCB/McGraw-Hill.
- [Pan *et al.*, 2004] Pan, J., Qiao, Y., & Sun, S. 2004. A Fast K Nearest Neighbors Classification Algorithm. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, **E87-A**.
- [Papa & Falcão, 2009] Papa, J. P., & Falcão, A. X. 2009. A Learning Algorithm for the Optimum-Path Forest Classifier. *7th IAPR Workshop on Graph-based Representations in Pattern Recognition*.
- [Papa *et al.*, 2009] Papa, J. P., Falcão, A. X., & Suzuki, C. T. N. 2009. Supervised Pattern Classification based on Optimum-Path Forest. *International Journal of Imaging Systems and Technology*, **19**(2), 120–131.
- [Raghu *et al.*, 1995] Raghu, P. P., Poongodi, R., & Yegnanarayana, B. 1995. A Combined Neural Network Approach for Texture Classification. *Neural Networks*, **8**, 975–987.
- [Schölkopf & Smola, 2002] Schölkopf, B., & Smola, A. J. 2002. *Learning with Kernels*. MIT Press.
- [Schölkopf *et al.*, 2000] Schölkopf, B., Williamson, R. C., & Bartlett, P. L. 2000. New Support Vector Algorithms. *Neural Computation*, 1207–1245.
- [Vapnik, 1999] Vapnik, V. N. 1999. An Overview of Statistical Learning Theory. *IEEE Transactions on Neural Networks*, **10**, 988–999.