UNIVERSIDADE FEDERAL DE ITAJUBÁ
PROGRAMA DE PÓS-GRADUAÇÃO
EM ENGENHARIA ELÉTRICA

RECONHECIMENTO DE PADRÕES UTILIZANDO O TEOREMA DO EIXO DE SEPARAÇÃO E REDES NEURAIS ARTIFICIAIS

RODRIGO MAXIMIANO ANTUNES DE ALMEIDA

ITAJUBÁ, MARÇO DE 2009

UNIVERSIDADE FEDERAL DE ITAJUBÁ
PROGRAMA DE PÓS-GRADUAÇÃO
EM ENGENHARIA ELÉTRICA

RODRIGO MAXIMIANO ANTUNES DE ALMEIDA

RECONHECIMENTO DE PADRÕES UTILIZANDO O TEOREMA DO EIXO DE SEPARAÇÃO E REDES NEURAIS ARTIFICIAIS

Dissertação submetida ao programa de Pós-Graduação em Engenharia Elétrica como parte dos requisitos para obtenção do Título de Mestre em Ciências em Engenharia Elétrica.

Área de Concentração: Automação e Sistemas Industriais.

Orientador: Prof. Dr. Leonardo de Mello Honório.

MARÇO DE 2009
ITAJUBÁ – MG\

Ficha catalográfica elaborada pela Biblioteca Mauá –
Bibliotecária Margareth Ribeiro- CRB_6/1700

A447r

Almeida, Rodrigo Maximiniano Antunes de
Reconhecimento de padrões utilizando o teorema do eixo de se_
paração e redes neurais artificiais / Rodrigo Maximiniano Antunes
de Almeida. -- Itajubá, (MG) : [s.n.], 2009.
55 p. : il.

Orientador: Prof. Dr. Leonardo de Mello Honório.
Dissertação (Mestrado) – Universidade Federal de Itajubá.

1. Redes neurais artificiais. 2. Teorema de eixos de separação. 3.
Classificação de padrões. I. Honório, Leonardo de Mello, orient. II.
Universidade Federal de Itajubá. III. Título.

CDU 004.032.26(043)

Dedicatória

Dedico este trabalho primeiramente a Deus, por me ter concedido a vida e as demais graças que me permitiram chegar aqui, e aos meus pais, Paulo e Carminha, pelo exímio exemplo de vida, sabedoria, retidão e amor.

Agradecimentos

Aos meus pais, por todo o apoio e incentivo para meus estudos.

Aos meus irmãos, Marcela e Daniel, simplesmente por fazerem parte da minha vida.

À Ana Paula Siqueira Silva, antes de tudo, pela constante presença, que a distância não pôde remover.

A todos os amigos, pelo incentivo e apoio demonstrados durante este tempo.

Aos colegas do CRTI, pela paciência e companheirismo, nas duvidas interessantes e, principalmente, nas não tão interessantes.

Ao Prof. Dr. Leonardo de Mello Honório, pelo convite, orientação e todo tempo disponibilizado para realização deste trabalho.

A todos aqueles que, direta ou indiretamente, colaboraram para que este trabalho conseguisse atingir os objetivos propostos.

"Entre todas as verdadeiras buscas humanas, a busca pela sabedoria é a mais perfeita, a mais sublime, a mais útil e a mais agradável"

São Tomás de Aquino

Resumo

Este trabalho apresenta uma nova técnica de treinamento de redes neurais artificiais baseado em técnicas geométricas. Este método tem a capacidade de gerar a topografia da rede sem necessidade de especificação de parâmetros. Técnicas similares, baseadas em fatores geométricos, vêm sendo desenvolvidas com altas taxas de acerto e boa capacidade de generalização. Neste trabalho adaptamos o teorema de eixos de separação para espaços n-dimensionais, expandindo o conceito de OBB-trees para OBHB-trees. O algoritmo é composto de quatro etapas: construção das OBHB's, busca dos hiperplanos de separação usando o teorema dos eixos de separação, seleção do melhor conjunto de hiperplanos e especificação da rede neural. Resultados experimentais demonstram que esta técnica pode ser utilizada com sucesso em problemas de classificação.

Palavras-chave: Redes neurais artificiais, teorema de eixos de separação, classificação de padrões.

Abstract

This work presents a new technique of artificial neural networks training based on geometric techniques. It is capable of generating the network topography without the necessity of parameters specification. Similar approaches, based on geometric factors, are being developed with high hit rate and generalization capacity. In this work we adapted the axis separation theorem for n-dimensional spaces, expanding the concept of OBB-trees to OBHB-trees. The algorithm is composed of four steps: OBHB's construction, search for the separation hiperplanes using axis separation theorem, select the best hiperplanes set and neural net specification. Experimental results show that this technique could be successful used in classification problems.

Keywords: Artificial neural nets, axis separation theorem, patterns classification.

Lista de figuras

Lista de tabelas

Lista de gráficos

Lista de abreviaturas

RNA	–	Rede Neural Artificial
AABB	–	*Axis Align Bounding Boxes*
OBB	–	*Oriented Bounding Boxes*
OBHB	–	*Oriented Bounding Hiperboxes*
SCL	–	Sistema de Coordenadas Local
SCG	–	Sistema de Coordenadas Global
SVM	–	*Support Vector Machine*
Id	–	Identificação
UR	–	*Uncertain Region*
MDL	–	Maximização de Distância entre Limites
MBC	–	Minimização de BiClassificação

Sumário

1 Introdução

O processo de classificação de objetos é uma necessidade real presente em diversas áreas(Brudzewski, et al., 2003) (Ding, et al., 2001) (Gentile, et al., 2001) (Lippmann, 1989). Ao longo da história diversas técnicas surgiram, mas a maior dificuldade era garantir uma taxa de acerto aceitável quando considerados os erros advindos dos dados utilizados. Esse problema foi em parte solucionado pelas técnicas de inteligência artificial, por heurísticas específicas para cada caso, abordagens para melhoria das capacidades de generalização e tratamento dos dados de entrada.

Duas abordagens são bastante utilizadas atualmente: uma baseada em técnicas de classificação geométricas, como *support vector machines* ou SVM (Chih, et al., 2002) (Fung, et al., 2005) (Gentile, et al., 2001), e a segunda em teorias de inteligência artificial, como as redes neurais artificiais(Lippmann, 1989) (Chen, et al., 1997) (Bose, et al., 1993).

Ambas abordagens possuem vantagens e desvantagens. SVM consegue altas taxas de acerto e tem uma boa fundamentação matemática, mas pode não ser adequada para problemas multi-classe. As redes neurais artificiais podem ser aplicadas em praticamente todos os problemas de classificação, sendo mais adequadas para sistemas multi-classes, mas possuem diversos parâmetros que precisam ser especificados à priori, principalmente a topologia da rede(Mukkamala, et al., 2002) (Beale, et al., 1990).

Este trabalho visa apresentar uma técnica para determinação de uma rede neural utilizando técnicas de segmentação geométricas de maneira automática. Para isto foi realizada uma adaptação do teorema de eixos de separação, já utilizado para detecção de colisões em espaços bi e tridimensionais(S Gosttchalk, 1996) (Gottschalk, 2000), para espaços N-dimensionais.

Para realização desta proposta dividimos o procedimento em quatro etapas. A primeira é responsável pela construção dos objetos virtuais que representarão as classes. Estes objetos foram definidos como hipercaixas pelas semelhanças que trazem com as caixas tridimensionais.

A segunda etapa consiste na busca dos hiperplanos de segmentação. Utilizando o teorema dos eixos de separação é possível encontrar o hiperplano, ou conjunto de hiperplanos que separem corretamente as classes.

Na terceira etapa realizamos uma redução no conjunto de hiperplanos encontrados. Pelos experimentos realizados é possível ver que para realizar a correta classificação das classes geralmente não são necessários todos os planos que foram encontrados.

A quarta etapa engloba a especificação e geração da rede neural que agrega todas as características encontradas nas etapas anteriores. Sua estrutura é definida apenas pelos hiperplanos encontrados, não exigindo a especificação de nenhum parâmetro.

Neste trabalho será apresentado brevemente a fundamentação teórica sobre redes neurais artificiais e testes de colisão no Capítulo 2.

O Capítulo 3 trata do desenvolvimento da técnica proposta OBHB-Trees, utilizando os conceitos definidos anteriormente.

No Capítulo 4 são apresentados os resultados obtidos com bancos de dados reais. São apresentados também os melhores resultados obtidos na literatura pesquisada para os mesmos bancos de dados.

O Capítulo 5 encerra o trabalho citando os resultados obtidos e apresentando futuras melhorias a serem implementadas.

2 Revisão bibliográfica

2.1 Redes neurais artificiais

Redes Neurais Artificiais (RNA's) são sistemas desenvolvidos com a intenção de solucionar problemas de uma maneira mais próxima possível àquela realizada pelo cérebro humano.

Os primeiros trabalhos nesta área foram originados do artigo publicado por McCulloch e Pitts em 1943 (McCulloch, et al., 1943), no qual estes propuseram um modelo de neurônio simplificado e de uma rede neural que imitasse alguns comportamentos do cérebro humano. Este modelo surgiu baseado nas maquinas de Turing, conforme apresenta Guatiero Piccinini (Piccinini, 2004) e foi a base dos estudos de Von Neumann para a criação de uma das primeiras arquiteturas de processadores. Apesar de ter tido um desenvolvimento inicial rápido, as RNA's só começaram a ter uso prático na década de 1980, em parte por causa do desenvolvimento do sistema de aprendizagem por retro-propagação de erro, mais conhecido como *back propagation*, e em parte pela crescente capacidade de processamento dos computadores na época.

As redes neurais podem ser caracterizadas como modelos computacionais que possuem propriedades particulares como a habilidade de se adaptar ou aprender, generalizar ou de organizar/classificar dados, e cuja operação é baseada num processamento paralelo(Beale, et al., 1990).

Uma das características mais importantes de uma rede neural é a sua capacidade de generalização. Ela é capaz de extrair as características de um banco de dados e utilizar esse "conhecimento" para inferir uma decisão ou resultado sobre um objeto não conhecido previamente.

2.1.1 Neurônio

A unidade básica das RNA's é o neurônio artificial. Este procura imitar o funcionamento de um neurônio biológico, ou seja, uma excitação é produzida no axônio (saída y) apenas se nos detritos (entradas x_i) houver excitação suficiente. Um modelo simples de neurônio com apenas duas entradas é apresentado na Figura 1.

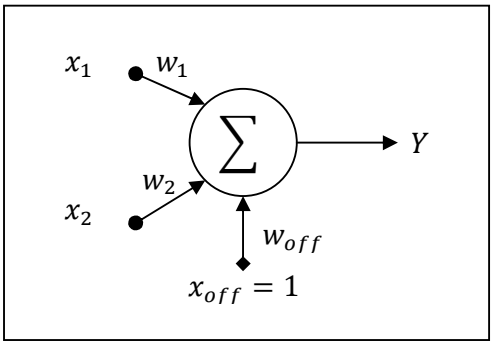


Figura 1 – Exemplo de neurônio artificial

Os neurônios são interligados através de conexões, que podem ser mais ou menos intensas. Isto é representado matematicamente por um peso w_i associado a cada conexão. Existe uma conexão, geralmente chamada de *offset* ou *bias*, que não esta conectada a nenhum outro neurônio e serve como um sistema de polarização para aquele. Pode ser representada por uma conexão com um peso w_{off} ligada a um neurônio sempre ativado com $x_{off} = 1$.

$$Y = F\left(\left(\sum_{i=1}^{n} x_i * w_i\right) + 1 * w_{off}\right) \quad \textbf{Eq. 1}$$

Onde $F(.)$ é a função de ativação.

2.1.2 Funções de ativação

As funções de ativação indicam para o neurônio qual é seu comportamento a respeito dos limiares necessários para ativação. Estas funções indicam qual o valor de entrada necessário para conseguir excitar a saída. Esta excitação pode ser discreta (verdadeiro/falso, zero/um), ou possuir um comportamento contínuo, ou seja, a excitação na saída é proporcional ao somatório das entradas multiplicadas pelos respectivos pesos.

As funções mais comumente utilizadas são a limiar, a pseudo-linear e a sigmoide representadas na Figura 2 (Beale, et al., 1990). A escolha destas funções depende da aplicação a qual será utilizada a RNA.

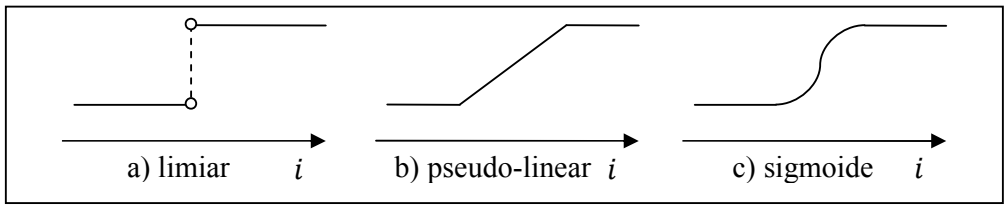


Figura 2 – Tipos de funções de ativação

2.1.3 Topologias das redes

Existem basicamente dois tipos de topologias, a direta (*feed-forward*) e a recursiva.

Na topologia *feed-forward* os "impulsos neurais" não retornam a um neurônio que já passou, ou seja, a saída de um neurônio não pode ser utilizada como a entrada de nenhum neurônio que contribua direta ou indiretamente para uma de suas entradas.

Normalmente essa topologia é estruturada através de camadas. Na Figura 3 podemos visualizar uma rede *feed-forward* com uma camada de entrada, uma camada intermediária e uma de saída. É comum omitir os neurônios da camada de entrada, como pode ser visto na Figura 4, pois estes não realizam nenhum processamento.

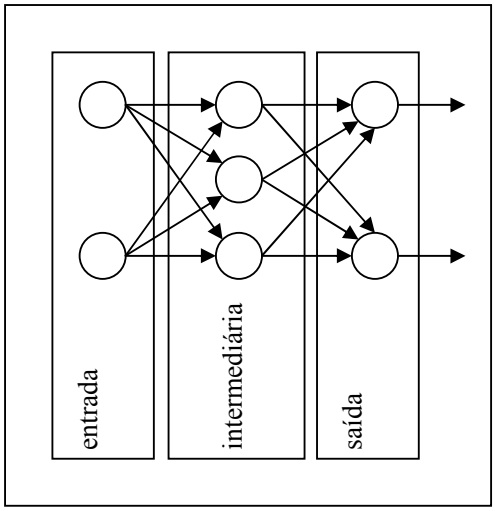


Figura 3 – Exemplo de RNA com topologia feed-forward

Nas estruturas recursivas os neurônios possuem ligações que retornam a informação destes para suas entradas de forma direta ou indireta, como pode ser visto na Figura 4.

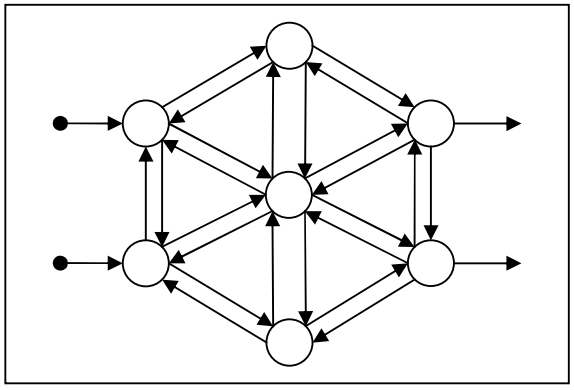


Figura 4 – Exemplo de RNA com topologia recursiva

Na estrutura recursiva, além do valor atual das entradas, os valores antigos de cada neurônio são utilizados no processamento. Desse modo os valores passados influenciam na saída atual. Por isso pode ser necessário que se efetue o processamento várias vezes até que as saídas se estabilizem. A rede de Hopfield é um exemplo de rede recursiva(Beale, et al., 1990).

Embora existam diversas arquiteturas disponíveis para utilização numa rede neural, nenhuma delas estipula a quantidade mínima de neurônios a serem utilizados.

Para o caso das redes *feed-forward*, é possível demonstrar que são necessárias no máximo 3 camadas de neurônios (entrada, intermediária e saída) para que a rede consiga aproximar qualquer tipo de função, isto é conhecido como teorema de Kolgomorov (Beale, et al., 1990). Apesar disto, não existe maneira determinística para definir a quantidade ótima de neurônios necessários na camada intermediária o bom funcionamento da rede. Algumas abordagens foram desenvolvidas com este intuito, obtendo apenas sucessos parciais (Mirchandani, 1989).

2.1.4 Treinamento

Segundo Hebb, "quando um axônio de uma célula A, está perto o suficiente para excitar a célula B é repetitivamente ou persistentemente a dispara, algum processo de crescimento ou mudança metabólica toma parte em uma ou nas das células de forma que a eficiência de A, em ativar a célula B, é aumentada" (Beale, et al., 1990). Este processo de ajuste na eficiência das sinápses dos neurônios, ou no caso das RNA's, o peso das conexões, é chamado treinamento.

O processo de treinamento nas RNA's pode ser dividido em dois tipos, o treinamento supervisionado, quando queremos treinar a rede com um banco de dados, e o não supervisionado, muito utilizado para agrupamento de objetos. Nos treinamentos supervisionados a rede é treinada para responder de acordo com uma série de casos, ou seja, dado uma entrada X_i a saída deve ser Y_i.

Um dos primeiros algoritmos utilizados foi a regra de Hebb. Dado uma conexão w_{ij} que liga o neurônio n_i ao neurônio n_j, o peso desta conexão é ajustado segundo a Eq. 2, onde o delta é calculado pela Eq. 3.

$$w_{ij}^{novo} = w_{ij}^{antigo} + \Delta w_{ij} \quad \textbf{Eq. 2}$$

$$\Delta w_{ij} = r * s_i * s_j \quad \textbf{Eq. 3}$$

Onde r: é a taxa de aprendizado, normalmente no intervalo]0,1];

w_{ij}: peso do arco que leva informação do neurônio n_i para o n_j;

s_x: valor na saída do neurônio n_x;

Este algoritmo tem diversas limitações: só pode ser utilizado em redes sem camadas intermediárias e com funções de ativação lineares.

A regra de Delta proposta em 1959 por Widrow e Hoff, retira a necessidade de se trabalhar com funções lineares embora ainda não possa ser utilizada em redes com mais de uma camada. Os pesos das conexões são atualizados a cada iteração de acordo com o erro entre o valor esperado para aquele neurônio e o valor calculado conforme Eq. 4.

$$w_{ij}^{novo} = w_{ij}^{antigo} + \Delta w_{ij}$$
$$\Delta w_{ij} = r * e * s_i \quad \textbf{Eq. 4}$$
$$e = s_{correto} - s_{calculado}$$

O problema de utilizar esse algoritmo em redes com camadas intermediárias é a impossibilidade de calcular o erro pertinente aos neurônios intermediários, pois o banco de dados contempla apenas os valores das entradas e das saídas da rede.

Na década de 80, Rumelhart, Hinton e Williams, apresentaram um artigo onde explicavam a regra de Delta Generalizada ou *backpropagation*, o que possibilitou então o treinamento de redes multicamadas com funções não lineares. Apesar deste ser considerado o ponto inicial do uso das redes neurais multicamadas, a teoria fora apresentada em 1974 por Paul Werbos em sua dissertação de Ph.D. intitulada "*Beyound Regression*" e de maneira independente por David Parker e David Rumelhart em 1982 (Chauvin, et al., 1995).

O algoritmo *backpropagation* corrige os pesos das conexões sinápticas utilizando o método do gradiente descendente, onde a quantidade de pesos define o tamanho do espaço de busca e o intuito é minimizar a função erro entre as saídas conhecidas e as calculadas. Este procedimento é apresentado no Algoritmo 1.

Algoritmo 1 – Treinamento em Backpropagation

- Dada uma entrada, calcula-se as saídas de acordo com a rede neural montada.
- Calcula-se o erro $e_s = (s_{real} - s_{calculado}) * F'(s_{calculado})$ para cada saída;

- Retroage-se o erro e_s de cada o neurônio n_s da saída, para os neurônios n_i da camada intermediária, que estão ligados à n_s através de um peso w_{is}, onde s_{n_i} é a saída do neurônio da camada intermediária e $e_{is} = e_s * \left(F'\left(s_{n_i}\right)\right) * w_{is}$ é o erro do neurônio n_i;
- Utiliza-se o somatório de todos os erros e_i retroagidos para computar o erro $e_j = \sum_{i=0}^{k} e_{ji}$,onde k é o numero de neurônios que recebem informação de n_j;
- Atualiza-se os valores de w_{ij} segundo a fórmula de delta;
- Repete-se o procedimento para cada um dos valores do banco de dados.

A Tabela 1 fornece um exemplo de entradas para treinamento de uma rede neural. Estas entradas representam o funcionamento de uma porta digital ou-exclusiva. Como possuímos apenas duas entradas podemos desenhar este modelo em duas dimensões como demonstrado na Figura 5.

Tabela 1 – Exemplo de entradas para treinamento de uma rede neural

e_1	e_2	s
0	0	0
0	1	1
1	0	1
1	1	0

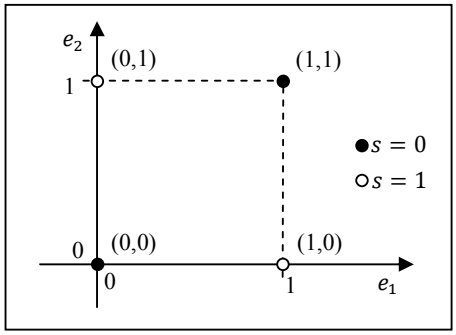


Figura 5 – Representação gráfica dos dados da Tabela 1

Pela Figura 5 pode-se perceber que é impossível separar os dois tipos de saídas utilizando apenas uma reta. Este tipo de problema é denominado não-linear. O atraso experimentado pelas RNA's nas décadas de 60 e 70 se deve justamente à ausência de um algoritmo de treinamento que pudesse trabalhar com mais de uma camada. A necessidade de se operar com mais de uma camada se deve à impossibilidade de resolver problemas não

lineares com uma camada, como é o exemplo do ou-exclusivo. Portanto a rede utilizada neste exemplo deve conter pelo menos uma camada intermediária conforme demonstramos na Figura 6.

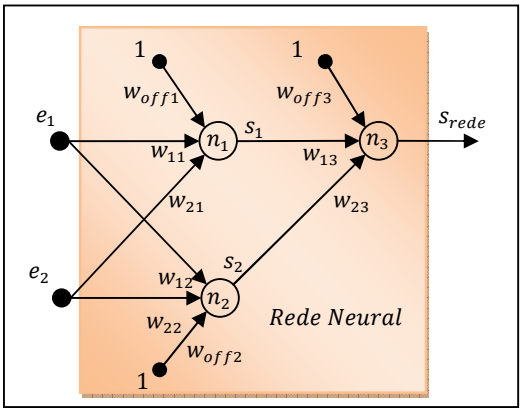


Figura 6 – Exemplo de rede neural

Os valores iniciais dos pesos foram sorteados aleatoriamente entre menos um e mais um ([1,1]) e podem ser vistos na Tabela 2. A função de ativação utilizada para todos os neurônios foi a sigmoide.

Tabela 2 – Pesos iniciais da rede neural

Conexão	Peso
w_{11}	−0,293
w_{21}	0,726
w_{off1}	0,827
w_{12}	−0.888
w_{22}	−0,810
w_{off2}	0,756
w_{13}	−0,536
w_{23}	0,183
w_{off3}	0,908

As equações que serão utilizadas na rede neural gerada neste exemplo são apresentadas em Eq. 5.

$$s_{n_1} = Sigmoid\left(w_{off1} * 1 + w_{11} * e_1 + w_{21} * e_2\right)$$
$$s_{n_2} = Sigmoid\left(w_{off2} * 1 + w_{12} * e_1 + w_{22} * e_2\right)$$
$$s_{rede} = Sigmoid\left(w_{off3} * 1 + w_{13} * s_{n_1} + w_{23} * s_{n_2}\right) \quad \textbf{Eq. 5}$$
$$Sigmoid(x) = \frac{1}{(1 - e^{-x})}$$

A Tabela 3 apresenta os valores iniciais obtidos pela rede neural para os pontos de treinamento. Na Figura 7 apresentamos uma representação de todos os pontos do espaço compreendidos entre 0 e 1 para as duas entradas. Percebemos que a figura apresenta uma tonalidade cinza praticamente constante em todo o espaço. Isto indica que a rede inicial apresenta uma resposta similar para qualquer entrada, que confirmamos na Tabela 3, onde todos os pontos apresentam uma saída próxima de 0,645.

Tabela 3 – Saídas iniciais da rede neural

e_1	e_2	s	s_1	s_2	s_{rede}
0	0	0	0,696	0,680	0,659
0	1	1	0,630	0,467	0,635
1	0	1	0,825	0,486	0,658
1	1	0	0,779	0,280	0,632

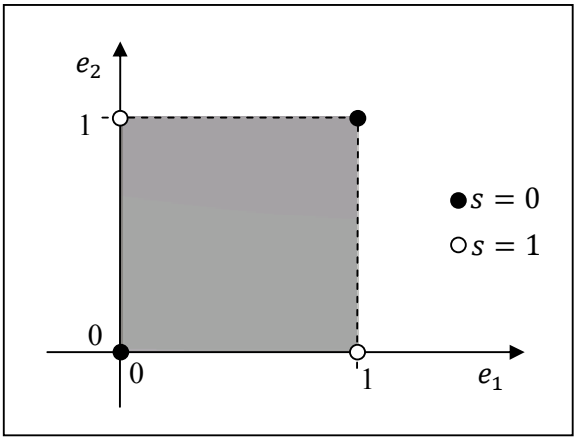


Figura 7 – Resposta da rede neural sem treinamento

O cálculo para retroação dos erros desta rede são apresentados na Eq. 6.

$$e_{rede} = (s - s_{rede}) * F'(s_{rede})$$
$$e_{n_1} = e_{rede*}\left(F'(s_1)\right) * w_{13} \qquad \textbf{Eq. 6}$$
$$e_{n_2} = e_{rede*}\left(F'(s_2)\right) * w_{23}$$

E a equação para ajuste dos pesos na Eq. 7

$$w_{ij}^{novo} = w_{ij}^{antigo} + r * e_j * s_i \qquad \textbf{Eq. 7}$$

Onde r: (taxa de aprendizado) foi escolhida como 1.

O processo de correção dos erros deve ser realizado para cada entrada. Para a primeira entrada ($e_1 = 0;\ e_2 = 0$) temos:

$$e_{rede} = -0{,}659 * F'(0{,}659) = -0{,}148 \qquad \textbf{Eq. 8}$$

Ajustando os pesos do neurônio de saída:

$$w_{13} = -0{,}536 - 0{,}148 * 0{,}696 * 1 = -0{,}639$$
$$w_{23} = 0{,}183 - 0{,}148 * 0{,}680 * 1 = 0{,}082$$
$$w_{off3} = 0{,}908 - 0{,}148 * 1 * 1 = 0{,}760$$

Eq. 9

Retroagindo os erros:

$$e_{n_1=} - 0{,}148 * F'(0{,}696) * -0{,}639 = 0{,}020$$
$$e_{n_2=} - 0{,}148 * F'(0{,}680) * -0{,}082 = -0{,}002$$

Eq. 10

Ajustando os demais pesos:

$$w_{11} = -0{,}293 + 0{,}020 * 0 * 1 = -0{,}293$$
$$w_{12} = 0{,}726 + 0{,}020 * 0 * 1 = -0{,}726$$
$$w_{off1} = 0{,}827 + 0{,}020 * 1 * 1 = 0{,}847$$
$$w_{21} = -0{,}888 - 0{,}002 * 0 * 1 = -0{,}888$$
$$w_{22} = -0{,}810 - 0{,}002 * 0 * 1 = -0{,}810$$
$$w_{off2} = 0{,}756 - 0{,}002 * 1 * 1 = -0{,}754$$

Eq. 11

Após esse primeiro ajuste, a saída para o par de entradas $(0; 0)$, que era (0,659), passa a ser (0,591), valor mais próximo do correto. Este processo é repetido para a mesma entrada até que a resposta convirja com uma precisão aceitável ou atinja um numero máximo de iterações. Para esta iteração em particular, em 2194 ajustes dos pesos, o valor converge para um erro menor que 1%.

Após o treinamento para a primeira entrada, realiza-se o mesmo para as demais. Após uma iteração de treinamento para todas as entradas temos a seguinte configuração:

Tabela 4 – Pesos da rede após 1ª iteração

Conexão	Peso
w_{11}	$-0{,}368$
w_{21}	1,888
w_{off1}	1,389
w_{12}	$-1{,}527$
w_{22}	$-0{,}800$
w_{off2}	0,450
w_{13}	$-3{,}726$
w_{23}	0,348
w_{off3}	$-1{,}016$

Com esta configuração de rede obtemos as saídas apresentadas na Tabela 5. Percebemos que os valores estão muito próximos de zero, apesar disto, a rede fornece resultados maiores (0,019 e 0,021) para entradas diferentes (0-1 e 1-0), e menores (0,015 e 0,010) para as iguais (0-0 e 1-1), o que representa o funcionamento de uma porta ou-exclusiva.

Tabela 5 – Saídas da rede após 1ª iteração

e_1	e_2	s	s_{rede}
0	0	0	0,015
0	1	1	0,019
1	0	1	0,021
1	1	0	0,010

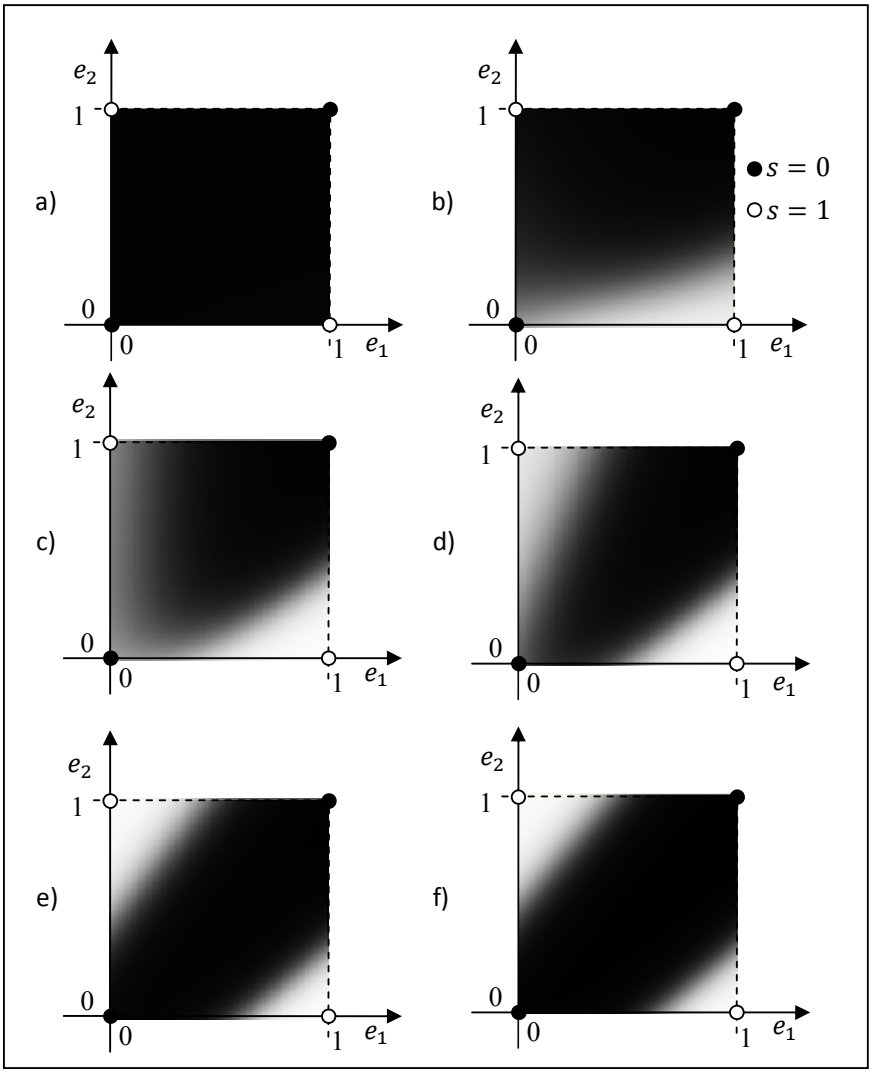


Figura 8 – Resposta da rede neural
após a) 2, b) 5, c) 7, d) 10, e) 15, f) 50 iterações.

Na Figura 8 é apresentado o resultado para a segunda, quinta, décima, décima quinta e qüinquagésima iteração. Podemos perceber que a rede aprende a classificar corretamente os pontos de entrada e aprende também a generalizar os resultados de maneira adequada para os demais pontos do espaço.

2.1.5 Representação geométrica dos neurônios

Outra maneira de encontrar os pesos w_{ij} das conexões é utilizar características geométricas dos dados de entrada. Como as entradas são valores numéricos podemos montar um vetor pertencente a um espaço euclidiano n-dimensional, onde n é o numero de entradas do sistema, conforme foi feito com os dados da Tabela 1 resultando no gráfico apresentado na Figura 5. Estes pontos podem ser agrupados formando grupos de saídas similares, podendo então ser separados por um conjunto de planos conforme demonstrado em (Bose, et al., 1993).

De posse destes planos é possível criar uma rede neural que classifique estes pontos corretamente, como demonstrado em (Gentile, et al., 2001) e (Chen, et al., 1997). Na Figura 9 é apresentado um neurônio e sua representação gráfica dando ênfase ao plano de separação gerado por ele.

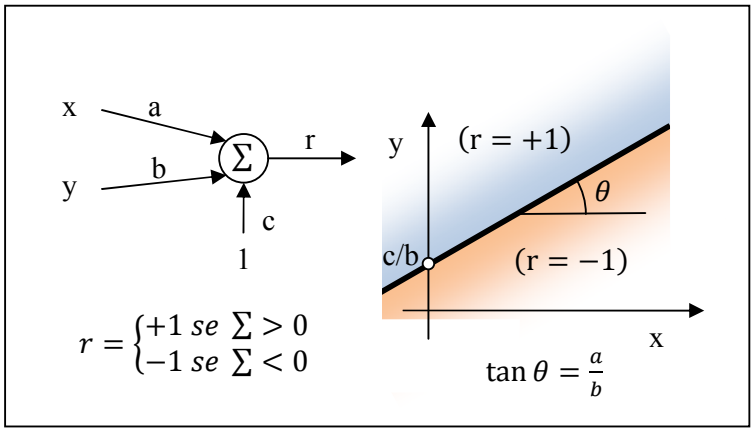


Figura 9 – Neurônio e respectiva representação gráfica

2.2 Testes para colisão de objetos

Em algumas operações computacionais, principalmente jogos e simulações de ambientes reais (Bourg, 2002), necessita-se de um sistema que detecte com precisão e velocidade adequadas as colisões entre os objetos.

A colisão entre dois objetos, A e B, se dá quando alguma região de A se encontra dentro do espaço delimitado por B conforme Figura 10.

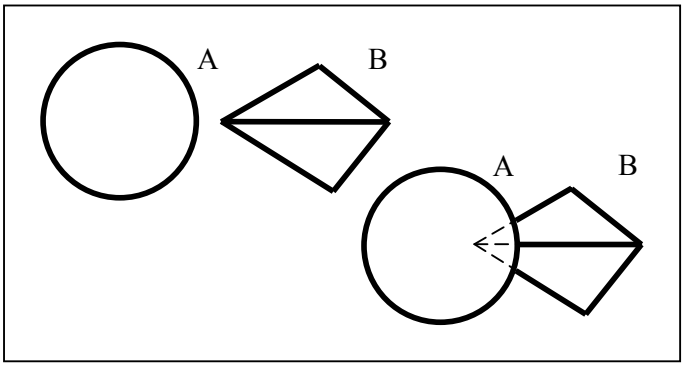


Figura 10 – Colisão de dois objetos

Para se verificar a colisão é necessário testar se existe algum ponto pertencente ao objeto A que se encontra dentro do espaço delimitado pelo objeto B, o que na maioria dos casos não pode ser realizado no espaço de tempo exigido pela aplicação.

Uma alternativa para realizar esse procedimento é a utilização de volumes envolventes. Esta é uma técnica amplamente utilizada por reduzir a quantidade de cálculos conseqüentemente aumentando a velocidade para detecção de colisões (Gottschalk, 2000).

2.2.1 Esferas

O volume envolvente mais simples para se detectar uma colisão entre dois objetos é a esfera. As esferas são criadas com o menor raio possível englobando todos os pontos pertinentes para cada objeto conforme pode ser visto na Figura 11.

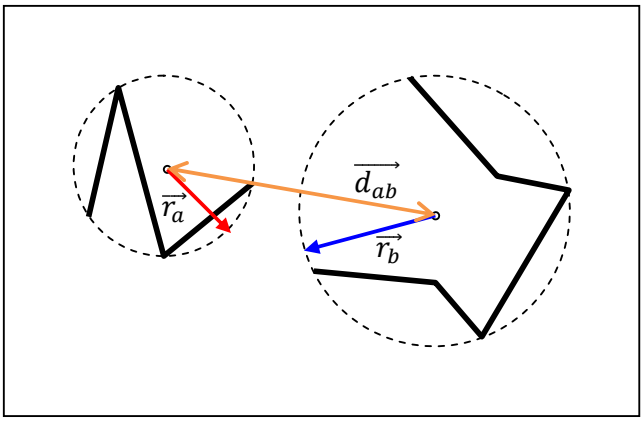


Figura 11 – Detecção de não colisão utilizando esferas

Após a criação das esferas, a detecção da colisão é reduzida a uma soma e uma comparação conforme Eq. 12.

$$Teste: \begin{cases} \left\|\overrightarrow{(r_a)}\right\| + \left\|\overrightarrow{(r_b)}\right\| > \left\|\overrightarrow{(d_{ab})}\right\| \rightarrow colide \\ \left\|\overrightarrow{(r_a)}\right\| + \left\|\overrightarrow{(r_b)}\right\| = \left\|\overrightarrow{(d_{ab})}\right\| \rightarrow tangencia \\ \left\|\overrightarrow{(r_a)}\right\| + \left\|\overrightarrow{(r_b)}\right\| < \left\|\overrightarrow{(d_{ab})}\right\| \rightarrow não\ colide \end{cases} \quad \textbf{Eq. 12}$$

Onde: r_a é o raio da esfera a e d_{ab} é a distancia entre os centros das esferas a e b.

Dessa forma o tempo necessário para determinação das colisões é reduzido drasticamente. Neste trabalho estaremos interessados em encontrar situações onde o sistema não colide para que possamos achar pelo menos um plano que separe os objetos. Por isso deste ponto em diante as situações de colisão e as quais os objetos apenas tangenciam-se serão tratadas conjuntamente. A desvantagem deste método é a grande quantidade de falsas colisões retornadas pelo sistema conforme pode ser observado na Figura 12.

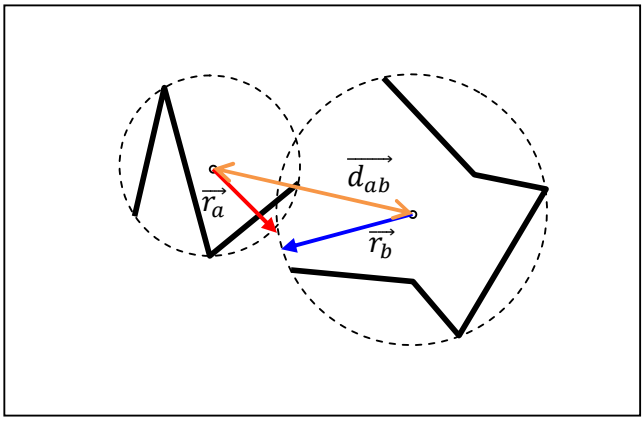


Figura 12 – Falsa colisão detectada utilizando esferas

2.2.2 Caixas alinhadas pelos eixos globais

Na abordagem de caixas alinhadas pelos eixos globais, ou AABB - *Axis Aligned Bounding Boxes*, uma caixa, com lados paralelos aos eixos globais, é gerada em volta dos objetos. A colisão acontece quando a distância entre os centros das caixas é menor que a soma das distâncias centro-face de cada caixa, para todas as direções. Isto pode ser observado quando as projeções das caixas nos eixos se encontram. Caso exista pelo menos um eixo no qual as sombras não se tocam as caixas não colidem conforme Figura 13.

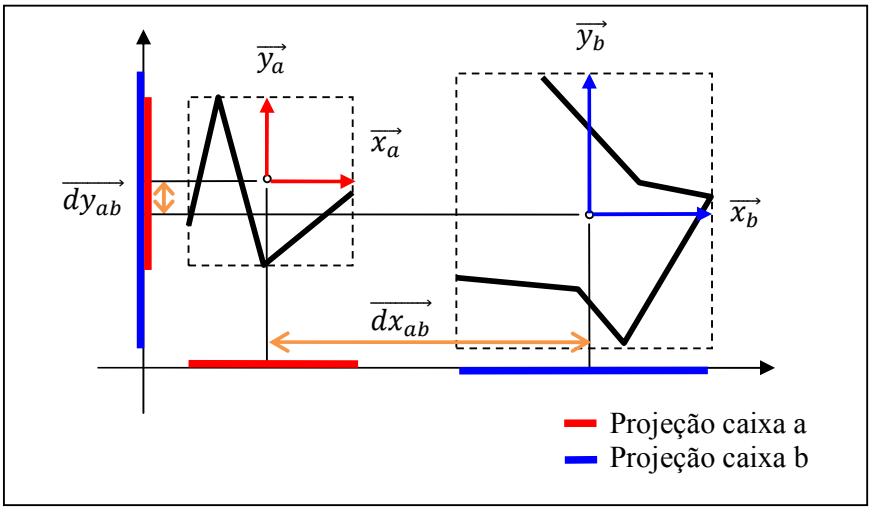


Figura 13 – Detecção de não colisão utilizando AABB

O cálculo realizado é similar àquele das esferas apresentado em Eq. 12. Repetimos o teste, conforme Eq. 13, para cada eixo do sistema. Caso exista pelo menos um eixo sem colisão, os objetos não colidem no espaço. Se em todos os eixos existe sobreposição das projeções os volumes envolventes estão colidindo.

$$Para\ o\ eixo\ k\ (k = x\ ou\ y): \begin{cases} \overrightarrow{k_a} + \overrightarrow{k_b} \geq \overrightarrow{d_{k_{ab}}} \rightarrow colide \\ \overrightarrow{k_a} + \overrightarrow{k_b} \geq \overrightarrow{d_{k_{ab}}} \rightarrow não\ colide \end{cases} \qquad \textbf{Eq. 13}$$

Onde k_a: é a distancia entre o centro e a face da caixa a no eixo k

$d_{k_{ab}}$: é a distancia entre os centros das caixas a e b no eixo k.

Na Figura 14 observamos um caso de falso positivo na detecção de colisão entre os dois objetos.

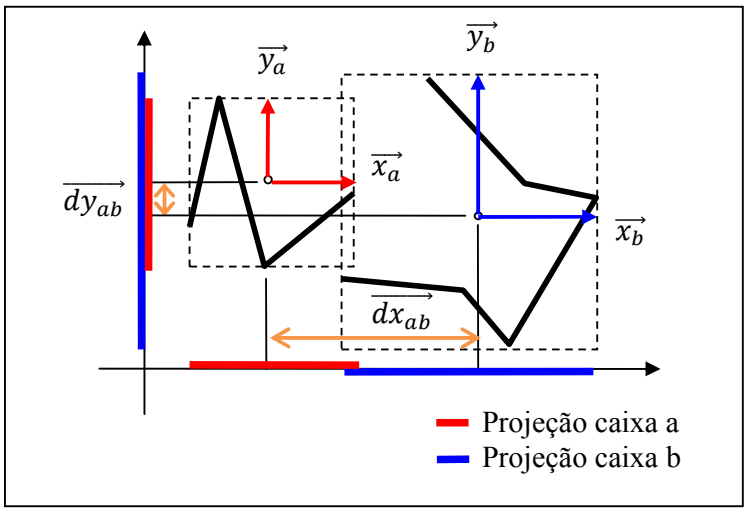


Figura 14 – Falsa colisão detectada utilizando AABB

Este método apresenta um baixo ajuste para objetos desalinhados com os eixos globais, mas normalmente gera volumes melhor ajustados aos objetos que as esferas(Gottschalk, 2000).

2.2.3 Caixas orientadas

O sistema de caixas orientadas, ou OBB – *Oriented Bounding Boxes*, é similar ao AABB com a diferença que agora as caixas são orientadas de acordo com os eixos de maior significância do objeto, ajustando-se de maneira mais eficiente na maioria dos casos (S Gosttchalk, 1996).

A diferença deste método para o AABB é que as distâncias que são utilizadas para verificação das colisões não são projetadas nos eixos globais, mas nos eixos locais, conforme pode ser visto na Figura 15.

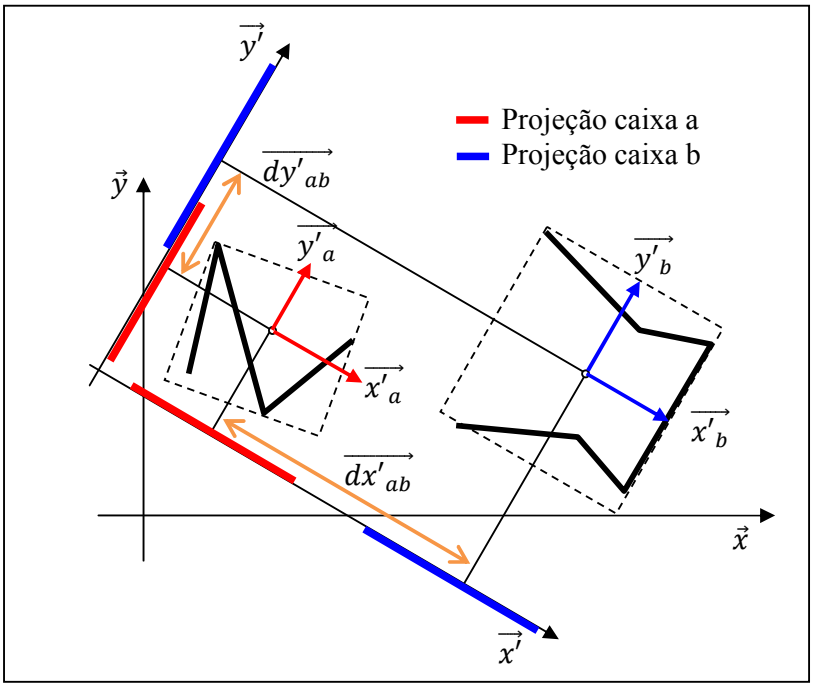


Figura 15 – Detecção de não colisão utilizando OBB

Com este método a quantidade de falsos positivos é reduzida, mas ainda pode acontecer, como podemos ver na Figura 16.

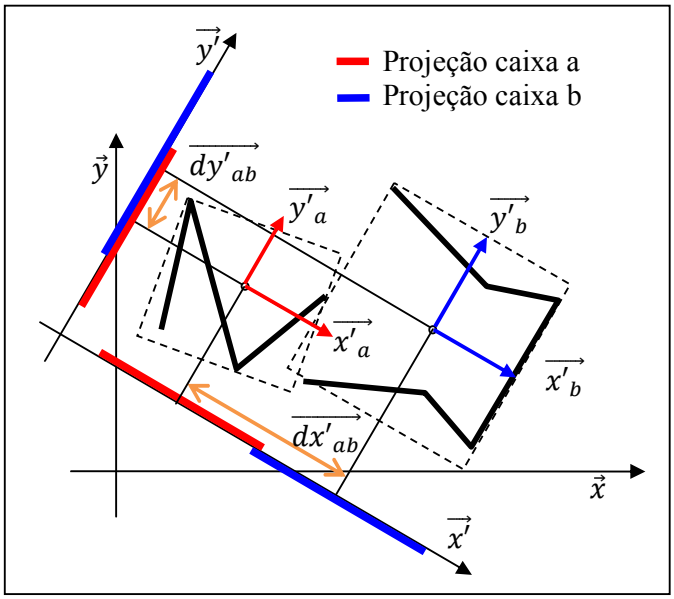


Figura 16 – Falsa colisão detectada utilizando OBB

As OBB's possuem um grau de ajuste muito superior aos demais volumes envolventes, perdendo apenas para as superfícies convexas, *convex hulls*, mas que devido à complexidade, tanto na montagem quanto nos testes, não será abordada neste trabalho (Gottschalk, 2000).

2.2.4 Teorema dos eixos de separação

O teorema dos eixos de separação ou *separating axis theorem* (Gottschalk, 2000) explicita que para garantir a não colisão dos sólidos é suficiente realizar os testes para os eixos da primeira caixa, os eixos da segunda e a combinação linear destes. Logo a complexidade dos testes aumenta na razão de n^2 onde n é a quantidade de dimensões.

Para simplificar este problema é possível excluir os testes com a combinação linear dos eixos das caixas. Como conseqüência desta simplificação a quantidade de falsos positivos tende a aumentar.

2.2.5 Árvores

Os métodos de detecção de colisão através de volumes envolventes fornecem uma alternativa para reduzir o tempo necessário para realizar os testes de colisão, mas acabam induzindo um erro considerável em muitas aplicações.

Para melhorar a precisão dos testes, sem aumentar demasiadamente o tempo de processamento, pode-se utilizar uma estrutura hierárquica de decisão denominada árvore (S Gosttchalk, 1996).

O procedimento a ser realizado em testes de colisão utilizando estrutura em árvore está representado na Figura 17.

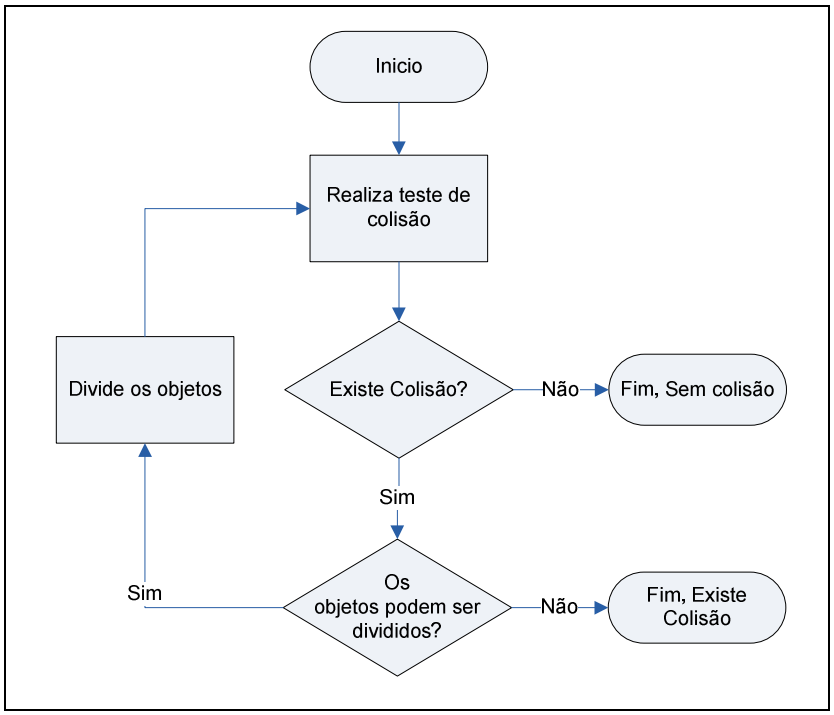


Figura 17 – Execução de um teste de colisão utilizando estrutura em árvore

Na Figura 18 é demonstrado como o falso positivo (Figura 12, Figura 14 e Figura 16) é evitado utilizando o algoritmo apresentado. No primeiro instante (Figura 18a) os objetos A e B não colidem, mas o sólido envolvente sim. Após uma iteração do algoritmo (Figura 18b), o objeto A é divido em dois (A1 e A2), dos quais apenas o segundo colide com o objeto B. Na segunda iteração (Figura 18b), em que o objeto B é divido em B1 e B2, percebemos que o objeto A2 não mais colide com nenhum outro objeto, finalizando assim o algoritmo.

Com esse exemplo vemos uma das vantagens em se utilizar o algoritmo de árvores. As iterações iniciais eliminam grandes seções dos objetos de tal forma que o algoritmo se concentra apenas nas áreas que necessitam de maior refinamento evitando cálculos desnecessários.

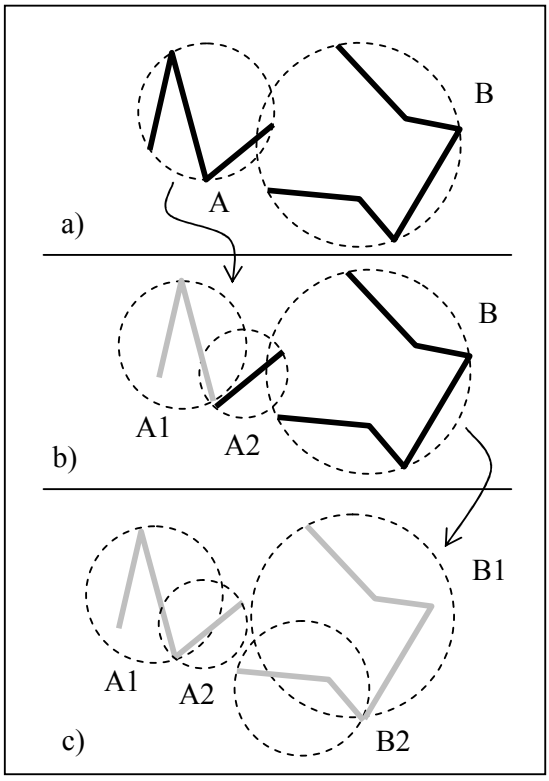


Figura 18 – Detecção de colisão com árvore de esferas

a) Condição inicial – falsa colisão, b) 1ª quebra – falsa colisão, c) 2ª quebra – não colisão;

2.2.6 Planos de segmentação

As técnicas de teste de colisão através de sólidos envolventes podem retornar um parâmetro interessante em seu resultado: o plano de segmentação.

A não-colisão é detectada através da verificação das distâncias entre os centros das caixas para um determinado eixo. Deste modo podemos inferir que existe um plano de segmentação que é perpendicular ao eixo e está entre as projeções dos volumes naquele eixo. Para o caso das esferas o vetor normal está sob o eixo que liga os dois centros. Já para a AABB ele repousa em um dos eixos globais no sistema, a passo que na OBB ele se encontra em um dos eixos locais de uma das caixas conforme podemos observar na Figura 19.

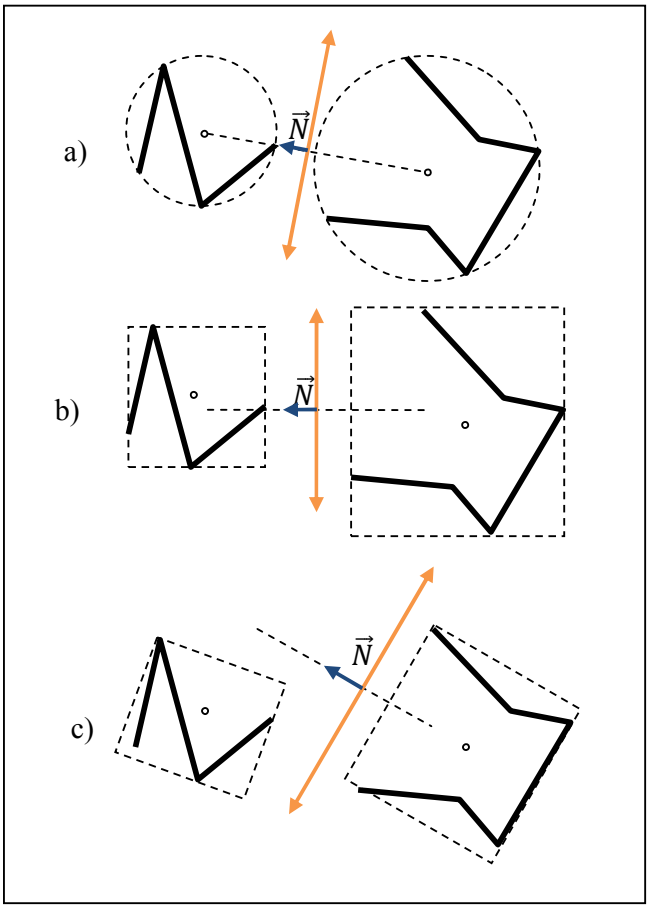


Figura 19 – Planos de separação: a) esferas, b) AABB's, c) OBB's

Cada plano divide o espaço em dois. A diferenciação destes espaços é dada pela direção apontada pelo vetor normal. Aquele apontado pelo vetor é definido positivo, o outro lado negativo. Definimos também que os pontos que se encontrem no semi-espaço positivo estão acima do plano, e aqueles no semi-espaço negativo estão abaixo.

O teste de colisão gera diversos planos de segmentação. Estes planos dividem o espaço em regiões que podem ser classificadas de maneira única. Esta propriedade será utilizada no programa para definir a segunda camada de neurônios, responsáveis por detectar os subespaços pré-classificados.

3 Metodologia

Neste trabalho foi proposto o treinamento de redes neurais a partir de uma abordagem geométrica dos pontos, mas de maneira diferente das literaturas atuais (Mangasarian, 1968), (Gentile, et al., 2001) e (Chen, et al., 1997). Através das técnicas de verificação de colisão, utilizando a teoria dos eixos de separação, temos uma maneira rápida e computacionalmente eficiente (S Gosttchalk, 1996) de calcular os planos de segmentação.

3.1 Teste de colisão utilizando arvores de OBB

Conforme demonstrado por (Gottschalk, 2000) e (S Gosttchalk, 1996) a OBB como volume envolvente é a melhor opção para testes de colisão. Neste trabalho fazemos a expansão do conceito de OBB para N-dimensões criando então uma hipercaixa envolvente e orientada, ou OBHB – *Oriented Bounding Hiperbox*. Para isso algumas alterações foram necessárias. Tais alterações estão resumidas na Tabela 6 e serão explicadas posteriormente nos subitens adequados.

Tabela 6 – Diferenças entre OBB-Tree e OBHB-Tree

	OBB-Tree	OBHB-Tree
Objetivo	Detectar colisão	Encontrar planos de segmentação
Dimensões	2 ou 3	N
Quantidade de testes	$(2N + N^2)$	$2N$
Volumes filhos	Sempre conectados	Podem ser separados
Abordagem de divisão	Ambas as caixas simultaneamente	Uma hipercaixa de cada vez, alternadamente

O algoritmo de detecção de colisão foi divido em quatro partes: criação da hipercaixa envolvente, busca por hiperplanos de segmentação, teste de colisão e seleção de hiperplanos. A estrutura de busca em árvore é implementada na terceira parte enquanto o algoritmo para divisão dos pontos se encontra na primeira.

Nos algoritmos que são apresentados, é feito uso de dois sistemas de coordenadas: o local e o global. O sistema de coordenadas global, SCG, indica os valores de coordenadas a respeito de uma origem fixa e comum a todos os objetos. O sistema de coordenadas local, SCL, realiza duas transformações de espaço: uma mudança de origem, na qual a nova origem passa a ser o centro da hipercaixa a ser analisada, e uma rotação, na qual os novos eixos são

os vetores centro-face da hipercaixa. Na Figura 20 temos a representação gráfica de ambos os sistemas para duas dimensões.

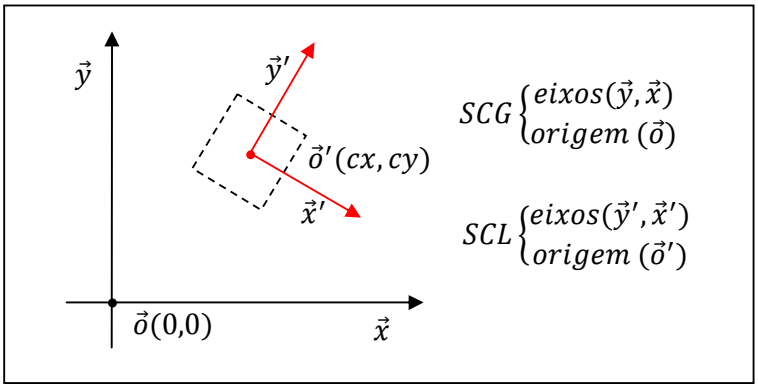


Figura 20 – Relação entre coordenadas locais e coordenadas globais

Nos itens que se seguem é feita uma exemplificação do funcionamento dos algoritmos baseados em um exemplo bidimensional com duas classes apenas: losangos com onze pontos, e círculos com seis. Tais pontos foram escolhidos arbitrariamente e podem ser visualizados na Tabela 7 ou na Figura 21.

Tabela 7 – Pontos das classes usadas como exemplo

Classe	Identificação	Posição no SCG	
		X	Y
Losangos $Y = sin(\pi * X) * X$	1	0,00	0,00
	2	0,10	0,03
	3	0,20	0,12
	4	0,30	0,24
	5	0,40	0,38
	6	0,50	0,50
	7	0,60	0,57
	8	0,70	0,57
	9	0,80	0,47
	10	0,90	0,28
	11	1,00	0,00
Círculos ***pontos arbitrários***	A	0,40	0,10
	B	0,40	0,00
	C	0,45	0,05
	D	0,55	0,05
	E	0,60	0,00
	F	0,60	0,10

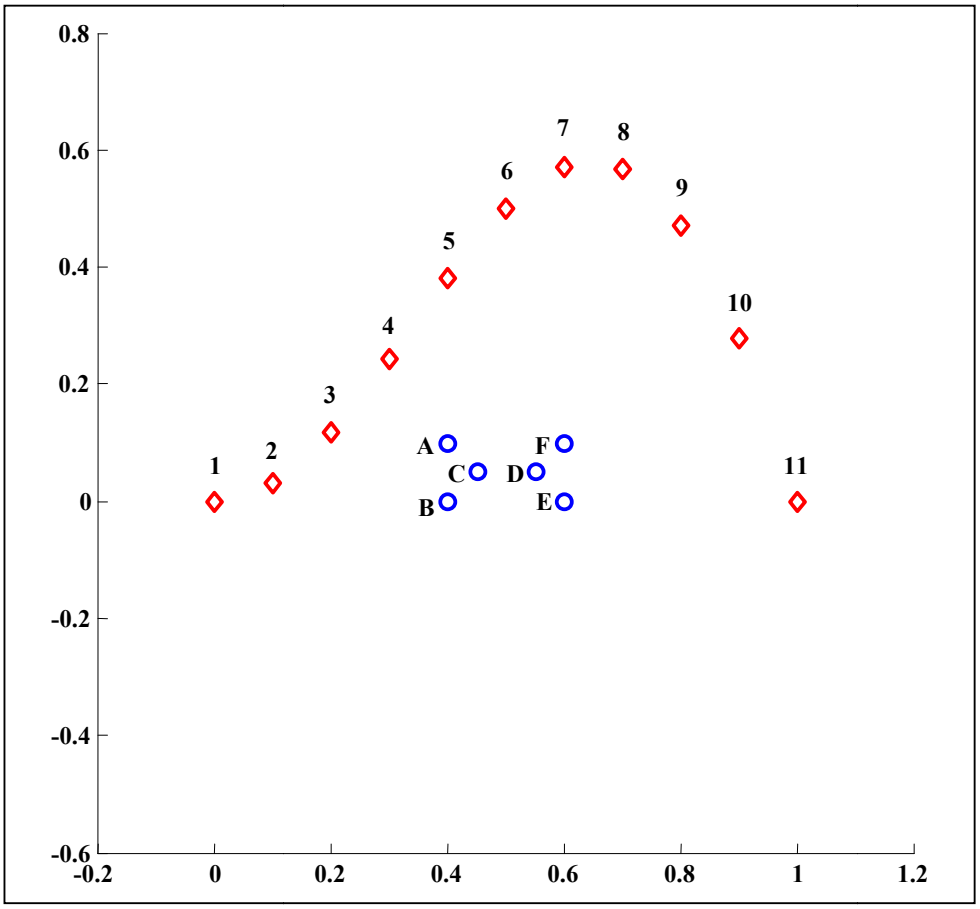


Figura 21 – Pontos das classes usadas como exemplo

3.2 Criação da hipercaixa envolvente orientado

Para criar uma OBHB é necessário encontrar os eixos sobre os quais a hipercaixa será criada. Dada uma matriz de covariância, gerada a partir dos valores das componentes dos pontos, os autovetores indicam as direções das distribuições de pontos no espaço. Pode-se visualizar na Figura 22 exemplos de autovetores se alinhando às distribuições de pontos. É possível notar que o autovetor com maior autovalor associado apresenta o eixo principal da OBHB que será gerada.

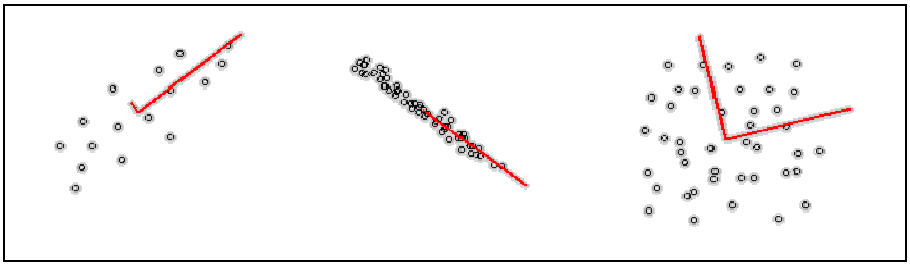


Figura 22 – Representação geométrica dos autovetores de uma matriz de covariância

Por serem ortogonais entre si, os autovetores advindos da matriz de covariância fornecem os eixos necessários para a construção da OBHB. Na Figura 23 são apresentados graficamente os passos para encontrar uma hipercaixa envolvente: encontrar os autovetores da matriz de covariância (b), projetar os pontos no sistema de coordenadas locais (b1) e definir dos limites da hipercaixa (b2).

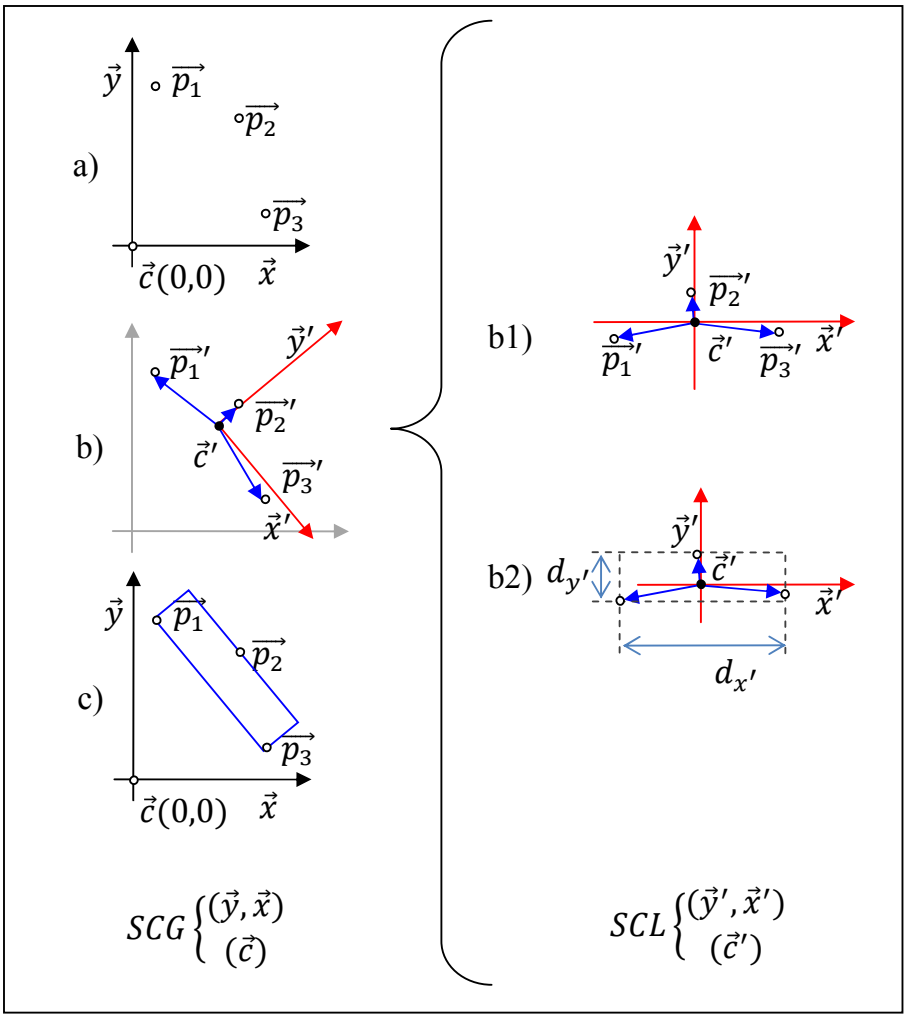


Figura 23 – Processo de construção de uma OBHB

De posse dos eixos do SCL e das projeções dos pontos nestas coordenadas, é possível encontrar todas as distâncias do centro da hipercaixa às faces. Estas distâncias serão utilizadas na detecção de colisões conforme podemos ver na Figura 15.

Para encontrar o centro e os limites do OBHB é necessário projetar todos os pontos no SCL e encontrar os valores máximos e mínimos para cada eixo local, encontrando assim uma distância em cada eixo. O centro da hipercaixa será a junção dos pontos médios de cada

distancia calculada. Assim os limites inferiores e superiores serão iguais. Observar que o ponto médio, que divide a quantidade de pontos ao meio, não é o mesmo que ponto central, que divide a caixa ao meio, não importando que a divisão dos pontos fique desbalanceada.

Realizando a transposição de todos os pontos da classe losango do SCG para o SCL temos que o máximo e mínimo para cada eixo são respectivamente: (-0,37 0,57) para o eixo x' e (-0,22 0,45) para o eixo y'. Pode-se assim encontrar as distâncias que compõe a hipercaixa conforme visualizado na Figura 24.

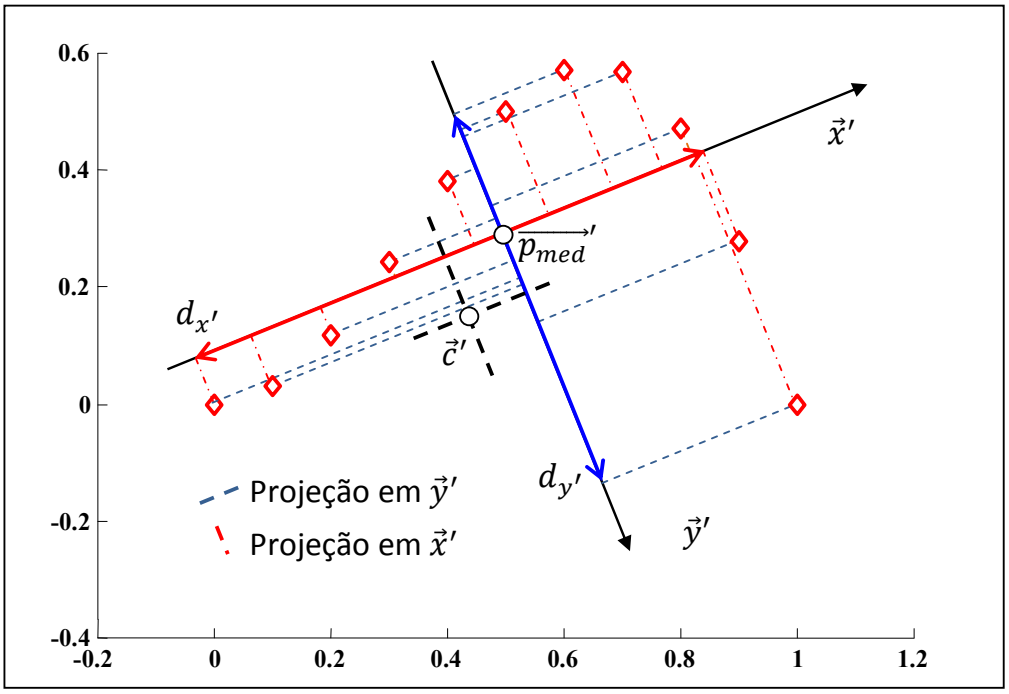


Figura 24 – Encontrando o centro e limites para uma OBHV

Todos os dados referentes à hipercaixa gerada para a classe losango podem ser visualizados na Tabela 8. O centro e os autovetores estão no SGC. Os limites estão descritos no SCL e representam as distâncias centro-face em cada eixo, pois as distâncias, superior e inferior, são iguais. A OBHB gerada pode ser observada graficamente na Figura 25.

Tabela 8 – Dados da hipercaixa para os pontos da classe losango do exemplo

Hipercaixa dos pontos de classe Losango	
Centro	(0,45 0,14)
Limites	(0,34 0,47)
Auto-Vetores (linhas)	$\begin{pmatrix} 0{,}38 & -0{,}93 \\ 0{,}93 & 0{,}38 \end{pmatrix}$

Como será utilizada detecção de colisões através de uma estrutura em árvore, é necessário classificar os pontos para uma posterior segmentação da OBHB. Convencionou-se segmentar a hipercaixa ao meio, passando pelo centro deste, através de um hiperplano normal ao eixo principal. Isto é feito projetando-se cada ponto para o SCL e analisando seu valor o eixo principal. Caso o valor for positivo, o ponto se encontra no semi-espaço superior, gerado pela segmentação do espaço pelo hiperplano. Em caso ele seja negativo, no semi-plano inferior.

Na Figura 25 podemos observar a divisão dos pontos do exemplo de acordo com a convenção adotada. Notar que a divisão, apesar de simétrica pelo centro da hipercaixa, nem sempre resulta em quantidades iguais de pontos em cada um dos lados. Isto traz algumas vantagens, pontos muito próximos serão analisados quase sempre numa só caixa e pontos afastados serão rapidamente isolados facilitando a procura de planos.

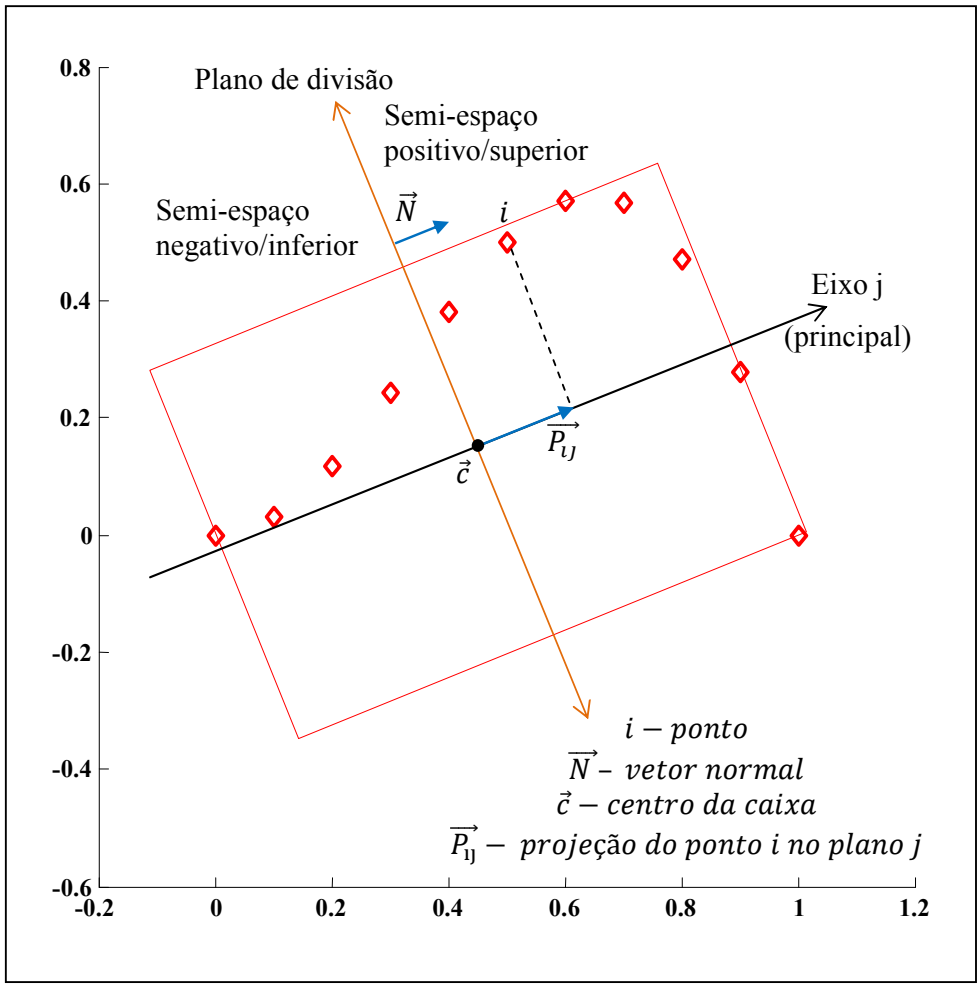


Figura 25 – Hipercaixa gerada para a classe losangos e classificação dos pontos

3.3 Busca por hiperplanos de segmentação

Através do teorema do eixo de separação (Gottschalk, 2000), sabe-se que se as projeções dos objetos num determinado eixo formarem duas sombras distintas, os objetos não colidem e o plano que separa estes objetos é normal àquele eixo. Segundo o teorema existe uma quantidade máxima de eixos que precisam ser verificados para garantir que os objetos não colidem.

Para o caso específico de OBB's, devem ser analisados os eixos de cada caixa e a combinação linear destes. No caso de duas dimensões temos: dois eixos para a primeira caixa, dois para a segunda e quatro (2x2) para as combinações, totalizando oito eixos, conforme podemos visualizar na Figura 26.

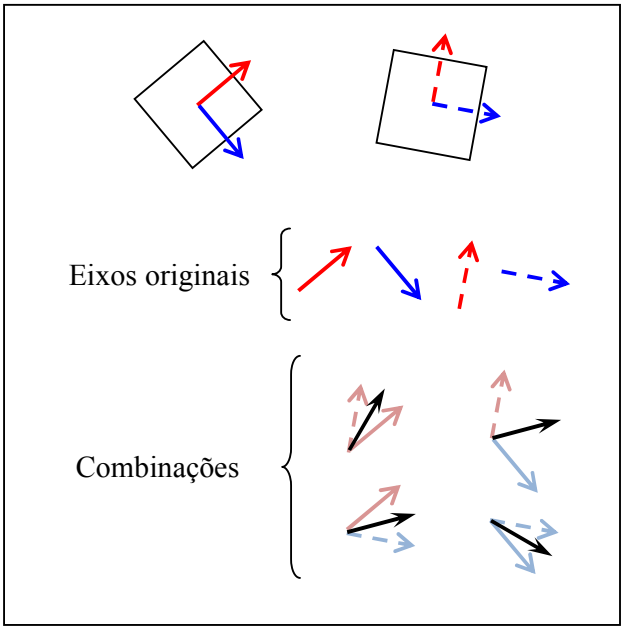


Figura 26 – Possíveis eixos para detecção de planos de segmentação

A quantidade de eixos a serem analisados é proporcional ao quadrado do número de dimensões do objeto, o que pode tornar o processamento inviável para sistemas com muitas variáveis.

A busca por hiperplanos de segmentação implementada não procura pelas combinações lineares dos eixos, apenas pelos eixos de ambas hipercaixas. Esta opção foi feita para simplificar os cálculos realizados, deste modo a complexidade cresce apenas linearmente com a quantidade de dimensões e não com o quadrado desta.

Para a realização do teste são verificados os hiperplanos que possuem como direção normal os eixos locais de cada hipercaixa. Utilizando o teorema do eixo de separação validamos quais destes planos são viáveis conforme algoritmo apresentado na Figura 27.

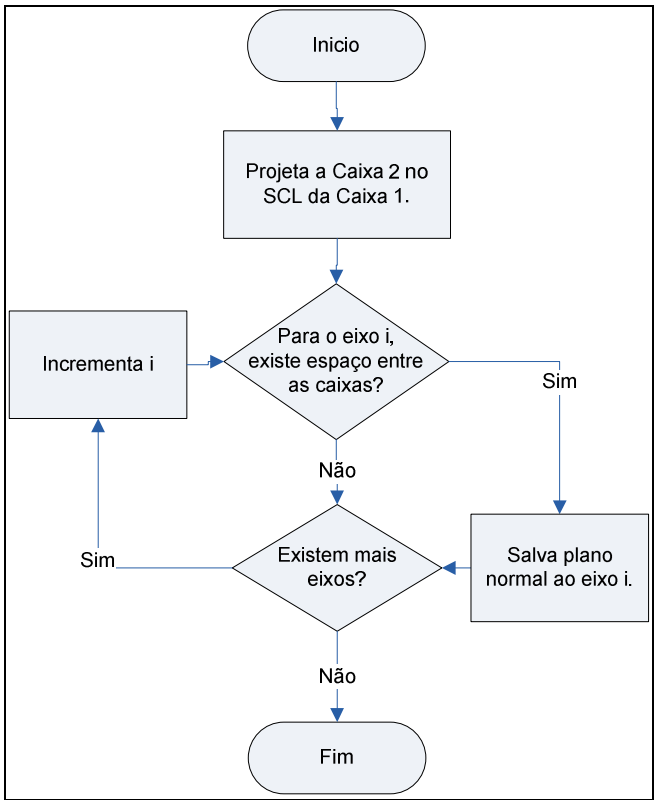


Figura 27 – Algoritmo para procura de planos

Para um determinado eixo, o hiperplano que separa as hipercaixas pode ser definido através de um vetor normal e um ponto. O vetor normal é qualquer vetor situado no eixo de análise. O ponto que o plano corta o eixo é definido como o ponto médio do espaço gerado entre as projeções das duas hipercaixas no eixo. Isto é demonstrado na Figura 28.

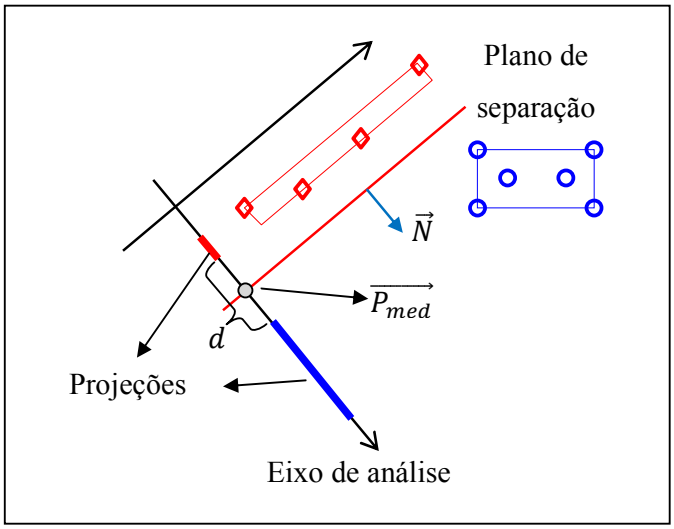


Figura 28 – Teste de eixo na procura de plano de separação

Para o exemplo apresentado, bidimensional, e excluindo a combinação linear de eixos, toda busca possui quatro hiperplanos de separação em potencial.

Na Figura 29 são apresentados os hiperplanos de segmentação encontrados para a comparação de duas hipercaixas, geradas através de um subconjunto dos pontos iniciais. Dos quatro eixos em potencial apenas dois separam efetivamente as caixas, enquanto os outros dois são descartados.

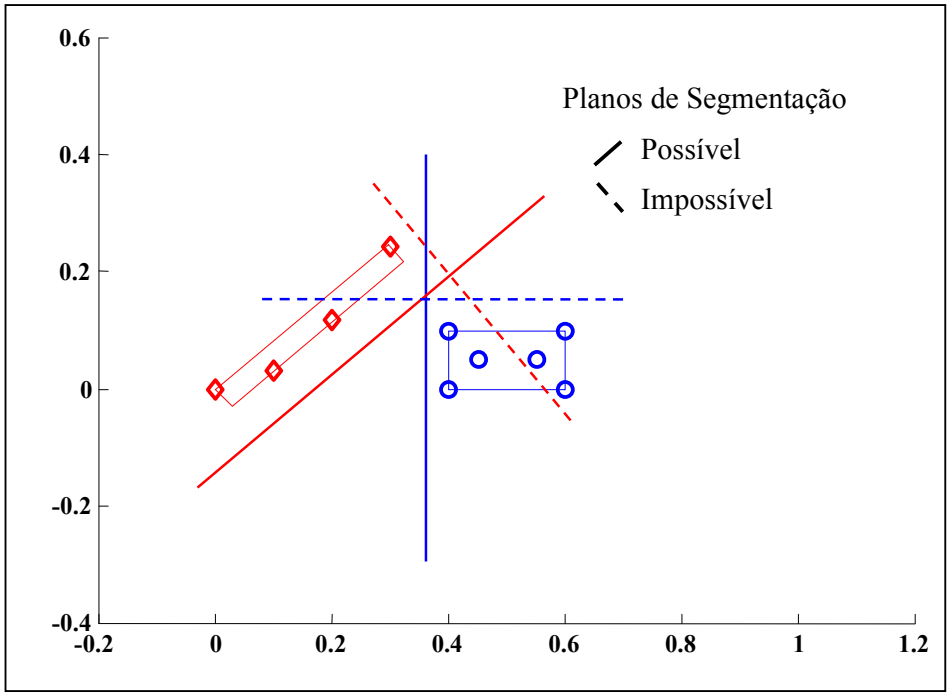


Figura 29 – Planos de segmentação

3.3.1 Teste de colisão

O algoritmo de teste de colisão verifica se existe algum plano de segmentação entre as duas hipercaixas através do algoritmo de busca por hiperplanos de segmentação. Caso não exista nenhum hiperplano ele divide uma das caixas e refaz o teste para cada um dos filhos da caixa segmentada com a segunda caixa. Este procedimento é repetido até que nenhum entre os objetos gerados colidam entre si, ou até não ser possível mais dividir os objetos e mesmo assim haver colisão. Neste último caso não é possível separar totalmente os objetos, pois existe um ponto que pertence a ambos.

Este teste é feito numa estrutura de árvore conforme algoritmo apresentado na Figura 17. Deste modo existem diversas técnicas de busca que podem ser aplicadas para decisão de qual objeto será dividido, quantas vezes será dividido e qual caminho seguir. Caso optemos pela divisão sempre do mesmo objeto, estaremos realizando uma busca em profundidade pela árvore. Isto não é interessante, pois deste modo o esforço computacional é concentrado, delineando em demasia o primeiro objeto, ao passo que algumas segmentações do segundo poderiam ser suficientes para provar a não colisão. Este fato pode ser visualizado na Figura 30, um caso particular em que as divisões dos pontos circulares, azuis, indicadas pelas setas cheias, não auxiliam em nada na busca da resposta, ao passo que apenas uma divisão dos losangos, vermelhos, é suficiente para verificar a separação.

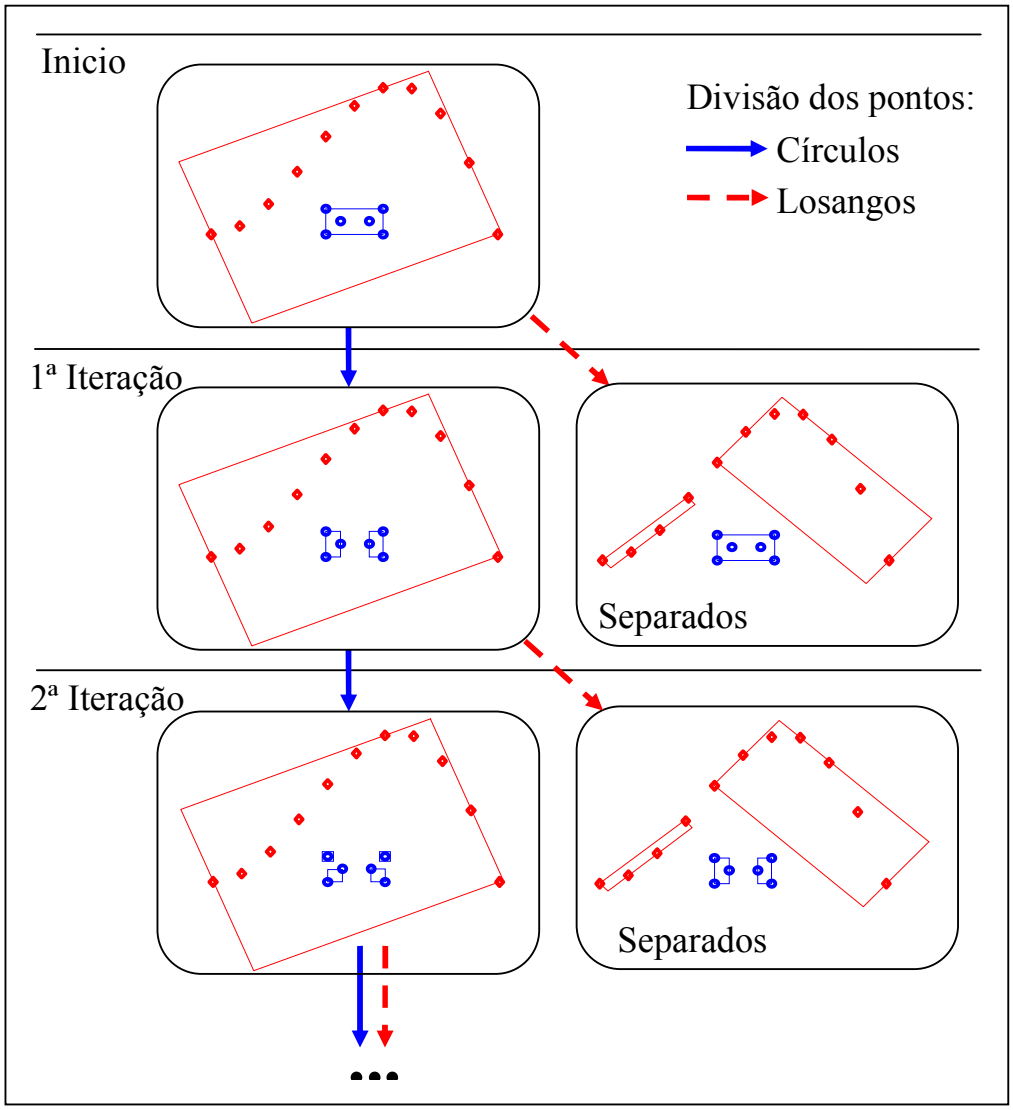


Figura 30 – Opções para divisão

Apesar de o exemplo apresentado possuir a peculiaridade de independer da quantidade de divisões de uma das hipercaixas, um sistema genérico também se beneficiará de uma divisão mais uniforme entre as duas hipercaixas. Isto acontece porque desta maneira grandes quantidades de pontos são eliminados, que geralmente se encontram bem afastados da área critica, em poucas iterações e ao mesmo tempo concentramos o esforço computacional nas áreas que precisam ser delineadas com maior precisão.

A busca pelos planos implementada é um misto da busca em profundidade, onde segue-se indefinidamente até o menor filho, sempre pelo mesmo caminho, e da busca em largura, onde todos os filhos de um mesmo nível são checados antes de passar para o próximo nível. O procedimento para esta busca é apresentado no fluxograma da Figura 31.

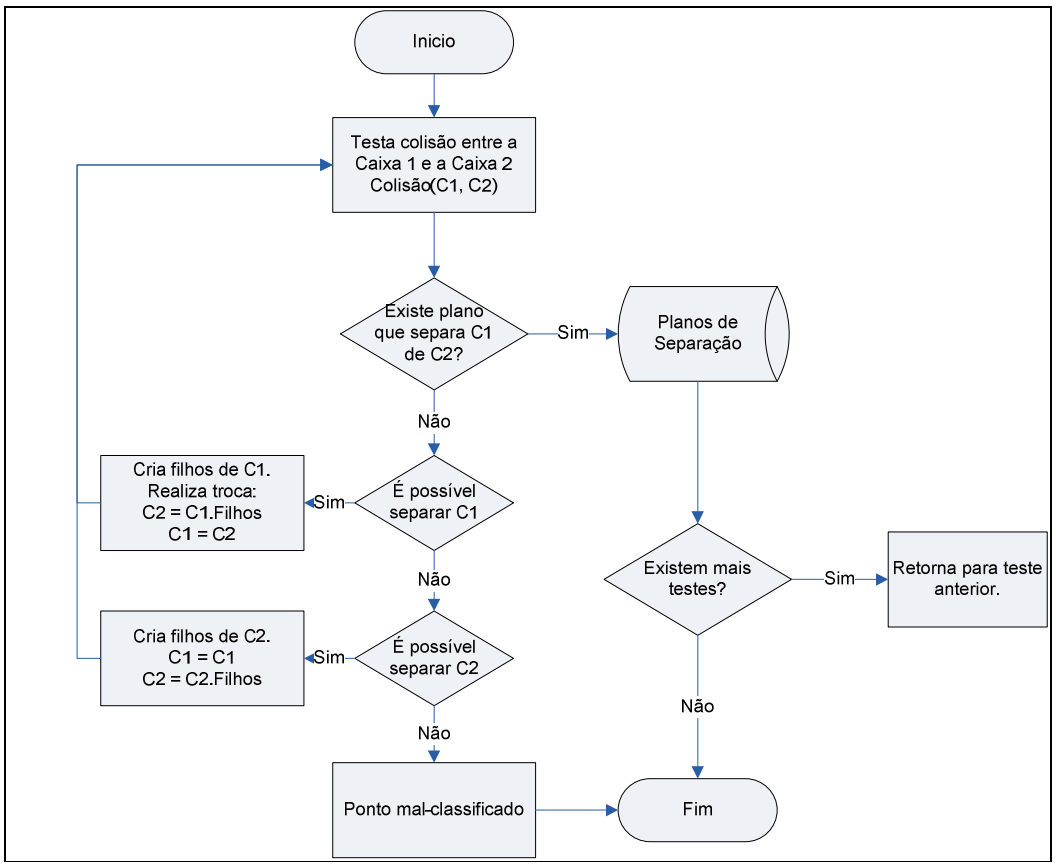


Figura 31 – Algoritmo para quebra das caixas na busca de planos de segmentação

Conforme pode ser visto na Figura 31 existe uma troca na posição das variáveis antes de chamar a função "Colisão(X,Y)". Isto garante que a cada chamada desta função uma hipercaixa diferente será dividida.

Na tabela 8 são apresentados os sete hiperplanos encontrados para separar as classes do exemplo apresentado. Na Figura 32 estes hiperplanos são apresentados de forma gráfica, assim como as divisões finais das hipercaixas.

Tabela 9 – Hiperplanos encontrados para as classes de exemplo

Hiperplano	Vetor normal	Ponto médio
P1	(−0,67 −0,74)	(0,63 0,12)
P2	(+1,00 0,00)	(0,44 0,05)
P3	(+0,64 −0,77)	(0,25 0,00)
P4	(+0,77 +0,64)	(0,32 0,24)
P5	(−0,67 −0,74)	(0,60 0,08)
P6	(−1,00 0,00)	(0,36 0,05)
P7	(+0,64 −0,77)	(0,22 0,04)

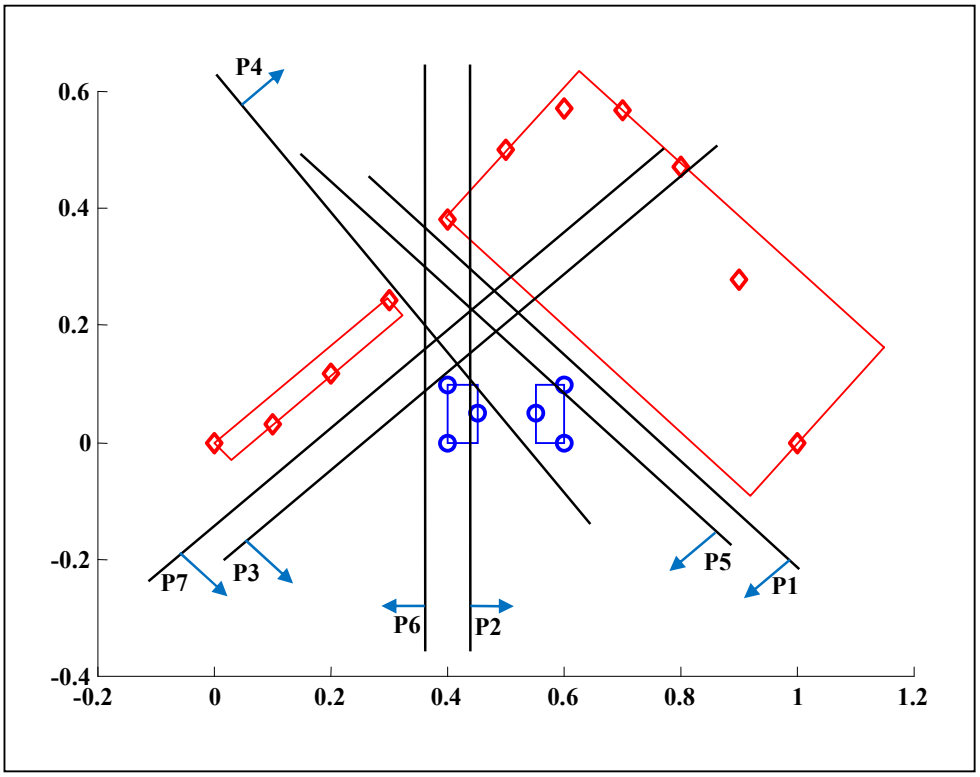


Figura 32 – Hiperplanos encontrados para o exemplo

3.4 Seleção dos hiperplanos

Após encontrar todos os hiperplanos é necessário realizar uma seleção com a finalidade de reduzir este número a uma quantidade que seja suficiente para separar os pontos utilizados para treinamento.

A partir da Figura 32 pode-se notar que os hiperplanos dividem o espaço em diversas regiões. Para que os hiperplanos consigam separar as classes corretamente, não pode haver pontos de classes distintas numa mesma região.

Para identificar essas regiões podemos rotulá-las com um vetor binário onde cada posição i, do vetor, representa em que lado do hiperplano i o ponto se encontra, sendo que o lado positivo ou superior é dado por aquele lado apontado pelo vetor normal. Desta forma podemos gerar uma matriz que identifica os pontos conforme apresentado na Tabela 10 para os pontos do tipo losango e na Tabela 11 para os pontos do tipo círculo.

Tabela 10 – Classificação dos pontos do tipo losango

Ponto	P1	P2	P3	P4	P5	P6	P7
1	+	−	−	−	+	+	−
2	+	−	−	−	+	+	−
3	+	−	−	−	+	+	−
4	+	−	−	−	+	+	−
5	−	−	−	+	−	−	−
6	−	+	−	+	−	−	−
7	−	+	−	+	−	−	−
8	−	+	−	+	−	−	−
9	−	+	−	+	−	−	+
10	−	+	+	+	−	−	+
11	−	+	+	+	−	−	+

Tabela 11 – Classificação dos pontos do tipo círculo

Ponto	P1	P2	P3	P4	P5	P6	P7
A	+	−	+	−	+	−	+
B	+	−	+	−	+	−	+
C	+	+	+	−	+	−	+
D	+	+	+	+	+	−	+
E	+	+	+	+	+	−	+
F	+	+	+	+	−	−	+

Para que as classes possam ser separadas, nenhum padrão (linha) da Tabela 10 pode repetir na Tabela 11. Com base nesta afirmativa, os hiperplanos são selecionados de forma que isso aconteça.

Para efetuar tal seleção inicia-se com o primeiro ponto da classe circulo (ponto A) e o primeiro da classe losango (ponto 1). Procura-se o primeiro hiperplano que os classifique de maneira diferente. O hiperplano P1 não satisfaz essa afirmação, pois classifica o ponto A e o ponto 1 como positivos. P2 também não pode ser selecionado pois ambos os pontos são classificados como negativos. O primeiro hiperplano que efetivamente separa os dois é P3.

Assim que o primeiro ponto for separado, incrementamos a posição passando para o segundo ponto, nesse caso o ponto 2. Neste momento é verificado se eles (A e 2) já não estão separados pelos hiperplanos selecionados anteriormente. Por já estarem separados prosseguimos comparando o ponto A com os demais: 3, 4, ... 10. Ao chegar no décimo ponto, é possível perceber que eles (A e 10) estão classificados da mesma maneira (positivo). Procura-se então um hiperplano que os separe. Selecionamos, por causa da ordem crescente, o plano P1. Agora o ponto A é classificado como (+,+) e o ponto 10 como (+,-).

Quando o primeiro ponto do primeiro grupo estiver completamente separado do segundo grupo o procedimento é refeito para o segundo ponto. Neste exemplo, apenas estes

dois hiperplanos, P3 e P1, são necessários para separar todos os pontos, conforme podemos visualizar na Figura 33.

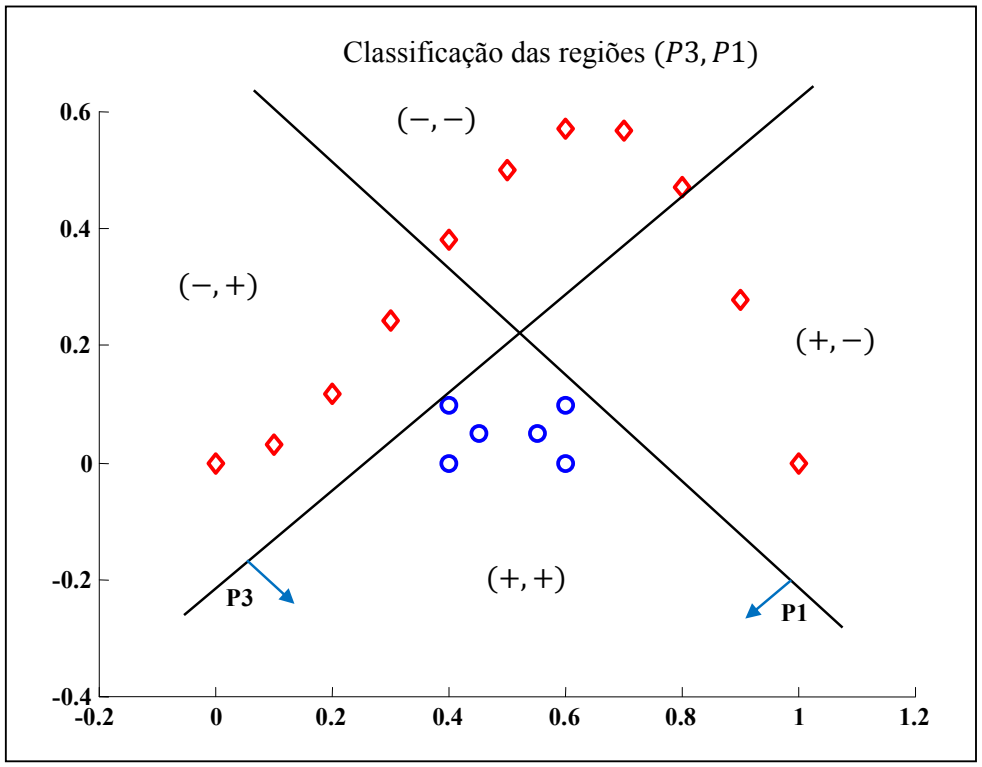


Figura 33 – Pontos do exemplo corretamente separados

Uma matriz reduzida é gerada a partir das matrizes apresentadas na Tabela 10 e na Tabela 11, utilizando apenas as colunas dos hiperplanos selecionados. Esta nova matriz possuirá diversas linhas idênticas. Retirando as linhas redundantes obtemos os seguintes padrões compilados na Tabela 12.

Tabela 12 – Padrões para os pontos das classes de exemplo

Tipo	Padrões
1 – Losangos	(+, −)
	(−, +)
	(−, −)
2 - Círculos	(+, +)

Estes padrões serão utilizados na montagem da rede neural para identificar as regiões delimitadas para cada grupo.

3.5 Geração da rede neural

No item 2.1.5, foi demonstrado que um neurônio representa um plano e cuja saída positiva representa o semi-espaço positivo, ou acima do plano, e uma saída negativa o semi-espaço negativo, ou abaixo do plano. Esta análise é facilmente generalizada, sendo que basta adicionar entradas a um neurônio para que ele possa representar corretamente um hiperplano. É possível então utilizar uma rede neural com apenas duas camadas para implementar o comportamento estabelecido na etapa anterior.

Na primeira camada os neurônios representam os hiperplanos que foram selecionados através da técnica de detecção de colisões, utilizando o teorema de eixos de separação. Já a segunda camada será a responsável por indicar em qual região o ponto se encontra, utilizando as informações derivadas dos padrões encontrados.

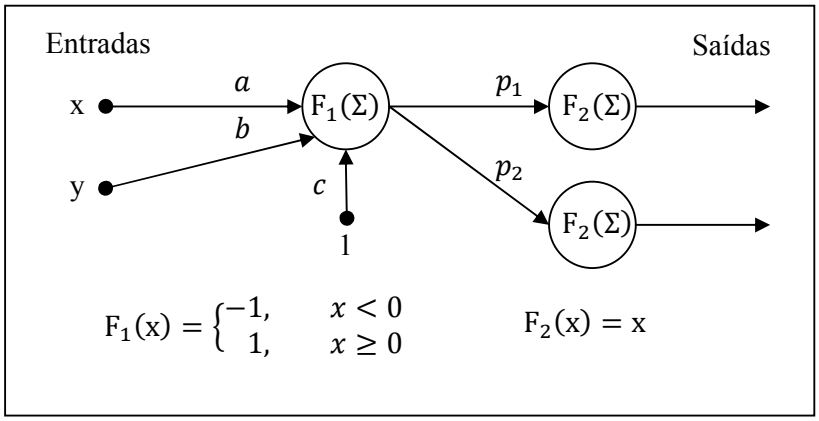


Figura 34 – Criação de uma rede neural com base nos hiperplanos de segmentação

Os valores a, b e c são obtidos através de um hiperplano P, que é definido por um vetor normal $\vec{N}$ e um ponto $\overrightarrow{P_{med}}$. Os valores de a e b são as componentes do vetor $\vec{N}$. O valor do peso c, ou *bias* é dado por $-\left(\overrightarrow{P_{med}} * \left(\vec{N}\right)^T\right)$. Para um hiperplano N-dimensional existem N entradas na rede neural, uma para cada dimensão, com peso igual à componente do vetor $\vec{N}$. naquela dimensão. O peso do *bias* continuará obedecendo à fórmula apresentada.

O valor dos pesos que ligam a primeira camada com a segunda, (p_1, p_2) podem ser positivos ou negativos e são dados pelo padrão correspondente à região que sensibiliza àquele neurônio. A magnitude destes valores é inversamente proporcional a quantidade de entradas que o neurônio da segunda camada possui. Por exemplo, caso o neurônio, possuindo três

entradas, seja sensibilizado por uma região $(+,+,-)$, os pesos (p_1, p_2, p_3) deste neurônio serão, respectivamente, $(+1,+1,-1) * \frac{1}{3} = (+0.33,+0.33,-0.33)$.

Desta forma garante-se que quando um ponto cair em uma região completamente conhecida, o neurônio responsável por aquela região indica 1 em sua saída. Caso o ponto recaia numa região não identificada, o neurônio que possuir o maior valor indicará a região mais próxima daquela, caracterizando assim a generalização do conhecimento.

Através dos resultados apresentados na Tabela 9 e na Tabela 12 pode-se montar a rede neural apresentada na Figura 35.

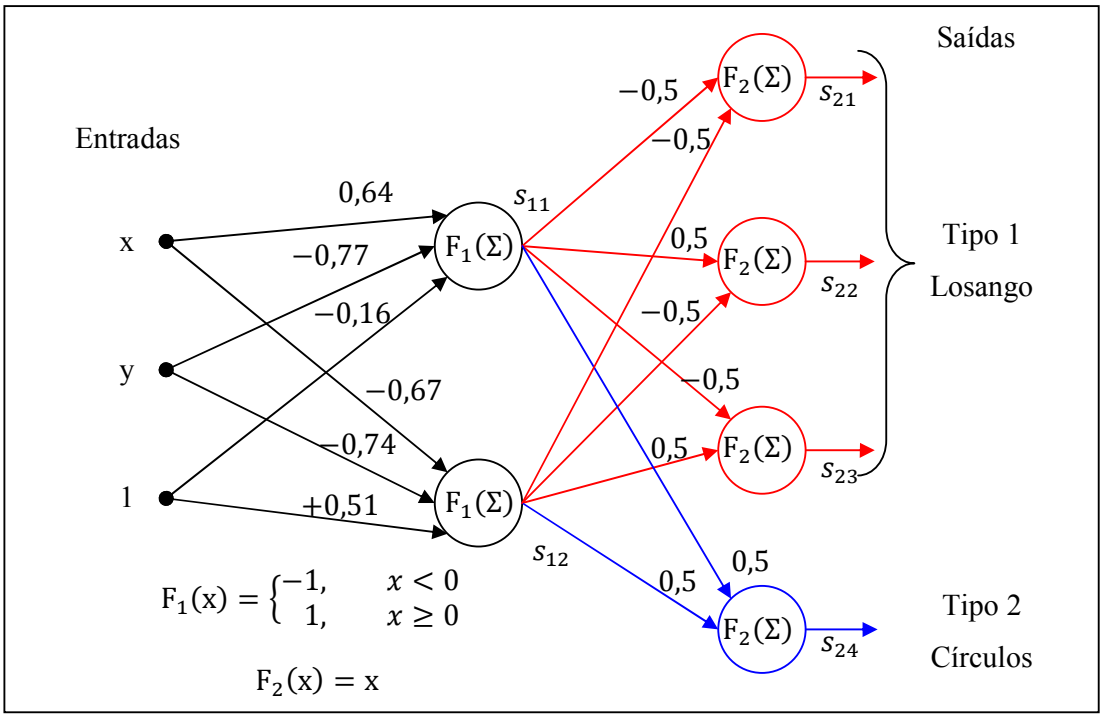


Figura 35 – Rede neural gerada para o exemplo apresentado

A Tabela 13 apresenta as contas para alguns pontos, com intuito de validar o funcionamento da rede neural gerada. Podemos notar que todos os pontos foram corretamente classificados.

Tabela 13 – Classificação de alguns pontos do exemplo através da RNA gerada

Ponto	x	y	Σn_1	s_{11}	Σn_2	s_{12}	s_{21}	s_{22}	s_{23}	s_{24}	Classificação
A	0,40	0,10	0,02	1	0,17	1	-1	0	0	**1**	Circulo
D	0,55	0,05	0,16	1	0,10	1	-1	0	0	**1**	Circulo
E	0,60	0,00	0,22	1	0,11	1	-1	0	0	**1**	Circulo
1	0,00	0,00	-0,16	-1	0,51	1	0	-1	**1**	0	Losango
4	0,30	0,24	-0,15	-1	0,13	1	0	-1	**1**	0	Losango
8	0,70	0,57	-0,16	-1	-0,37	-1	**1**	0	0	-1	Losango
11	1,00	0,00	0,48	1	-0,16	-1	0	**1**	-1	0	Losango

3.6 Modificações para problemas com mais de uma classe

O algoritmo apresentado foi desenvolvido para utilização com duas classes apenas. Para adaptar classificadores binários para sistemas multi-classes, existem algumas maneiras apresentadas pela literatura (Chih, et al., 2002) (Ding, et al., 2001), entre elas uma parece ser apropriada e facilmente implementada: *one-against-all.*

Nesta técnica são gerados N problemas, onde N é o número de classes que se desejamos separar. Em cada subproblema elege-se uma classe principal diferente. As demais classes serão agrupadas sob um único rótulo e o problema torna uma forma binária. Ao fim do procedimento teremos N redes neurais, cada qual especializada para uma determinada classe.

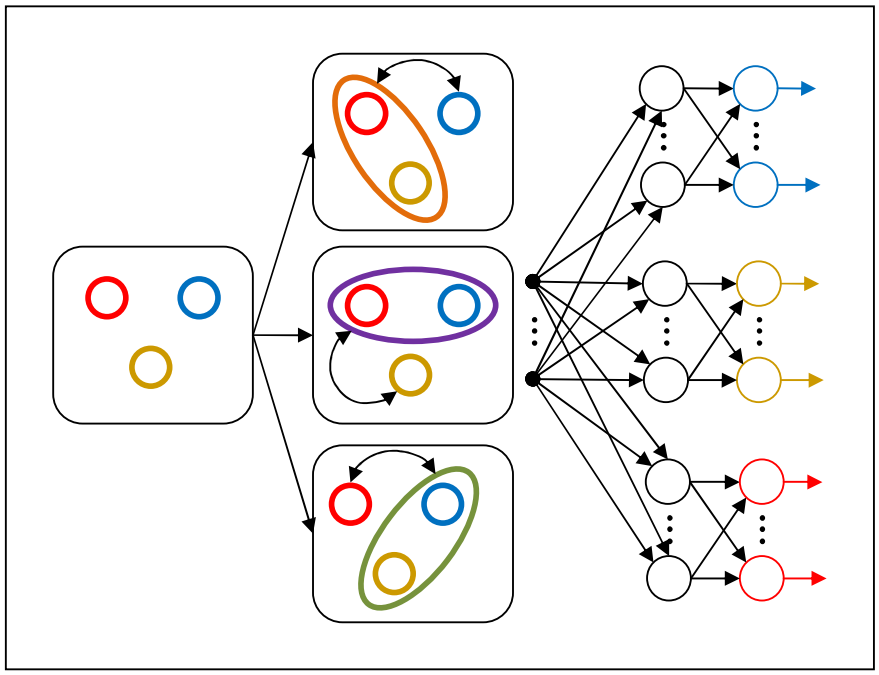


Figura 36 – Técnica de classificação multi-classe: *one-against-all*

Para cada subproblema geramos uma RNA que possui neurônios que são ativados quando o ponto pertence à classe principal e neurônios que são ativados quando este pertence ao conjunto das demais classes. Apenas os neurônios gerados que se sensibilizam com a classe principal de cada subproblema são mantidos, os demais são descartados.

3.7 Regiões de incerteza

Para que um plano possa ser encontrado através do teste de colisão, é necessário que exista um espaço entre as caixas. Como as caixas delimitam a região que a classe ocupa, qualquer espaço fora das caixas possui uma incerteza associada.

Os dados utilizados representam, geralmente, apenas uma amostra de todas as opções possíveis, por isso é uma tarefa difícil decidir em que ponto o plano deverá passar. Nas considerações feitas até o momento optou-se por colocar os planos passando pelo ponto médio do espaço que divide as caixas. Uma alternativa mais interessante é definir uma região de incerteza, *Uncertain Region* ou UR, onde o ponto não será mais classificado apenas como positivo ou negativo, mas receberá um valor representando a incerteza associada à sua posição em relação às caixas conforme pode ser visto na Figura 37.

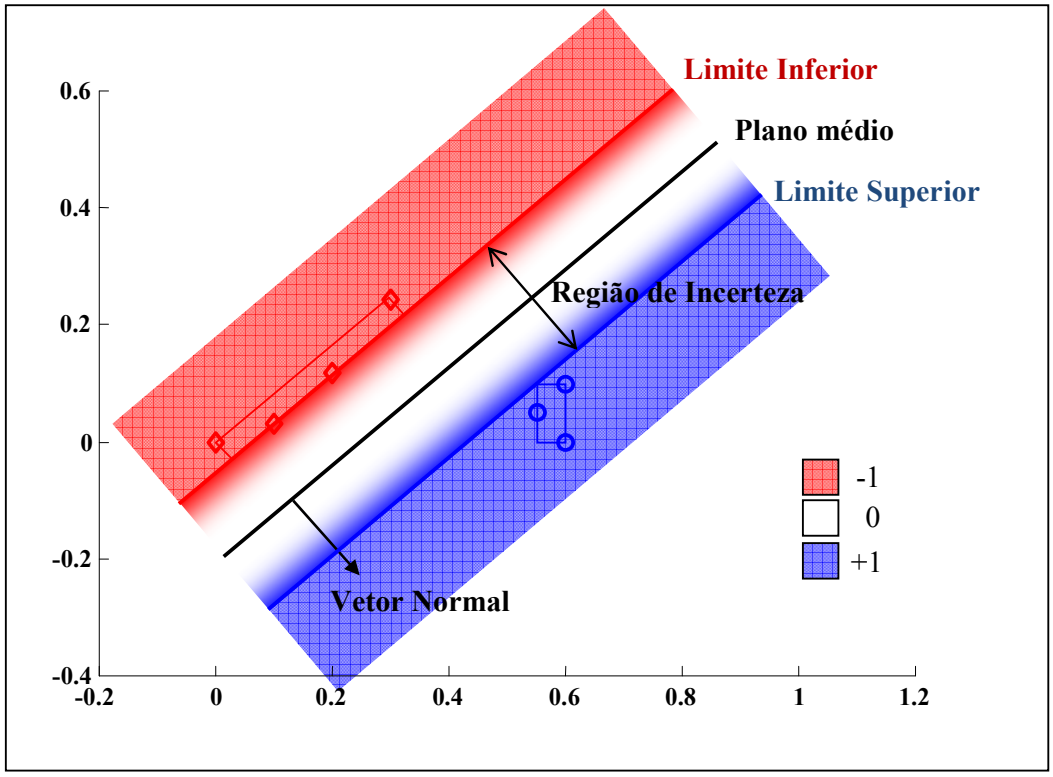


Figura 37 – Definição da área de incerteza

Com esta alteração a saída dos neurônios da primeira camada poderá variar linearmente entre +1 e -1 dentro da região de incerteza. Acima do limite superior, *uper bound* ou UB, que é indicado pelo vetor normal, os pontos recebem valor +1, representando 100% de certeza que o ponto pertence à classe positiva. Procedimento análogo é realizado para os pontos além do limite inferior, *lower bound* ou LB.

Utilizando a região de incerteza, os neurônios de saída poderão indicar valores intermediários, indicando quanto um ponto está próximo da região delimitada por aquele.

Os limites superiores e limites inferiores são definidos à mediada que os planos são encontrados no teste de colisão.

3.8 Qualidade e classificação dos hiperplanos

Como estamos buscando um conjunto mínimo de hiperplanos que melhor classifique os conjuntos de dados, podemos elencar os planos de forma que a escolha deste conjunto priorize os melhores hiperplanos.

Duas métricas nos auxiliam no procedimento de definir quão bom é um plano: a largura da região de incerteza e quantos pontos o plano consegue classificar corretamente.

Maximizar a distância entre os limites pode ajudar a melhorar a capacidade de generalização da rede o que é justamente o objetivo dos classificadores SVM's. Utilizando esta opção os planos são classificados de forma decrescente em relação à largura da região de incerteza. Esta técnica foi denominada Maximização de Distância entre Limites ou MDL.

É esperado que um plano P, dividindo o espaço em dois, consiga colocar todos os dados de um grupo A de um lado, e os do grupo B do outro. Partindo desta premissa, testamos todos os pontos contra o plano P de forma que PA+ é a porcentagem de pontos da classe A que foram classificados como positivos e PA- indica a porcentagem que foi rotulada como negativo. Os mesmos valores são calculados para a classe B: PB+ e PB-. A magnitude deste índice I é calculada pela Eq. 14.

$$I = Max(PA+, PA-) * Max(PB+, PB-) \qquad \textbf{Eq. 14}$$

O sinal deste índice I é positivo quando a maioria dos pontos de A, está de um lado do hiperplano e a maioria dos pontos de B, do outro lado.

Com esta definição I assume +1 quando o plano consegue separar totalmente as classes e -1 quando todos os pontos se encontram do mesmo lado do plano. Esta técnica foi denominada Minimização de BiClassificação ou MBC.

Quando as técnicas são utilizadas em conjunto o melhor plano é aquele que possuir a melhor distância classificando corretamente os pontos, deste modo ponderamos o incide MDL multiplicando-o por MBC.

3.9 Considerações sobre o teste de colisão

Existem situações onde a complexidade é muito grande, sendo necessário descer muitos níveis até obter a precisão desejada. Nestas situações pode ser inviável, computacionalmente, realizar todos os cálculos. Por isso inserimos um controle de nível na decisão dos testes de colisão. Quando o nível crítico é atingido, são calculados os hiperplanos daquele nível e caso exista alguma colisão esta não é tratada e o algoritmo retorna para os níveis anteriores.

Quando a busca atinge o limite de profundidade especificado, alguns pontos não poderão ser separados pelos planos encontrados. Deste modo é preciso realizar algumas alterações na etapa de seleção de planos.

A alternativa encontrada é ignorar o ponto em questão caso este não possa ser separado por nenhum hiperplano. Deste modo não teremos 100% de acerto para os pontos utilizados no treinamento, mas garantiremos uma resposta em tempo hábil, tão mais precisa quanto maior o nível crítico escolhido.

Quando utilizamos a região de incerteza, existe uma dificuldade maior para garantir 100% de acerto para o banco de testes, pois nem sempre os pontos recaem em regiões completamente classificadas. Para minimizar esse problema ajustamos os limites superiores e inferiores no momento de seleção dos hiperplanos.

4 Resultados

4.1 Banco de dados utilizados

Para validação da técnica desenvolvida é necessário testar o modelo com bancos de dados conhecidos. Neste estudo foram utilizados seis bancos de dados obtidos do repertório disponibilizado pela UCI (Asuncion, et al., 2007), sendo que dois destes são de autoria da Statlog. Estes bancos foram escolhidos por serem utilizados na literatura pesquisada (Chih, et al., 2002) e (Fung, et al., 2005). Na Tabela 14 são apresentadas algumas informações destes bancos de dados.

Tabela 14 – Banco de dados utilizados no teste

Nome	Origem	Qntd. de classes	Qntd. de dados	Atributos
Iris	UCI Database	3	150	4
Glass	UCI Database	6	214	9
Wine	UCI Database	3	178	13
Vowel	UCI Database	10	528	11
Image Segmentation	Statlog Database	7	2310	19
Vehicle Silhouettes	Statlog Database	4	846	18

Dividimos também os bancos de dados em dois grupos, um considerado simples e outro complexo. Esta divisão foi realizada levando em conta a quantidade de classes e de atributos de cada banco. O algoritmo OBHB cresce linearmente com o número de atributos das classes, com o número de classes e com o número de instâncias. De base destes valores e observando os resultados obtidos, em termos de tempo, *Iris*, *Vehicle* e *Wine* são considerados simples e *Vowel*, *Image Segmentation* e *Vehicle Silhouettes*, complexos.

4.1.1 Iris

O objetivo do banco de dados Iris é classificar corretamente uma flor entre três espécies diferentes utilizando quatro parâmetros: comprimento e largura da pétala, comprimento e largura da sépala. São três espécies de flores: Iris-Setosa, Iris-Virgínica e Iris-Versicolor. Este é um dos problemas mais conhecidos da literatura de reconhecimento de padrões. O primeiro artigo publicado é “The use of multiple measurements in taxonomic problems” de autoria de RA Fisher no Annals of Eugenics em 1936. Este é um problema de

fácil resolução, pois uma classe (*Iris Setosa*) é linearmente separável das demais e as duas últimas são facilmente separadas utilizando poucos hiperplanos(Asuncion, et al., 2007).

4.1.2 Glass

No banco de dados *Glass*, classificamos um tipo de vidro entre seis classes diferentes. Os atributos de cada instância são: índice de refração, porcentagem de sódio, magnésio, alumínio, silício, potássio, cálcio, bário e ferro. Este é o banco de dados que apresenta piores resultados nas literaturas pesquisadas. Este fato se deve pelo fato das classes possuírem muitas regiões de interseção em todas as variáveis de análise.

4.1.3 Wine

O banco de dados *Wine* visa determinar o tipo de vinho com base nos atributos: quantidade de álcool, acido málico, cinzas, alcalinidade da cinza, magnésio, total de *phenols*, quantidade de flavanóides, quantidade de *phenols* não flavanóides, intensidade e saturação da cor, OD280/OD315 dos vinhos diluídos e *proline*. Existem três tipos de vinhos no banco de dados utilizado. Este é outro banco de dados que gera bons resultados de classificação segundo a literatura.

4.1.4 Vowel

O banco de dados *Vowel* tem o intuito de classificar o som entre 10 tipos de vogais: /i/, /O/, /I/, /C:/, /E/, /U/, /A/, /u:/, /a:/, /3:/, /Y/, segundo notação fonética internacional. Foram coletadas amostras destas vogais entre 15 pessoas. Estes sinais foram amostrados por um filtro passa baixa de 4.7kHz e digitalizados para 12 bits com uma taxa de amostragem de 10kHz. Uma análise preditiva linear de decimal segunda ordem foi realizada em seis segmentos de 512 amostras numa janela de Hamming, retirada da seção estável da vogal. Os coeficientes de reflexão foram usados para calcular dez parâmetros de área logaritmos, retornando um espaço de entrada de dez dimensões.

Para cada vogal existem seis trechos de fala diferente de cada pessoa num total de 90 amostras em cada classe.

4.1.5 Image Segmentation

Os dados do banco *Image Segmentation* foram retirados de sete imagens externas. Estas imagens foram segmentadas manualmente para classificar cada pixel conforme a região que este pertencia: pedra, céu, folhagem, cimento, janela, caminho e grama. A imagem foi então redividida em blocos de 3x3 pixels. Com base nestes blocos e na região que o pixel pertence foram gerados os seguintes valores:

1. region-centroid-col: a coluna do pixel central da região;
2. region-centroid-row: a linha do pixel central da região;
3. region-pixel-count: o número de pixels na região (sempre igual a nove);
4. short-line-density-5: quantas linhas de tamanho cinco ou menor (qualquer orientação), com baixo contraste, que atravessa a região;
5. short-line-density-2: igual ao algoritmo short-line-density-5 mas conta as linhas de alto contraste com tamanho maior que cinco;
6. vedge-mean: mede o contraste dos pixels adjacentes horizontais da região, que são seis. São apresentados a média e o desvio padrão. É utilizada como detecção de bordas verticais;
7. vegde-sd: desvio padrão do item 6;
8. hedge-mean: similar ao item 6 mas para pixels horizontais;
9. hedge-sd: desvio padrão do item 8;
10. intensity-mean: valor médio da região = (R + G + B)/3;
11. rawred-mean: valor médio da componente vermelha da região;
12. rawblue-mean: valor médio da componente azul da região;
13. rawgreen-mean: valor médio da componente verde da região;
14. exred-mean: medida do excesso de vermelho = (2R - (G + B));
15. exblue-mean: medida do excesso de azul = (2B - (G + R));
16. exgreen-mean: medida do excesso de verde = (2G - (R + B));
17. value-mean: primeira componente da transformação não-linear dos valores RGB em VSH. (Algoritmo pode ser encontado em Foley and VanDam, Fundamentals Interactive Computer Graphics)
18. saturatoin-mean: segunda componente do item 17.
19. hue-mean: terceira componente do item 17.

O objetivo do banco de dados é classificar corretamente cada bloco 3x3, indicando a qual região/classe ele pertence.

4.1.6 Vehicle Silhouettes

O banco de dados *Vehicle Silhouettes* visa reconhecer qual o tipo de veículo está representado na imagem bidimensional apresentada. São quatro tipos de veículos representados: opel(modelo de carro), saab(modelo de carro), ônibus e vans. Estes veículos foram escolhidos com a expectativa que as vans e ônibus fossem rapidamente diferenciados dos carros. As imagens foram adquiridas por uma câmera olhando para baixo com um ângulo fixo de 34.2 graus com a horizontal. Os veículos foram colocados num fundo iluminado difuso. Os veículos foram pintados de preto fosco para minimizar reflexos. As imagens foram capturadas utilizando uma câmera CRS4000 com resolução de 128x128 pixels com 64 níveis de cinza. As imagens foram binarizadas e processadas digitalmente para retirada de ruídos. Destas imagens foram retiradas 18 características.

4.2 Metodologia de teste

Conforme adotado por (Chih, et al., 2002) e (Fung, et al., 2005) foi utilizado *tenfold* como regra para apresentar os resultados obtidos, através das redes neurais geradas, para os bancos de dados selecionados.

Nesta metodologia, o banco de dados é dividido em dez partes. Para realizar esta divisão cada classe foi dividida aleatoriamente em dez, de modo que cada parte do banco de dados geral possua aproximadamente a mesma distribuição de dados que o banco original. Uma destas partes do banco é então separada e armazenada conforme pode ser visualizado na Figura 38. Com as nove partes restantes a rede neural é construída/treinada com base na técnica utilizada. A rede é então testada com aqueles dados separados inicialmente.

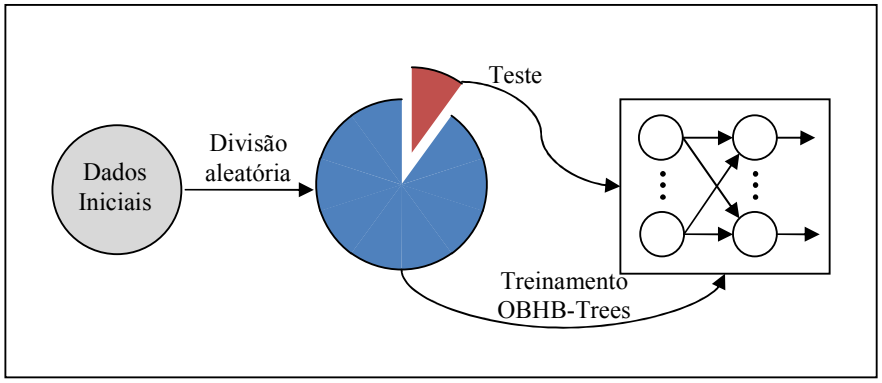


Figura 38 – Método de divisão dos dados para treino/teste

Para garantir que a rede será testada contra todos os pontos do banco de dados, este procedimento é repetido com cada uma das dez partes conforme se pode visualizar na Figura 39. Apresentam-se então os resultados, com base nas dez medidas.

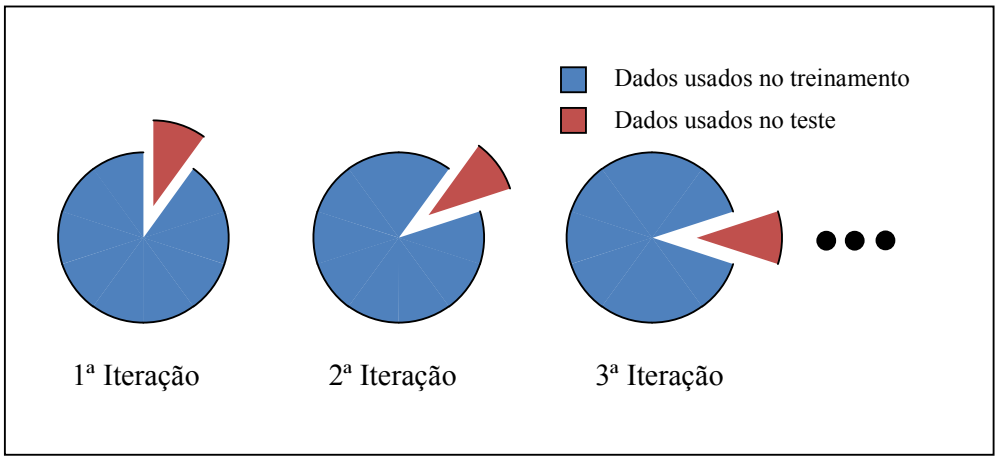


Figura 39 – Metodologia de teste por *tenfold*

4.3 Testes

Foram testados todos os bancos de dados utilizando *ten fold* com a técnica proposta: OBHB. A Tabela 15 apresenta os resultados obtidos para o banco de dados Iris. Nela a coluna "Profundidade" indica até que nível as hipercaixas foram quebradas. Já a coluna "Planos" indica a quantidade de planos que foram necessários para separar as classes em questão, ou seja, a quantidade de neurônios na primeira camada da rede. A coluna Padrões indica a quantidade de regiões foram classificadas, referenciando também a quantidade de neurônios necessários na segunda camada da rede neural.

Tabela 15 – Resultados obtidos para *Iris* com OBHB

Classificação dos pontos do treinamento		Classificação dos pontos do teste		Profundidade	Planos	Padrões
Corretos	Errados	Corretos	Errados			
135	0	15	0	5	22	39
135	0	15	0	5	22	33
135	0	15	0	6	18	40
135	0	15	0	6	20	37
135	0	15	0	5	14	27
135	0	15	0	6	22	39
135	0	14	1	5	21	42
135	0	15	0	6	20	35
135	0	14	1	5	22	38
135	0	14	1	5	21	32

Pela Tabela 15 é possível notar que a rede neural gerada para a classe Iris se mostrou eficiente, acertando todos os dados utilizados no treinamento. Nos dados de testes a rede apresentou uma média de acerto de 98%, sendo que em 7 casos esse valor foi de 100%.

Na Tabela 16, é apresentado um resumo dos resultados obtidos para todos os bancos de dados utilizando a técnica proposta: OBHB. Em todos os testes obtivemos 100% de acerto para os dados utilizados no treinamento. É possível notar que apenas os bancos de dados *Iris* e *Wine* apresentaram boas taxas de detecção chegando, em alguns casos, a 100% de acerto. Os demais possuíram taxas mais baixas que podem ser explicados de duas maneiras: bancos de difícil classificação e má capacidade de generalização da rede.

Tabela 16 – Resultados obtidos com OBHB

	Iris	*Glass*	*Wine*	*Vowel*	*Segmentation*	*Vehicle*
Máximo	100,0%	78,9%	100,0%	88,9%	74,5%	69,5%
Média	98,0%	68,0%	95,0%	78,8%	68,7%	63,6%
Mínimo	93,3%	52,6%	87,5%	68,9%	62,8%	55,6%

Para reconhecer em qual dos dois grupos os bancos se encaixam comparamos os resultados obtidos com alguns apresentados na literatura, (Fung, et al., 2005) e (Chih, et al., 2002), através da Tabela 17.

O banco de dados *Glass* se encaixa no primeiro problema. Foi obtida uma taxa de acerto comparável com a dos artigos, de modo que o banco pode ser definido como de difícil classificação.

Os bancos de dados *Vowel* e *Segmentation* possuem boas taxas de acerto nestes artigos, o que nos indica que o problema deva ser com o algoritmo apresentado.

O banco de dados *Vehicle* possui ambas as características. A literatura verificada não possui uma taxa de acerto muito grande para ele. Mesmo assim nosso resultado ainda esta muito aquém do esperado.

Tabela 17 – Resultados de OBHB versus literatura

	Iris	*Glass*	*Wine*	*Vowel*	*Segmentation*	*Vehicle*
OBHB	98,0%	68,0%	95,0%	78,8%	68,7%	63,6%
(Chih, et al., 2002)	97,33%	73,8%	99,44%	99,05%	97,58%	87,47%
(Fung, et al., 2005)	98,7%	72,9%	100,0%	98,5%	97,0%	82,2%

4.4 Modificações e testes adicionais

Analisando os primeiros resultados foi notado que bancos de dados maiores apresentam resultados inferiores aos apresentados na literatura. No intuito de resolver esta situação foram apresentadas duas possibilidades de modificação no sistema básico: utilização da região de incerteza e classificação dos melhores planos. Estas melhorias foram propostas com o intuito de melhorar a capacidade de generalização da rede e aproveitar melhor as informações advindas da seção de busca dos hiperplanos.

Para a melhoria no método de seleção de planos foram propostas duas alternativas: Maximização da Distância entre os Limites da região de incerteza, MDL, e a Minimização da BiClassificação dos pontos, MBC.

Na Tabela 18 apresentamos os resultados obtidos pelo sistema proposto combinando MDL, MBC e UR. São apresentados também os melhores resultados obtidos por (Fung, et al., 2005) e (Chih, et al., 2002), que utilizam SVM como técnicas de classificação. Optou-se por escolher estes trabalhos dado a atualidade dos resultados e o grande progresso que as técnicas de SVM vêm conseguindo na área de Reconhecimento de Padrões (Mavroforakis, et al., 2006). Em negrito temos os melhores resultados gerais, e em itálico os melhores entre os propostos.

Tabela 18 – Tabela comparativa de resultados

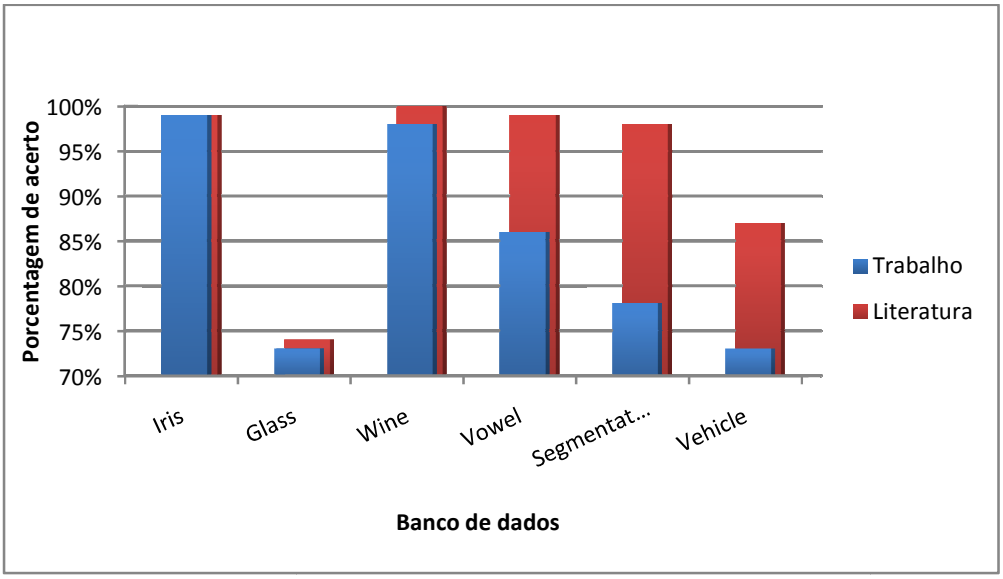
OBHB	*Iris*	*Glass*	*Wine*	*Vowel*	*Segmentation*	*Vehicle*
Original	98,0%	68,0%	95,0%	78,8%	68,7%	63,6%
MDL	***98,7%***	*73,1%*	92,3%	80,3%	61,9%	50,9%
MBC	***98,7%***	69,0%	*97,5%*	83,4%	74,9%	*72,6%*
MDL+MBC	98,0%	69,9%	94,4%	84,0%	63,0%	68,1%
UR	***98,7%***	69,6%	95,6%	82,7%	70,8%	67,9%
UR+MDL	***98,7%***	70,0%	96,6%	82,2%	69,8%	66,0%
UR+MBC	***98,7%***	70,6%	97,2%	85,1%	*78,1%*	71,2%
UR+MDL+MBC	***98,7%***	70,8%	97,2%	*85,6%*	66,7%	69,9%
(Chih, et al., 2002)	97,33%	**73,8%**	99,44%	**99,05%**	**97,58%**	**87,47%**
(Fung, et al., 2005)	**98,7%**	72,9%	**100,0%**	98,5%	97,0%	82,2%

Quanto à taxa de acerto para os dados usados no treinamento, no banco de dados *Segmentation*, utilizando as técnicas MBC/UR, foi obtido 99,97%. Em todos os demais casos esta taxa foi de 100%.

Para os bancos de dados *Iris*, *Glass* e *Wine* os melhores resultados obtidos ficam bem próximos aos melhores apresentados pela literatura, com resultado igual para *Iris*, 0,7%

menor para o Glass e 2,5% menor para *Wine*. Para os demais bancos a diferença aumenta quanto maior a complexidade do banco de dados a ser testado, chegando a 13,45%, 19,48% e 14,87% para *Vowel*, *Segmentation* e *Vehicle*, respectivamente. Estas diferenças podem ser observadas no Gráfico 1.

Gráfico 1 – Melhores taxas de acerto para os bancos de dados testados



Na Tabela 19 é apresentado o quanto cada alteração impactou nos resultados obtidos. A região de incerteza impactou de maneira positiva em todos os resultados. Isto indica uma melhoria na capacidade de generalização da rede neural quando utilizado UR.

Tabela 19 – Melhoria nos resultados através das alterações

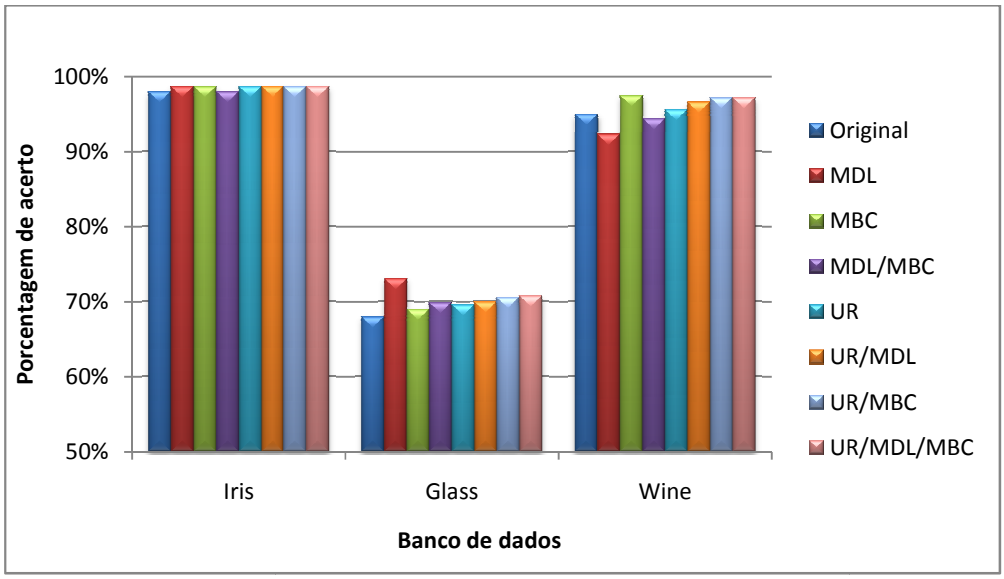
	Iris	Glass	Wine	Vowel	Segmentation	Vehicle
UR	0,35%	0,25%	1,85%	2,28%	4,23%	4,95%
MDL	0,00%	1,65%	-1,20%	0,53%	-7,78%	-5,10%
MBC	0,00%	-0,10%	1,70%	3,53%	2,88%	8,35%

A técnica MDL impactou negativamente em alguns resultados e isso se deve pelo fato que os planos com maior distância, apesar de permitirem uma maior capacidade de generalização, podem não refletir a melhor orientação para os dados apresentados. Conforme apresentado na Tabela 19, esta técnica não produziu melhorias em todos os bancos de dados, podendo prejudicar muito os resultados em alguns casos como *Vehicle,* onde a taxa de acerto do algoritmo normal é 13,7% maior que utilizando MDL.

Ao contrário da MDL, a técnica de seleção MBC apresentou melhoria dos resultados em praticamente todos os bancos de dados. Para bancos de dados mais simples, como Iris ou Glass, não houve melhoria, pois nestes bancos o algoritmo retorna poucos planos, não existindo deste modo uma grande necessidade de melhor classificá-los. Já para os bancos de dados mais complexos, onde a quantidade de planos encontrados é muito grande, a melhoria se apresenta de maneira mais significativa.

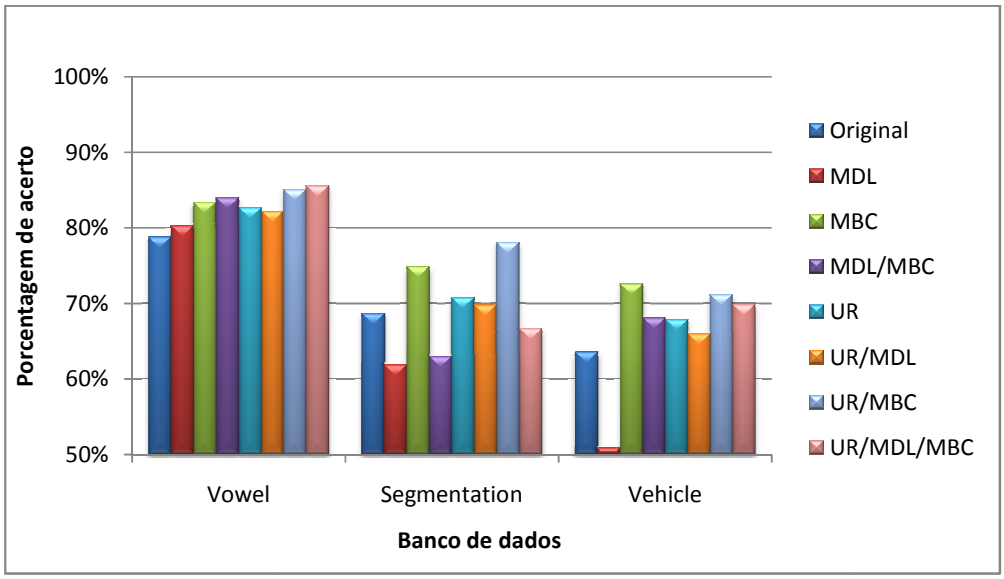
Para os bancos de dados mais simples percebemos que as alterações não tiveram um impacto significativo conforme pode-se visualizar no Gráfico 2, embora seja nítido que todos os melhores resultados foram gerados a partir de uma rede com alguma das modificações.

Gráfico 2 – Taxas de acerto das técnicas propostas para os bancos de dados simples



Nos bancos de dados mais complexos é nítida a diferença nos resultados utilizando as diferentes alterações, conforme demonstra o Gráfico 3. Novamente o melhor resultado foi obtido a partir de uma rede gerada com alguma das alterações.

Gráfico 3 – Taxas de acerto das técnicas propostas para os bancos de dados complexos



Conforme apresentado, principalmente no Gráfico 1, ainda existe um hiato considerável entre os bancos de dados complexos quando comparados os resultados obtidos com aqueles presentes na literatura. Foi demonstrado também que as alterações propostas, embora simples, melhoram os resultados. Desta maneira, técnicas mais eficazes para auxiliar a busca, seja ela inteligente ou determinística, podem impactar de maneira positiva.

5 Conclusão

Este trabalho apresentou uma nova técnica de treinamento de redes neurais, OBHB-Trees, baseada na teoria dos eixos de segmentação. Ela é capaz de definir a estrutura, tamanho e pesos das conexões de uma rede neural de forma analítica, baseado apenas nos dados de entrada, sem nenhum outro conhecimento à priori.

Com base nos resultados demonstrados, conclui-se que a técnica de especificação e treinamento de uma rede neural baseada em segmentação de elementos apresenta valores de acerto comparáveis com os apresentados pela literatura.

As melhorias apresentadas impactaram de maneira positiva nos resultados obtidos, em todos os bancos de dados estudados. As melhores alterações foram UR e MBC sendo que em cinco dos seis exemplos estudados os melhores resultados foram conseguidos a utilização de pelo menos uma destas alterações.

As técnicas apresentadas, para melhoria na seleção dos planos, podem ser modificadas utilizando outros algoritmos, como sistemas de análise combinatória ou buscas inteligentes. Existe também a possibilidade de não garantir a totalidade no acerto dos dados utilizados no treinamento, dando margem para redes mais generalistas.

A técnica utilizada para resolver problemas multi-classes (*one-against-all*) se mostrou satisfatória. A utilização de outras técnicas seria interessante para trabalhos futuros, visando um melhor aproveitamento do processamento, não desperdiçando parte do conhecimento adquirido.

6 Referências bibliográficas

Asuncion, A e Newman, D J. 2007. UCI Machine Learning Repository. [Online] University of California, School of Information and Computer Science., 2007. [Citado em: 23 de 10 de 2008.] http://www.ics.uci.edu/~mlearn/MLRepository.html. Irvine, CA.

Beale, R e Jackson, T. 1990. *Neural computing: an introduction.* s.l. : Hilger, 1990. p. 240.

Bose, N K e Garga, A K. 1993. Neural Network Desing Using Voronoi Diagrams. *IEEE Trasactions on Neural Networks.* Setembro de 1993, Vol. 4, 5, pp. 778-787.

Bourg, D M. 2002. *Physics for Game Developers.* Sebstopol : O'Reilly, 2002. p. 326.

Brudzewski, K., Osowski, S. e Markiewicz, T. 2003. Classification of milk by means of an electronic nose and SVM neural network. *Sensors and Actuators B: Chemical.* 2003, Vol. 98, pp. 291-298.

Chauvin, Yves e Rumelhart, David E. 1995. *Backpropagation: theory, architectures, and applications.* s.l. : Lawrence Erlbaum Associates, 1995. p. 561. ISBN 0805812598.

Chen, Y Q, Damper, R I e Nixon, M S. 1997. On Neural-Network Implementations of k-Nearest Neighbor Pattern Classifiers. *IEEE Transactions on Circuits and Systems—I: Fundamental Theory and Applications.* Julho de 1997, Vol. 44, 7.

Chih, Wei H. e Chih, Jen. L. 2002. A Comparison of Methods for Multiclass Support Vector Machines. *IEEE Transactions on Neural Networks.* Março de 2002, Vol. 13, 2, pp. 415-25.

Ding, Chris H Q e Dubchack, Inna. 2001. Multi-class protein fold recognization using support vetor machines and neural networks. *Bioinformatics.* Abril de 2001, Vol. 17, 4, pp. 349-358.

Fung, Glenn M e Mangasarian, O L. 2005. Multicategory Proximal Support Vector Machine Classifiers. [ed.] Shai Ben David. *Machine Learning.* 59, 2005, pp. 77-97.

Gentile, C e Sznaier, M. 2001. An Improved Voronoi-Diagram-Based Neural Net for Pattern Classification. *IEEE Transactions on Neural Networks.* Setembro de 2001, Vol. 12, 5, pp. 1227-1234.

Gottschalk, S. 2000. *Collision Queries using Oriented Bounding Boxes.* Departamento de Ciência da Computação, Universidade da Carolina do Norte. Chappel Hill : s.n., 2000. Tese de Doutorado.

Lippmann, R.P. 1989. Pattern classification using neural networks. *Communications Magazine, IEEE.* Novembro de 1989, Vol. 27, 11, pp. 47-50, 59-64.

Mangasarian, O L. 1968. Multisurface Method of Pattern Separation. *IEEE Transactions on Information Theory.* Novembro de 1968, Vol. 14, 6, pp. 801-807.

Mavroforakis, E Michael e Theodoridis, Sergios. 2006. A Geometric Approach to Support Vector Machine (SVM) Classification. *IEEE TRANSACTIONS ON NEURAL NETWORKS.* 17, Maio de 2006, Vol. 3, pp. 671-682.

McCulloch, W S e Pitts, W. 1943. A logical Calculus of the ideas immanent in nervous activities. *Journal Bulletin of Mathematical Biology.* Dezembro de 1943, Vol. 5, 4, pp. 115-133.

Mirchandani, G. 1989. On Hidden Nodes for Neural Nets. *IEEE Transactions on Circuits and Systems.* Maio de 1989, Vol. 36, 5, pp. 661-664.

Mukkamala, Srnivas, Janoski, Guadalupe e Sung, Andrew. 2002. Intrusion Detection Using Neural Networks and Support Vector Machines. *Proceedings of IEEE International Joint Conference on Neural.* 2002, pp. 1702-1707.

Piccinini, G. 2004. The First Computational Theory of Mind and Brain: A Close Look at Mcculloch and Pitts's “Logical Calculus of Ideas Immanent in Nervous Activity”. *Synthese.* Agosto de 2004, Vol. 141, 2, pp. 175-215.

S Gosttchalk, M C Lin, D Manocha. 1996. OBB-Tree:A Hierarchical Structure for Rapid Interference Detection. *ACM Siggraph'96.* 1996, pp. 171-180.

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo