



Universidade Federal de Juiz de Fora
Faculdade de Engenharia Elétrica

Lucas Corrêa Netto Machado

MÉTODO DE SEGMENTAÇÕES GEOMÉTRICAS SUCESSIVAS PARA
TREINAMENTO DE REDES NEURAS ARTIFICIAIS

Dissertação de Mestrado

Juiz de Fora
2013

Lucas Corrêa Netto Machado

Método de Segmentações Geométricas Sucessivas para Treinamento de Redes Neurais
Artificiais

Dissertação apresentada ao Programa de Pós-graduação em Energia Elétrica, área de concentração: Sistemas de Energia, da Universidade Federal de Juiz de Fora como requisito parcial para obtenção do grau de mestre.

Orientador: Prof. Leonardo de Mello Honório

Juiz de Fora
2013

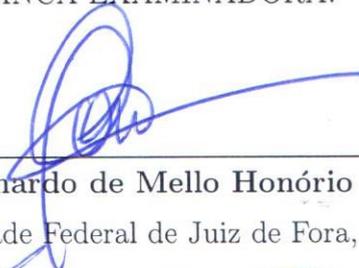
Lucas Corrêa Netto Machado

Método de Segmentações Geométricas Sucessivas para Treinamento de Redes Neurais
Artificiais

Dissertação apresentada ao Programa de Pós-graduação em Energia Elétrica, área de concentração: Sistemas de Energia, da Universidade Federal de Juiz de Fora como requisito parcial para obtenção do grau de mestre.

Aprovada em 22 de novembro de 2013.

BANCA EXAMINADORA:



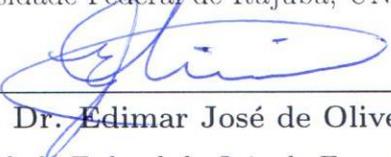
Prof. Dr. Leonardo de Mello Honório - Orientador
Universidade Federal de Juiz de Fora, UFJF



Prof. Dr. Augusto Santiago Cerqueira - Coorientador
Universidade Federal de Juiz de Fora, UFJF



Prof. Dr. Carlos Henrique Valério de Moraes
Universidade Federal de Itajubá, UNIFEI



Prof. Dr. Edimar José de Oliveira
Universidade Federal de Juiz de Fora, UFJF

AGRADECIMENTOS

Agradeço meus pais Alvaro e Roberta, e minha irmã Caroline por todo o apoio que sempre recebi vindo deles, por sempre terem confiado em mim. A Todos meus amigos pelos momentos de trabalho e por aqueles de distração proporcionados.

Aos amigos do Grupo de Robótica Inteligente - GRIn, Ana Sophia, Elias, Leandro, Exuperry, Wolmar, Amadeu, Alexandre, Murillo, pelo apoio no desenvolvimento dos vários trabalhos durante o período que estivemos juntos.

Em especial, aos professores Leonardo de Mello Honório, Augusto Santiago Cerqueira, pela possibilidade de crescimento profissional dada por eles.

RESUMO

Este trabalho apresenta uma técnica para treinamento de Redes Neurais Artificiais (RNA), capaz de obter os parâmetros da rede através dos dados disponíveis para treinamento, sem necessidade de estabelecer a arquitetura da rede *a priori*, denominado Método de Segmentações Geométricas Sucessivas (MSGs). O MSGs agrupa os dados de cada classe em Hipercaixa (HC) onde cada caixa é alinhada de acordo com os eixos de maior distribuição de seu conjunto de pontos. Sendo as caixas linearmente separáveis, um hiperplano de separação é identificado originando um neurônio. Caso não seja possível a separação por um único hiperplano, uma técnica de quebra é aplicada para dividir os dados em classes menores para obter novas HCs. Para cada subdivisão novos neurônios são adicionados à rede. Os resultados dos testes realizados apontam para um método rápido e com alta taxa de sucesso.

Palavras chave: Rede Neural Artificial. Reconhecimento de padrões. Teorema do Hiperplano de Separação. Hipercaixas. Segmentação Geométrica Sucessiva. Auto-geração de Neurônio.

ABSTRACT

This work presents a technique for Artificial Neural Network (ANN) training, able to get the network parameters from the available data for training, without establishing the network architecture a priori, called Successive Geometric Segmentation Method (SGSM). The SGSM groups the data of each class into hyperboxes (HB) aligned in accordance with the largest axis of its points distribution. If the HB are linearly separable, a separating hyperplane may be identified resulting a neuron. If it is not, a segmentation technique is applied to divide the data into smaller classes for new HB. For each subdivision new neurons are added to the network. The tests show a rapid method with high success rate.

Keywords: Artificial neural nets. Patterns classification. Hyperplane separation theorem. Hyperbox. Successive Geometric Segmentation. Self-generation of neuron.

LISTA DE ILUSTRAÇÕES

| | | |
|----|--|----|
| 1 | Exemplo de classes não separáveis em duas dimensões | 18 |
| 2 | Exemplo de classes não-convexas | 19 |
| 3 | Exemplo de classes não contínuas no espaço de soluções | 19 |
| 4 | Ilustração de um neurônio, elemento básico de uma rede neural biológica (PACKTER, 2011) | 22 |
| 5 | Modelo adotado para um neurônio artificial com <i>bias</i> | 24 |
| 6 | Modelo adotado para um neurônio artificial com <i>bias</i> como peso sináptico | 24 |
| 7 | Neurônio representação gráfica para um neurônio | 25 |
| 8 | função de ativação com limiar | 26 |
| 9 | função de ativação linear por partes com coeficientes $a = 0,5$, $a = 1$, $a = 1,8$ | 27 |
| 10 | função de ativação sigmóide com coeficientes $a = 1,5$, $a = 2$, $a = 5$ | 28 |
| 11 | Curvas bipolares utilizadas como função de ativação | 29 |
| 12 | exemplo de rede neural em camada única | 30 |
| 13 | Exemplo de rede neural multicamada 5-3-2 | 31 |
| 14 | camada competitiva | 32 |
| 15 | Diagrama do processo de aprendizado supervisionado | 33 |
| 16 | Diagrama do processo de aprendizado sem supervisão | 34 |
| 17 | Separação linear devido a um neurônio ($w_0 = -0,5$, $w_1 = 0,6$ e $w_2 = 0,3$) | 35 |
| 18 | Rede Neural multicamadas com 2 neurônios na camada oculta | 35 |
| 19 | Separação linear devido a um neurônio ($w_0 = 0,5$, $w_1 = 0,6$ e $w_2 = 0,3$) | 36 |
| 20 | Saída de uma rede neural com duas camadas (pesos sinápticos do neurô- nio da camada de saída: $w_0 = -1$, $w_1 = -1$ e $w_2 = 1$) | 36 |

| | | |
|----|---|----|
| 21 | Rede neural artificial com 3 camadas 3-6-2-1 | 37 |
| 22 | Representação da saída de uma rede neural 3 camadas | 38 |
| 23 | gaussiana com média $\mu = 0$ e desvio padrão $\sigma = 1$ | 39 |
| 24 | Exemplo de histogramas de um mesmo conjunto de dados com número de bins variável | 39 |
| 25 | Mistura de Gaussiana | 40 |
| 26 | Estimador de densidade e pontos centrais de cada kernel | 41 |
| 27 | Região R utilizada para formulação da estimação de densidade | 42 |
| 28 | Densidade estimada utilizando diferentes larguras do <i>kernel</i> , e tamanhos diferentes do conjunto de eventos | 44 |
| 29 | Visualização de diferentes funções kernel | 45 |
| 30 | Colisão entre dois objetos | 47 |
| 31 | Diferentes formas de volumes envolventes | 47 |
| 32 | Quebra em diferentes volumes envolventes | 48 |
| 33 | teste de colisão entre esferas | 48 |
| 34 | teste de colisão entre caixas alinhadas pelos eixos globais | 49 |
| 35 | teste de colisão entre caixas orientadas | 50 |
| 36 | Fluxograma do <i>MSGs</i> | 53 |
| 37 | Projeções de \vec{X}^{1L} definindo os limites da Caixa Envolvente $\vec{\mathcal{H}}^{1L}$ | 56 |
| 38 | Caixa envolvente $\vec{\mathcal{H}}$ e suas projeções parciais em π_1 e π_2 | 56 |
| 39 | Identificação do Eixo de Separação | 58 |
| 40 | Definição do Hiperplano de Separação | 60 |
| 41 | Hiperplano de Separação | 61 |
| 42 | Oriented Bounding Hyper-Box (OBHB) com três pontos gerados pertencentes à mesma casa | 62 |
| 43 | distribuição gaussiana para três pontos de mesma fonte e somatório das distribuições | 62 |

| | | |
|----|--|----|
| 44 | Instâncias de duas casas diferentes | 63 |
| 45 | distribuição de probabilidade da HC | 64 |
| 46 | quebra da HC | 64 |
| 47 | Exemplo de Geração de Rede Neural utilizando o TES | 66 |
| 48 | Otimizações aplicadas ao TES | 66 |
| 49 | <i>Toolbox</i> de Reconhecimento de Padrões no <i>Matlab</i> | 70 |
| 50 | Exemplo de conjunto de teste gerado | 73 |
| 51 | Exemplo utilizado para teste | 73 |
| 52 | Conjunto para o problema da espiral dupla | 76 |
| 53 | Hipercaixas no problema da espiral dupla | 76 |
| 54 | Regiões de separação no problema da espiral dupla pelo MSGS | 77 |
| 55 | Função tabuleiro | 86 |

LISTA DE TABELAS

| | | |
|---|--|----|
| 1 | Diferentes funções para kernel | 44 |
| 2 | Resultados comparativos de tempo de treinamento com FT (em segundos) | 71 |
| 3 | Resultados comparativos com FT - dados aleatórios seguindo mesma distribuição dos dados de treino, 50 pontos por casa | 72 |
| 4 | Resultados Comparativos com FT - dados seguindo distribuição pro- posta na Figura 50 | 72 |
| 5 | Resultados Comparativos com FT - dados seguindo distribuição pro- posta na Figura 51 | 72 |
| 6 | Descrição dos Bancos Públicos | 74 |
| 7 | Resultados Quantitativos | 75 |

LISTA DE ABREVIATURAS E SIGLAS

AC Árvore de Colisão

HC Hipercaixa

MLP Multilayer Perceptron

MSGs Método de Segmentações Geométricas Sucessivas

OBHB Oriented Bounding Hyper-Box

RNA Rede Neural Artificial

TES Teorema do Eixo de Separação

THS Teorema do Hiperplano de Separação

LISTA DE SIMBOLOS

| | |
|---|---|
| $\vec{x} = \{x_1, \dots, x_n\}$ | ponto no espaço \mathbb{R}^n no sistema cartesiano de coordenadas. |
| $\vec{X} = \{\vec{x}^1, \dots, \vec{x}^m\}$ | conjunto de m pontos \vec{x} . |
| $\vec{Y} = \{\vec{X}^1, \dots, \vec{X}^q\}$ | conjunto de q classes que devem ser identificadas. |
| \vec{x}_c | vetor em \mathbb{R}^n representando o ponto médio de \vec{X} . |
| $\vec{w} = \{w_1, \dots, w_n\}$ | vetor de escalares no espaço \mathbb{R}^n . |
| b | escalar no espaço \mathbb{R}^1 . |
| $\mathcal{X}(\vec{w}, b, \vec{x})$ | hiperplano no espaço \mathbb{R}^n . Contém o ponto \vec{x} , possui normal \vec{w} e parâmetro de ajuste b . |
| $cov(\vec{X})$ | vetor pertencente a $\mathbb{R}^{n \times n}$ representando a matriz de covariância do conjunto de pontos \vec{X} . |
| $\vec{\Lambda} = \{\lambda_1, \dots, \lambda_n\}$ | vetor em \mathbb{R}^n contendo os autovalores de $cov(\vec{X})$ |
| λ_i | escalar representando o autovalor da dimensão i de $cov(\vec{X})$ |
| $\vec{\Pi} = \{\vec{\pi}_1, \dots, \vec{\pi}_n\}$ | vetor em \mathbb{R}^n contendo os autovetores de $cov(\vec{X})$ |
| $\vec{\pi}_i$ | vetor em \mathbb{R}^n representando o autovetor associado ao autovalor λ_i de $cov(\vec{X})$. |
| $\mathcal{C}(\vec{X})$ | sistemas de coordenadas em \mathbb{R}^n com centro em \vec{x}_c e eixos dados por $\vec{\Pi}$. |
| π_i | eixo do sistema de coordenadas $\mathcal{C}(\vec{X})$ associado ao vetor $\vec{\pi}_i$ |
| $\vec{v} = \{v_1, \dots, v_n\}$ | ponto no espaço \mathbb{R}^n utilizando o sistema de coordenadas $\mathcal{C}(\vec{X})$. |
| \vec{V} | conjunto de pontos \vec{X} no sistema de coordenadas $\mathcal{C}(\vec{X})$ |
| $Pr(\vec{x}^k, \pi_i)$ | escalar contendo o valor da projeção de um vetor $\vec{x}^k \in \vec{X}$ sobre o eixo i do sistema de coordenadas $\mathcal{C}(\vec{X})$ |
| $L\vec{P}r(\vec{V}, \pi_i)$ | vetor em \mathbb{R}^2 constituído do maior (v_{max}) e do menor (v_{min}) valor da projeção dos pontos de \vec{V} no eixo i de $\mathcal{C}(\vec{X})$. |
| $\vec{\mathcal{H}} = \{L\vec{P}r(\vec{V}, \vec{\Pi})\}$ | hipercaixa representada pelos valores de menor e maior projeção dos pontos de \vec{V} sobre cada um dos eixos de $\vec{\Pi}$. |

| | |
|--|---|
| $\Delta_{\vec{\mathcal{H}}_{1,i}}$ | escalar representando o comprimento da projeção da hipercaixa sobre o eixo i , ou seja, $v_{max} - v_{min}$. |
| $\delta_{\vec{\mathcal{H}}_{1,i}}$ | escalar referente a projeção parcial da hipercaixa sobre o eixo i , ou seja, $\Delta_{\vec{\mathcal{H}}_{1,i}}/2$ |
| $\vec{\delta}_{\vec{\mathcal{H}}_{1,i}}$ | vetor na direção de π_i e de módulo $\delta_{\vec{\mathcal{H}}_{1,i}}$ |
| $\mathcal{S}(\vec{\mathcal{H}}_r, \vec{\mathcal{H}}_s, i)$ | segmento de reta no eixo i do sistema de coordenadas $\mathcal{C}(\vec{X}^r)$ situado entre $\vec{P}r(\vec{V}^r, \vec{\pi}_i)$ e $\vec{P}r(\vec{V}^s, \vec{\pi}_i)$, onde $r \leq q$, $s \leq q$ e $r \neq s$. |
| $\vec{\mathcal{H}}_0$ | vetor em \mathbb{R}^n representando o centro da hipercaixa no sistema de coordenadas $\mathcal{C}(\vec{X})$. |

SUMÁRIO

| | | |
|----------|---|-----------|
| 1 | Introdução | 17 |
| 1.1 | Objetivos | 18 |
| 2 | Fundamentação Teórica | 21 |
| 2.1 | Redes Neurais | 21 |
| 2.1.1 | Redes Neurais Biológicas | 22 |
| 2.1.2 | Modelo de um Neurônio Artificial | 23 |
| 2.1.3 | Redes Neurais Artificiais | 24 |
| 2.1.3.1 | Tipos de Função de Ativação | 25 |
| 2.1.3.2 | Arquiteturas de Redes Neurais | 29 |
| | Redes Neurais em Camada Única | 29 |
| | Redes Neurais Multicamadas | 30 |
| | Rede Neural Recorrente | 31 |
| 2.1.3.3 | Aprendizado | 32 |
| | Aprendizado Supervisionado | 32 |
| | Aprendizado sem supervisão | 33 |
| 2.1.4 | Interpretação Geométrica de um Neurônio | 34 |
| 2.1.4.1 | Combinações de neurônios | 35 |
| 2.1.4.2 | Redes com três camadas | 37 |
| 2.2 | Estimador de densidade | 37 |
| 2.2.1 | Kernel Density Estimator | 40 |
| 2.2.1.1 | Formulação | 41 |
| 2.2.1.2 | Kernel | 43 |

| | | |
|----------|---|-----------|
| 2.2.1.3 | Smoothing Kernels | 43 |
| 2.3 | Teste de Colisão em Objetos | 46 |
| 2.3.1 | Esfera | 48 |
| 2.3.2 | Caixas Alinhadas Pelos Eixos | 49 |
| 2.3.3 | Caixas orientadas | 50 |
| 2.3.4 | Teorema do Eixo de Separação | 51 |
| 3 | Descrição do Método de Segmentações Geométricas Sucessivas | 52 |
| 3.1 | Introdução | 52 |
| 3.2 | Construção de hipercaixas | 52 |
| 3.3 | Teste de Colisão | 57 |
| 3.3.1 | Teorema do Eixo de Separação (TES) | 57 |
| 3.3.2 | Teorema do Hiperplano de Separação (THS) | 59 |
| 3.4 | Segmentação Geométrica dos OBHB | 60 |
| 3.4.1 | Ponto de Quebra | 62 |
| 3.5 | Reagrupamento de OBHB | 64 |
| 3.6 | Otimização dos Hiperplanos | 66 |
| 3.7 | Montagem da Rede Neural | 67 |
| 4 | Resultados | 69 |
| 4.1 | Testes utilizando a função Tabuleiro | 69 |
| 4.2 | Testes utilizando bancos de dados públicos | 74 |
| 4.3 | Problema da Espiral Dupla | 75 |
| 5 | Conclusões | 78 |
| 5.1 | Considerações Finais | 78 |
| 5.2 | Trabalhos Futuros | 79 |
| | Referências | 80 |

| | |
|--|----|
| Apêndice A - Interpretação geométrica de um neurônio | 83 |
| Apêndice B - Função Tabuleiro | 85 |

1 INTRODUÇÃO

Para se treinar uma Rede Neural Artificial (RNA) é necessário, na maioria dos algoritmos disponíveis, definir à priori a topologia a ser utilizada (quantidade de camadas, quantidade de neurônios em cada camada, existência ou não de realimentação e as funções de ativação dos neurônios). Para o caso das redes feed-forward, foi demonstrado ser necessário uma rede de três camadas (entrada, duas intermediárias e saída) para que a rede consiga aproximar qualquer tipo de função, isto é conhecido como teorema de Komogorov (BEALE; JACKSON, 1990). Apesar disto, não existe maneira determinística para definir a quantidade ótima de neurônios necessários na camada intermediária para garantir o bom funcionamento da rede. Algumas abordagens foram desenvolvidas com este intuito, obtendo apenas sucessos parciais (HAN; KAMBER; PEI, 2011).

Já o processo de classificar é uma necessidade real presente em diversas áreas (ABE, 2010) (HSU; LIN, 2002) e (SILVA; FERREIRA; VELASQUEZ, 2008), sendo duas abordagens bastante utilizadas atualmente: uma baseada em técnicas de classificação geométricas, como *support vector machines* ou SVM (ABE, 2010), e (CESA-BIANCHI; GENTILE; ZANIBONI, 2006) e a segunda em teorias de inteligência computacional, como as redes neurais artificiais (SILVA; FERREIRA; VELASQUEZ, 2008). No caso da técnica baseada em SVM, obtém-se altas taxas de acerto com boa fundamentação matemática, no entanto, apresentam dificuldades para escolha da função kernel e determinação de parâmetros e margem. Por outro lado, as redes neurais artificiais podem ser aplicadas em praticamente todos os problemas de classificação. Entretanto necessita da especificação de diversos parâmetros, principalmente os relacionados à topologia da rede (HUANG; FENG; CAO, 2008), como o número de camadas e o número de neurônios por camada, que são dependentes do tipo de problema a ser tratado.

Existem alguns fatores que mostram a qualidade de um algoritmo de classificação, entre eles pode-se destacar: o desempenho do método utilizado, a velocidade de resposta, a complexidade do modelo. Apesar de ambos serem importantes, o fator determinante da escolha de um entre os diversos métodos apresentados, consiste no

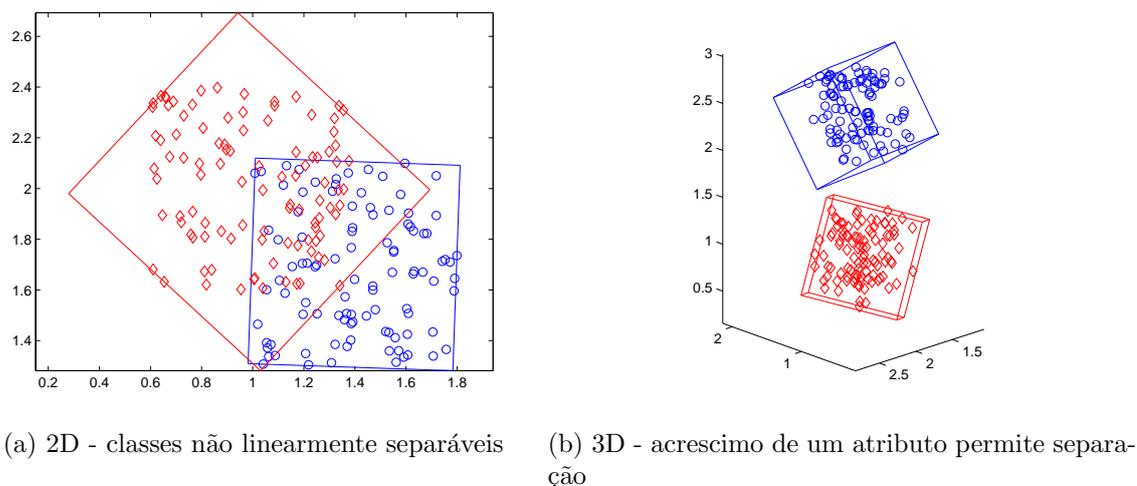


Figura 1: Exemplo de classes não separáveis em duas dimensões

resultado que se pretende alcançar com o sistema produzido.

Uma das metodologias mais rápidas de geração de classificadores é a técnica de separação geométrica através de intervalos (LEUNG et al., 2008). Estes métodos utilizam funções lineares ou não-lineares (FUNG; MANGASARIAN, 2005) para diferenciar as classes. Quando funções lineares são utilizadas o treinamento é rápido, entretanto, dependendo do sistema escolhido, a resposta não é satisfatória devido a existência de dados com características similares, espaço de parâmetros não contendo todos os elementos necessários para uma correta desagregação (Figura 1), classes não-convexas que não podem ser separadas linearmente (Figura 2) e classes não contínuas no espaço de soluções (Figura 3). Por outro lado, funções não-lineares apresentam uma melhor resposta, contudo, assim como metodologias de treinamento de redes neurais (HAN; KAMBER; PEI, 2011), necessitam de muito tempo de treinamento.

1.1 OBJETIVOS

Buscando uma metodologia eficiente e rápida para problemas de classificação, este trabalho apresenta o desenvolvimento de uma técnica para construção topológica e de pesos de uma rede neural utilizando técnicas de segmentações geométricas. O Método de Segmentações Geométricas Sucessivas (MSGs) mostrado neste trabalho apresenta como vantagem o tratamento de multi-classes de dados, sem a necessidade de especificar os parâmetros e a topologia da rede.

O MSGs utiliza o princípio de que um neurônio pode ser representado através de um

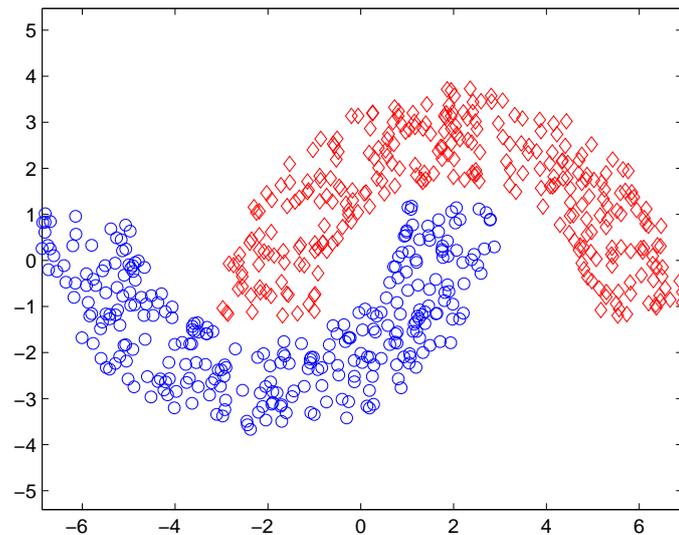


Figura 2: Exemplo de classes não-convexas

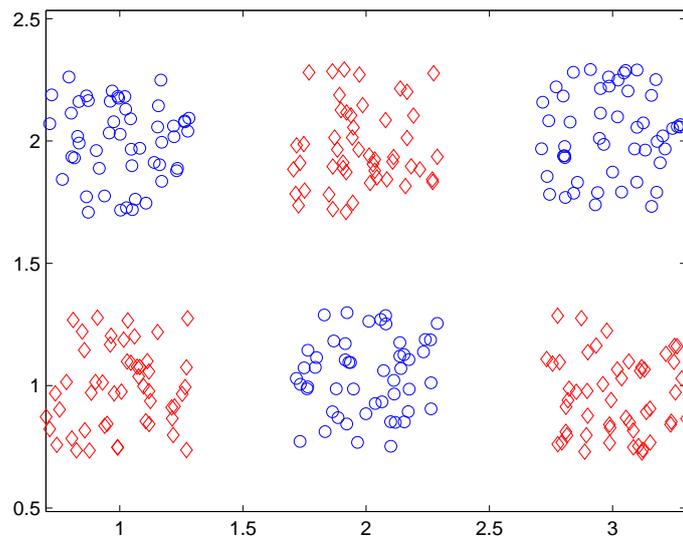


Figura 3: Exemplo de classes não contínuas no espaço de soluções

hiperplano associado a uma função de ativação. Desta forma a busca é direcionada para encontrar uma combinação de hiperplanos que consiga separar as classes analisadas. O método inicia representando cada classe de dados através de um único envelope. Caso os envelopes sejam linearmente separáveis é possível definir pelo menos um hiperplano cujos parâmetros são utilizados para estipular os pesos de um neurônio associado a esta classificação. No caso das classes não serem linearmente separáveis, os dados são divididos em subconjuntos gerando novos envelopes, menores e mais precisos. Este processo ocorre até que todos os envelopes de uma classe estejam separados das demais

classes ou até que um critério de parada seja atingido.

Após o fim deste procedimento, cada hiperplano dá origem a um neurônio da primeira camada oculta da rede. O processamento da informação proveniente da ativação destes neurônios é realizado pelas camadas posteriores que definem a saída da rede, ou seja, a classificação de um dado de entrada. Desta forma, o MSGS encontra o número de neurônios da rede e seus parâmetros, resolvendo um dos grandes problemas no treinamento de redes neurais.

Este trabalho está dividido da seguinte maneira para facilitar na leitura das seções subsequentes, a Seção apresenta toda a nomenclatura utilizada. O Capítulo 2 trata os fundamentos teóricos necessários para melhor entendimento de algumas partes do método. O Capítulo 3 mostra um fluxograma do método onde cada passo é explicado em detalhes. O Capítulo 4 mostra alguns resultados, primeiro com uma função bidimensional cujo desempenho é comparado com a *toolbox* de reconhecimento de padrões do *Matlab*, o segundo é formado por conjuntos de dados genéricos utilizados na literatura com os resultados mostrados em comparação com dois dos melhores e mais citados algoritmos de classificação, e por último é mostrado o comportamento do código ao ser apresentado ao problema da espiral dupla que possui característica altamente não linear. Por fim, as conclusões são mostradas no Capítulo 5.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 REDES NEURAIS

O funcionamento da mente sempre foi, e continua sendo um mistério que instiga grandes pensadores. A primeira abordagem moderna realizada acerca do assunto é atribuída a Descartes, que associou ao cérebro a capacidade de pensar, refletir, tomar decisões. Apesar de ainda existirem dúvidas sobre o funcionamento do cérebro em muitos aspectos, os avanços conseguidos nesta área do conhecimento nos últimos séculos permitiram que houvesse o surgimento de outras linhas de pesquisa, como a neurobiologia, psicologia experimental, e inteligência artificial (IA) (PERETTO, 1992).

O ramo de inteligência artificial diz sobre sistemas capazes de perceber o ambiente e tomar ações tais que maximizem a chance de sucesso. Várias técnicas foram desenvolvidas com esse intuito, e podemos citar algumas como: inteligência de enxames, lógica nebulosa, e rede neurais artificiais.

A inspiração para o surgimento de Redes Neurais Artificiais (RNA) vem do funcionamento do cérebro humano, e foram desenvolvidas com o intuito de generalizar modelos matemáticos e dessa forma, ser capaz de realizar determinadas tarefas como regressão de funções, reconhecimento de padrões, classificação, dentre outras (FAUSETT, 1994).

Entre as principais características almeçadas, está a capacidade do cérebro em aprender com dados apresentados e assim, ser capaz de se adaptar a certas necessidades particulares de determinadas aplicações.

Para isso, uma RNA é desenvolvida utilizando vários elementos que realizam tarefas matemáticas simples e efetuam processamento distribuído e paralelo, denominados “neurônios”. A analogia com o cérebro continua considerando a forma como esses neurônios são interconectados, podendo utilizar diferentes pesos para cada uma das entradas, aos quais é dado o nome de “pesos sinápticos”(HAYKIN, 1999).

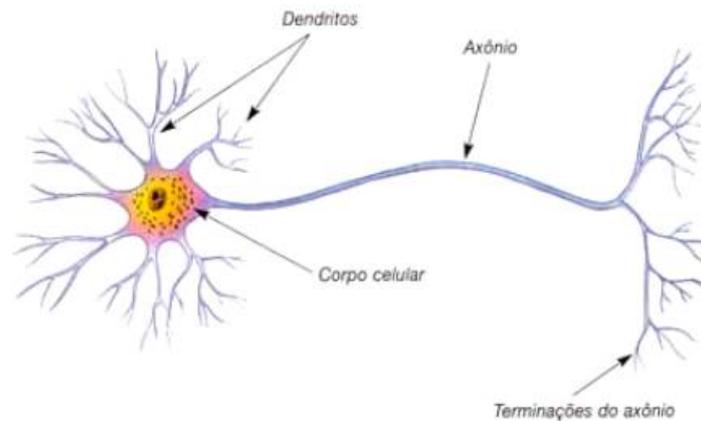


Figura 4: Ilustração de um neurônio, elemento básico de uma rede neural biológica (PACKTER, 2011)

Utilizando dessas características inspiradas pelo comportamento do cérebro humano, uma RNA consegue se assemelhar deste principalmente pelos seguintes aspectos:

- A rede adquire conhecimento do ambiente através de um processo de aprendizado.
- A força das interconexões entre os neurônios são usadas para armazenar o conhecimento adquirido

O processo de aprendizado é realizado através do chamado “algoritmo de aprendizado” e tem como função alterar os pesos sinápticos das conexões entre os neurônios de forma a alcançar um determinado objetivo.

2.1.1 REDES NEURAIIS BIOLÓGICAS

Como citado anteriormente, a inspiração para o desenvolvimento de Redes Neurais Artificiais vem de sistemas neurais biológicos, e conseqüentemente, existem muitas semelhanças entre o neurônio que se vê em estruturas como cérebros e nervos em seres vivos, e o principal elemento de uma RNA, chamado também de neurônio.

Um neurônio biológico é constituído de várias partes, porém, para o entendimento dos neurônios artificiais usados em uma RNA existem três principais que devem ser citadas: *dendritos*, *corpo celular* e *axônios* (Figura 4).

Estima-se que o córtex humano (camada mais externa do cérebro, com aproximadamente 2-4mm de espessura) é formado por cerca de 10 bilhões de neurônios e 60 trilhões de conexões entre estes neurônios (HAYKIN, 1999).

As conexões entre neurônios, que recebem o nome de sinapses, ocorrem entre o

axônio de um neurônio, e o dendrito do neurônio seguinte. São em geral processos eletro-químicos que ocorrem com a liberação de neurotransmissores na junção sináptica entre dois neurônios. A força das sinapses é variável, e é essa característica que juntamente com a criação de novas sinapses permite que o sistema nervoso armazene informações e aprenda.

Os muitos dendritos de um neurônio recebem informações de vários neurônios anteriores, e a ação resultante de cada sinal varia de acordo com a força da respectiva sinapse.

O axônio leva para os próximos neurônios o estado do neurônio, se foi ou não disparado.

O corpo do neurônio é capaz de somar os sinais de entrada e caso seja recebido uma entrada suficiente o neurônio dispara, o que significa levar para os próximos neurônios um sinal através do axônio.

Uma forma de aprendizado pode ser vista como, se uma sinapse é muito utilizada ela se torna mais forte, e caso seja pouco utilizada se torna mais fraca (HEBB, 1949).

2.1.2 MODELO DE UM NEURÔNIO ARTIFICIAL

Uma Rede Neural Artificial é um sistema capaz de processar informações, e que tem seu comportamento baseado em sistema nervoso biológico (Seção 2.1.1), e para isso apresenta uma estrutura na qual o processamento ocorre de forma distribuída ao longo da rede, em elementos básicos chamados *neurônios* (FAUSETT, 1994).

Os neurônios de uma RNA se comunicam com outros neurônios através de conexões, que permitem a um neurônio ter acesso à saída do neurônio anterior.

Cada conexão entre neurônios tem um peso associado, que em uma típica rede neural, multiplica o sinal transmitido.

A Figura 5 mostra o modelo de um neurônio k que forma a base para o projeto de uma Rede Neural Artificial. O modelo mostrado nesta figura conta com uma entrada independente que atribui o valor do *bias*. Uma outra alternativa é mostrada na Figura 6, na qual o valor do *bias* é dado através de um peso sináptico ligado à uma entrada fixa. Onde x_1, x_2, \dots, x_m são os sinais de entrada vindos do neurônio anterior, $w_{k1}, w_{k2}, \dots, w_{km}$ são os pesos sinápticos, v_k é chamado campo local induzido (Equação (2.1)), $\varphi(\cdot)$ é a função de ativação (geralmente não linear) aplicada ao campo local induzido para gerar a saída do neurônio y_k .

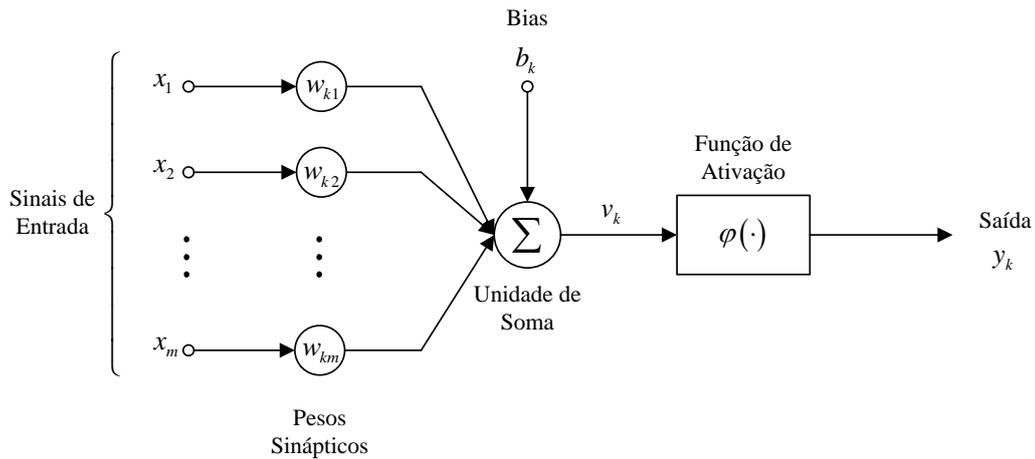


Figura 5: Modelo adotado para um neurônio artificial com *bias*

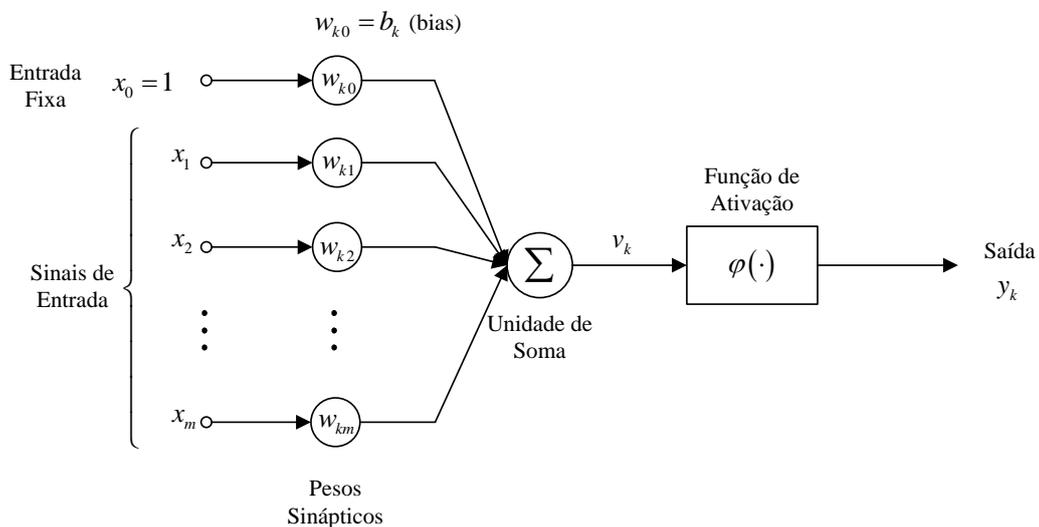


Figura 6: Modelo adotado para um neurônio artificial com *bias* como peso sináptico

O modelo do neurônio mostrado na Figura 6 permite ao *bias* ser tratado como apenas outro peso sináptico.

$$v_k = \sum_{j=1}^m w_{kj} \cdot x_j + b_k \quad (2.1)$$

2.1.3 REDES NEURAIS ARTIFICIAIS

Rede Neural Artificial (RNA) é um conjunto de neurônios artificiais interligados que utilizam informações do meio para se adaptar, buscando a melhor forma de realizar alguma tarefa, sendo essa de reconhecimento de padrões, mapeamento, regressão, classificação, previsão, entre outras.

Uma rede neural é caracterizada pela função de ativação utilizada nos neurônios,

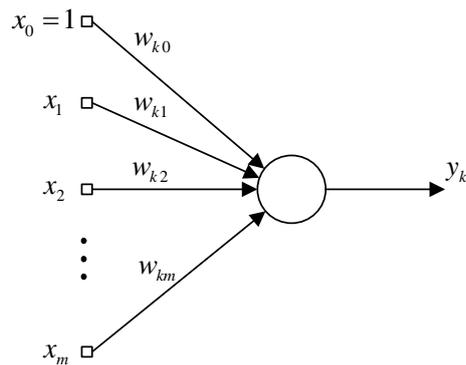


Figura 7: Neurônio representação gráfica para um neurônio

pelo padrão de conexão entre os neurônios e o método utilizado para se obter os pesos entre as conexões.

A função de ativação fornece a saída de um determinado neurônio devido ao campo local induzido pelos sinais de entrada no neurônio e o peso das conexões sinápticas.

O padrão de conexão entre os neurônios, também chamado de arquitetura ou topologia da rede neural é caracterizado pela organização dos neurônios em camadas, e como eles estão ligados entre si e com outras camadas.

Já o método utilizado para obter os pesos diz respeito à forma como ocorre o treinamento de uma rede neural, como ela é capaz de utilizar informações do ambiente para adquirir conhecimento.

A representação gráfica de um neurônio k com m entradas será feita seguindo a Figura 7, que se baseia no modelo apresentado na Figura 6.

2.1.3.1 TIPOS DE FUNÇÃO DE ATIVAÇÃO

A operação básica de um neurônio é somar as entradas multiplicadas pelos pesos sinápticos relativos a cada uma delas, e sujeitar esse valor (chamado de campo local induzido) a uma função de ativação que determina sua real saída.

A função de ativação, definida como $\varphi(v)$, pode assumir diferentes formas, algumas são mostradas a baixo

1 - Função linear A função de ativação linear faz com que o valor do campo local induzido seja transformado para uma saída contínua e sem não-linearidades seguindo a Equação (2.2).

$$\varphi(v) = a \cdot v \quad (2.2)$$

com $a \in \mathbb{R}$.

2 - Função limiar Uma função de ativação com limiar (*threshold function*) atribui valor 1 para entradas acima de um determinado limiar, e 0 caso esteja abaixo. Considerando que o neurônio recebe uma entrada referente ao *bias*, o valor de limiar é normalmente considerado 0 (Equação (2.3)).

$$\varphi(v) = \begin{cases} 1 & \text{se } v \geq 0 \\ 0 & \text{se } v < 0 \end{cases} \quad (2.3)$$

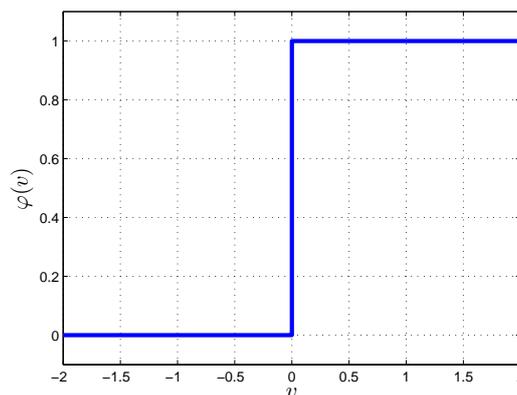


Figura 8: função de ativação com limiar

Desta forma, para um neurônio k com função de ativação com limiar, a saída y_k devido a um campo local induzido v_k é dado pela Equação (2.4)

$$y_k = \begin{cases} 1 & \text{se } v_k \geq 0 \\ 0 & \text{se } v_k < 0 \end{cases} \quad (2.4)$$

A forma da função de ativação com limiar pode ser vista na Figura 8, que mostra uma das características principais dessa função: a presença de uma descontinuidade para $v = 0$.

3 - Linear por partes A equação que define a função de ativação linear por partes pode ser vista na Figura 2.5.

$$\varphi(v) = \begin{cases} 1 & \text{se } v \geq a \\ \frac{v}{2a} + \frac{1}{2} & \text{se } -a < v < a \\ 0 & \text{se } v \leq -a \end{cases} \quad (2.5)$$

A Figura 9 mostra a função de ativação linear por partes com coeficientes $a = \{0,5; 1; 1,8\}$, e é possível perceber que, a medida que a aumenta, a função tende a uma função linear sem saturação. E a medida que a diminui a função tende a função limiar.

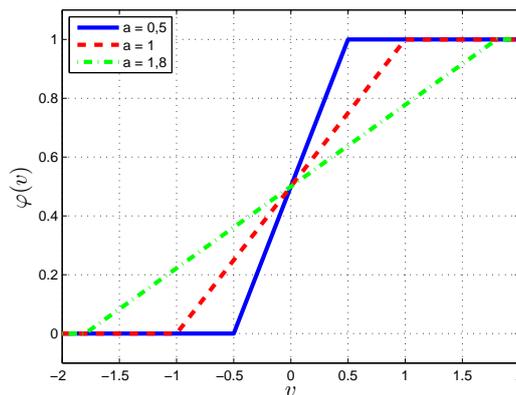


Figura 9: função de ativação linear por partes com coeficientes $a = 0,5$, $a = 1$, $a = 1,8$

A Figura 9 mostra que similarmente à função limiar, a função linear por partes apresenta descontinuidades em $v = a$ e $v = -a$.

4 - Sigmóide Constantemente para a o papel de $\varphi(v)$ são usadas funções sigmóides, que são funções em formato de S , e um exemplo desta família de funções é a função logística que segue a Equação (2.6).

$$\varphi(v) = \frac{1}{1 + e^{-av}} \quad (2.6)$$

Sendo a um parâmetro capaz de controlar a inclinação da curva.

A Função logística apresenta comportamento similar às duas funções apresentadas anteriormente pelo ponto de vista de apresentar uma saturação para valores que distanciam de $v = 0$ (Como pode ser visto na Figura 10). Porém a maior diferença está no fato de apresentar comportamento contínuo, fato que se mostra importante em muitas arquiteturas de redes neurais multicamadas como será mostrado na Seção 2.1.3.2.

A Figura 10 mostra o comportamento da função logística para diferentes coeficientes de inclinação a , e é possível perceber que quanto maior o valor de a , menos a

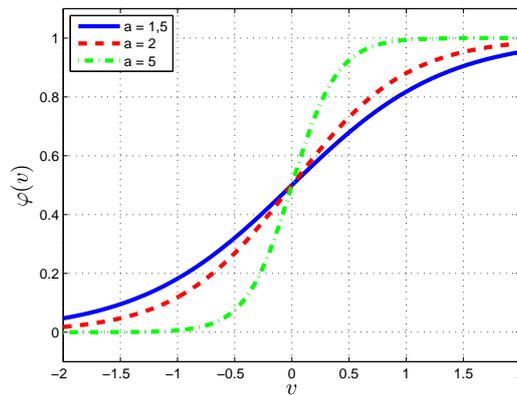


Figura 10: função de ativação sigmóide com coeficientes $a = 1,5$, $a = 2$, $a = 5$

curva apresenta a saturação. Enquanto a medida que a diminui, a curva se aproxima da função limiar.

Funções bipolares As funções apresentadas nos parágrafos anteriores têm comportamento variando entre valores no intervalo $[0,1]$.

Em algumas situações é preferível um comportamento bipolar de forma que a função de ativação apresente valores variando de $[-1,1]$, apresentando um comportamento anti-simétrico em relação à origem, para valores de campo local induzido.

Desta forma, a função limiar pode ser representada conforme a Equação (2.7), a parcialmente linear conforme a Equação (2.8), e a sigmóide utilizada passa a ser a tangente hiperbólica (Equação (2.9))

$$\varphi(v) = \begin{cases} 1 & \text{se } v > 0 \\ 0 & \text{se } v = 0 \\ -1 & \text{se } v < 0 \end{cases} \quad (2.7)$$

$$\varphi(v) = \begin{cases} 1 & \text{se } v \geq a \\ \frac{v}{a} & \text{se } -a < v < a \\ -1 & \text{se } v \leq -a \end{cases} \quad (2.8)$$

$$\varphi(v) = \frac{e^v - e^{-v}}{e^v + e^{-v}} = \tanh(v) \quad (2.9)$$

A Figura 11 mostra as funções bipolares apresentadas que são utilizadas como função de ativação para neurônios em redes neurais artificiais.

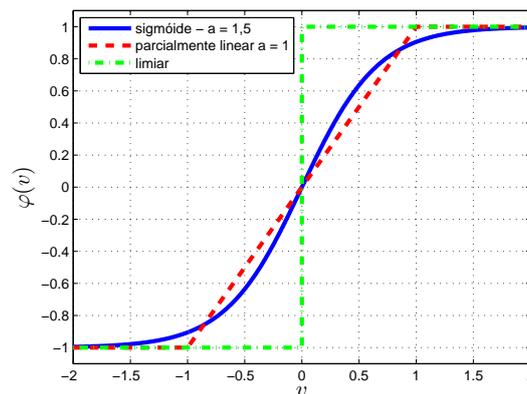


Figura 11: Curvas bipolares utilizadas como função de ativação

2.1.3.2 ARQUITETURAS DE REDES NEURAIAS

Dentre as principais características qualitativas observáveis sobre a arquitetura de uma RNA pode-se destacar para a aplicação deste trabalho: dimensão do sinal de entrada, topologia de conexões entre neurônios (direta, cruzada, inversa, etc.), número de camadas, número de neurônios em cada camada (GALUSHKIN, 2007).

Uma camada em uma rede neural apresenta neurônios com um comportamento igual, considerando sua função de ativação e padrão das conexões entre outros neurônios.

Em geral, a primeira camada de uma rede neural, é representada por unidades que retornam para as camadas ocultas ou de saída os valores dos sinais de entrada.

Uma camada de saída de uma RNA é aquela que interage com o meio externo fornecendo a saída devido ao funcionamento da rede neural como um todo.

Como a camada de entrada de uma RNA não desempenha nenhuma computação, não é considerada para determinar o número de camadas de uma rede neural, e desta forma, uma RNA de camada única é aquela com uma camada de entrada e uma de saída.

Redes neurais nas quais o fluxo de dados da entrada para saída segue apenas um sentido, são ditas *feedforward*, ou seja, apresentam apenas conexões diretas entre neurônios (HAYKIN, 1999).

REDES NEURAIAS EM CAMADA ÚNICA Uma RNA em camada única (*Single-Layer*) apresenta uma camada de entrada que reflete os sinais vindos do am-

biente, e uma camada de saída que fornece a resposta da rede neural aos sinais de entrada.

O exemplo mais simples de uma rede neural em camada única é um neurônio, que seria uma rede neural em camada única com apenas um neurônio na camada de saída. A quantidade de neurônios de saída deve ser a necessária para cada tipo de problema.

A Figura 12 mostra um exemplo de uma rede neural em camada única com dois neurônios de saída.

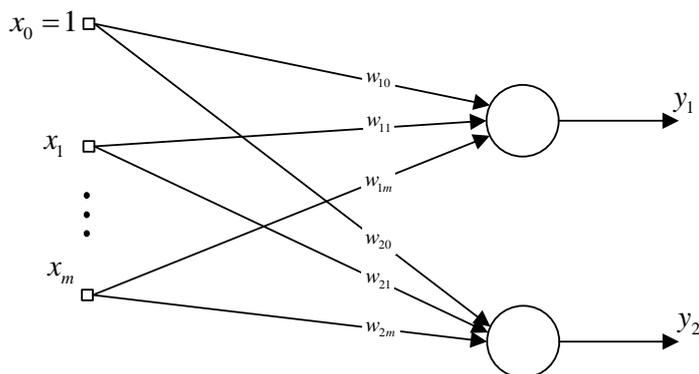


Figura 12: exemplo de rede neural em camada única

REDES NEURAIAS MULTICAMADAS Uma RNA multicamada (*Multi-layer*) se distingue de uma em camada única pela presença de pelo menos uma camada oculta.

Camada oculta é aquela que não tem contato com o exterior, ou seja, não recebe sinais do meio, nem fornece saídas. Ela apenas recebe e fornece sinais para outras camadas da rede neural.

A função dos neurônios ocultos (neurônios de uma camada oculta) é de intervir entre a camada de entrada e a de saída de uma maneira útil. A inserção de camadas ocultas possibilita à rede extrair estatísticas de maior ordem de um determinado processo, o que permite que uma rede multicamada resolva problemas mais complexos do que uma rede de camada única.

Em contrapartida, o treinamento de uma rede neural multicamada é mais complexo do que o treinamento para uma rede de camada única, devido ao número de pesos sinápticos. Além disso, alterar o valor de um peso sináptico, interfere na saída dos neurônios seguintes.

Em uma rede multicamadas do tipo *feedforward*, os neurônios presentes na primeira camada (camada de entrada) recebem o sinal de entrada para a rede. A saída desta ca-

mada serve de entrada para a segunda camada de neurônios (primeira camada oculta), e de forma similar, a saída da segunda camada segue para a terceira. O processo se mantém até a camada de saída, cuja saída representa a resposta global da rede para o padrão de entrada.

Uma rede neural multicamada na qual todos os neurônios de uma camada estão conectados à todos os neurônios da camada seguinte é dita rede completamente conectada, em oposição a uma rede parcialmente conectada (HAYKIN, 1999).

A Figura 13 mostra o exemplo de uma rede neural multicamada, *feedforward*, completamente conectada, com cinco neurônios de entrada, 3 neurônios na camada oculta, e 2 neurônios de saída.

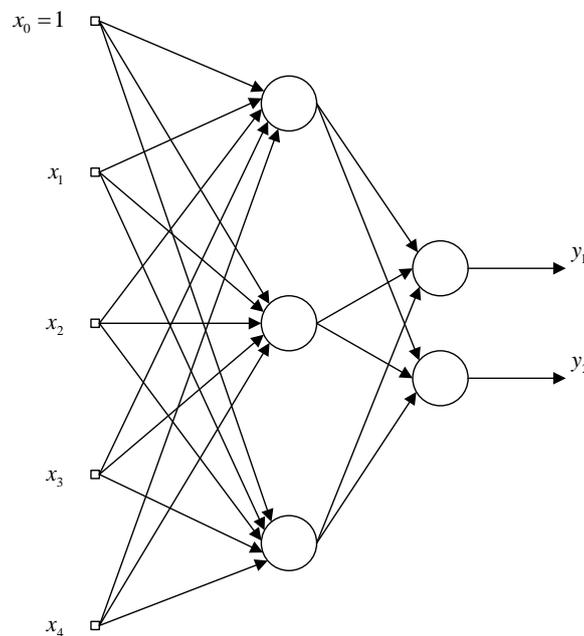


Figura 13: Exemplo de rede neural multicamada 5-3-2

REDE NEURAL RECORRENTE Rede neural recorrente é aquela que apresenta pelo menos um laço de realimentação, que faz com que a saída de um determinado neurônio, de uma camada, de uma rede, dependa da saída de um neurônio no instante anterior.

Um exemplo de uma rede neural recorrente pode ser vista na Figura 14, que mostra a estrutura básica de uma camada competitiva do tipo *winner-take-all*, conhecida como MAXNET, que faz com que apenas o maior sinal de excitação apareça como saída (FAUSETT, 1994).

A camada mostrada na Figura 14 pode ser usada como uma sub-rede em uma RNA

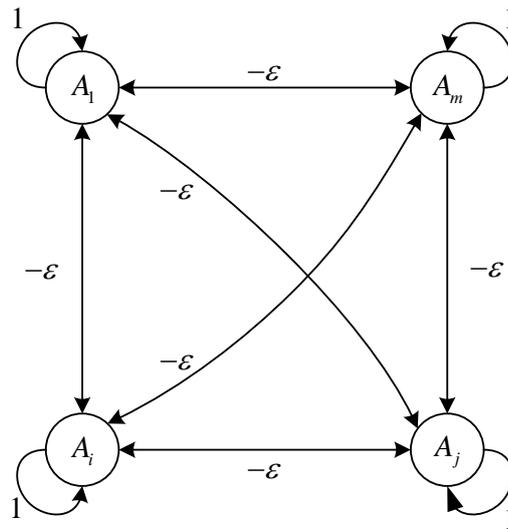


Figura 14: camada competitiva

capaz de selecionar o neurônio com maior entrada. Os m neurônios na sub-rede são completamente interconectados com pesos simétricos $-\varepsilon$.

2.1.3.3 APRENDIZADO

Uma outra característica que diferencia uma RNA de outra é a forma como ela é treinada. A habilidade de aprender através de informações do meio e melhorar o seu funcionamento, é de grande importância para uma rede neural.

Aprendizagem é o processo pelo qual os parâmetros livres de uma rede neural são adaptados através de estímulos do meio no qual a rede se encontra (HAYKIN, 1999).

Assim, o processo de aprendizado pode ser resumido como o seguintes passos:

1. A rede recebe um estímulo vindo do ambiente.
2. A rede sofre mudanças nos seus parâmetros livres como resultado do estímulo.
3. A rede é capaz de responder de uma forma diferente ao meio devido às mudanças sofridas na estrutura interna.

Em geral, são citadas principalmente duas classes de aprendizado: supervisionado, e não supervisionado.

APRENDIZADO SUPERVISIONADO O aprendizado supervisionado, também chamado de aprendizado com um professor, considera um elemento (*tutor*) que

tem certo conhecimento sobre o ambiente, representado como um conjunto com exemplos de entrada-saída. Por outro lado, a rede neural não tem esse conhecimento.

Sendo assim, pode-se utilizar um vetor do conjunto de exemplos entrada-saída como vetor para treinamento.

Desta forma, quando o vetor é apresentado para a rede neural, o *tutor* é capaz de fornecer a resposta desejada para este exemplo. Com isso, é possível calcular um sinal de erro, entre a resposta desejada e a resposta da rede neural para o vetor de treinamento (Figura 15). Uma vez que a resposta desejada fornece aquela preferível para a rede neural, o sinal de erro possibilita um ajuste dos parâmetros livres da rede.

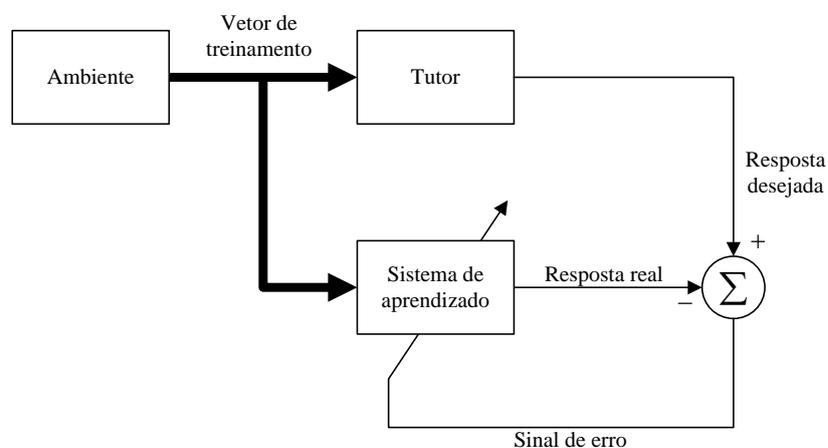


Figura 15: Diagrama do processo de aprendizado supervisionado

Este ajuste ocorre de forma iterativa buscando fazer com que a rede neural seja capaz de aproximar do comportamento do *tutor* sobre algum critério estatístico.

Desta maneira, o conhecimento disponível sobre o meio é passado do *tutor* para a rede neural, o que capacita a rede a partir deste ponto a lidar com o ambiente sem necessidade do *tutor*.

APRENDIZADO SEM SUPERVISÃO No aprendizado sem supervisão não existe a figura de um *tutor* orientando o aprendizado, que implica em não haver exemplos rotulados para treinamento da rede.

Para promover o aprendizado é fornecida uma sequência de vetores de entrada, porém sem especificação de valores desejados. Desta forma, os pesos se modificam buscando associar entradas similares a uma mesma saída.

Para o funcionamento de aprendizado sem supervisão em geral se torna necessário o uso de algoritmos que consideram competição, como exemplo a camada mostrada na

Figura 14.

A Figura 16 apresenta o diagrama de funcionamento do aprendizado não-supervisionado, destacando a presença do meio fornecendo padrões de entrada, porém não existe nenhuma associação deste com uma saída desejada.

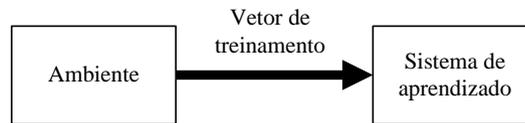


Figura 16: Diagrama do processo de aprendizado sem supervisão

2.1.4 INTERPRETAÇÃO GEOMÉTRICA DE UM NEURÔNIO

Tanto um neurônio quanto uma rede neural artificial podem ser interpretados de forma geométrica. Para uma rede neural, o número de camadas bem como o número de neurônios em cada camada permitem a adaptação a diferentes tipos de problemas (XIANG; DING; LEE, 2005).

Considerando o modelo de neurônio mostrado na Figura 6, para qual o campo local induzido é dado pela Equação 2.1 como $v_k = \sum_{j=0}^m w_{kj} \cdot x_j$, é possível perceber que, para funções de ativação bipolares (Subseção 2.1.3.1) $v_k = 0$ define um hiperplano em \mathbb{R}^m .

Para elucidar esse fato, em um caso particular de um problema bidimensional, o campo local induzido v de um neurônio que recebe x_1 e x_2 como entrada pode ser visto como:

$$v = w_0 + w_1x_1 + w_2x_2$$

e desta forma, $v = 0$ define uma reta em \mathbb{R}^2 .

Um hiperplano gerado por $v = 0$ é capaz de separar um espaço em dois subespaços, com valores de $v > 0$ e $v < 0$, como pode ser visto em um exemplo na Figura 17.

O exemplo na Figura 17 ilustra uma nuvem de pontos e uma reta separando-a em duas partes. Neste exemplo, a reta na figura pode ser gerada através de um neurônio que recebe como entrada x_1 (abscissas) e x_2 (ordenadas), e possui pesos sinápticos $w_0 = -0,5$, $w_1 = 0,6$ e $w_2 = 0,3$.

Assim, os pontos mostrados como "O" são alguns daqueles que induziriam uma saída positiva no neurônio, caso fossem submetidos como entrada. Enquanto os mostrados como "X", induziriam saída negativa.

Como um neurônio forma uma reta em \mathbb{R}^2 , uma rede neural em camada única

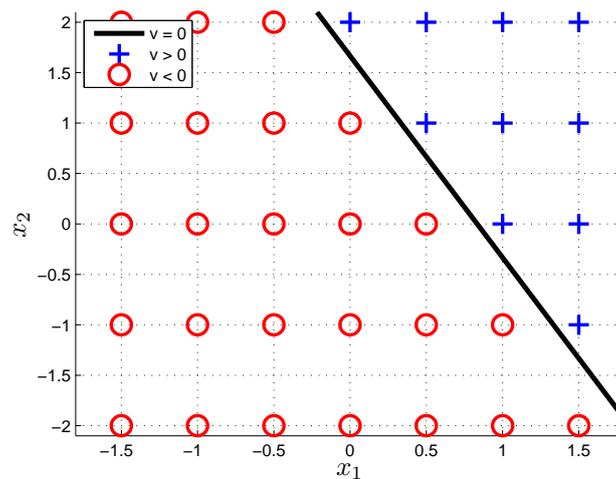


Figura 17: Separação linear devido a um neurônio ($w_0 = -0,5$, $w_1 = 0,6$ e $w_2 = 0,3$)

só é capaz de separar conjuntos que sejam linearmente separáveis. Existem porém, casos como os mostrados no Capítulo 1 que não são linearmente separáveis, e por esse motivo, problemas de classificação incluindo conjuntos com essas características precisam de redes multicamadas para serem solucionados.

2.1.4.1 COMBINAÇÕES DE NEURÔNIOS

A fim de solucionar as limitações em resolver problemas de classificação entre conjuntos não separáveis linearmente impostas por redes neurais em camadas únicas, pode-se utilizar redes multicamadas.

A Figura 18 mostra uma rede neural que possibilita a separação de conjuntos não separáveis linearmente, uma vez que apresenta uma camada oculta com 2 neurônios e uma camada de saída com 1 neurônio.

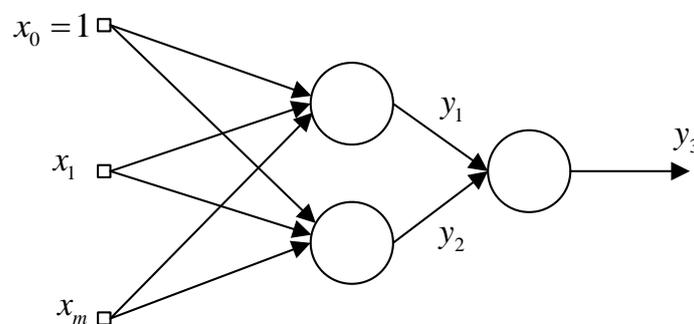


Figura 18: Rede Neural multicamadas com 2 neurônios na camada oculta

Como um neurônio é capaz de separar o espaço amostral em dois subespaços, cada um dos neurônios da camada oculta da rede neural exposta na Figura 18 pode gerar

um hiperplano diferente. Desta forma, o neurônio da camada de saída apenas age como uma função AND fazendo uma combinação entre as duas saídas dos neurônios da camada oculta y_1 e y_2 .

Assim, é possível fazer com que para um ponto no espaço que gere $y_1 < 0$ E $y_2 > 0$ a saída y_3 seja positiva, enquanto para outras combinações seja negativa.

Considerando então a saída y_1 de um neurônio da camada oculta representada pela Figura 17, e a saída y_2 como a Figura 19, um neurônio na camada de saída com $w_0 = -1$, $w_1 = -1$ e $w_2 = 1$, teria a saída y_3 como a mostrada na Figura 20.

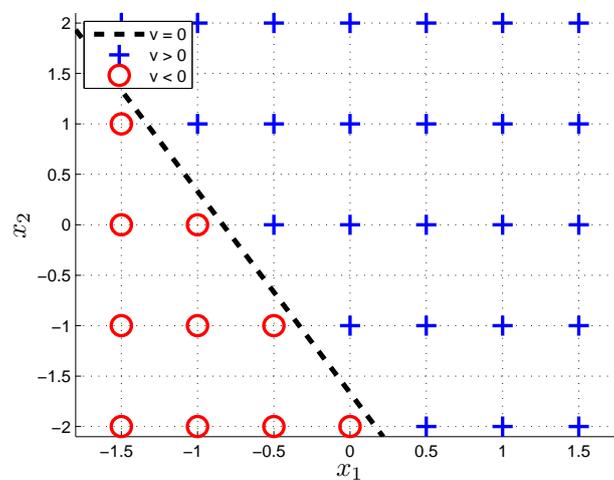


Figura 19: Separação linear devido a um neurônio ($w_0 = 0,5$, $w_1 = 0,6$ e $w_2 = 0,3$)

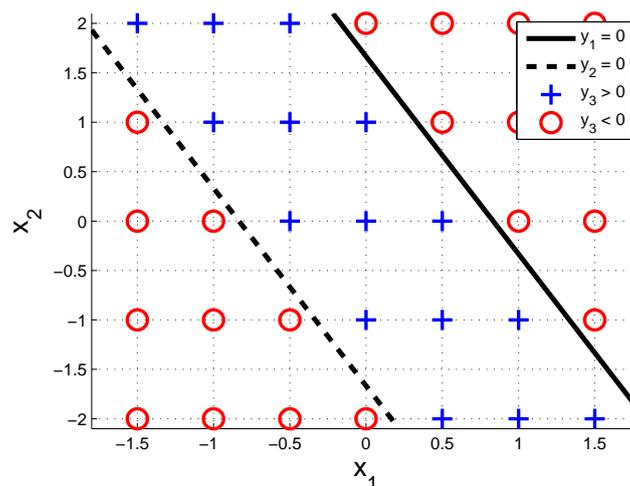


Figura 20: Saída de uma rede neural com duas camadas (pesos sinápticos do neurônio da camada de saída: $w_0 = -1$, $w_1 = -1$ e $w_2 = 1$)

Desta forma, nota-se que uma rede neural com pelo menos uma camada oculta é capaz de fazer separação de conjuntos não separáveis linearmente.

2.1.4.2 REDES COM TRÊS CAMADAS

Estendendo o conceito, é possível demonstrar que uma rede neural com 3 camadas pode separar regiões fechadas no espaço de soluções (MACLEOD, 1999).

A RNA mostrada na Figura 21 possui duas camadas ocultas, e assim é capaz de classificar regiões separadas no espaço de solução conforme mostrado na Figura 22.

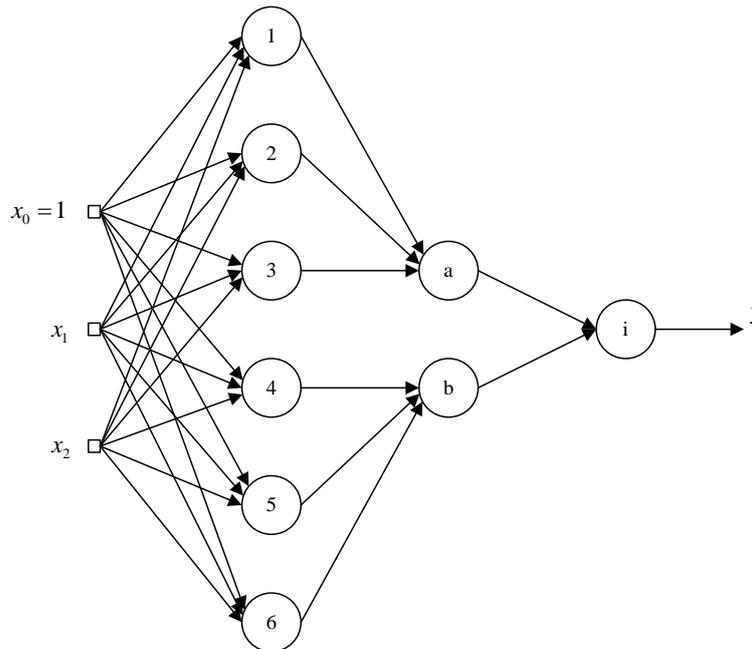


Figura 21: Rede neural artificial com 3 camadas 3-6-2-1

Considerando o que foi dito na Subseção 2.1.4.1 os neurônios da primeira camada oculta geram retas separando o espaço de soluções (1 – 2 – 3 – 4 – 5 – 6), a segunda camada oculta é capaz de utilizar essas retas para criar regiões fechadas ($a - b$), e cabe à camada de saída analisar se um determinado vetor de entrada pertence a alguma das regiões consideradas como desta classe. Ou seja, seguindo o princípio de funcionamento de uma função OU é capaz de julgar se um vetor está na região a ou na região b que fazem parte de uma certa classe.

2.2 ESTIMADOR DE DENSIDADE

A estimação de densidade de probabilidade se faz necessária em várias aplicações atuais, uma vez que em muitas áreas diferentes tem-se acesso a informações no formato de conjunto de dados numéricos que representam alguma variável aleatória de um dado processo (ARCHAMBEAU et al., 2006).

Considerando que uma variável aleatória é completamente caracterizada pela sua

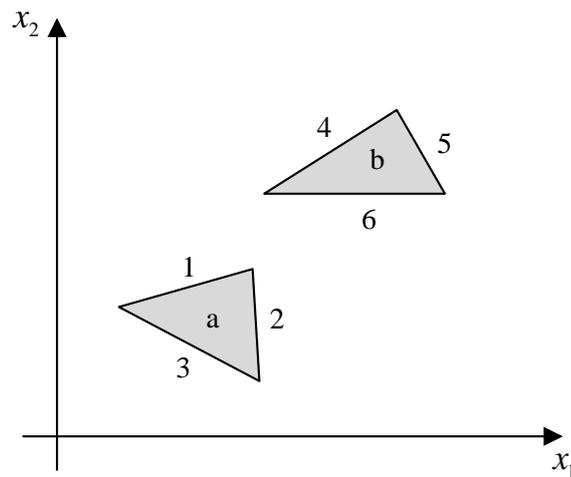


Figura 22: Representação da saída de uma rede neural 3 camadas

função densidade de probabilidade (fdp), que fornece a probabilidade de um evento qualquer x ocorrer, ser capaz de estimar essa função tendo acesso apenas a valores amostrais de realizações dessa variável aleatória é de grande importância em muitas aplicações.

Estimação de densidade de probabilidade pode ser descrita como modelar uma função densidade de probabilidade $p(x)$ de uma variável aleatória x tendo um conjunto finito x_1, \dots, x_N de observações (BISHOP, 2006). Porém, por utilizar um número finito de observações, este acaba sendo um problema que aceita infinitas soluções, já que qualquer distribuição $p(x)$ que tem valores diferentes de zero para cada um dos pontos amostrais x_1, \dots, x_N é um candidato em potencial.

Para estimação de densidade três alternativas podem ser consideradas:

A primeira delas é a estimação de densidade paramétrica, que assume que os dados seguem uma distribuição conhecida, e o ajuste a um modelo pré-determinado é feito através da otimização de alguns poucos parâmetros, como por exemplo média e variância para o caso de uma distribuição gaussiana (Figura 23, Equação (2.10)). Pode ser apontada com uma desvantagem desse método, o fato de que a função escolhida pode não ser capaz de representar a real distribuição da variável.

$$p(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)} \quad (2.10)$$

Uma outra forma é a não-paramétrica, que não assume função $p(x)$ conhecida, estima-se a forma da distribuição considerando as observações da variável aleatória x .

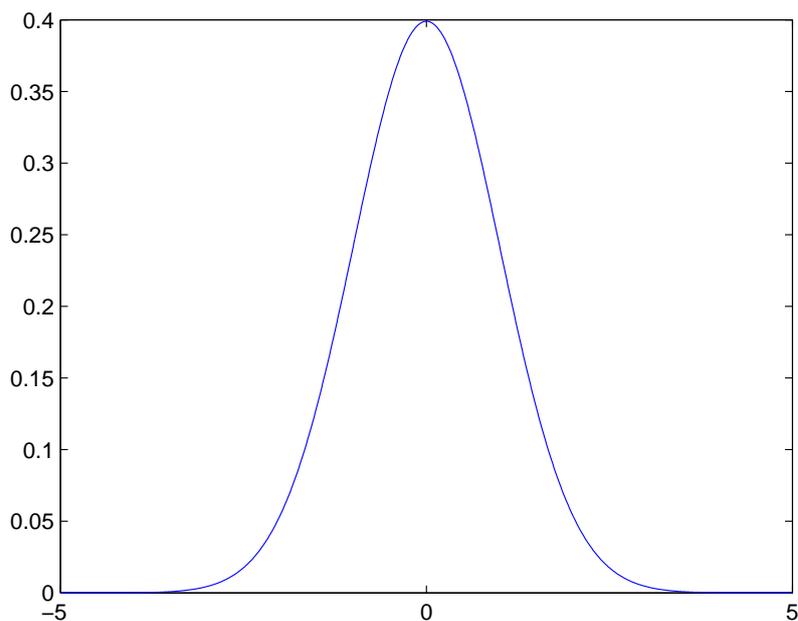


Figura 23: gaussiana com média $\mu = 0$ e desvio padrão $\sigma = 1$

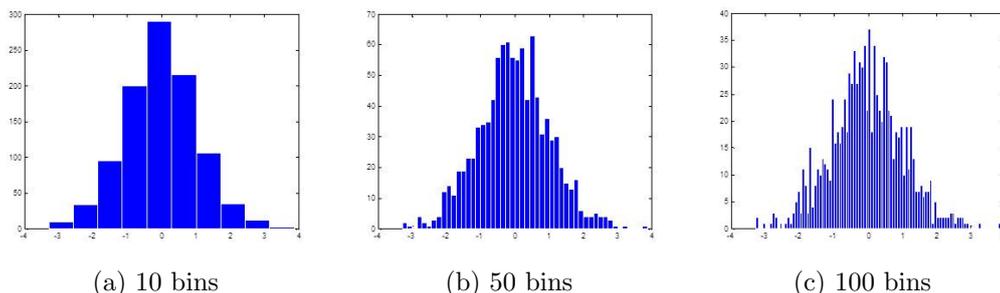


Figura 24: Exemplo de histogramas de um mesmo conjunto de dados com número de bins variável

Um exemplo da estimação não-paramétrica é o histograma, no qual o espaço amostral é dividido em *bins* com uma largura pré-definida, e a quantidade de eventos que ocorrem na região limitada por um dado *bin* fornece o valor que ele recebe (Figura 24). Uma desvantagem do método é o aumento do número de parâmetros a medida que aumenta o tamanho do conjunto de dados, que se torna pesado rapidamente.

A última alternativa a se considerar é a semi-paramétrica, que busca unir vantagens das duas formas citadas anteriormente. Bem como a técnica não-paramétrica, não se assume conhecimento *a priori* da forma da distribuição, mas por outro lado, a complexidade do modelo é fixa a fim de impedir que cresça indefinidamente a medida que se aumenta o tamanho do conjunto de dados. Com esse propósito pode-se citar o método de mistura de gaussiana (Figura 25), que aproxima a função densidade de

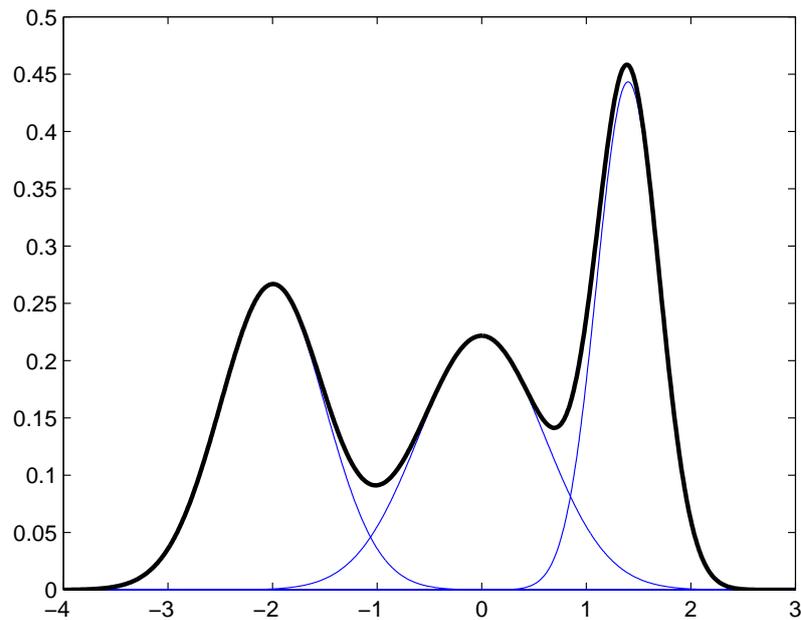


Figura 25: Mistura de Gaussiana

probabilidade por um número K de componentes de densidade (Equação 2.11), sendo $K \ll N$.

$$\hat{p}(x) = \sum_{k=1}^K P(k)p(x|k) \quad (2.11)$$

2.2.1 KERNEL DENSITY ESTIMATOR

O *Kernel Density Estimator* é um método não-paramétrico de estimação da função densidade de probabilidade. Logo, tem como uma das principais características o fato de a função não pressupor uma forma inicial, permitindo assim que a fdp seja inteiramente determinada pelos pontos do conjunto de dados.

A obtenção do estimador é realizada utilizando uma função *kernel*, de forma que cada observação da variável aleatória acrescenta uma função ao estimador (Figura 26). Como será detalhado na Subseção 2.2.1.1.

O fato de poder alterar a função *kernel*, usando várias formas diferentes, permite que a função densidade de probabilidade adquira inclusive formas suaves (sem descontinuidades), diferente das estimadas utilizando histogramas.

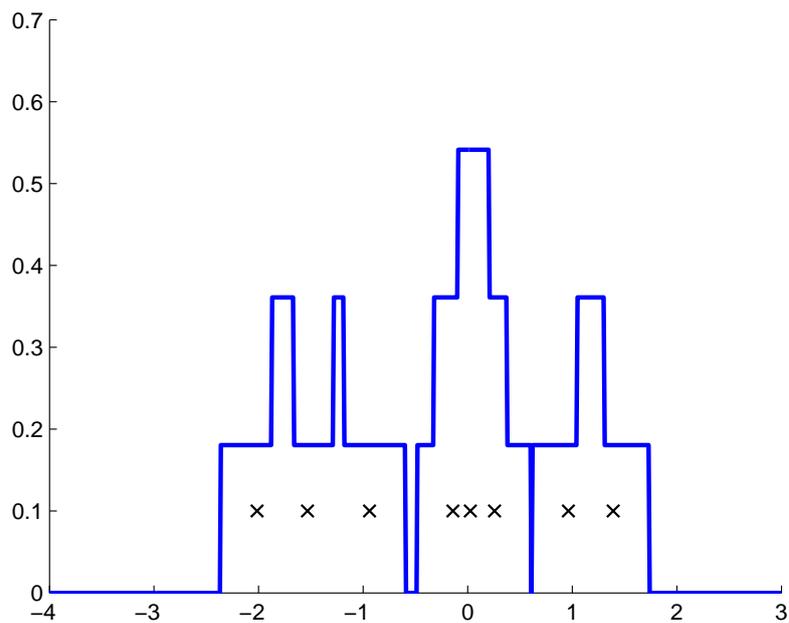


Figura 26: Estimador de densidade e pontos centrais de cada kernel

2.2.1.1 FORMULAÇÃO

Supondo um conjunto de observações de uma variável aleatória X , tal que a função densidade de probabilidade $p(x)$ seja desconhecida em um espaço D -dimensional. Deseja-se estimar $p(x)$ tendo acesso apenas aos valores das realizações de X .

Considerando uma pequena região R que esteja contida no espaço amostral x (Figura 27), pode-se calcular a probabilidade de um evento ocorrer nessa região de acordo com a Equação (2.12).

$$P = \int_R p(x) dx \quad (2.12)$$

Sendo então $\vec{X} = \{x_1, \dots, x_N\}$ um conjunto de N observações da variável aleatória X , como cada evento tem uma probabilidade P de estar na região R , o número total K de pontos que pertence à esta região segue uma distribuição binomial (Equação 2.13).

$$\text{Bin}(K|N,P) = \frac{N!}{K!(N-K)!} P^K (1-P)^{1-K} \quad (2.13)$$

Para uma distribuição binomial, tem-se que o valor esperado é dado pela Equação (2.14) e a variância é dada pela Equação (2.15). E desta forma, analisando a fração K/N de eventos que se encontram na região R , é possível encontrar que o valor esperado

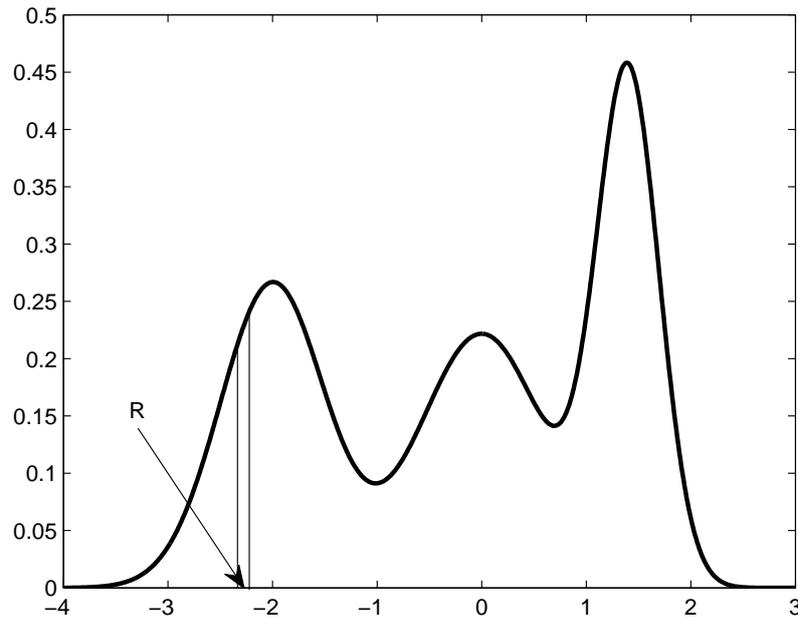


Figura 27: Região R utilizada para formulação da estimação de densidade

para essa fração é dado por $E\left[\frac{K}{N}\right] = P$ (BISHOP, 2005). E de forma análoga, a variância de m é dada por $\text{var}\left[\frac{K}{N}\right] = \frac{P(1-P)}{N}$.

$$E[m] = \sum_{m=0}^N m \cdot \text{Bin}(m|N, \mu) = N\mu \quad (2.14)$$

$$\text{var}[m] = \sum_{m=0}^N (m - E[m])^2 \cdot \text{Bin}(m|N, \mu) = N\mu(1 - \mu) \quad (2.15)$$

Assim, como a variância é inversamente proporcional à N , para um grande número de observações de X , a distribuição de K forma um pico em torno de:

$$K \simeq NP$$

Assumindo ainda que a região R é pequena o suficiente para que se possa afirmar que a densidade $p(x)$ é contínua na região, tem-se que:

$$P \simeq p(x)V$$

Sendo V o volume de R . E desta forma, combinando-se essas equações, o valor de $p(x)$ pode ser dado por:

$$p(x) = \frac{K}{NV} \quad (2.16)$$

O resultado encontrado na Equação (2.16) pode ser explorado de duas maneiras diferentes. Ao fixar o valor de K e determinar V pelas amostras, tem-se o método *K-Nearest-Neighbour*. Por outro lado, ao fixar o valor de V e determinar K pelas amostras, o método é *Kernel Density Estimator*, que foi o método mais aprofundado neste trabalho (Subseção 2.2.1.2).

2.2.1.2 KERNEL

Como foi mostrado na Subseção 2.2.1.1, o método *Kernel Density Estimator* ocorre ao se fixar um valor V para o volume da região R , e determinar K número de pontos dentro da região R pelos dados (Equação (2.16)).

A fim de calcular quantos são os eventos pertencentes à região R considera-se esta região como um hipercubo D -dimensional centrado no ponto x . Para isso se define a função:

$$k(u) = \begin{cases} 1, & |u_i| \leq 1/2 \quad i = 1, \dots, D \\ 0, & c.c \end{cases} \quad (2.17)$$

E desta forma, $k((x - x_n)/h)$ que é um hipercubo de largura h centrado em x_n , terá valor 1 quando x_n pertencer à região R e 0 quando não pertencer.

O número total de eventos na região R pode ser encontrado através de:

$$K = \sum_{n=1}^N k\left(\frac{x - x_n}{h}\right)$$

e conseqüentemente, utilizando a Equação (2.16)

$$p(x) = \frac{1}{N} \sum_{n=1}^N \frac{1}{h^D} k\left(\frac{x - x_n}{h}\right) \quad (2.18)$$

sendo h^D o volume de um cubo D -dimensional de lado h .

2.2.1.3 SMOOTHING KERNELS

Para resolver o problema de descontinuidade nas bordas de cada *kernel* (Figura 26) pode-se escolher outras funções como *kernel*. Em geral, uma forma para esta função

Tabela 1: Diferentes funções para kernel

| Kernel | $K(x)$ |
|--------------|---|
| Uniforme | $1/2 \cdot k(x/2)$ |
| Triangular | $(1 - x) \cdot k(x/2)$ |
| Epanechnikov | $\frac{3}{4} (1 - x^2) k(x/2)$ |
| Quartic | $\frac{15}{16} (1 - x^2)^2 k(x/2)$ |
| Triweight | $\frac{35}{32} (1 - x^2)^3 k(x/2)$ |
| Gaussiana | $\frac{1}{\sqrt{2\pi}} e^{(-\frac{1}{2}x^2)}$ |
| Cosseno | $\frac{\pi}{4} \cos\left(\frac{\pi}{2}x\right) \cdot k\left(\frac{x}{2}\right)$ |

que é bastante usada, é a gaussiana (MURPHY, 2012) (Equação 2.19).

$$p(x) = \frac{1}{N} \sum_{n=1}^N \frac{1}{\sqrt{2\pi h^2}} e^{\left\{-\frac{\|x-x_n\|^2}{2h^2}\right\}} \quad (2.19)$$

Sendo h o desvio padrão da gaussiana que serve como parâmetro para suavizar a curva (Figura 28).

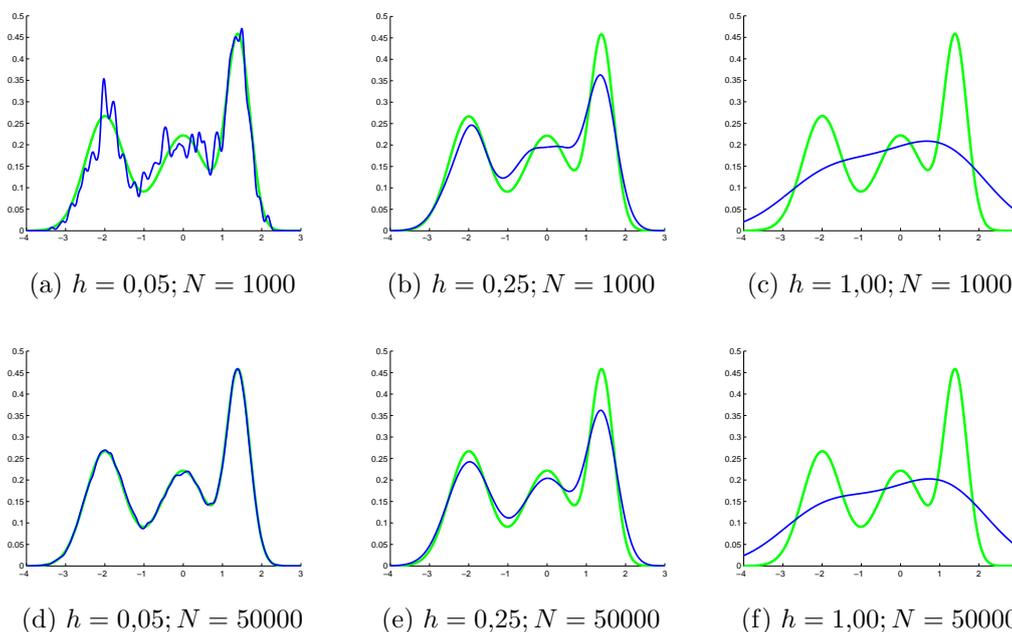
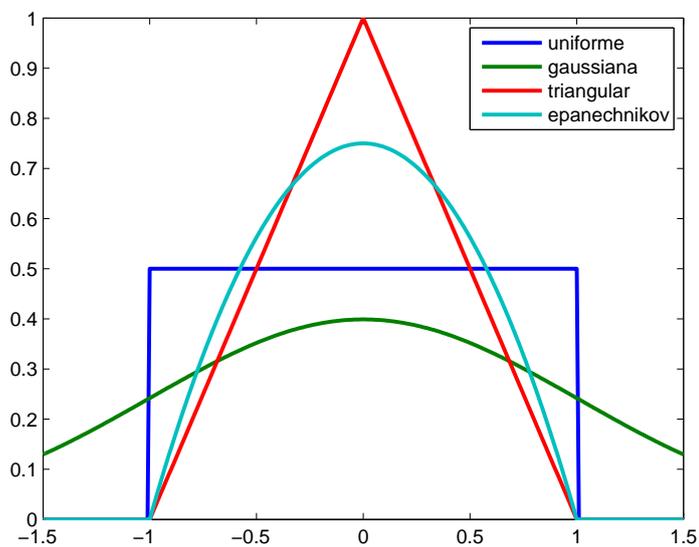


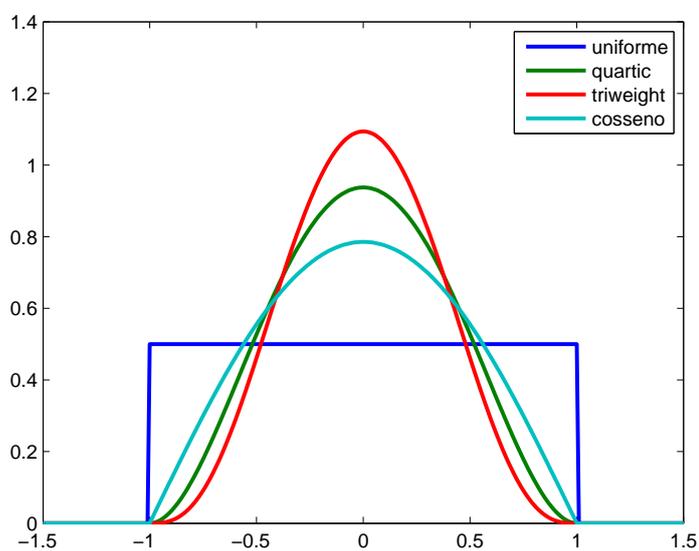
Figura 28: Densidade estimada utilizando diferentes larguras do *kernel*, e tamanhos diferentes do conjunto de eventos

Existem ainda outras funções frequentemente usadas como *kernel* que são citadas na Tabela 1 e mostradas na Figura 29 (HÄRDLE et al., 2005).

A função *Kernel* pode ser qualquer função que possua propriedades de funções densidade de probabilidade, ou seja:



(a) Kernels: Uniforme, Gaussiana, Triangular e Epanechnikov



(b) Kernels: Uniforme, Quartic, Triweight e Cosseno

Figura 29: Visualização de diferentes funções kernel

$$k(x) \geq 0$$

$$\int_{-\infty}^{\infty} k(x)dx = 1$$

E assim, para qualquer função utilizada pode-se ainda acrescentar o parâmetro h , que altera a largura de banda, fazendo com que a função estimada tenha curvas mais ou menos suaves.

$$k_h(x) \triangleq \frac{1}{h}k\left(\frac{x}{h}\right)$$

A estimação de função densidade de probabilidade através das funções *kernel* possibilita que a estimação seja feita considerando apenas observações da variável aleatória, por ser uma estimação não-paramétrica. As Figuras 28a-28c mostram a estimação da fdp realizada para diferentes valores de largura de banda e uma mesma quantidade de observações da v.a., o que pode ser percebido é a largura de banda afetando na suavidade da curva que passa de sensível à ruídos e pequeno números de amostras (no caso de σ pequeno), para suavização exagerada (σ grande) .

Por outro lado, as Figuras 28d-28f mostram que para um valor de $N \rightarrow \infty$, a estimação tende à densidade de probabilidade que gerou os dados desde que o valor de h não seja grande o suficiente para fazer com que a curva seja suavizada de forma exagerada.

2.3 TESTE DE COLISÃO EM OBJETOS

A detecção de colisão entre objetos é uma tarefa importante em ambientes de simulação de sistemas de realidade virtual ou jogos (BOURG, 2002). A colisão entre objetos pode ser observada facilmente caso haja o conhecimento de sua geometria.

A colisão entre dois objetos A e B ocorre quando parte da região ocupada por A coincide com a região ocupada por B (Figura 30).

Considerando que diversas vezes a geometria dos objetos assume formas que não são descritas facilmente por equações, uma solução sugerida é a utilização de volumes envolventes (GOTTSCALK, 2000). Desta forma, o objeto em questão é todo envolvido por um volume de geometria conhecida, e o teste de colisão segue uma hierarquia de

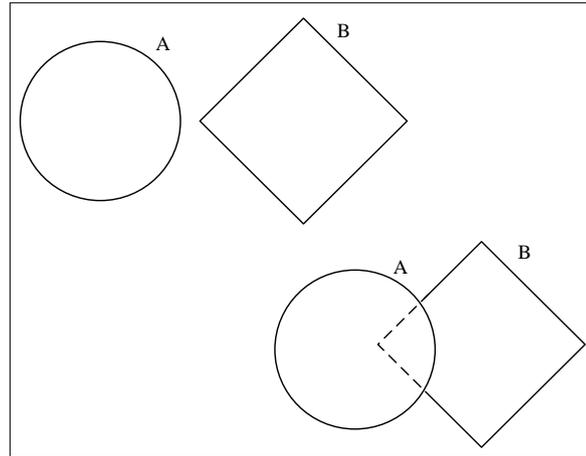


Figura 30: Colisão entre dois objetos

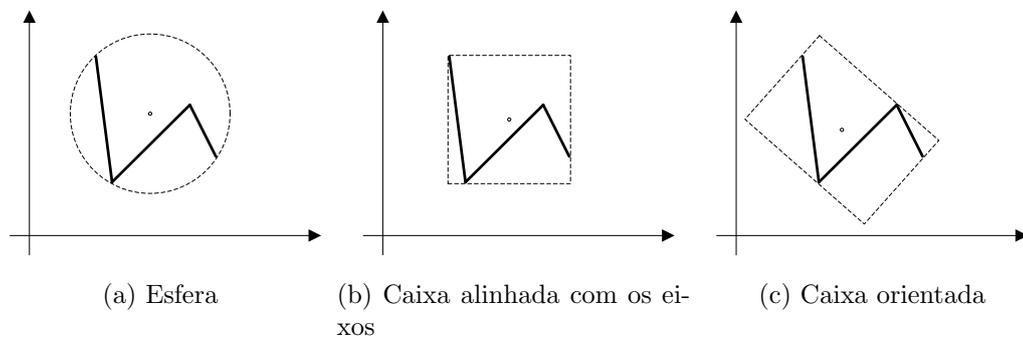


Figura 31: Diferentes formas de volumes envolventes

volumes buscando uma melhor representação do objeto. Assim, quando necessário, o objeto é representado por mais de um volume de menor tamanho do que o primeiro considerado.

Existem diversas topologias baseadas em volumes geométricos convexos cuja matemática de cálculo de interatividade é de rápida avaliação. Logo, com pequena quantidade de operações é possível descobrir se há colisão entre dois volumes.

Dentre muitas das formas possíveis, pode-se destacar esferas (Figura 31a), caixas alinhadas com os eixos globais (Figura 31b), caixas orientadas (Figura 31c), entre outras.

Havendo necessidade de uma representação do objeto mais específica, as Figuras 32a - 32c mostram o comportamento de diferentes formas de volumes envolventes após o procedimento de quebra do volume.

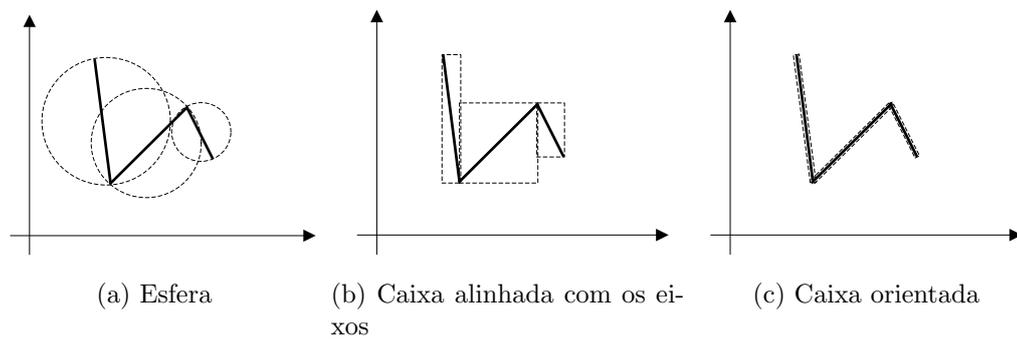


Figura 32: Quebra em diferentes volumes envolventes

2.3.1 ESFERA

A representação de um objeto através de uma esfera trás uma forma simples de observar uma colisão entre dois objetos.

As esferas são criadas com o menor raio possível capaz de envolver todos os pontos de um objeto. E a análise para colisão se resume à comparação entre o raio das esferas envolventes e a distância entre o centro delas.

A Figura 33 mostra o teste de colisão para objetos representados por esferas, aonde estão destacados os raios r_a e r_b de cada uma delas, e a distância entre os centros d_{ab} .

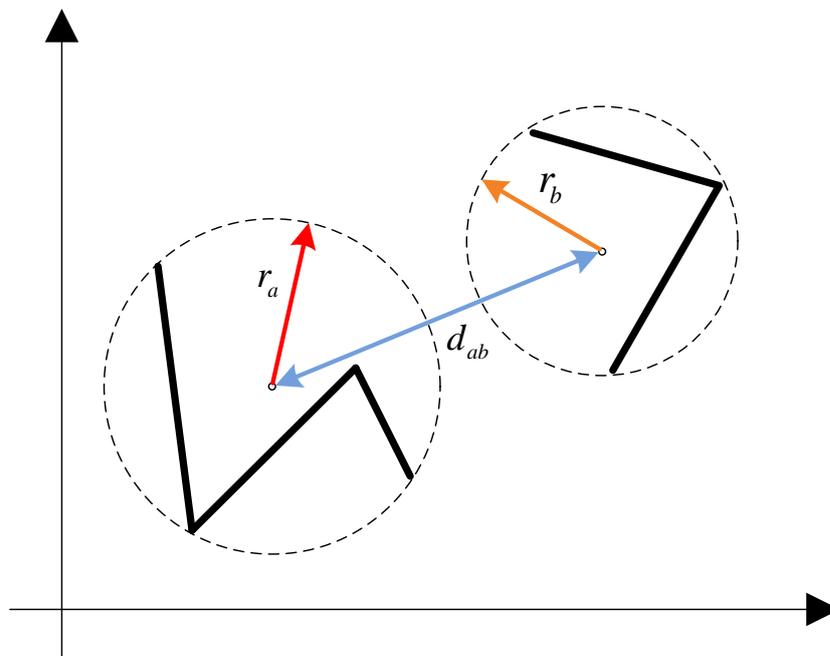


Figura 33: teste de colisão entre esferas

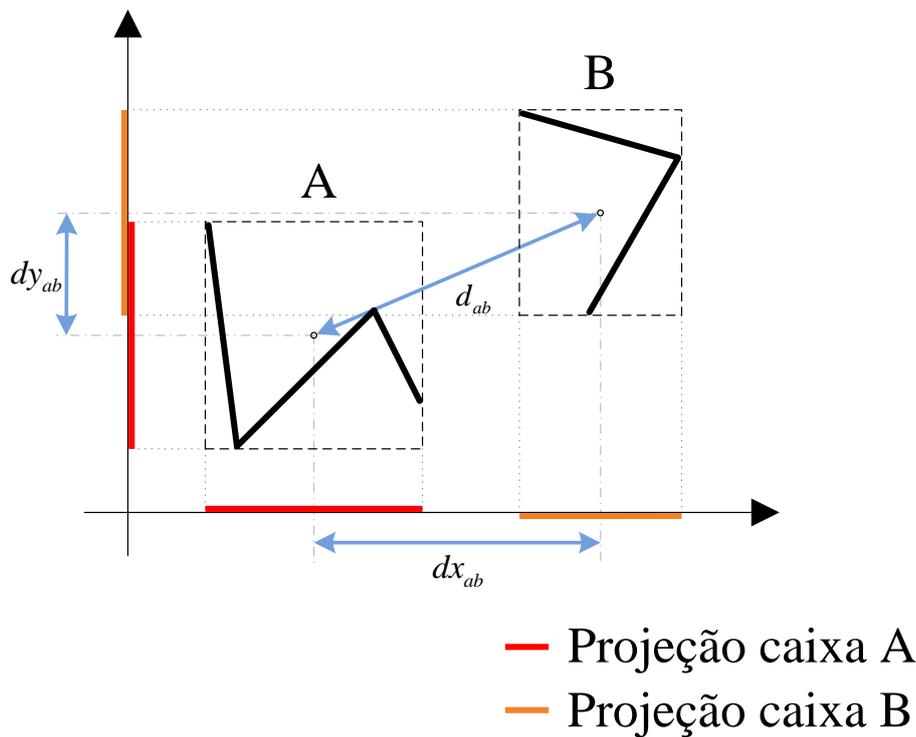


Figura 34: teste de colisão entre caixas alinhadas pelos eixos globais

E desta forma, pode-se interpretar a colisão entre duas esferas como:

$$\begin{cases} r_a + r_b > d_{ab} & \rightarrow \text{colide} \\ r_a + r_b = d_{ab} & \rightarrow \text{tangencia} \\ r_a + r_b < d_{ab} & \rightarrow \text{n\~ao colide} \end{cases}$$

2.3.2 CAIXAS ALINHADAS PELOS EIXOS

De forma semelhante à esfera, as caixas alinhadas devem envolver todos os pontos do objeto.

Como elas são alinhadas de acordo com os eixos globais do sistema, é simples obter o volume envolvente apenas com os valores de mínimo e máximo sobre os eixos.

A maneira de se detectar a colisão entre dois volumes é simples, leva em consideração apenas os valores da distância entre os centros d_{ab} e a largura das caixas nos eixos globais (Figura 34).

A Figura 34 mostra como os objetos utilizados anteriormente seriam representados como caixas alinhadas com os eixos principais. Sendo d_{ab}^i a projeção da distância d_{ab} sobre o eixo i , δ_a^i a metade do módulo da projeção da caixa A também sobre o eixo

i , e δ_b^i a metade do módulo da projeção da caixa B sobre o mesmo eixo, a análise de colisão pode ser vista como:

$$\begin{cases} \delta_a^i + \delta_b^i \geq d_{ab}^i & \rightarrow \text{colide} \\ \delta_a^i + \delta_b^i < d_{ab}^i & \rightarrow \text{não colide} \end{cases}$$

2.3.3 CAIXAS ORIENTADAS

As caixas orientadas têm a mesma forma das caixas alinhadas com os eixos globais. A diferença básica é que elas podem ter orientação arbitrária (Figura 35).

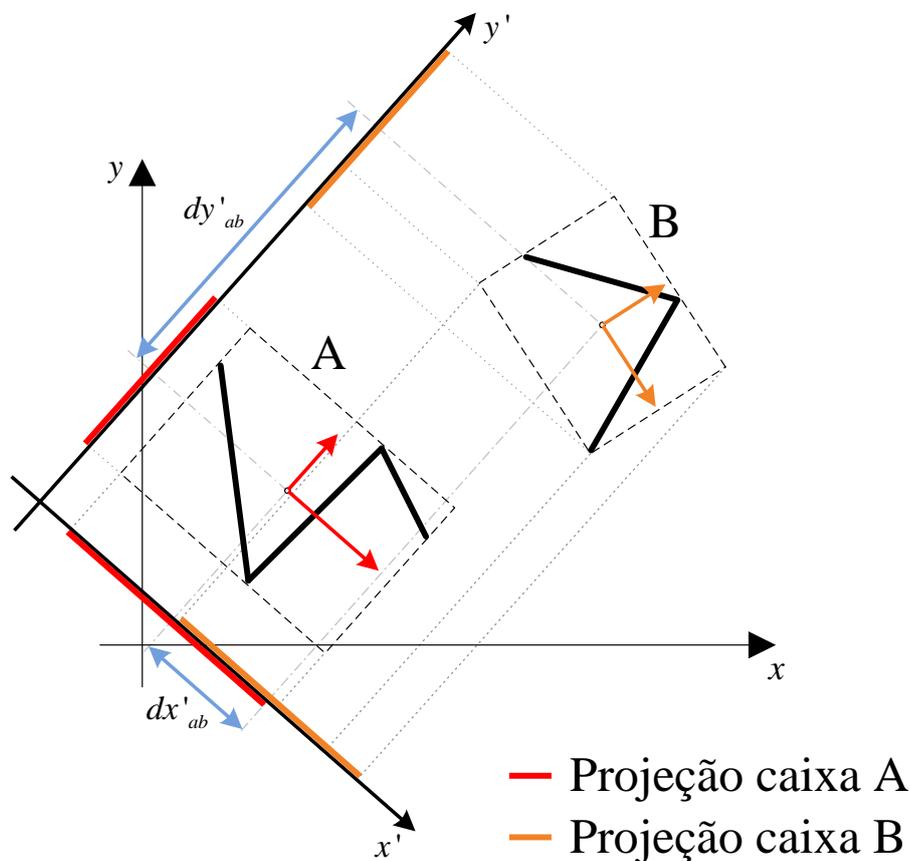


Figura 35: teste de colisão entre caixas orientadas

O fato de a orientação da caixa não estar atrelada à direção dos eixos globais, permite com que as dimensões delas sejam reduzidas, sendo capazes de representar melhor um conjunto de dados que sigam formas variadas, como pode ser visto na Figura 32c.

A detecção de colisões acontece de forma similar às caixas alinhadas, exceto pelos eixos que são considerados. São analisadas projeções das caixas sobre os eixos das

próprias caixas orientadas ao invés dos eixos globais, e desta forma:

$$\begin{cases} \delta_a^{i'} + \delta_b^{i'} \geq d_{ab}^{i'} & \rightarrow \text{colide} \\ \delta_a^{i'} + \delta_b^{i'} < d_{ab}^{i'} & \rightarrow \text{não colide} \end{cases}$$

2.3.4 TEOREMA DO EIXO DE SEPARAÇÃO

Considerando o que foi mostrado nas Subseções anteriores, o teorema do eixo de separação (GOTTSCHALK, 2000) permite garantir que duas caixas são separáveis desde que elas não colidam em um dos eixos de qualquer uma das caixas, ou em combinações lineares desses.

Neste trabalho apenas são consideradas as análises nos eixos de cada caixa, não considerando pois, as combinações lineares. O que aumenta a necessidade de quebras de caixas, porém diminui a quantidade de eixos analisados.

3 DESCRIÇÃO DO MÉTODO DE SEGMENTAÇÕES GEOMÉTRICAS SUCESSIVAS

3.1 INTRODUÇÃO

O Método de Segmentações Geométricas Sucessivas (MSGs) é proposto como uma solução para o treinamento de redes neurais artificiais para classificação de padrões sem a necessidade de especificar uma topologia para rede. Desta forma é capaz de se adaptar a inúmeros problemas diferentes sem que seja preciso a análise de cada um isoladamente, uma vez que é capaz de se adaptar às particularidades na distribuição de um determinado problema.

O método se baseia numa estrutura de rede neural similar à mostrada na Subseção 2.1.4.2, capaz de separar regiões fechadas, e na busca para determinar essas regiões.

Para isso, é necessário fazer com que as classes que se deseja classificar se tornem separáveis umas das outras e após isso encontrar os hiperplanos capazes de separá-las (Seção 2.1.4).

O fluxograma na Figura 36 ilustra o processo do treinamento. Cada classe do problema é agrupada (Seção 3.2). É realizada uma verificação para julgar se elas podem ou não ser separadas por um hiperplano (Seção 3.3). Enquanto não for possível separar todas as classes é realizado um procedimento de separação dos eventos de um determinada classe em conjuntos menores (Seção 3.4). A partir do momento que não existe sobreposição entre os conjuntos de diferentes classes, se torna possível criar um conjunto de hiperplanos capaz de separar-los (Seção 3.6).

3.2 CONTRUÇÃO DE HIPERCAIXAS

Como primeira parte para o funcionamento do MSGs necessita-se do agrupamento de um conjunto de dados para buscar uma forma de separá-los de dados pertencentes à outras classes.

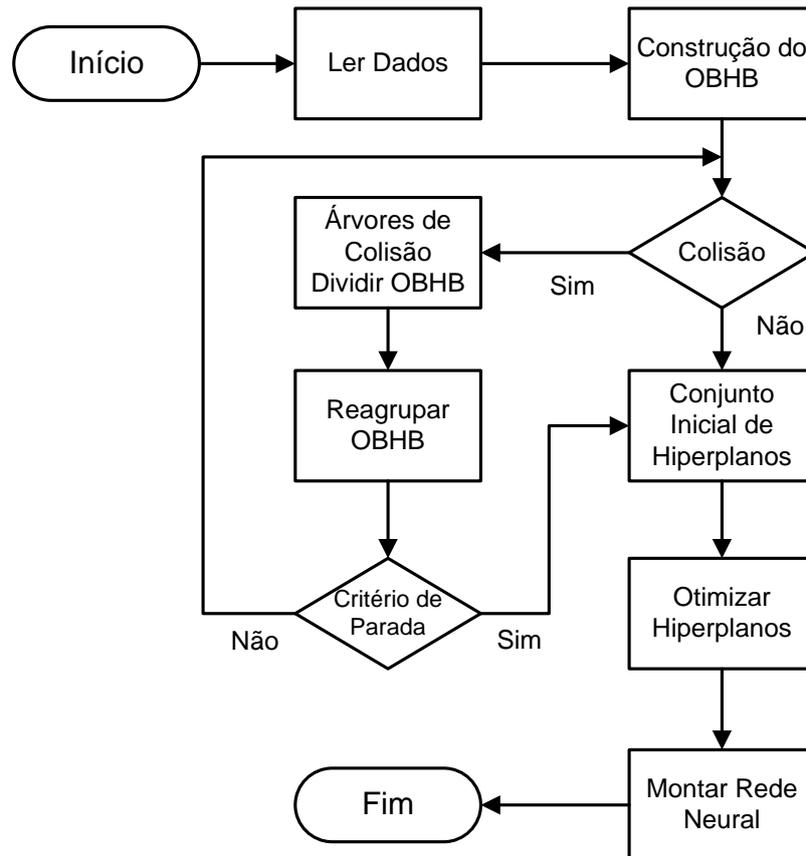


Figura 36: Fluxograma do *MSGS*.

O *MSGS* utiliza suporte algébrico para detectar colisões, a fim de separar as classes. Para tanto existem diversas topologias baseadas em volumes geométricos convexos cuja matemática de cálculo de interação é de rápida avaliação. Logo, com pequena quantidade de operações é possível descobrir se há colisão entre dois volumes.

As formas exemplificadas nas Figuras 31a-31c para representação de objetos em teste de colisão, precisam inicialmente abranger todo o objeto ou conjunto de dados de uma distribuição a ser analisada por uma rede neural.

A colisão entre dois volumes ocorre quando há alguma interseção entre eles como foi mostrado nas Figuras 33-35. No caso de haver uma colisão, um dos volumes é segmentado, fazendo com que a representação de um conjunto de dados por volumes seja mais próxima ao que se deseja representar, como será mostrado na Seção 3.4.

A referência (GOTTSCHALK, 2000) sugere que a forma mais eficaz para detectar colisões através de volumes envolventes é utilizando caixas orientadas (*OBB - Oriented Bounded Box*), a qual foi generalizada para um maior número de dimensões como hipercaixa orientada (*OBHB - Oriented Bounded Hyper-Box*).

O método *OBB* foi inicialmente desenvolvido para o espaço tridimensional, visando

exclusivamente testes de colisão em sistemas de realidade virtual. A seguir uma generalização desta técnica é apresentada para \mathbb{R}^n . A forma utilizada para obtenção dos eixos para uma caixa orientada foi através dos autovetores da matriz de covariância dos dados.

Seja \vec{X} um conjunto de m pontos em \mathbb{R}^n . A construção de uma hipercaixa envolvente alinhada com a distribuição de \vec{X} se inicia através do cálculo do conjunto de autovetores $\vec{\Pi}$ e autovalores $\vec{\Lambda}$ da matriz de covariância $Cov(\vec{X})$ dada pela Equação (3.1). Onde $\vec{\Pi}$ determina vetores alinhados com a dispersão dos pontos ao longo de \mathbb{R}^n e $\vec{\Lambda}$ determina a magnitude da dispersão em cada um destes eixos. Desta forma os autovetores representam as direções no espaço $n - dimensional$ com a maior taxa de variação de dados.

$$Cov = \frac{1}{m} \times \begin{bmatrix} \sum_{k=1}^m (x_1^k - \bar{x}_1)^2 & \dots & \sum_{i=1}^m (x_1^k - \bar{x}_1)(x_r^k - \bar{x}_r) & \dots & \sum_{k=1}^m (x_1^k - \bar{x}_1)(x_n^k - \bar{x}_n) \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \sum_{k=1}^m (x_r^k - \bar{x}_r)(x_1^k - \bar{x}_1) & \dots & \sum_{k=1}^m (x_r^k - \bar{x}_r)^2 & \dots & \sum_{k=1}^m (x_r^k - \bar{x}_r)(x_n^k - \bar{x}_n) \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \sum_{k=1}^m (x_n^k - \bar{x}_n)(x_1^k - \bar{x}_1) & \dots & \sum_{k=1}^m (x_n^k - \bar{x}_n)(x_p^k - \bar{x}_p) & \dots & \sum_{k=1}^m (x_n^k - \bar{x}_n)^2 \end{bmatrix} \quad (3.1)$$

Com isso é possível gerar um novo sistema de coordenadas $\mathcal{C}(\vec{X})$ cujos eixos são definidos de acordo com a orientação do conjunto $\vec{\Pi}$ de autovetores e a origem se encontra no ponto médio \vec{x}_c dos elementos de \vec{X} onde

$$\vec{x}_c = \frac{1}{m} \cdot \sum_{k=1}^m \vec{x}^k \quad (3.2)$$

Logo, para uma dada classe de pontos \vec{X} é realizada a transformação de cada um de seus pontos para o novo sistema de coordenadas $\mathcal{C}(\vec{X})$ obtendo-se $\vec{V} = \{\vec{v}^1, \dots, \vec{v}^m\}$ onde para um dado $\vec{v}^k \in \vec{V}$ têm-se

$$\vec{v}^k = \{Pr(\vec{x}^k, \pi_1), \dots, Pr(\vec{x}^k, \pi_n)\} \quad (3.3)$$

e

$$Pr(\vec{x}^k, \pi_i) = \frac{(\vec{x}^k - \vec{x}_c) \cdot \vec{\pi}_i}{|\vec{\pi}_i|} \quad (3.4)$$

Assim sendo, utilizando \vec{V} é possível obter os limites da projeção (valor máximo e mínimo) dos elementos de \vec{X} sobre cada um dos eixos existentes em $\vec{\Pi}$. Logo pode-se definir uma função de projeção responsável por determinar um segmento de reta

formado pelos limites de \vec{X} sobre um dado eixo π_p como sendo

$$L\vec{P}r(\vec{X}, \pi_p) = \{v_p^{min}, v_p^{max}\} \quad (3.5)$$

onde

$$v_p^{min} = MIN(v_p^K), v_p^{max} = MAX(v_p^K)$$

e $K = \{k \in \mathbb{N} : 1 \leq k \leq m\}$. Utilizando estes segmentos de reta como arestas é possível definir uma hipercaixa $\vec{\mathcal{H}}$ que envolve todos os pontos de \vec{X} . Onde

$$\vec{\mathcal{H}} = \{L\vec{P}r(\vec{X}, \pi_1), \dots, L\vec{P}r(\vec{X}, \pi_n)\} = \{L\vec{P}r(\vec{X}, \vec{\Pi})\} \quad (3.6)$$

contém os limites mínimos e máximos das projeções de \vec{X} sobre cada eixo presente em $\vec{\Pi}$ e representados no sistema de coordenadas $\mathcal{C}(\vec{X})$.

Como exemplo da metodologia descrita até o momento e no restante desta seção, considere dois conjuntos em \mathbb{R}^2 formados por losangos (números) representando o conjunto de dados $\vec{X}^{1L} = \{x_1^{1L}, \dots, x_{11}^{1L}\}$ e círculos (letras) representando o conjunto $\vec{X}^{2C} = \{x_a^{2C}, \dots, x_f^{2C}\}$.

A Figura 37 ilustra o exemplo e mostra a construção da hipercaixa $\vec{\mathcal{H}}^{1L}$ que circunscreve o conjunto de pontos \vec{X}^{1L} . Os eixos π_1 e π_2 são construídos a partir do ponto médio \vec{x}_c segundo os autovetores da matriz de covariância de \vec{X}^{1L} . Cada ponto $\vec{x} \in \vec{X}^{1L}$ é transformado para este novo sistema de coordenadas como pode ser visualizado pelas linhas tracejadas. Finalmente os valores mínimo e máximo da projeção dos pontos em cada eixo é identificado, onde neste exemplo são, para o eixo π_1 as projeções dos pontos 1 e 11 e para o eixo π_2 as projeções dos pontos 11 e 7.

Entretanto, como pode ser visualizado na Figura 38, o centro de $\vec{\mathcal{H}}^{1L}$ não é o mesmo que o ponto médio \vec{x}_c que representa o novo centro do sistema de coordenadas. Desta forma é necessário seu cálculo e também o das projeções parciais e totais da caixa em cada um dos eixos do novo sistema de coordenadas para que a caixa tenha seu centro referenciado na nova origem. O motivo deste passo é facilitar a detecção de colisão que será mostrada na próxima subseção.

Com o centro da hipercaixa calculado pode-se definir Δ_p e δ_p , sendo Δ_p o comprimento da projeção completa, resultado do tamanho total da caixa projetada sobre o eixo π_p , e δ_p o tamanho da projeção parcial, comprimento do centro da caixa à sua borda sobre o eixo π_p como

$$\Delta_{\vec{\mathcal{H}}, p} = |L\vec{P}r(\vec{X}, \pi_p)| = \sqrt{(v_p^{max} - v_p^{min})^2}. \quad (3.7)$$

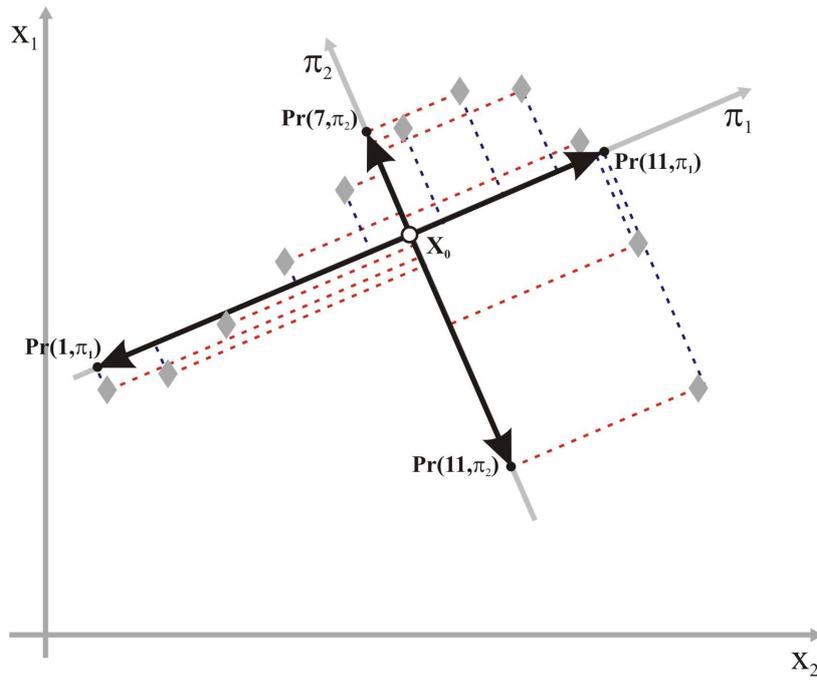


Figura 37: Projeções de \vec{X}^{1L} definindo os limites da Caixa Envolvente $\vec{\mathcal{H}}^{1L}$

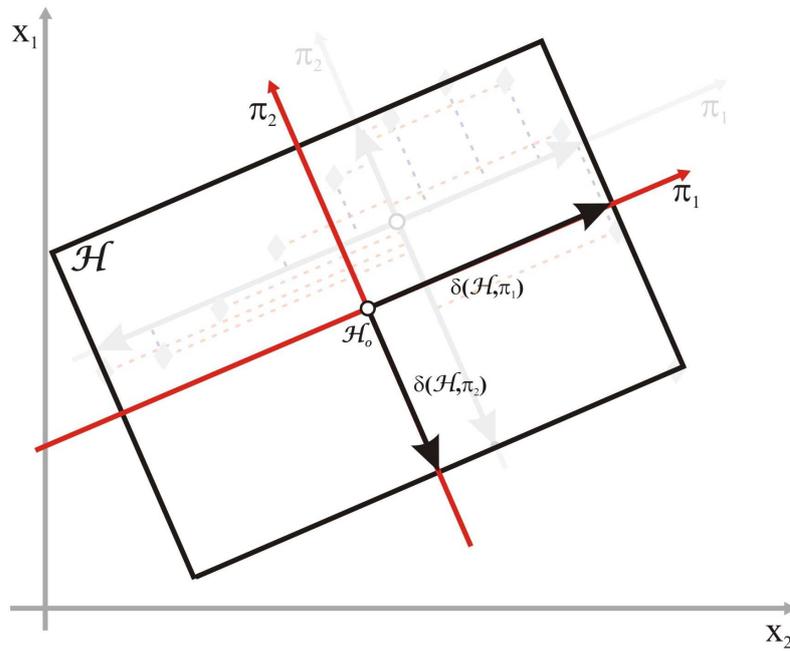


Figura 38: Caixa envolvente $\vec{\mathcal{H}}$ e suas projeções parciais em π_1 e π_2

e

$$\delta_{\vec{\mathcal{H}},p} = \frac{\Delta_{\vec{\mathcal{H}},p}}{2} \tag{3.8}$$

$$\vec{\delta}_{\vec{\mathcal{H}},p} = \delta_{\vec{\mathcal{H}},p} \times \vec{\pi}_p \tag{3.9}$$

e o centro da hipercaixa representado por $\vec{\mathcal{H}}_0$ é dado por

$$\vec{\mathcal{H}}_0 = \left\{ \frac{v_1^{max} + v_1^{min}}{2}, \dots, \frac{v_n^{max} + v_n^{min}}{2} \right\} \quad (3.10)$$

Como \vec{X} e $\vec{\mathcal{H}}$ são vetores formados por pontos em \mathbb{R}^n com $\vec{\mathcal{H}}$ sendo uma projeção de \vec{X} em $\vec{\Pi}$ pode-se ainda definir que para o sistema de coordenada \mathcal{C} , têm-se que as projeções de \vec{X} e $\vec{\mathcal{H}}$ são iguais, ou seja:

$$Pr(\vec{X}, \vec{\Pi}) = Pr(\vec{\mathcal{H}}, \vec{\Pi}) \quad (3.11)$$

3.3 TESTE DE COLISÃO

Seguindo o desenvolvimento da Seção 3.2 é necessário um procedimento para identificar se duas hipercaixas $\vec{\mathcal{H}}^1$ e $\vec{\mathcal{H}}^2$ ocupam o mesmo lugar no espaço. A proposta inicialmente demonstrada em (GOTTSCHALK, 2000) apresenta um método denominado *Teorema do Eixo de Separação (TES)* para realizar o teste de colisão. Entretanto, como será demonstrado a seguir, o *TES* tem como objetivo encontrar apenas um eixo onde as projeções das caixas não estão sobrepostas. Para a finalidade de classificação é necessário expandir este conceito e encontrar todos os eixos de separação e definir hiperplanos normais a estes eixos, denominados hiperplanos de separação, que fornecem uma função capaz de determinar se um dado ponto pertence a uma ou outra classe. Isto é de vital importância, pois como é demonstrado na Seção 2.1.4, os parâmetros de um hiperplano são utilizados para a criação de um neurônio correspondente. Para apresentar estas teorias, a Subseção 3.3.1 demonstra a aplicação do TES, originalmente desenvolvido para \mathbb{R}^3 , em um espaço \mathbb{R}^n e, na sequência, a Subseção 3.3.2 demonstra o Teorema do Hiperplano de Separação (THS).

3.3.1 TEOREMA DO EIXO DE SEPARAÇÃO (TES)

Sejam duas hipercaixas $\vec{\mathcal{H}}^1$ e $\vec{\mathcal{H}}^2$ em \mathbb{R}^n com seus respectivos centros em $\vec{\mathcal{H}}_0^1$ e $\vec{\mathcal{H}}_0^2$ formando o conjunto de pontos $\vec{X}_c = \{\vec{\mathcal{H}}_0^1, \vec{\mathcal{H}}_0^2\}$. Se existir um eixo $\pi_p \in \mathbb{R}^n$ que atenda à inequação

$$| LPr(\vec{X}_c, \pi_p) | \geq \frac{| LPr(\vec{\mathcal{H}}^1, \pi_p) | + | LPr(\vec{\mathcal{H}}^2, \pi_p) |}{2} \quad (3.12)$$

ou, de forma análoga,

$$\Delta_{\vec{X}_c, p} \geq \delta_{\vec{\mathcal{H}}^1, p} + \delta_{\vec{\mathcal{H}}^2, p}. \quad (3.13)$$

onde

$$\delta_{\vec{H},p} = \sum_{i=1}^n \left| \frac{\vec{\delta}_{\vec{H},i} \cdot \vec{\pi}_p}{|\vec{\pi}_p|} \right| \quad (3.14)$$

os conjuntos representados por \vec{H}^1 e \vec{H}^2 estão devidamente separados. Logo, para se determinar que não existe colisão, é necessário encontrar um eixo onde a projeção das imagens das hipercaixas não é contínua, ou seja, um eixo sobre o qual a projeção do vetor que liga os centros das hipercaixas tenha módulo menor que as projeções parciais das hipercaixas somadas.

A Figura 39 ilustra este procedimento onde $\{\pi_1, \pi_2\}$ e $\{\pi_3, \pi_4\}$ são, respectivamente, as bases dos sistemas de coordenadas \mathcal{C}^1 e \mathcal{C}^2 que contêm duas hipercaixas \vec{H}^1 e \vec{H}^2 . Seja $\vec{\Delta}_{\vec{H}_0^1, \vec{H}_0^2}$ um vetor que vai do centro de \vec{H}^1 para \vec{H}^2 . As projeções parciais de \vec{H}^1 sobre os eixos π_1 e π_2 e as projeções parciais de \vec{H}^2 sobre π_3 e π_4 são representadas por $\delta_{1,1}, \delta_{1,2}, \delta_{2,3}, \delta_{2,4}$ respectivamente. Aplicando o *TES* para o eixo $\pi_3 = x_2$ têm-se que a soma de $|\vec{\delta}_{\vec{H}^1,3}|$ com $|\vec{\delta}_{\vec{H}^2,3}|$ é inferior a projeção dos centros das caixas $|\vec{\Delta}_{\vec{H}_0^1, \vec{H}_0^2}|$, atendendo à Equação 3.13. Desta forma o eixo π_3 é considerado um eixo de separação.

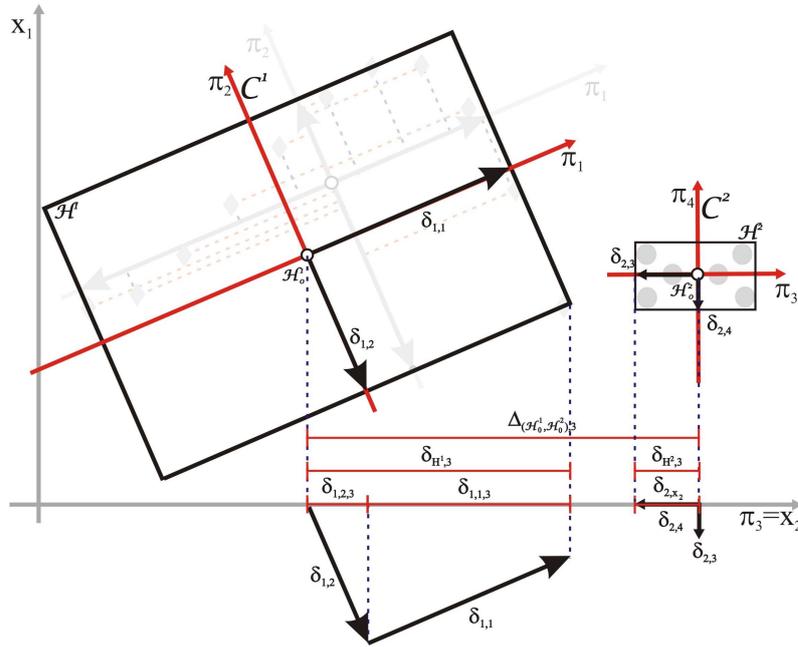


Figura 39: Identificação do Eixo de Separação

É possível visualizar na figura que para os eixos restantes, apenas π_1 é um eixo de separação, já que π_2 e π_4 não atendem às condições necessárias.

O algoritmo original em um espaço tridimensional sugere que o teste seja realizado em pelo menos 15 eixos (os 3 eixos de cada objeto mais a combinação linear entre estes) e, se algum destes possuir uma descontinuidade na projeção os elementos não

estão em colisão. Entretanto realizar a combinação linear de eixos de espaços com muitas dimensões irá aumentar a complexidade do problema de forma exponencial. Entretanto, como foi demonstrado em (HONÓRIO et al., 2008), utilizando busca em árvore de colisão é possível obter o mesmo resultado qualitativo executando mais vezes o teste, porém com apenas os eixos originais, sem considerar suas combinações lineares.

Outra diferença apresentada por este trabalho é que, apesar da identificação de apenas um eixo que atenda às exigências do teorema já caracterize uma separação, para o fim proposto é necessário a identificação de todos os eixos de separação, desta forma, para cada conjunto de 2 hipercaixas $\in \mathbb{R}^n$ devem ser executados $2 \times n$ testes. Isso permite encontrar todos os hiperplanos e numa etapa posterior definir a melhor forma para separar todas as hipercaixas do problema.

3.3.2 TEOREMA DO HIPERPLANO DE SEPARAÇÃO (THS)

Este trabalho ainda mostra uma extensão do Teorema do Eixo de Separação apresentado anteriormente. Considerando duas hipercaixas $\vec{\mathcal{H}}^1$ e $\vec{\mathcal{H}}^2$ no espaço \mathbb{R}^n , se as caixas não estiverem em colisão, existe um eixo de separação π_p que contém as projeções $\delta_{\vec{\mathcal{H}}^1,p}$ e $\delta_{\vec{\mathcal{H}}^2,p}$, conforme demonstrado na Subseção 3.3.1.

Considerando π_p como sendo o eixo de separação é possível definir um segmento de reta \mathcal{S} sobre este eixo tal que

$$|\mathcal{S}| = \Delta_{\vec{x}_{c,p}} - \delta_{\vec{\mathcal{H}}^1,p} - \delta_{\vec{\mathcal{H}}^2,p}. \quad (3.15)$$

Isto significa que existem pontos pertencentes ao eixo π_p que estão localizados entre as projeções das hipercaixas. Utilizando assim o segmento de reta \mathcal{S} é possível então definir um hiperplano $\chi = 0 \in \mathbb{R}^n$ que seja normal à π_p e que contenha um ponto \vec{x}^r sobre o segmento de reta \mathcal{S} , logo

$$\chi = \vec{\pi}_p \cdot \vec{X} - \vec{\pi}_p \cdot \vec{x}^r \quad (3.16)$$

onde \vec{X} é um conjunto de pontos pertencentes a \mathbb{R}^n que atendem a equação 3.16, formando assim o hiperplano χ . Logo $\chi(\vec{\mathcal{H}}^i, \vec{\mathcal{H}}^j)$ é definido como sendo um hiperplano de separação entre as hipercaixas $\vec{\mathcal{H}}^i$ e $\vec{\mathcal{H}}^j$.

A Figura 40 exemplifica este processo onde $\vec{\mathcal{H}}^1$ e $\vec{\mathcal{H}}^2$ são duas hipercaixas em \mathbb{R}^2 . O segmento de reta \mathcal{S} é definido considerando os valores das projeções $\delta_{\vec{\mathcal{H}}^1,p}$ e $\delta_{\vec{\mathcal{H}}^2,p}$ sobre o eixo de separação π_p , e os centros das caixas $\vec{\mathcal{H}}_0^1$ e $\vec{\mathcal{H}}_0^2$. Utilizando o ponto \vec{x}^r sobre

o segmento de reta \mathcal{S} , criou-se o hiperplano de separação $\chi(\vec{\mathcal{H}}_i, \vec{\mathcal{H}}_j)$ perpendicular ao eixo π_p . Uma observação importante é que *pode existir mais de um eixo de separação entre duas hipercaixas* pois qualquer ponto \vec{x}^r sobre \mathcal{S} é candidato para a construção de $\chi(\vec{\mathcal{H}}_i, \vec{\mathcal{H}}_j)$. Neste trabalho o ponto considerado para análise foi o ponto médio de \mathcal{S} . Finalizando, é possível utilizar os mesmos parâmetros encontrados através da Equação

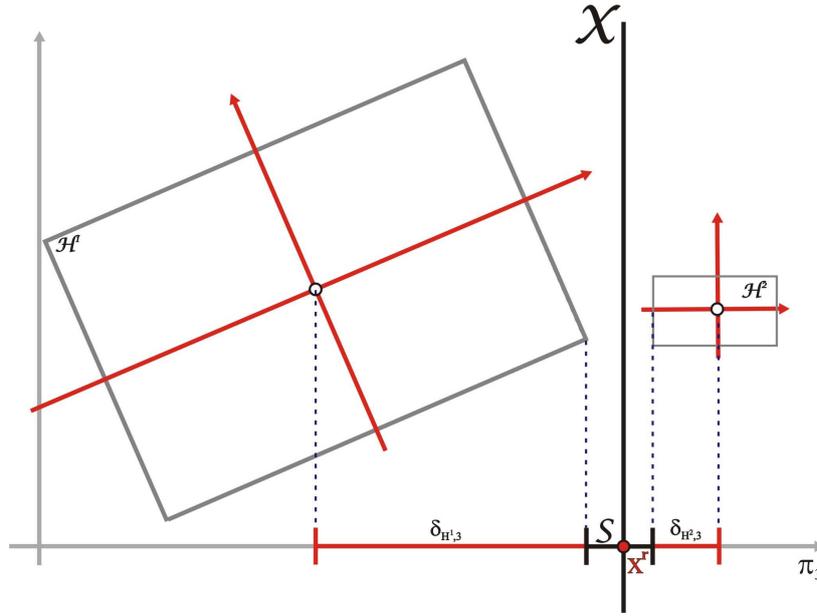


Figura 40: Definição do Hiperplano de Separação

(3.16) para definir um neurônio conforme a Equação (A.4) apresentada no *Apêndice-A*.

A Figura 41 apresenta a construção de um hiperplano de separação descrito pela equação $\chi = \vec{\pi}_p \cdot \vec{X} - \vec{\pi}_p \cdot \vec{x}^r = 0$ para um caso no qual \vec{X} é um conjunto de pontos pertencentes à \mathbb{R}^3 a fim de ratificar que o conceito pode ser expandido para $\vec{x} \in \mathbb{R}^n$.

3.4 SEGMENTAÇÃO GEOMÉTRICA DOS OBHB

Dado que ocorreu colisão, torna-se necessário aumentar a representatividade da hipercaixa com a base de dados em análise. Uma técnica apresentada na literatura (GOTTSCHALK; LIN; MANOCHA, 1996) é através da segmentação geométrica dos dados, gerando uma busca denominada de *Árvore de Colisão (AC)*. Similar às tradicionais buscas em árvore, a técnica de AC divide os dados de uma classe em uma forma hierárquica onde a cada novo nível o conjunto total de pontos é dividido e, para cada divisão, uma nova hipercaixa é criada aumentando a precisão do envelope.

Existem diversas técnicas de busca que podem ser aplicadas para decisão de qual massa de dados será dividida, (CHANG; WANG; KIM, 2010) e (JAMES; PAI, 2004), por

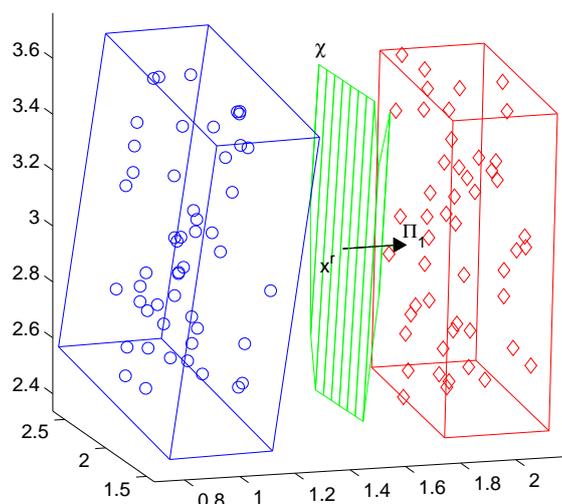


Figura 41: Hiperplano de Separação

quantas vezes e qual a ordem desta divisão. Por exemplo, se a divisão for realizada sempre no mesmo objeto, caracteriza uma busca em profundidade pela árvore. Apesar de factível, esta heurística não é interessante por concentrar todo o esforço computacional na delimitação apenas do primeiro objeto.

No algoritmo do MSGS, a segmentação ou quebra de uma hipercaixa $\vec{\mathcal{H}}_n$ que envolve o conjunto de pontos \vec{X}^n , a divide em duas novas hipercaixas $\vec{\mathcal{H}}_{n1}$ e $\vec{\mathcal{H}}_{n2}$. A quebra ocorre ao longo do eixo π_p associado ao maior autovalor de $\vec{\Lambda}$, ou seja, $\lambda_p = \text{MAX}(|\vec{\Lambda}|)$.

O método escolhido para definir o ponto de quebra leva em consideração como os dados estão distribuídos ao longo da hipercaixa. Para isso foi utilizado o estimador de densidade de probabilidade *kernel*, mostrado na Subsecção 2.2.1.2 (MACHADO; HONÓRIO; CERQUEIRA, 2013).

Esta forma de segmentação permite escolher pontos de quebra capazes de separar os dados de uma hipercaixa em caixas resultantes que melhor se adaptem a forma como os dados estão distribuídos, melhorando a representação deles no processo.

Como exemplo para ilustrar esta quebra, é utilizado um conjunto de três pontos envolvidos por uma caixa orientada, mostrado na Figura 42.

O valor de desvio padrão escolhido como largura do *kernel* para gerar a estimação de densidade, está intimamente ligado à interferência entre os dados de um conjunto na construção da densidade (Equação (2.19)), e desta forma quanto menor o valor de σ mais próximo devem estar os pontos para que haja contribuição entre as gaussianas, e

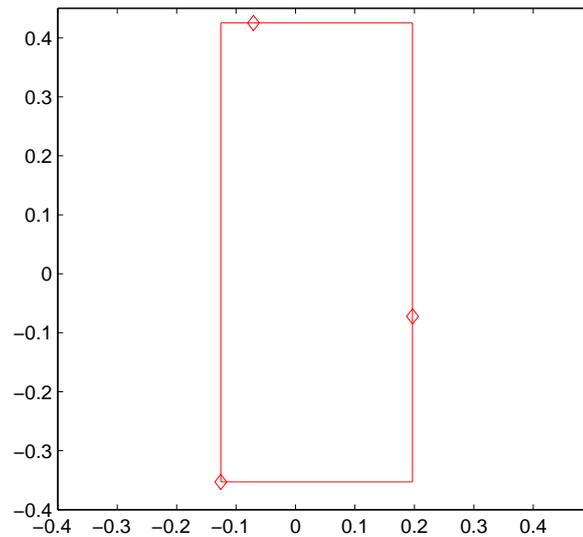
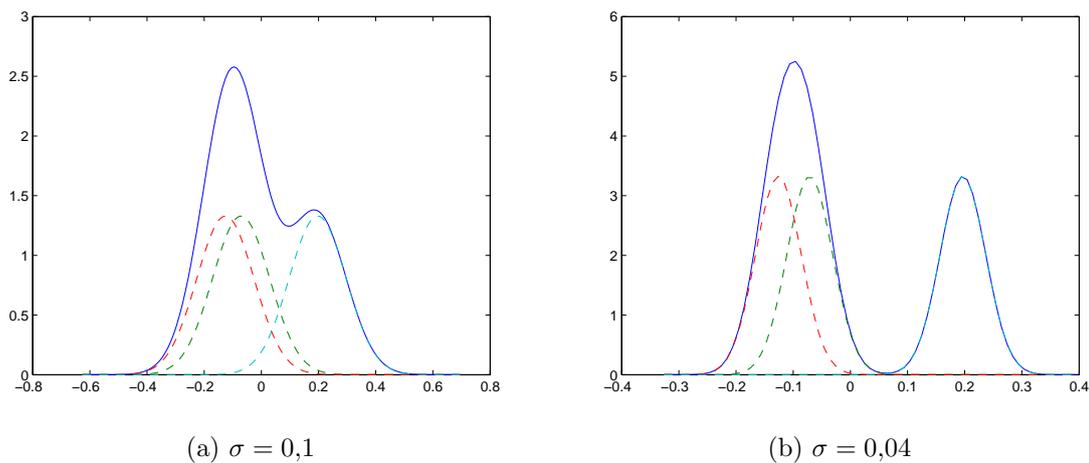


Figura 42: OBHB com três pontos gerados pertencentes à mesma casa



(a) $\sigma = 0,1$

(b) $\sigma = 0,04$

Figura 43: distribuição gaussiana para três pontos de mesma fonte e somatório das distribuições

assim, gerar uma distribuição contínua, sem vales. A Figura 43 mostra um comparativo com curvas geradas usando dois valores diferentes de σ para estimação da densidade de probabilidade dos pontos pertencentes à caixa mostrada na Figura 42, na qual fica possível perceber a diferença na interação entre as curvas realçando a descontinuidade na distribuição dos pontos dentro da hipercaixa.

3.4.1 PONTO DE QUEBRA

A quebra de uma Hipercaixa considerando a estimação da densidade de probabilidade é realizada utilizando um valor de limiar, sendo assim, valores na curva de

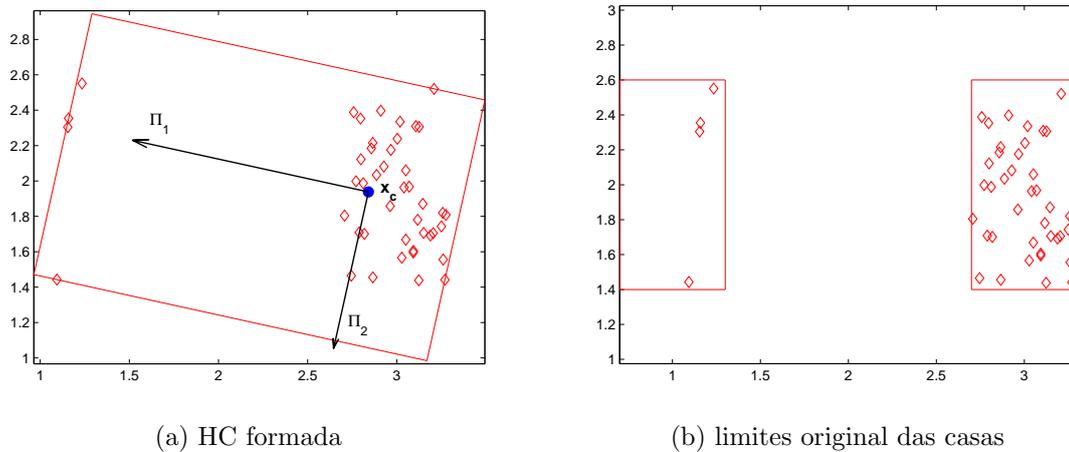


Figura 44: Instâncias de duas casas diferentes

densidade menores que este limiar são tratados como pertencentes a um intervalo para quebra da OBHB.

A Figura 44a mostra um caso frequente no qual a HC agrupa um conjunto de dados \vec{X} que pertence a uma mesma classe porém apresentam características diferentes que fazem com que seja descontínua no espaço de soluções. Neste caso o conjunto é gerado através da função tabuleiro (*Apendice-B*) pertencendo à duas casas diferentes. A Figura 44b mostra os limites originais das casas utilizadas circundando as instâncias pertencentes à cada uma delas. As casas estão centradas em $x = 1$ e $x = 3$ e cada uma contribui com um número diferente de pontos: quatro na casa em $x = 1$ e quarenta naquela em $x = 3$.

O fato da hipercaixa contar com contribuições diferentes das duas casas e o modo como os pontos estão separados dentro da HC, faz com que o ponto médio \vec{x}_c se encontre dentro dos limites de uma das casas (Figura 44a).

A quebra da Hipercaixa é realizada tomando o eixo de maior autovalor π_1 . Sendo levado em conta a densidade de probabilidade da Hipercaixa, o ponto de quebra escolhido fica entre os dois picos da distribuição o que favorece a quebra (Figura 45 mostra a caixa em relação ao sistema de coordenadas $\mathcal{C}(\vec{X})$). Desta forma, as caixas filhas resultantes do processo se adaptam melhor à formação dos dados, evitando que uma HC englobe pontos de duas casas diferentes como poderia ocorrer em outras formas de quebra (Figura 46b).

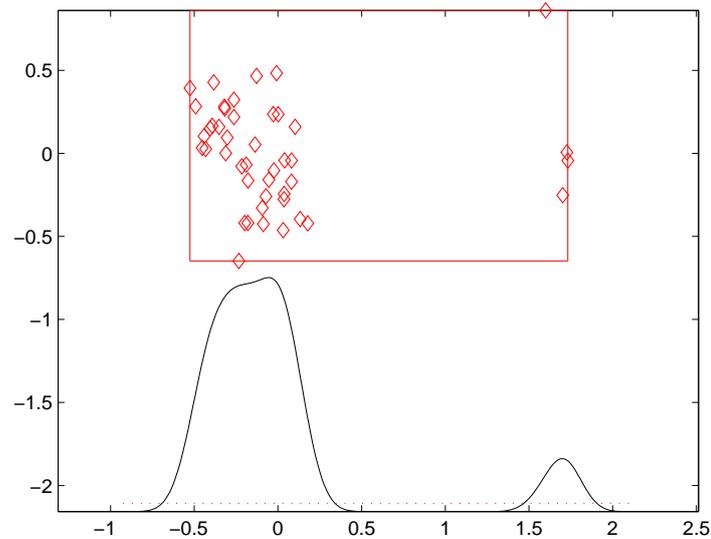
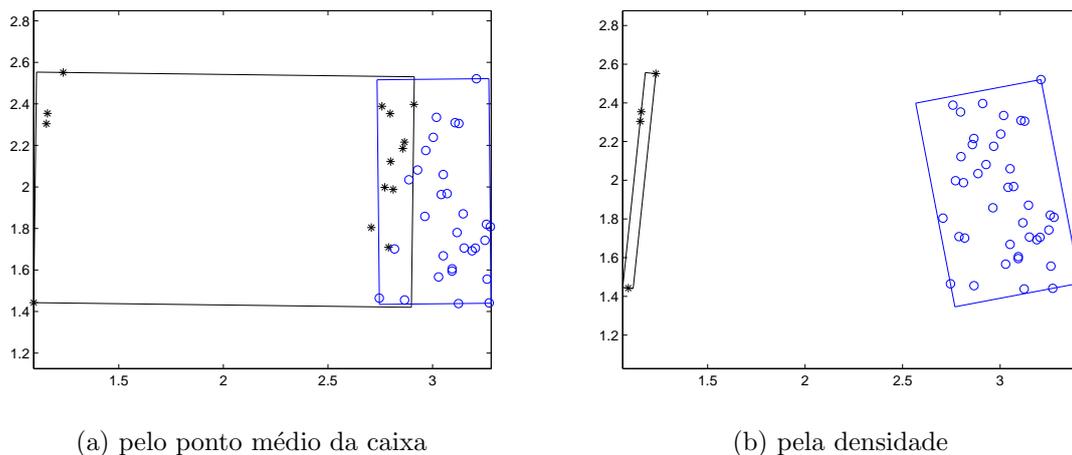


Figura 45: distribuição de probabilidade da HC



(a) pelo ponto médio da caixa

(b) pela densidade

Figura 46: quebra da HC

3.5 REAGRUPAMENTO DE OBHB

Neste ponto as hipercaixas já segmentadas em caixas filhas apresentam separação umas das outras. Com isso consegue-se obter um conjunto de neurônios que divide o espaço em diversas regiões de uma classe.

Para separar todas estas regiões de forma a identificar corretamente as classes desejadas, não podem haver pontos de classes distintas em uma mesma região bem como é interessante que não haja redundância nos hiperplanos. Desta forma, após realizadas quebras nas maiores caixas de forma que não se colidam mais, é necessário efetuar algumas operações para que não ocorram situações indesejadas. A exemplificação destes

conceitos será realizada novamente através da função apresentada no *Apêndice-B*. Tal sistema apresenta dois conjuntos de dados não linearmente separáveis onde regiões que possuem pontos de cada um dos conjuntos se alternam no espaço de solução \mathbb{R}^2 (Figura 47a).

A Figura 47b mostra a construção inicial das caixas onde pode-se visualizar a existência de colisão. Seguindo o procedimento descrito na Seção 3.4, a quebra continua sempre pela maior caixa que apresenta colisão até que o conjunto esteja separado. A Figura 47c mostra o final do processo onde as caixas não apresentam nenhuma colisão. Entretanto, apesar de todas as caixas estarem separadas, é possível observar que algumas regiões não foram corretamente separadas, considerando a forma conhecida da função. Apesar desta situação não ser necessariamente um erro, seria mais desejável uma melhor especialização de cada região.

São adotadas estratégias na tentativa de solucionar a falha na identificação ressaltada: algumas caixas são quebradas novamente por apresentarem indícios de não serem representativas de uma só região e, após isso, as caixas resultantes são reagrupadas de uma outra forma.

Uma análise escolhida para detectar caixas cuja quebra possa favorecer o funcionamento do método faz com que para cada caixa contendo um número mínimo de dados, uma nova quebra é realizada e caso a soma das áreas resultantes das duas novas caixas for menor que uma porcentagem da caixa original (e.g. 50%), isso indica a possibilidade da existência de pontos provenientes de diferentes regiões estarem sendo representados pela mesma hipercaixa, desta forma a quebra se mantém, caso contrário a caixa original permanece inalterada.

Posteriormente um reagrupamento é realizado da seguinte forma: uma lista das regiões é indexada da menor para a maior área. Seguindo esta ordem, cada elemento da lista é reagrupado com o seu par mais próximo pertencente a mesma classe de saída. Se esta nova hipercaixa não apresentar colisões, a união se mantém, uma nova lista indexada é criada e o processo recomeça do menor elemento da lista. Entretanto, caso a caixa esteja em colisão, as caixas originais se mantêm e o processo continua normalmente para o restante da lista. Isto é realizado até que nenhuma caixa mais possa ser reagrupada. A Figura 47d mostra o resultado final deste processo.

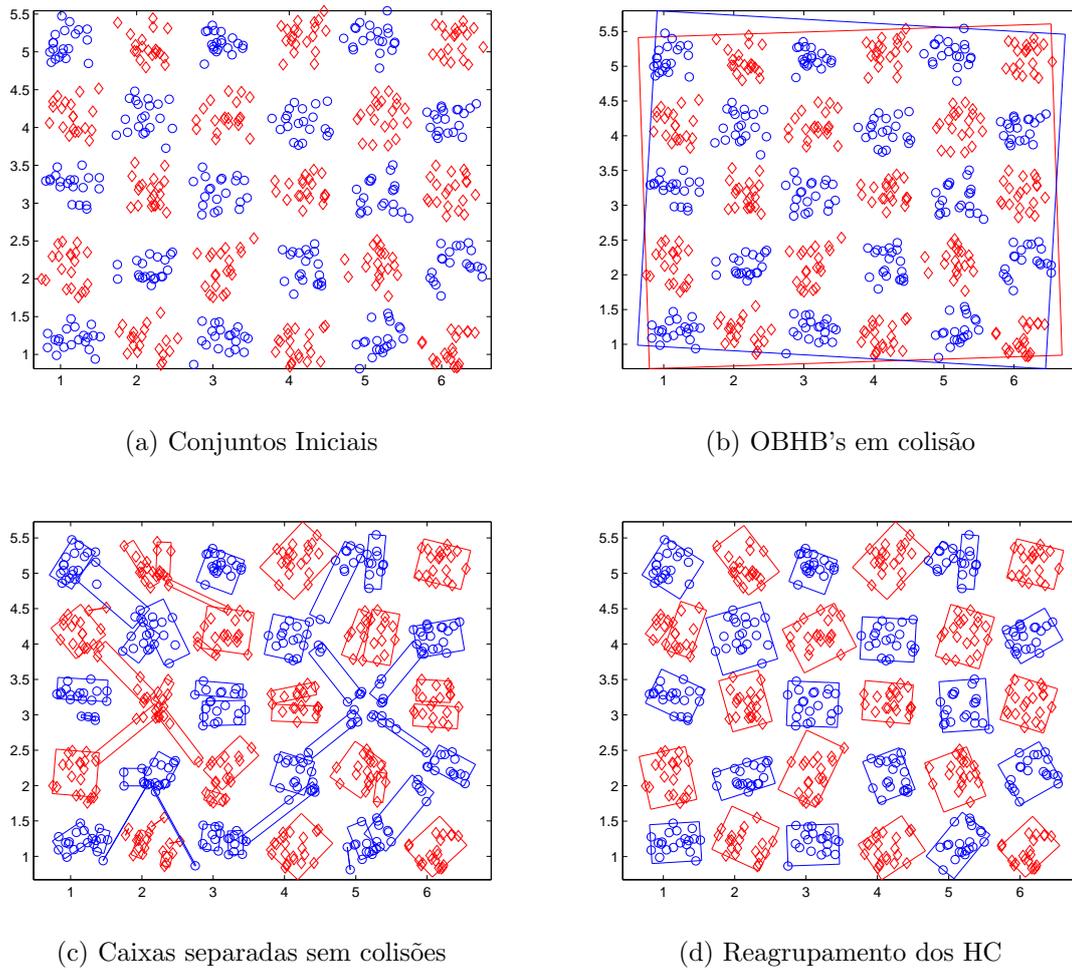


Figura 47: Exemplo de Geração de Rede Neural utilizando o TES

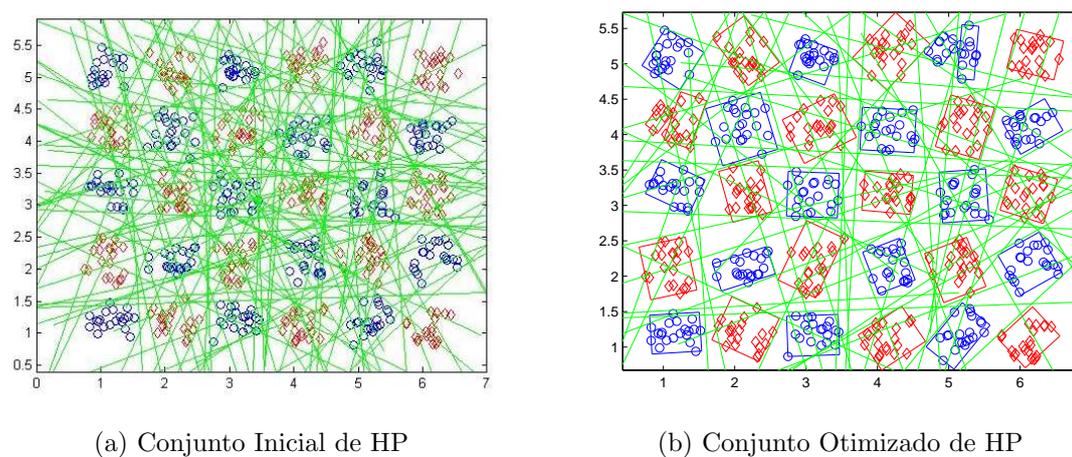


Figura 48: Otimizações aplicadas ao TES

3.6 OTIMIZAÇÃO DOS HIPERPLANOS

Com o prosseguimento do algoritmo de busca chega-se à um ponto onde as caixas, já segmentadas em caixas filhas, apresentam separação umas das outras. Neste ponto é

possível encontrar todos os planos de separação conforme apresentados na Figura 48a. Pode-se observar na figura que o excesso de hiperplanos encontrados iria gerar uma rede neural muito mais complexa e especializada do que o necessário.

Para simplificar a rede neural gerada, é necessário encontrar qual o menor conjunto de hiperplanos que separe todas as regiões pertencentes à diferentes classes. Isto é um problema de otimização discreta e pode ser abordado por diversos métodos. Entretanto, como a principal finalidade deste trabalho é mostrar a eficácia do método de treinamento da rede neural, a solução apresentada aqui baseia-se em um critério de busca em profundidade simples sem que nenhuma otimização mais elaborada tenha sido aplicada.

O critério de escolha de um dado hiperplano é um índice determinando quantas regiões ele separa. Desta forma todos os hiperplanos são indexados de acordo com este índice, e a cada iteração do algoritmo de busca, o hiperplano que é capaz de separar mais regiões é adicionado à lista dos hiperplanos selecionados para a montagem final da rede neural. Para evitar redundância, as regiões separadas são marcadas não sendo necessária a adição de novos hiperplanos para a sua classificação. Este processo continua até que todas as hipercaixas estejam separadas. Aplicando este procedimento no resultado mostrado na Figura 48a o número de hiperplanos diminuiu de 798 para apenas 54 (Figura 48b).

3.7 MONTAGEM DA REDE NEURAL

A rede neural gerada pela metodologia proposta é composta de uma camada de entrada, três camadas ocultas e uma camada de saída. A funcionalidade de cada camada é descrita abaixo:

Camada de Entrada: responsável por distribuir as entradas para os neurônios da primeira camada oculta. Na proposta adotada, apresenta-se uma camada de entrada cuja única função topológica é a organização e distribuição dos dados para a camada oculta. Possui número de unidades igual aos atributos de entrada.

Primeira Camada Oculta: tem o objetivo de fazer a avaliação do ponto de entrada para cada hiperplano de acordo com a função expressa na Equação (3.16). Destaca-se a correspondente Equação (A.4) representativa do neurônio tornando possível a equiparação dos parâmetros do hiperplano com os parâmetros da rede. Esta funcionalidade retorna um valor real que, caso nulo o ponto está no hiperplano, caso positivo está

acima do hiperplano, e caso negativo está abaixo. Relembrando que os valores positivos e negativos são determinados pelo sentido da normal que define o hiperplano, tem-se que a camada é formada por um número de neurônios igual ao número de hiperplanos que além de indicar o lado do ponto em relação a um dado hiperplano, calcula sua distância para o mesmo.

Segunda Camada Oculta: responsável por processar o conjunto de hiperplanos que definem a pertinência de um dado ponto a uma região definida do espaço. Cada hipercaixa é separada por um conjunto de hiperplanos que modelaram um subconjunto de neurônios da primeira camada oculta. Desta forma, cada hipercaixa irá definir um neurônio na segunda camada oculta da rede neural que terá como saída um valor real indicando a distância mínima do ponto testado à sua fronteira e irá retornar positivo caso o ponto esteja dentro da região e negativo caso esteja fora. Uma observação importante é que, embora seja possível, não é necessário propagar todos os neurônios da primeira para a segunda camada oculta uma vez que apenas o conjunto que forma a fronteira da região são exigidos como entrada. O funcionamento dos neurônios desta camada se assemelha ao de um 'operador E ', uma vez que retornará positivo apenas no caso em que o ponto estiver entre todos os hiperplanos que delimitam uma dada região.

Terceira Camada oculta: possui quantidade de neurônios igual ao número de diferentes classes que se deseja classificar, e retorna valores reais para cada uma. Caso a saída seja positiva, significa que o ponto testado se encontra em alguma das regiões delimitadas pelos hiperplanos, caso seja negativa, indica a distância deste ponto para a região classificada mais próxima. A necessidade desta camada é fornecer um *índice de pertinência* individual para cada instância de saída que será processado pela próxima camada. Funcionamento similar ao de um 'operador OU ', já que a saída será positiva quando o ponto pertencer à alguma das regiões definidas como sendo de uma determinada classe.

Camada de Saída: processa os valores da terceira camada oculta. Esta camada possui o mesmo número de neurônios que a camada anterior e além de examinar os resultados, retorna um valor positivo para o neurônio que representa a classe a qual o ponto testado foi classificado.

4 RESULTADOS

Três conjuntos de testes serão realizados. Primeiro utilizando a função tabuleiro mostrada no *Apêndice-B*, uma comparação de desempenho do algoritmo proposto será realizada com a *toolbox* de reconhecimento de padrões do *Matlab 2011* (MRP). O segundo conjunto de testes será realizado com bancos de dados conhecidos. Neste estudo foram utilizados cinco bancos de dados obtidos do repertório disponibilizado UCI (ASUNCION; NEWMAN, 2007). O terceiro conjunto, mostrará resultados comparativos com o problema de espiral dupla. Para todos os casos o algoritmo MSGS foi desenvolvido em *Matlab*, e as simulações realizadas num computador com processador Intel core i7, 8Gb RAM e 2,93GHz.

4.1 TESTES UTILIZANDO A FUNÇÃO TABULEIRO

A função utilizada para testes apresentada no *Apêndice-B* define uma função tabuleiro (FT) na qual elementos pertencentes à duas classes se alternam no espaço de solução com *offset* e *distr* definidos pelo usuário. Em todos os testes descritos a seguir os valores utilizados foram $offset(\delta) = 2$ e $distr = 0,3$.

A *toolbox* de reconhecimento de padrões foi utilizada para gerar RNA's *feedforward* do tipo perceptron de múltiplas camadas (Multilayer Perceptron (MLP)) treinadas com o algoritmo Levenberg-Marquardt (LEVENBERG, 1944)(MARQUARDT, 1963), que são utilizadas como controle para comparação de resultados (Figura 49). Para isso, são treinadas RNA's com diferentes topologias, porém todas de 3 camadas. As topologias escolhidas foram: 2-4-4-1, 2-8-9-1, 2-18-30-1.

Cada tabela mostra resultados para diferentes configurações utilizadas da Função Tabuleiro para o treinamento da rede. Estas variam conforme o número de “casas”(Coluna *Cs*) e o número de instâncias em cada “casa”(Coluna *Ins*).

Para cada uma dessas configurações os resultados mostrados são obtidos através de uma média aritmética dos resultados individuais de teste das redes treinadas por

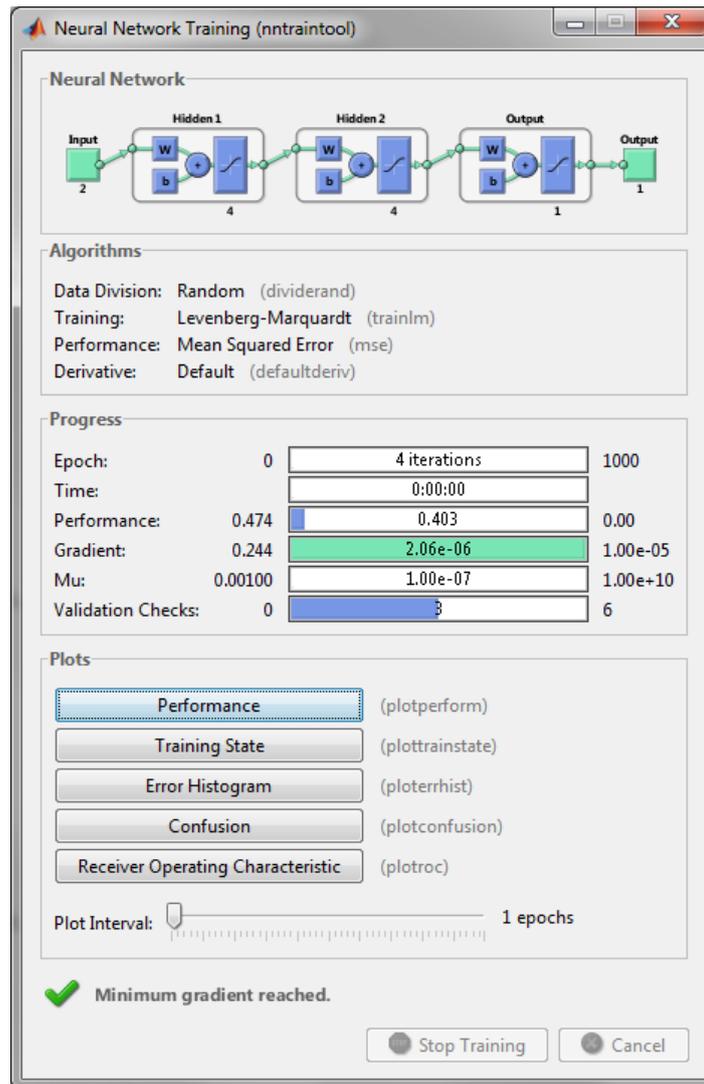


Figura 49: *Toolbox* de Reconhecimento de Padrões no *Matlab*

cinco conjuntos de dados diferentes gerados aleatoriamente (segundo a FT).

A quantidade de neurônios atribuída à cada camada oculta das redes neurais utilizadas para comparação foi escolhida considerando o número de “casas” por coluna e por linha da função tabuleiro. Desta forma, o número de neurônios na primeira camada oculta é dado seguindo a Equação 4.1, que fornece o dobro do número mínimo necessário de neurônios para separar corretamente as regiões (conforme mostrado na Seção 2.1.4).

$$nNeur_1 = 2 \cdot [(nLin - 1) + (nCol - 1)] \quad (4.1)$$

Sendo $nNeur_1$ o número de neurônios na primeira camada oculta, $nLin$ o número de linhas e $nCol$ o número de colunas da função tabuleiro utilizada.

Já o número de neurônios na segunda camada oculta é escolhido de forma a obter o número suficiente para separar cada uma das “casas” conforme foi mostrado na Subseção 2.1.4.1. Assim, o número de neurônios na segunda camada oculta é igual ao número de “casas” geradas.

A Tabela 2 mostra um comparativo de tempo para o treinamento das redes analisadas. Os valores apresentados se referem à uma média dos tempos gastos nos treinamentos realizados com diferentes conjuntos de dados. Percebe-se que os tempos encontrados para o treinamento utilizando o MSGS em todos os casos é menor do que os tempos de treinamento das redes MLP.

Tabela 2: Resultados comparativos de tempo de treinamento com FT (em segundos)

| Cs | Ins | 2-4-4-1 | 2-8-9-1 | 2-18-30-1 | MSGS |
|----|-----|---------|---------|-----------|--------|
| 4 | 5 | 0,4091 | 0,3760 | 0,5291 | 0,0089 |
| | 10 | 0,3875 | 0,3769 | 0,5787 | 0,0100 |
| | 20 | 0,4205 | 0,3826 | 0,6485 | 0,0114 |
| 9 | 5 | 0,4428 | 0,4211 | 0,7381 | 0,0399 |
| | 10 | 0,4887 | 0,4636 | 0,7479 | 0,0401 |
| | 20 | 0,5795 | 0,5077 | 0,8873 | 0,0451 |
| 30 | 5 | 0,5096 | 0,5305 | 1,4850 | 0,3167 |
| | 10 | 0,6317 | 0,6841 | 1,5922 | 0,3144 |
| | 20 | 1,1142 | 1,0987 | 2,7016 | 0,3512 |

Nas Tabelas 3 - 5, o desempenho é mostrado através da taxa de sucesso na classificação para cada configuração utilizada da FT, e para cada rede neural considerada.

Os resultados mostrados na Tabela 3 são referentes aos testes realizados com conjuntos de dados gerados aleatoriamente seguindo a mesma distribuição dos dados de treino. Para cada “casa” são gerados cinquenta pontos.

Na Tabela 4 o conjunto utilizado para teste tem mesma quantidade de “casas”, e mesmo valor de *offset* e *distr* que os dados de treino. Porém não são obtidos aleatoriamente, e sim cobrindo toda a área disponível pela FT considerada, como pode ser visto na Figura 50.

Já os resultados expostos na Tabela 5 utiliza um conjunto de teste com pontos igualmente espaçados ao longo do espaço de soluções como pode ser visto na Figura 51.

Os dois testes cujos resultados se encontram nas Tabelas 4 e 5 se tornam possíveis de realizar por conta da forma como os dados são distribuídos ao longo do espaço ser conhecida matematicamente. A realização destes visa alcançar uma forma de mostrar

a generalização obtida para dados não presentes no treinamento.

Tabela 3: Resultados comparativos com FT - dados aleatórios seguindo mesma distribuição dos dados de treino, 50 pontos por casa

| Cs | Ins | 2-4-4-1 | 2-8-9-1 | 2-18-30-1 | MSGS |
|----|-----|---------|---------|--------------|--------------|
| 4 | 5 | 84,10 | 96,40 | 86,45 | 93,90 |
| | 10 | 98,60 | 87,60 | 96,05 | 99,40 |
| | 20 | 93,70 | 83,20 | 99,95 | 99,90 |
| 9 | 5 | 80,62 | 91,47 | 85,78 | 93,90 |
| | 10 | 82,62 | 95,24 | 98,73 | 98,30 |
| | 20 | 92,22 | 98,40 | 95,09 | 99,90 |
| 30 | 5 | 61,43 | 70,91 | 94,73 | 92,44 |
| | 10 | 61,01 | 85,21 | 97,81 | 98,37 |
| | 20 | 78,56 | 94,19 | 99,09 | 99,64 |

Tabela 4: Resultados Comparativos com FT - dados seguindo distribuição proposta na Figura 50

| Cs | Ins | 2-4-4-1 | 2-8-9-1 | 2-18-30-1 | MSGS |
|----|-----|---------|---------|--------------|--------------|
| 4 | 5 | 81,64 | 95,24 | 84,74 | 92,27 |
| | 10 | 95,78 | 85,97 | 95,31 | 98,14 |
| | 20 | 91,28 | 82,41 | 99,45 | 99,29 |
| 9 | 5 | 78,55 | 88,45 | 83,32 | 92,32 |
| | 10 | 80,08 | 93,09 | 97,41 | 97,23 |
| | 20 | 90,26 | 96,87 | 93,82 | 99,45 |
| 30 | 5 | 60,48 | 69,96 | 92,82 | 90,06 |
| | 10 | 59,98 | 83,44 | 96,27 | 96,99 |
| | 20 | 76,85 | 91,54 | 97,97 | 98,64 |

Tabela 5: Resultados Comparativos com FT - dados seguindo distribuição proposta na Figura 51

| Cs | Ins | 2-4-4-1 | 2-8-9-1 | 2-18-30-1 | MSGS |
|----|-----|---------|--------------|--------------|--------------|
| 4 | 5 | 70,21 | 75,75 | 69,40 | 74,45 |
| | 10 | 75,95 | 70,41 | 76,90 | 77,17 |
| | 20 | 70,10 | 66,08 | 78,81 | 78,25 |
| 9 | 5 | 65,01 | 68,28 | 68,74 | 71,58 |
| | 10 | 64,97 | 69,43 | 72,43 | 73,26 |
| | 20 | 70,27 | 72,55 | 71,96 | 74,25 |
| 30 | 5 | 56,02 | 58,89 | 68,07 | 68,38 |
| | 10 | 55,23 | 64,21 | 71,01 | 70,77 |
| | 20 | 60,78 | 68,42 | 70,65 | 70,94 |

Analisando apenas as redes neurais MLP's, percebe-se que cada topologia leva vantagem em uma configuração da função tabuleiro. Em geral, os melhores resultados

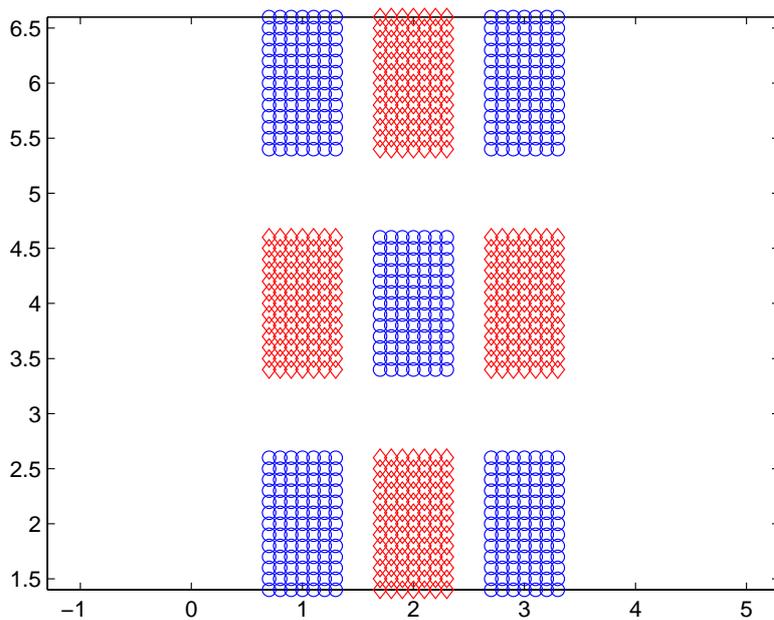


Figura 50: Exemplo de conjunto de teste gerado

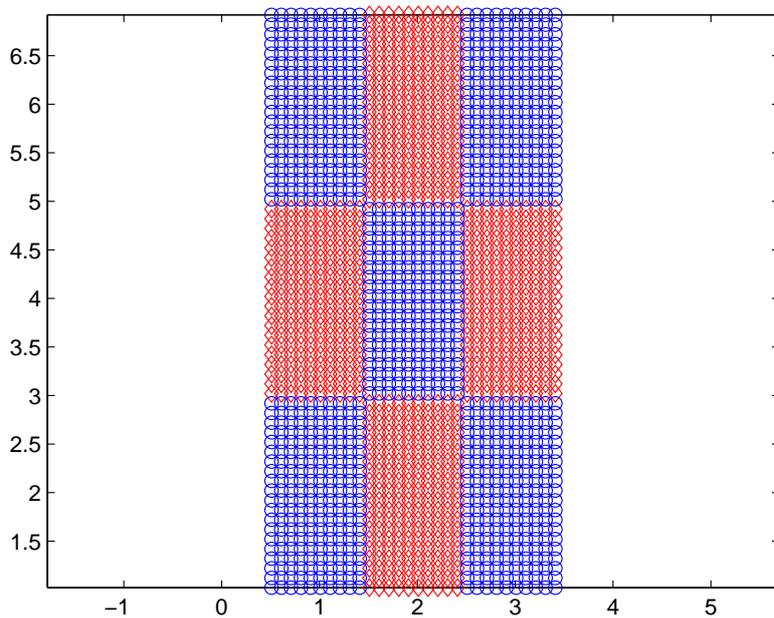


Figura 51: Exemplo utilizado para teste

apresentados pela rede neural 2-4-4-1 são para os testes com 4 casas, e os valores obtidos de taxa de sucesso superam, ou pelo menos são próximos aos obtidos com as redes mais complexas. Já a rede 2-8-9-1 apresenta os melhores resultados nos testes com 9 casas. Para os testes com 30 casas se mostra necessário o uso de redes mais complexas, o que pode ser visto por conta da rede 2-18-30-1 apresentar os melhores resultados.

Com os resultados obtidos nos diferentes testes é possível perceber que o MSGS é capaz de treinar a rede neural atingindo níveis de sucesso na classificação maiores do que os MLP's treinados com o algoritmo Levenberg-Marquardt, na maioria dos casos. O que se deve ao fato de para cada configuração diferente do conjunto de treino, o MSGS se adapta à quantidade necessária de neurônios por camada, evitando assim gerar redes mais complexas do que o necessário, o que pode afetar a capacidade de generalizar ao serem apresentados a dados de teste diferentes dos de treino.

4.2 TESTES UTILIZANDO BANCOS DE DADOS PÚBLICOS

Estes bancos foram escolhidos por serem utilizados na literatura (FUNG; MANGASARIAN, 2005) e (HSU; LIN, 2002). Na Tabela 6 são apresentadas algumas informações destes bancos de dados.

Tabela 6: Descrição dos Bancos Públicos

| | N. Classes | Instâncias | Atributos |
|----------------------------|------------|------------|-----------|
| <i>Iris</i> | 3 | 150 | 4 |
| <i>Glass</i> | 6 | 214 | 9 |
| <i>Wine</i> | 3 | 178 | 13 |
| <i>Vowel</i> | 10 | 528 | 11 |
| <i>Vehicle Silhouettes</i> | 4 | 846 | 18 |

Como dito na Seção 3.4, a metodologia utilizada para a seleção do hiperplano de separação definida aqui refere-se ao ponto central do segmento \mathcal{S} . Também é importante mencionar que a metodologia de trabalho para multiclasse escolhida foi a *one-against-all*.

Na Tabela 7, é apresentado um resumo dos resultados obtidos para todos os bancos de dados utilizando a técnica proposta (MSGS). Foram testados todos os bancos de dados utilizando a metodologia *ten fold*, no qual as instâncias são divididas em 10 conjuntos com 10% da quantidade total de amostras cada, e a cada rodada de treinamento, 9 dos conjuntos são usados para treinar a rede neural, e o outro conjunto é usado para testar a rede. Os melhores resultados são apresentados em negrito.

A análise dos resultados mostrados na Tabela 7 mostra que o método apresentado foi o mais robusto para os casos analisados, com uma melhor taxa de identificação em 3 dos 5 conjuntos testados. Nos casos em que a taxa de acerto do MSGS é menor, o desempenho é próximo aos resultados mostrados nas referências. É interessante

Tabela 7: Resultados Quantitativos

| | MSGS | Hsu | Fung |
|---------|-------------|-------------|--------------|
| | s(%) | s(%) | s(%) |
| Iris | 99,3 | 97,3 | 98,7 |
| Glass | 71,3 | 73,8 | 70,0 |
| Wine | 94,3 | 99,4 | 100,0 |
| Vowel | 99,2 | 99,1 | 98,5 |
| Vehicle | 90,8 | 87,5 | 82,2 |

ressaltar que mesmo tendo apresentado altas taxas de acerto, o algoritmo testado ainda é uma versão inicial do método MSGS, exigindo ainda uma análise mais aprofundada em algumas etapas do processo. Análise essa que tende a melhorar tanto o desempenho computacional quanto quantitativo do MSGS.

4.3 PROBLEMA DA ESPIRAL DUPLA

A espiral dupla é um problema de classificação que apresenta um comportamento altamente não linear de separação entre as classes, frequentemente usado como teste para comparação entre resultados de diferentes algoritmos de treinamento supervisionado e arquiteturas de redes neurais artificiais (SáNCHEZ, 1999).

O conjunto gerado, é composto de 194 amostras pertencentes à duas classes diferentes seguindo a Equação (4.2) para o sistema de coordenadas polar, e conseqüentemente, a Equação (4.3) no sistema de coordenadas cartesianas. Logo, o conjunto de dados pode ser visto na Figura (52).

$$\begin{cases} \theta_i = \frac{\pi}{2} + \frac{\pi i}{16} \\ \rho_i = (-1)^{k-1} \cdot \frac{1}{3}\theta_i \end{cases} \quad (4.2)$$

sendo $i = 0, 1, \dots, 96$ referente à amostra, e $k = 1, 2$ referente à classe da amostra.

$$\begin{cases} x_i = \rho_i \cdot \cos(\theta_i) \\ y_i = \rho_i \cdot \text{sen}(\theta_i) \end{cases} \quad (4.3)$$

Apesar de, em teoria, ser possível resolver este problema de classificação utilizando um MLP com uma camada oculta, em geral redes treinadas com *backpropagation* não apresentam resultados satisfatórios (DAQI; HAO; YUNFAN, 2006).

O Método de Segmentações Geométricas Sucessivas se mostrou capaz de resolver o problema da espiral dupla separando as duas classes. O agrupamento inicial dos

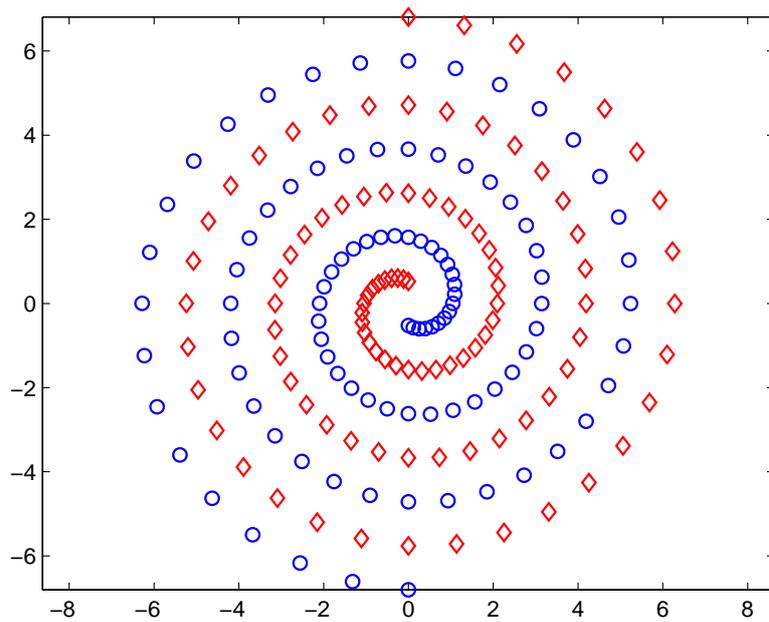


Figura 52: Conjunto para o problema da espiral dupla

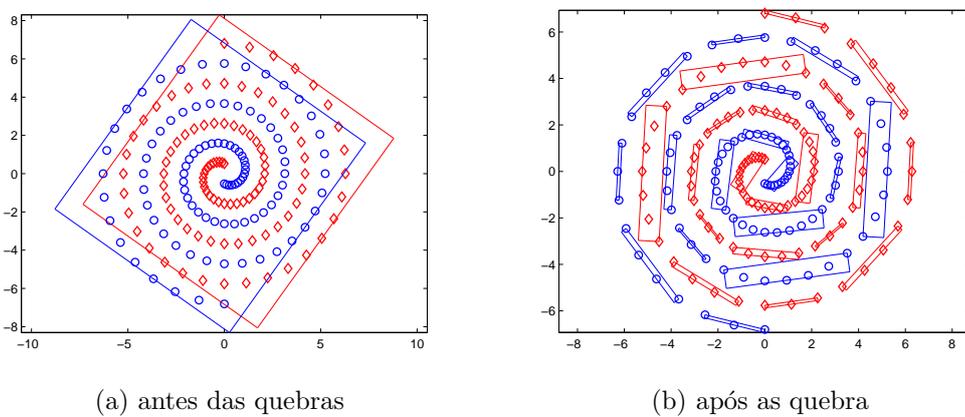


Figura 53: Hipercaixas no problema da espiral dupla

dados em hipercaixas pode ser visto na Figura 53a, e após 38 épocas, o conjunto final de hipercaixas encontradas é visto na Figura 53b, que resulta nas regiões classificadas mostradas na Figura 54.

Para a classificação através do MSGS foram encontrados 36 hiperplanos separando 37 hipercaixas, resultando em uma RNA 2-36-37-2-2.

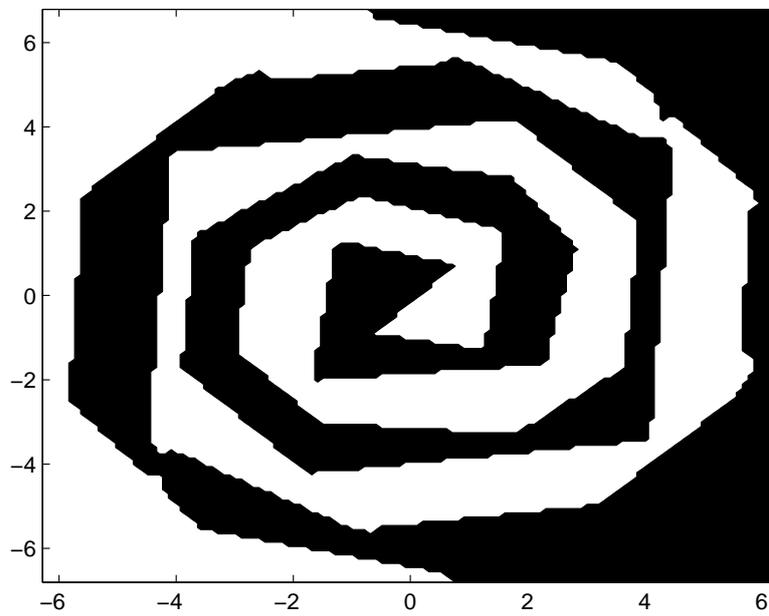


Figura 54: Regiões de separação no problema da espiral dupla pelo MSGS

5 CONCLUSÕES

5.1 CONSIDERAÇÕES FINAIS

O trabalho mostrou uma técnica para treinamento de redes neurais artificiais multicamadas, denominada Método de Segmentações Geométricas Sucessivas (MSGs). O Método possibilita a geração da rede neural sem especificação de topologia, o que permite a criação de RNA's mais adequadas à cada problema, evitando a utilização de redes mais complexas que o necessário.

Através dos desenvolvimentos mostrados neste trabalho, foi possível a publicação de dois trabalhos em congressos, o primeiro no XI Congresso Brasileiro de Inteligência Computacional (CBIC) intitulado "Otimização do Processo de Construção de Redes Neurais Artificiais com o Método de Segmentações Geométricas Sucessivas Utilizando Projeções de Estimador de Densidade para Quebra em Hipervolumes.- (MACHADO, L. C. N. ; HONÓRIO, L. de M. ; CERQUEIRA, A. S.). O segundo no XI Simpósio Brasileiro de Automação Inteligente (SBAI) intitulado "Diagnóstico de Falhas em Transformadores de Potência Através do Método de Segmentações Geométricas Sucessivas."(VALENTE, W. A. G. ; OLIVEIRA, E. J. ; HONÓRIO, L. de M. ; MACHADO, L. C. N.).

Apesar de em alguns resultados, as RNA's resultantes do MSGs apresentarem grande número de neurônios, pelo fato de ser uma rede parcialmente conectada, os modelos gerados não possuem grande complexidade.

Baseado nos resultados apresentados na Seção anterior, alguns aspectos podem ser destacados:

- O *MSGs* é capaz de definir a estrutura, quantidade de neurônios por camada e pesos das conexões de uma rede neural de forma analítica, baseando-se apenas nos dados de entrada e saída, sem a necessidade da especificação de parâmetros topológicos;
- O fato dos neurônios serem obtidos de forma analítica possibilita ainda uma ade-

quação da rede para priorizar diferentes pontos de atuação. Sendo os pesos da primeira camada de neurônios relativos a um hiperplano, este hiperplano pode facilmente ser movido ao longo do espaço de soluções se adaptando melhor às características de um dado problema.

- O método não exige o fornecimento da topologia da rede a ser treinada, e ainda atinge taxas de acerto elevadas comparando com os outros testes efetuados.

- O MSGS se mostra capaz de treinar redes neurais que alcançam altos níveis de desempenho, apta a aplicar o conhecimento adquirido dos dados de treinamento de forma generalista.

5.2 TRABALHOS FUTUROS

- Uma perspectiva positiva é a possibilidade de aplicação de outras técnicas para otimizar o processo e melhorar o desempenho do classificador possibilitando treinar redes não supervisionadas;

- O método de otimização de hiperplanos apresentado foi heurístico e ainda assim os resultados provenientes foram satisfatórios. No entanto, a metodologia permite a utilização de um otimizador discreto;

- A nova metodologia proposta abre espaço para aplicação com outras técnicas já existentes, tais como clusterização e otimização;

- Pode ser adaptada para saídas contínuas abrindo ainda mais a possibilidade de novas aplicações.

- Trabalhos futuros serão realizados buscando separar hipercaixas que não são linearmente separáveis evitando a quebra excessiva das mesmas. Diminuindo assim a construção de redes neurais muito especialistas.

REFERÊNCIAS

- ABE, S. *Support vector machines for pattern classification*. [S.l.]: Springer-Verlag New York Inc, 2010.
- ARCHAMBEAU, C. et al. Assessment of probability density estimation methods: Parzen window and finite gaussian mixtures. In: *Circuits and Systems, 2006. ISCAS 2006. Proceedings. 2006 IEEE International Symposium on*. [S.l.: s.n.], 2006. p. 4 pp.–.
- ASUNCION, A.; NEWMAN, D. Uci machine learning repository [<http://www.ics.uci.edu/~mllearn/mlrepository.html>]. irvine, ca: University of california. *School of Information and Computer Science*, 2007.
- BEALE, R.; JACKSON, T. *Neural computing: an introduction*. [S.l.]: Hilger, 1990. ISBN 9780852742624.
- BISHOP, C. M. *Neural Networks for Pattern Recognition*. [S.l.: s.n.], 2005.
- BISHOP, C. M. *Pattern Recognition and Machine Learning*. [S.l.: s.n.], 2006.
- BOSE, N.; GARGA, A. Neural network design using voronoi diagrams. *Neural Networks, IEEE Transactions on, IEEE*, v. 4, n. 5, p. 778–787, 1993.
- BOURG, D. *Physics for game developers*. [S.l.]: O’Reilly Media, 2002.
- CESA-BIANCHI, N.; GENTILE, C.; ZANIBONI, L. Incremental algorithms for hierarchical classification. *The Journal of Machine Learning Research, JMLR. org*, v. 7, p. 31–54, 2006.
- CHANG, J.; WANG, W.; KIM, M. Efficient collision detection using a dual obb-sphere bounding volume hierarchy. *Computer-Aided Design, Elsevier*, v. 42, n. 1, p. 50–57, 2010.
- CHEN, Y.; DAMPER, R.; NIXON, M. On neural-network implementations of k-nearest neighbor pattern classifiers. *Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions on, IEEE*, v. 44, n. 7, p. 622–629, 1997.
- DAQI, G.; HAO, L.; YUNFAN, Y. Task decomposition and modular perceptrons with sigmoid activation functions for solving the two-spirals problem. In: *Networking, Sensing and Control, 2006. ICNSC ’06. Proceedings of the 2006 IEEE International Conference on*. [S.l.: s.n.], 2006. p. 218–223.
- FAUSETT, L. *Fundamentals of Neural Networks: Architectures, Algorithms, and Applications*. [S.l.]: Prentice Hall, 1994.
- FUNG, G.; MANGASARIAN, O. Multicategory proximal support vector machine classifiers. *Machine Learning, Springer*, v. 59, n. 1, p. 77–97, 2005.

- GALUSHKIN, A. I. *Neural Networks Theory*. [S.l.: s.n.], 2007.
- GENTILE, C.; SZNAIER, M. An improved voronoi-diagram-based neural net for pattern classification. *Neural Networks, IEEE Transactions on*, IEEE, v. 12, n. 5, p. 1227–1234, 2001.
- GOTTSCHALK, S. *Collision queries using oriented bounding boxes*. Tese (Doutorado) — Citeseer, 2000.
- GOTTSCHALK, S.; LIN, M.; MANOCHA, D. Obbtree: a hierarchical structure for rapid interference detection. In: ACM. *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*. [S.l.], 1996. p. 171–180.
- HAMEL, L. *Knowledge discovery with support vector machines*. [S.l.]: Wiley Online Library, 2009.
- HAN, J.; KAMBER, M.; PEI, J. *Data mining: concepts and techniques*. [S.l.]: Morgan Kaufmann Pub, 2011.
- HAYKIN, S. *Neural Networks: A comprehensive Foundation*. [S.l.]: Prentice Hall, 1999.
- HEBB, D. O. *The Organization of Behavior: A Neuropsychological Theory*. [S.l.]: McGill University, 1949.
- HONÓRIO, L. M. et al. Ambiente de manufatura virtual 3d: Desenvolvimento de equipamentos e testes de colisões em tempo real. In: CBA2006. *XVI Congresso Brasileiro de Automática*. [S.l.], 2008. p. 911–916.
- HÄRDLE, W. et al. *Nonparametric and Semiparametric Models: An Introduction*. [S.l.: s.n.], 2005.
- HSU, C.; LIN, C. A comparison of methods for multiclass support vector machines. *Neural Networks, IEEE Transactions on*, IEEE, v. 13, n. 2, p. 415–425, 2002.
- HUANG, H.; FENG, G.; CAO, J. Robust state estimation for uncertain neural networks with time-varying delay. *Neural Networks, IEEE Transactions on*, IEEE, v. 19, n. 8, p. 1329–1339, 2008.
- JAMES, D.; PAI, D. Bd-tree: output-sensitive collision detection for reduced deformable models. In: ACM. *ACM Transactions on Graphics (TOG)*. [S.l.], 2004. v. 23, n. 3, p. 393–398.
- LEUNG, Y. et al. A rough set approach for the discovery of classification rules in interval-valued information systems. *International Journal of Approximate Reasoning*, Elsevier, v. 47, n. 2, p. 233–246, 2008.
- LEVENBERG, K. A method for the solution of certain problems in least squares. *Quart. Appl. Math.*, v. 2, p. 164–168, 1944.
- MACHADO, L. C. N.; HONÓRIO, L. de M.; CERQUEIRA, A. S. Otimização do processo de construção de redes neurais artificiais com método de segmentações geométricas sucessivas utilizando projeções de estimador de densidade para quebra em hipervolumes. *11º Congresso Brasileiro de Inteligência Computacional - CBIC*, 2013.

MACLEOD, C. *The synthesis of Artificial Neural Networks using Single String Evolutionary Techniques*. Tese (Doutorado) — School of Electronic and Electrical Engineering, 1999.

MARQUARDT, D. W. An algorithm for least-square estimation of nonlinear parameters. *Quart. Appl. Math.*, v. 11, p. 431–441, 1963.

MURPHY, K. P. *Machine Learning: A probabilistic Perspective*. [S.l.]: the MIT Press, 2012.

PACKTER, I. *Neurônio biológico*. 2011. Disponível em: <<http://www.filosofiaadistancia.com.br/gravações%20agosto/Neurociências/Neurociência%20%20apostila%20I.htm>>. Acesso em: 29 out. 2013.

PERETTO, P. *An Introduction to the Modeling of Neural Networks*. [S.l.]: Cambridge University Press, 1992.

SILVA, A. Alves da; FERREIRA, V.; VELASQUEZ, R. Input space to neural network based load forecasters. *International Journal of Forecasting*, Elsevier, v. 24, n. 4, p. 616–629, 2008.

SÁNCHEZ, J. R. Álvarez. Injecting knowledge into the solution of the two-spiral problem. *Neural Computing & Applications*, Springer-Verlag London Limited, v. 8, n. 3, p. 265–272, 1999. ISSN 0941-0643. Disponível em: <<http://dx.doi.org/10.1007/s005210050029>>.

XIANG, C.; DING, S.; LEE, T.-H. Geometrical interpretation and architecture selection of mlp. *Neural Networks, IEEE Transactions on*, v. 16, n. 1, p. 84–96, 2005. ISSN 1045-9227.

APÊNDICE A - INTERPRETAÇÃO GEOMÉTRICA DE UM NEURÔNIO

Os neurônios de uma rede neural artificial possuem uma interpretação geométrica bem definida. Seja \vec{x}^1 e \vec{x}^2 pontos no espaço \mathbb{R}^2 , e com \vec{w} um vetor de escalares também em \mathbb{R}^n e $b \in \mathbb{R}$. Sendo $\vec{x}^1 \neq \vec{x}^2$ sinais de entrada no neurônio é possível afirmar a existência de pelo menos um vetor de pesos \vec{w} e um escalar b que produzirão saídas de sinais diferentes, como mostrado nas Equações A.1 e A.2.

$$\text{sgn}(\vec{x}^1 \cdot \vec{w} + b) \neq \text{sgn}(\vec{x}^2 \cdot \vec{w} + b) \quad (\text{A.1})$$

onde para $z \in \mathbb{R}$

$$\text{sgn}(z) = \begin{cases} +1, & \text{se } z > 0, \\ -1, & \text{se } z < 0 \\ 0, & \text{se } z = 0 \end{cases} \quad (\text{A.2})$$

Considerando \vec{w} e b pesos atribuídos à entrada de um neurônio com uma função de ativação representada pela Equação A.2, sendo propriamente ajustados, \vec{w} e b irão gerar um hiperplano entre \vec{x}^1 e \vec{x}^2 de forma que a função $f(\vec{w}, b, \vec{x}) \in \mathbb{R}$ dada por

$$f(\vec{w}, b, \vec{x}) = \text{sgn}(\vec{x} \cdot \vec{w} + b) \quad (\text{A.3})$$

o que representa uma superfície de separação e irá possuir sinais diferentes para \vec{x}^1 e \vec{x}^2 . Generalizando, qualquer conjunto de pontos $\vec{x}^k \in \mathbb{R}^n$ que estejam na mesma direção em relação à normal do hiperplano $\mathcal{X}(\vec{w}, b, \vec{x})$ dado por

$$\mathcal{X}(\vec{w}, b, \vec{x}) = \vec{x} \cdot \vec{w} + b \quad (\text{A.4})$$

irão apresentar o mesmo valor de $f(\mathcal{X}(\vec{w}, b, \vec{x}))$.

Desta forma, não considerando a parte topológica das diversas camadas, pode-se dizer sem perda de generalidade que a união de dois ou mais neurônios gera uma rede

neural. Expandindo o raciocínio anterior, um conjunto de neurônios com seus pesos devidamente calculados pode gerar uma saída que identifica pontos em uma região específica do espaço de estados. A referência (BOSE; GARGA, 1993) demonstra esta característica mostrando que dois ou mais conjuntos de pontos podem ser separados por um conjunto de hiperplanos. Se for possível então agrupar os dados de entrada em conjuntos e encontrar esses planos de maneira geométrica é possível criar uma rede neural que classifique-os corretamente, como demonstrado em (GENTILE; SZNAIER, 2001), (CHEN; DAMPER; NIXON, 1997) e (HAMEL, 2009).

APÊNDICE B - FUNÇÃO TABULEIRO

```

input :  $NIns$ ,  $NCasasx$ ,  $NCasasy$ ,  $\delta$ ,  $Distr$ 
output:  $SET_1$ ,  $SET_2$ 
 $n_1 = 0$ ,  $n_2 = 0$ ;
for  $k \leftarrow 1$  to  $NInst$  do
  for  $i \leftarrow 1$  to  $NCasasx$  do
    for  $j \leftarrow 1$  to  $NCasasy$  do
       $conjunto = \text{mod}(i + j, 2)$ 
       $dx = -Distr + 2 \times Distr \times \text{RANDOM}$ 
       $dy = -Distr + 2 \times Distr \times \text{RANDOM}$ 
      if  $conjunto == 1$  then
         $n_1 = n_1 + 1$ ;
         $SET_1(n_1, :) = [i, \delta \times j] + [dx, \delta \times dy]$ ;
      else
         $n_2 = n_2 + 1$ ;
         $SET_2(n_2, :) = [i, \delta \times j] + [dx, \delta \times dy]$ ;
      end
    end
  end
end

```

A Figura 55 ilustra a construção de um conjunto de dados seguindo a função tabuleiro. Os valores de i se referem ao eixo das abcissas, enquanto j fazem menção ao eixo das ordenadas, desta forma, as casas com menor abcissa têm $i = 1$, e as de menor ordenada têm $j = 1$.

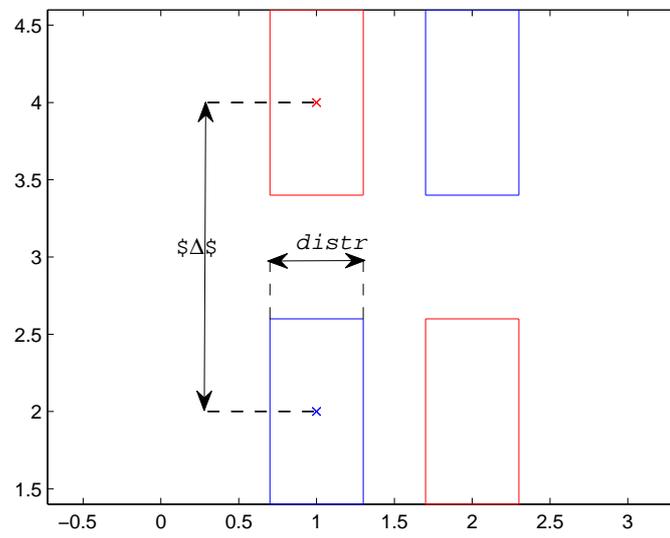


Figura 55: Função tabuleiro