Universidade Federal do Pampa

Autor: Eliezer Soares Flores

PROCESSAMENTO DIGITAL DE IMAGENS NA IDENTIFICAÇÃO DE VARIEDADES DE SOJA

Trabalho de Conclusão de Curso II

BAGÉ
2012

ELIEZER SOARES FLORES

PROCESSAMENTO DIGITAL DE IMAGENS NA IDENTIFICAÇÃO DE VARIEDADES DE SOJA

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Computação da Universidade Federal do Pampa, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Computação.

Orientador: Fábio Ronei Rodrigues Padilha

Bagé
2012

ELIEZER SOARES FLORES

PROCESSAMENTO DIGITAL DE IMAGENS NA IDENTIFICAÇÃO DE VARIEDADES DE SOJA

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Computação da Universidade Federal do Pampa, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Computação.

Trabalho de Conclusão de Curso defendido e aprovado em: 22 de novembro de 2012
Banca examinadora:

_____

Prof. Me. Fábio Ronei Rodrigues Padilha
Orientador
UNIPAMPA

_____

Profª. Drª. Ana Paula Lüdtke Ferreira
UNIPAMPA

_____

Prof. Dr. Sandro da Silva Camargo
UNIPAMPA

Dedico este trabalho aos meus amados pais, Ildefonso e Mariza, maiores incentivadores e fontes inesgotáveis de apoio, amor e compreensão.

AGRADECIMENTO

Ao Prof. Me. Fábio Ronei Rodrigues pela dedicação e paciência durante os quase dois anos de orientação.

Aos professores, minha gratidão pela forma de conduzir o curso em todas as etapas.

A todos os colegas pelo convívio e pelos momentos de amizade.

A todas as pessoas que, direta ou indiretamente contribuiram para a realização desta pesquisa.

É muito melhor arriscar coisas grandiosas, alcançar triunfos e glórias, mesmo expondo-se a derrota, do que formar fila com os pobres de espírito que nem gozam muito, nem sofrem muito, porque vivem nessa penumbra cinzenta que não conhece vitória nem derrota.

Theodore Roosevelt

RESUMO

Este trabalho apresenta uma alternativa para a caracterização de cultivares de soja, visando diminuir os gastos tipicamente agregados na realização desta tarefa. Para isso, foi utilizado um conjunto de técnicas de processamento digital de imagens e redes neurais artificiais, conforme sugerido em Padilha (2007). A diferença fundamental do presente trabalho é o uso da cor dos grãos como fator principal para o reconhecimento das variedades. O uso da cor teve como objetivo melhorar a precisão do processo. Neste sentido, foram utilizadas para estudo de caso as seguintes cultivares de soja: AL 83, BRS 133, BRS 184, BRS 214, CD 205, CD 206, CD 215, EMB 48, MERCEDES 70, MSOY 5826, NK 8350, RS 10 e MSOY 8000 RR. As imagens destas variedades foram fornecidas por Padilha (2007) e passaram por uma série de pré-processamentos até que os grãos individuais pudessem ser localizados de forma precisa nas imagens. Então, os coeficientes *RGB* dos grãos nas imagens forneceram os dados quantitativos necessários para alimentar uma rede neural. Por fim, diversas arquiteturas *feed forward* com uma camada oculta foram treinadas pelo algoritmo de *back propagation* e foi selecionado o modelo que mostrou uma melhor capacidade de generalização na solução do problema.

Palavras-chave: Identificação de variedades de soja. Processamento digital de imagens. Redes neurais artíficiais.

ABSTRACT

This work presents an alternative to the characterization of soybean cultivars in order to reduce spending typically aggregated in this task. For this, it was used a set of digital image processing techniques and artificial neural networks, as suggested in Padilha (2007). The fundamental difference of this work from the work of Padilha (2007) is the use of grain color as the main factor for the recognition of varieties. The use of color aims to improve the process. In this sense, it was used a case study for the following soybean cultivars: AL 83, BRS 133, BRS 184, BRS 214, CD 205, CD 206, CD 215, EMB 48, MERCEDES 70, MSOY 5826, NK 8350, RS 10 and MSOY 8000 RR. The images were supplied by Padilha (2007) and they were pre-processed until individual grains could be precisely located in the images. Then, the grain RGB coefficients in the images provided quantitative data needed to feed a neural network. Finally, several architectures of feed forward single hidden layer were trained by the back propagation algorithm and there was selected the model which shown better generalization ability to solve the problem.

Keywords: Characterization of soybean cultivars. Digital image processing. Artificial neural networks.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

a.C. – antes de Cristo

cm – centímetro

d.C. – depois de Cristo

Dr. – Doutor

Drª – Doutora

f. – folha(s)

m – metro

Me. – Mestre

n. – número

nm – nanômetro

nov. – novembro

p. – página(s)

Prof. – Professor

Profª. – Professora

s – segundo

v. – volume

LISTA DE SIGLAS

2-D – Duas Dimensões

3-D – Três Dimensões

COODETEC – Cooperativa Central de Pesquisa Agrícola

EMBRAPA – Empresa BRAsileira de Pesquisa Agropecuária

FEPAGRO – Fundação Estadual de Pesquisa e Agropecuária

LPC – Lei de Proteção de Cultivares

MATLAB – MATrix LABoratory®

MLP – Multi Layer Perceptron

RNA – Redes Neurais Artificiais

UNICAMP – Universidade Estadual de Campinas

UNIJUI - Universidade Regional do Noroeste do Estado do Rio Grande do Sul

UNIPAMPA – Universidade Federal do Pampa

LISTA DE SÍMBOLOS

$\vec{p}$ – ponto no plano *xy*

$f(\vec{p})$ – brilho, amplitude, tom ou nível de cinza no ponto $\vec{p}$

$(\vec{p}, f(\vec{p}))$ – pixel (*picture element*)

$\tau(x,y)$ – iluminância no ponto (x,y)

$\gamma(x,y)$ – refletância no ponto (x,y)

f_{min} – mínimo da função $f(\vec{p})$

f_{max} – máximo da função $f(\vec{p})$

$[f_{min}, f_{max}]$ – escala de cinza da imagem (*grayscale*)

ξ – branco em uma *grayscale* deslocada para $[0,\xi]$ e com $\xi+1$ níveis de cinza

M – número de linhas da matriz que representa uma imagem digital

N – número de colunas da matriz que representa uma imagem digital

κ – parâmetro de quantização

β – quantidade de bits necessários para armazenar uma imagem digital

R – quantidade(s) de vermelho (*red*) no modelo *RGB*

G – quantidade(s) de verde (*green*) no modelo *RGB*

B – quantidade(s) de azul (*blue*) no modelo *RGB*

c – velocidade da luz

λ – comprimento de onda

u – frequência

$|\nabla f|$ – módulo do gradiente

x_j – sinal ou estímulo na entrada *j*

w_{kj} – peso sináptico da entrada *j* em um neurônio *k*

$[0,1]$ – intervalo binário

$[-1,1]$ – intervalo bipolar

υ_k – potêncial de ativação

φ – função de transferência ou ativação

y_k – saída do neurônio *k*

a – parâmetro da função logística

b – parâmetro da função tangente hiperbólica

e_k – sinal de erro na saída do neurônio *k*

E – energia total do erro

d_k – resposta desejada para o neurônio *k*

Δw_{kj} – fator de sensibilidade

η – parâmetro da taxa de aprendizagem

SUMÁRIO

1 INTRODUÇÃO

1.1 Contextualização

Atualmente, são consumidos em todo o mundo aproximadamente 264 milhões de toneladas de soja por ano. O Brasil possui a segunda maior produção mundial, com uma safra estimada em 75 milhões de toneladas (EMBRAPA, 2011). Segundo o Ministério da Agricultura, Pecuária e Abastecimento (2011), dentre todas as culturas produzidas no país, a soja apresenta o maior número de variedades (cultivares) protegidas pela Lei n. 9.456/97, conhecida como Lei de Proteção de Cultivares (BRASIL, 1997).

O primeiro passo para proteger uma nova variedade é a sua identificação por meio de descritores (características morfológicas, fisiológicas, bioquímicas ou moleculares). De acordo com Millach (2005), a identificação baseada em descritores de DNA embora precisa requer um alto investimento por parte dos produtores e a caracterização baseada em descritores morfológicos é geralmente utilizada pelos melhoristas. Uma alternativa, proposta por Padilha (2007), foi a realização de um processamento de imagens digitais, que sob o auxílio de Redes Neurais Artificiais (RNA) tenta distinguir variedades de soja com base na forma e no tamanho dos grãos. Neste mesmo trabalho, foi concluído que a precisão do processo de identificação poderia ser melhorada por meio do uso de outros caracteres morfológicos, como por exemplo a cor dos grãos.

Visto que a maioria das cultivares de soja comercializadas possui coloração amarelada, a distinção destas através da cor pode ser uma tarefa complicada. Entretanto, um sistema de visão artificial (computador) muitas vezes percebe diferenças sutis, que passariam despercebidas pelo sistema visual humano (MARQUES FILHO; VIEIRA NETO, 1999). Koschan e Abidi (2008) evidenciam o fato de que a cor é costuma ser o descritor fundamental para o reconhecimento de imagens por meio de um sistema de visão artificial.

1.2 Objetivo

O objetivo deste trabalho é o desenvolvimento de um software, que baseado em um conjunto de técnicas de processamento de imagens e redes neurais artificiais possibilite a caracterização de variedades de soja com base na cor dos grãos.

2 REVISÃO BIBLIOGRÁFICA

2.1 Processamento de imagens

De acordo com Gonzales e Woods (2002, p. 1), o processamento de imagens digitais, ou simplesmente processamento de imagens pode ser definido como o processo de modificação de imagens digitais. Estas modificações podem ter por objetivo a interpretação humana ou ainda a percepção automática de máquinas. A definição de imagem digital é baseada no conceito de imagem monocromática ou simplesmente imagem, que é dado a seguir.

Uma imagem monocromática, também denominada imagem em tons (ou níveis) de cinza pode ser vista matematicamente como uma função f: $\mathbb{R}^2 \to \mathbb{R}$. Para cada ponto $\vec{p} = (x,y) \in \mathbb{R}^2$, f associa um valor $f(\vec{p}) = f((x,y)) \in \mathbb{R}$. Para simplificar a notação, os dois parênteses são substituídos por um só, assim, $f(x,y)$ denota $f((x,y))$ que é denominado brilho, amplitude, tom ou nível de cinza da imagem no ponto $\vec{p}$.

O par ordenado $(\vec{p},f(\vec{p}))$ é normalmente referido como pixel, sendo pixel uma abreviação de *picture element* (elemento da figura). Uma imagem monocromática f pode ser definida em termos de duas componentes: a iluminância, isto é, a quantidade de luz que incide sobre um objeto em um ponto $\vec{p}$ e a fração desta luz incidente que o objeto reflete, denominada refletância ou transmitância.

Sejam τ: $\mathbb{R}^2 \to \mathbb{R}$ e γ: $\mathbb{R}^2 \to \mathbb{R}$ duas funções, representando respectivamente as componentes de iluminância e refletância de uma cena qualquer. Podemos definir uma imagem monocromática como:

$$f(x,y) = \tau(x,y).\gamma(x,y) \tag{2.1}$$

onde:

$$0 < \tau(x,y) < \infty \tag{2.2}$$

$$0 < \gamma(x,y) < 1 \tag{2.3}$$

Se existirem os valores mínimo e máximo da função, $f_{min}, f_{max} \in \mathbb{R}$, de forma que $f_{min} \leq f(x,y) \leq f_{max}$, $\forall\ (x,y)$, então o intervalo $[f_{min},f_{max}]$ será denominado *grayscale* (escala de cinza da imagem). É comum deslocar este intervalo para $[0,\xi]$, onde $\xi \in \mathbb{R}$, de forma que $f(x,y) = 0$ representa o preto e $f(x,y) = \xi$ representa o branco, estando assim, todos os níveis de cinza entre o preto e o branco.

As Tabelas 1 e 2 aprestam exemplos de valores para $\tau(x,y)$ e $\gamma(x,y)$.

Tabela 1 – Exemplos de valores para $\tau(x,y)$ (em *lux* ou *lúmem*/m^2)

Ambiente	$\tau(x,y)$
Dia ensolarado	900
Dia dublado	100
Iluminação média de escritório	10
Noite clara de lua cheia	0,001

Fonte: (MARQUES FILHO; VIEIRA NETO, 1999, p. 20)

Tabela 2 – Exemplos de valores para $\gamma(x,y)$

Objeto	$\gamma(x,y)$
Neve	0,93
Parede branco-fosca	0,80
Aço inoxidável	0,65
Veludo preto	0,01

Fonte: (MARQUES FILHO; VIEIRA NETO, 1999, p. 20)

Uma imagem monocromática é de natureza analógica, ou seja, é contínua tanto na sua variação espacial no plano $\mathbb{R}^2$ quanto nos níveis de cinza. Porém, visando o seu processamento computacional, esta deve ser digitalizada e representada por alguma estrutura de dados (geralmente uma matriz).

Uma imagem ao ser digitalizada torna-se discreta, tanto no espaço, onde a discretização recebe o nome de amostragem, quanto na amplitude, onde a discretização recebe o nome de quantização. Basicamente, a digitalização converte uma imagem analógica em uma matriz de M linhas e N colunas, conforme é mostrado abaixo:

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & \dots & f(0,N-1) \\ f(1,0) & f(1,1) & \dots & f(1,N-1) \\ \vdots & \vdots & \vdots & \vdots \\ f(M-1,0) & f(M-1,1) & \dots & f(M-1,N-1) \end{bmatrix} \tag{2.4}$$

Atualmente, a digitalização é normalmente feita pelo próprio dispositivo de aquisição da imagem. Maiores valores de M e N implicam em uma imagem de maior resolução espacial. Por outro

lado, a quantização faz com que cada um desses pontos assuma um valor inteiro, na faixa de 0 a 2^{κ} - 1, onde $\kappa \in \mathbb{N} - \{0\}$. O caso particular em que $\kappa = 1$ define uma imagem binária.

Um par de índices (linha e coluna) da matriz identificam um ponto na imagem, o valor da matriz no ponto identifica o nível de cinza e os elementos da matriz identificam os pixeis. O processo de digitalização envolve decisões a respeito dos valores de M, N e κ, variando muito de dispositivo para dispositivo. A prática comum em processamento de imagens é assumir que M e N são potências inteiras de dois (GONZALES; WOODS, 2002, p. 56). O número β de bits necessários para armazenar uma imagem é:

$$\beta = M.N.\kappa \tag{2.5}$$

A visão e as ações humanas são influenciadas pela abundância de informações contidas tanto na geometria (forma) quanto nas cores de objetos. A estrutura do olho humano permite que todas as cores sejam vistas como combinações das três cores primárias aditivas (vermelho, verde e azul). Estas cores podem ser adicionadas produzindo as chamadas cores secundárias aditivas, ou seja, magenta (mistura de vermelho e azul), ciano (mistura de verde e azul) e amarelo (mistura de vermelho e verde). A mistura das três cores primárias aditivas ou de uma secundária aditiva com sua cor primária aditiva oposta produz o branco. Também é importante destacar que todas estas misturas devem ser feitas em quantidades corretas, de forma a produzir o resultado desejado.

As cores secundárias aditivas podem ser definidas como cores primárias subtrativas, também conhecidas por pigmentos ou corantes. Uma cor primária subtrativa absorve do branco uma cor primária aditiva, definida nesse contexto, como uma cor secundária subtrativa.

A Figura 1 ilustra como as cores primárias aditivas formam as cores secundárias aditivas, o branco e o preto.

Figura 1 – Cores primárias aditivas e suas combinações

Fonte: (GONZALES; WOODS, 2002, p. 302)

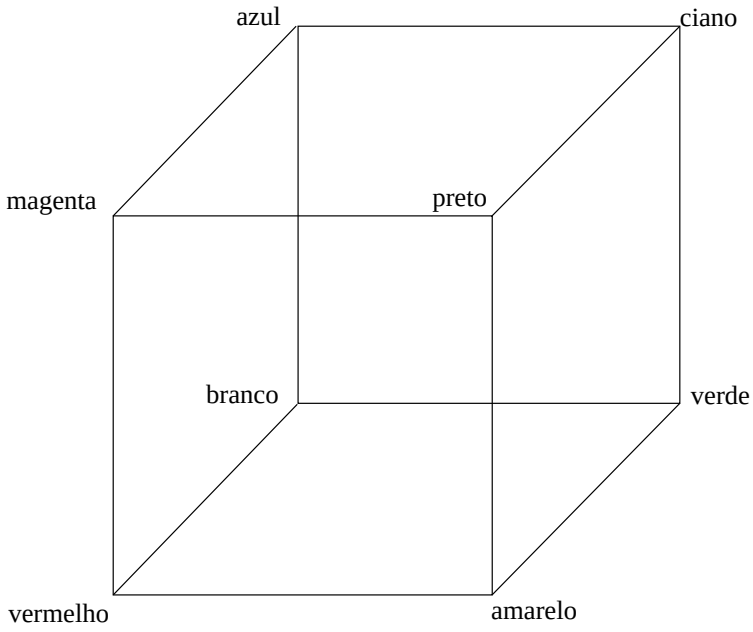
Um modelo de cores tem por objetivo proporcionar uma forma padrão de representação para

as cores, podendo ser visto como o espaço gerado por uma base, geralmente no $\mathbb{R}^3$, onde cada cor é representada por um ponto neste espaço.

Os modelos de cor podem ser orientados ao hardware ou orientados a aplicação. Dentre os modelos orientados ao hardware mais comumente utilizados na prática está o *RGB* (*Red Green Blue*). Este modelo é utilizado em monitores coloridos e também por uma ampla classe de câmeras de vídeo (GONZALES; WOODS, 2002, p. 290).

No modelo *RGB* cada cor é formada a partir dos seus componentes vermelho, verde e azul. Esse modelo baseia-se num sistema de coordenadas cartesianas. Por conveniência, assume-se que todos os valores de cor foram normalizados, de modo que as cores estejam em um cubo unitário, conforme mostrado na Figura 2. Neste modelo, as cores são pontos no cubo e podem ser vistas como vetores no $\mathbb{R}^3$.

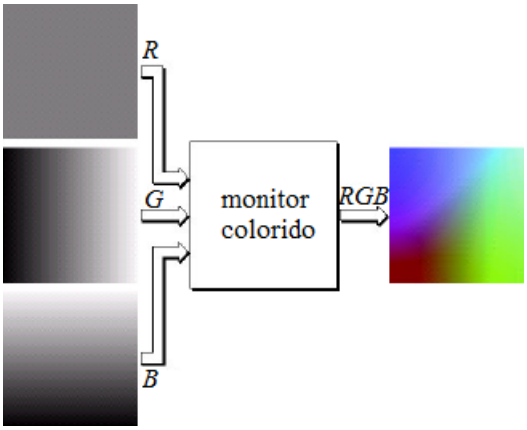
Figura 2 – Cubo de cores *RGB*



Fonte: (GONZALES; WOODS, 2002, p. 290)

Imagens no modelo *RGB* consistem em três planos de imagens independentes (camadas). Quando alimentadas num monitor *RGB*, estas camadas combinam-se sobre a tela para produzir uma imagem de cores compostas, como ilustrado na Figura 3.

Figura 3 – Reprodução de cores em um monitor *RGB*

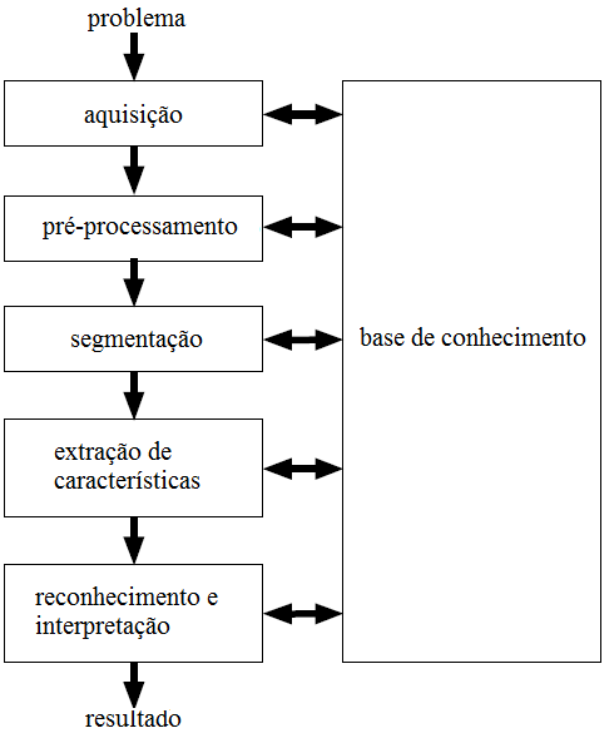


Fonte: (GONZALES; WOODS, 2002, p. 292)

2.1.1 Etapas do processamento de imagens

O processamento de imagens é normalmente dividido em: aquisição da imagem, pré-processamento, segmentação, extração de características, reconhecimento e interpretação da imagem. A Figura 4 ilustra estas etapas, que são descritas a seguir e baseadas em (MARQUES FILHO; VIEIRA NETO, 1999, p. 9-10).

Figura 4 – Etapas do processamento de imagens



Fonte: (MARQUES FILHO; VIEIRA NETO, 1999, p. 9)

a) Aquisição da imagem: A primeira etapa é a aquisição das imagens. Para tanto, são necessários alguns sensores e um digitalizador. Os sensores convertem a informação óptica em sinais elétricos e o digitalizador transformará a imagem analógica em imagem digital. Dentre os aspectos de projeto envolvidos nesta etapa, pode-se mencionar: a escolha do tipo do sensor, o conjuntos de lentes a utilizar, as condições de iluminação da cena, a resolução espacial e o número de níveis de cinza da imagem. Esta etapa produz uma imagem digitalizada como saída.

b) Pré-processamento: A imagem resultante do passo anterior pode apresentar diversas imperfeições, tais como: presença de pixeis ruidosos além de contraste ou brilho inadequado. A função da etapa de pré-processamento é aprimorar a qualidade da imagem para as etapas subsequentes. A imagem resultante desta etapa é uma imagem digitalizada de melhor qualidade que a original.

c) Segmentação: A tarefa básica da segmentação é dividir uma imagem digital em suas unidades significativas, ou seja, nos objetos de interesse que a compõem. Esta tarefa, apesar de simples de descrever, é das mais difíceis de implementar.

d) Extração das características: Esta etapa procura extrair características das imagens resultantes da segmentação através de descritores que permitam caracterizar estes objetos. Estes descritores devem ser representados por uma estrutura de dados adequada ao algoritmo de reconhecimento. É importante observar que nesta etapa a entrada ainda é uma imagem, mas a saída é um conjunto de dados correspondentes àquela imagem.

e) Reconhecimento e interpretação: Nesta última etapa, denominamos reconhecimento, o processo de atribuição de um rótulo a um objeto baseado em suas características, traduzidas por seus descritores. A tarefa de interpretação, por outro lado, consiste em atribuir significado a um conjunto de objetos reconhecidos.

Os processos de reconhecimento e interpretação em conjunto são comumente denominados de análise de imagens. Algumas aplicações requerem uma análise automática de imagens, ou seja, o entendimento automático de padrões que sejam relevantes para realizar uma determinada tarefa. As redes neurais artificiais, apresentadas em detalhes na próxima seção, fornecem uma abordagem para análise automática de imagens.

f) Base de conhecimento: Todas as tarefas das etapas descritas acima pressupõem a existência de um conhecimento sobre o problema a ser resolvido, armazenado em uma base de

dados, cujo tamanho e complexidade podem variar enormemente. Idealmente, esta base de conhecimento deveria não somente guiar o funcionamento de cada etapa, mas também permitir a realimentação entre elas. Esta integração entre as várias etapas através da base de conhecimento ainda é um objetivo difícil de alcançar e não está presente na maioria dos sistemas de visão artificial existentes atualmente.

2.1.2 Técnicas de processamento de imagens

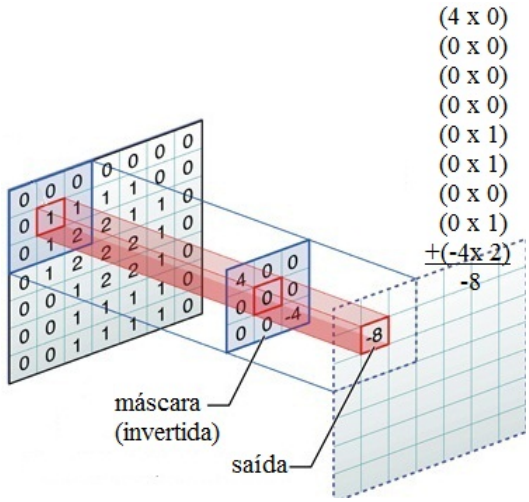
Diversas técnicas de processamento de imagens são empregadas durante as etapas de pré-processamento e segmentação, algumas dessas são apresentadas em detalhes a seguir.

2.1.2.1 Operações orientadas a vizinhança

As operações orientadas a vizinhança utilizam o conceito de convolução ou correlação com máscaras, também denominadas janelas ou ainda *templates*. Uma máscara pode ser definida como uma matriz quadrada qualquer, onde naturalmente tem-se um pixel central. Será feita uma multiplicação elemento a elemento entre a máscara e uma sub-área da imagem, de mesmo tamanho que a máscara, o resultado de cada uma destas multiplicações é somado e o resultado final desta soma é então colocado na posição central da máscara em uma nova imagem de saída.

Este processo se repete de forma que a máscara percorra toda a imagem, desde o canto superior esquerdo até o canto inferior direito da imagem. A Figura 5 dá um exemplo de geração de um pixel de saída no processo de convolução, a única diferença entre o processo de convolução e o processo de correlação é que no primeiro a máscara será invertida (posição das linhas e posição das colunas). Nas bordas existem várias abordagens a serem dadas, uma vez que algumas posições da máscara ficarão de fora da imagem de entrada e diferentes tratamentos são possíveis, variando desde o uso de 0 nestas posições até soluções mais complexas.

Figura 5 – Processo de convolução



Fonte: nada (a)

2.1.2.2 Suavização pela média

Uma das formas mais simples de suavização de imagens dá-se pela média, por exemplo, pode-se utilizar uma máscara 3 x 3 com todos os coeficientes iguais a 1 e dividir o resultado da convolução por um fator de normalização, neste caso igual a 9. Na escolha do tamanho da máscara, deve-se ter em mente que quanto maior esta, mais borrada será a imagem resultante.

Tabela 3 – Alguns operadores utilizados para a suavização de imagens

3 x 3	5 x 5	7 x 7
$\frac{1}{9}\cdot\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}$	$\frac{1}{25}\cdot\begin{bmatrix}1&1&1&1&1\\1&1&1&1&1\\1&1&1&1&1\\1&1&1&1&1\\1&1&1&1&1\end{bmatrix}$	$\frac{1}{49}\cdot\begin{bmatrix}1&1&1&1&1&1&1\\1&1&1&1&1&1&1\\1&1&1&1&1&1&1\\1&1&1&1&1&1&1\\1&1&1&1&1&1&1\\1&1&1&1&1&1&1\\1&1&1&1&1&1&1\end{bmatrix}$

Fonte: (MARQUES FILHO; VIEIRA NETO, 1999, p. 86)

2.1.2.3 Realce por diferenciação

Sabendo que o cálculo da média remove os componentes de alta frequência e que o mesmo pode ser visto como análogo à operação de integração, é razoável esperar que a diferenciação produza o efeito oposto e, portanto, enfatize os componentes de alta frequência presentes em uma imagem. O método mais usual de diferenciação em aplicações de processamento de imagens é baseado no módulo do gradiente, denotado por $|\nabla f|$. Em termos contínuos, o módulo do gradiente de $f(x,y)$ em um certo ponto $\vec{p} = (x,y)$ é dado por:

$$|\nabla f| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2} \tag{2.6}$$

para uma imagem digital, o mesmo pode ser aproximado por:

$$|\nabla f| \approx \sqrt{[f(x,y) - f(x+1,y)]^2 + [f(x,y) - f(x,y+1)]^2} \tag{2.7}$$

ou ainda:

$$|\nabla f| \approx |f(x,y) - f(x+1,y)| + |f(x,y) - f(x,y+1)| \tag{2.8}$$

A Equação 2.8 pode ser implementada usando operadores (máscaras) de tamanho 3 x 3. Alguns destes operadores são apresentados na Tabela 4:

Tabela 4 – Alguns operadores usados no realce de imagens

Operador	Vertical	Horizontal
Roberts	$\begin{bmatrix} 0 & 0 & -1 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$
Sobel	$\frac{1}{4}\cdot\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$	$\frac{1}{4}\cdot\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$
Prewitt	$\frac{1}{3}\cdot\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$	$\frac{1}{3}\cdot\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$

Fonte: (MARQUES FILHO; VIEIRA NETO, 1999, p. 38)

2.1.2.4 Morfologia matemática

O princípio básico da morfologia matemática consiste em extrair informações relativas a geometria e a topologia de um conjunto desconhecido (imagem) pela transformação através de outro conjunto completamente definido, chamado elemento estruturante.

Sejam A e B dois conjuntos em $\mathbb{Z}^2$, sejam x_1, $x_2 \in \mathbb{Z}$ dois escalares, seguem as seguintes definições:

1. Translação de A por $\vec{x} = (x_1,x_2)$:

$$(A)_{\vec{x}} = \{(a_1 + x_1, a_2 + x_2) : \ (a_1, a_2) \in A\} \tag{2.9}$$

2. Reflexão de A:

$$\hat{A} = \{(-a_1, -a_2) : \ (a_1, a_2) \in A\} \tag{2.10}$$

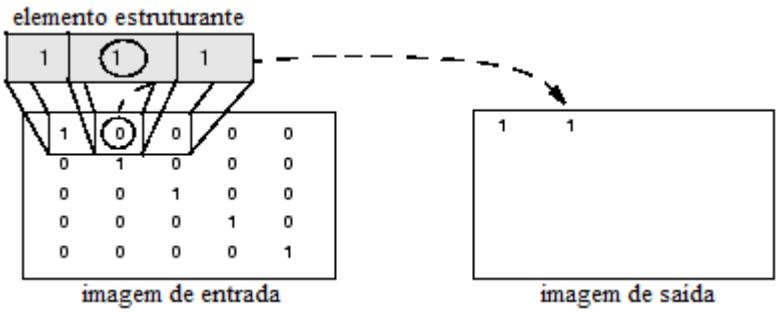
A dilatação de A por B, $A \oplus B$, é definida como:

$$A \oplus B = \{\vec{x} = (x_1, x_2) : \quad [(\hat{B})_{\vec{x}} \cap A] \neq \emptyset\} \tag{2.11}$$

Portanto, o processo de dilatação consiste em obter a reflexão de B sobre sua origem e depois deslocar esta reflexão de $\vec{x} = (x_1, x_2)$. A dilatação de A por B é, então, o conjunto de todos os deslocamentos para os quais a intersecção de $(\hat{B})_{\vec{x}}$ por A inclui pelo menos um elemento diferente de zero.

A Figura 6 ilustra o processo de dilatação em uma imagem binária. A função de dilatação define o valor do pixel de saída para 1 porque pelo menos um dos elementos na vizinhança definida pelo elemento estruturante é 1. Neste caso, a imagem será dilatada horizontalmente pois o elemento estruturante é uma linha horizontal. Se este fosse uma linha vertical, dilatariamos a imagem verticalmente.

Figura 6 – Dilatação em uma imagem binária



Fonte: nada (b)

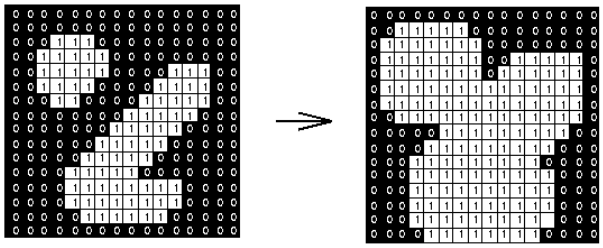
A erosão de A por B, $A \ominus B$, é dada por:

$$A \ominus B = \{\vec{x} = (x_1, x_2)) : \ (\hat{B})_{\vec{x}} \subset A\} \tag{2.12}$$

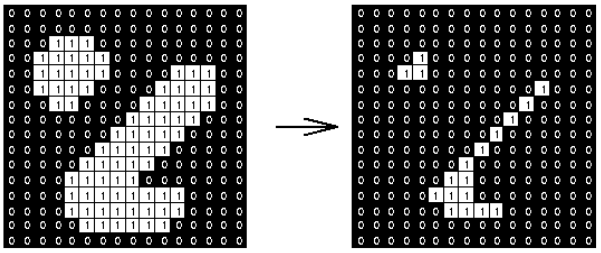
Em outras palavras, a erosão de A por B resulta no conjunto de todos os pontos $\vec{x} = (x_1, x_2)$, tais que B, transladado de $\vec{x}$, está contido em A.

A Figura 7 ilustra os efeitos de aplicar uma matriz 3 x 3 como elemento estruturante nos processos de dilatação e erosão.

Figura 7 – Matriz 3 x 3 como elemento estruturante



(a) Dilatação



(b) Erosão

Fonte: nada (c)

2.2 Redes Neurais Artificiais

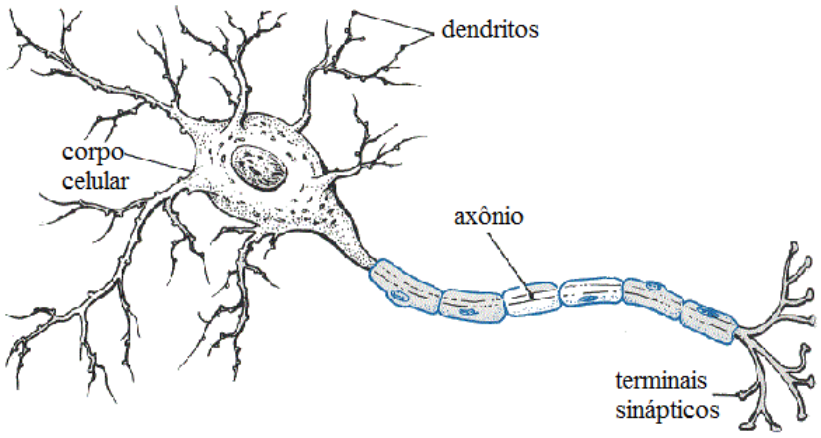
Redes Neurais Artificiais (RNA), segundo Braga, Ludemir e Carvalho (2000, p. 1) são sistemas paralelos distribuídos, compostos por unidades de processamento simples, também conhecidos como nodos ou neurônios, que por sua vez, são interligados por um grande número de conexões a fim de determinar funções matemáticas.

2.2.1 Neurônio biológico

Com base em Veelenturf (1995, p. 2), a unidade básica das RNA é uma versão simplificada do que acredita-se ser o comportamento funcional de um neurônio orgânico. O cérebro humano contém aproximadamente 10^{11} neurônios. De forma grosseira, é possível distinguir anatomicamente três partes em um neurônio: um conjunto de fibras de entrada (dendritos), o corpo da célula (soma) e uma fibra de saída (axônio). O axônio divide-se em diferentes terminais, denominados terminais sinápticos. Cada um destes faz contato com dendritos de outros neurônios. A região de contato entre os neurônios é conhecida como sinapse. Um neurônio pode receber até 10000 entradas de outros neurônios. Impulsos elétricos podem ser gerados pelos neurônios e são transmitidos pelo axônio até as sinapses. Quando a atividade elétrica é transferida pela sinapse para outro neurônio, pode contribuir para a excitação ou inibição daquele neurônio. As sinapses possuem um papel importante porque sua eficiência de transmissão de impulsos elétricos pode

ser modificada. Hebb (1949) postulou que a capacidade de aprendizagem dos seres humanos é, provavelmente, incorporada pela mudança da eficiência de transmissão das sinapses.
A Figura 8 mostra o neurônio biológico e seus componentes.

Figura 8 – Neurônio biológico e seus componentes



Fonte: (NEUROSCIENZE, 2010)

2.2.2 Modelo matemático de um neurônio

O modelo neural utilizado em RNA foi introduzido por McCulloch e Pitts (1943). De acordo com Haykin (2001, p. 36), três elementos básicos definem este modelo:

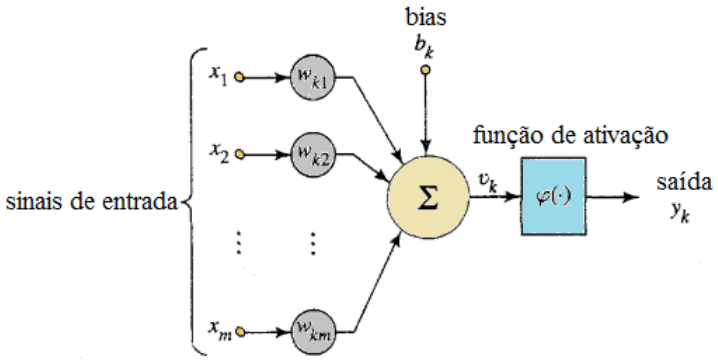
1. Um conjunto de sinapses, as quais são caracterizadas por um peso (positivo se o estímulo for excitatório ou negativo se for inibitório). Um sinal (estímulo) x_j na entrada da sinapse j conectada ao neurônio k é multiplicado pelo peso sináptico w_{kj}.

2. Um somador, que faz a combinação linear dos sinais de entrada, ponderados pelos pesos sinápticos correspondentes.

3. Uma função de transferência, também denominada função de ativação, para limitar a amplitude da saída do neurônio. Esta amplitude é normalmente limitada no intervalo unitário fechado $[0,1]$ (binário) ou alternativamente $[-1,1]$ (bipolar).

A Figura 9 ilustra o modelo neural descrito.

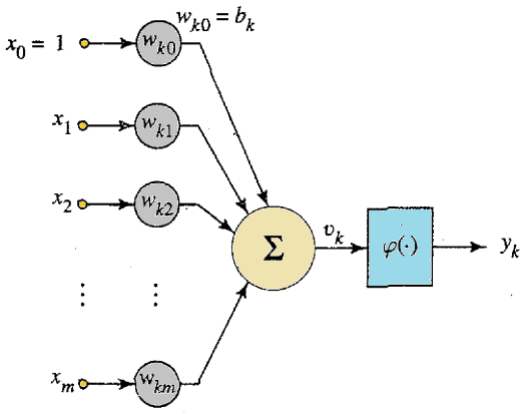
Figura 9 – Modelo neural



Fonte: (HAYKIN, 2001, p. 36)

O modelo inclui também um bias b_k, que é somado a entrada da função de ativação. Ao adicionar um novo sinal de entrada $x_0 = 1$, transforma-se o bias em um peso adicional w_{k0}, conforme ilustrado na Figura 10.

Figura 10 – Bias como peso no modelo neural



Fonte: Haykin (2001, p. 38)

A Equação 2.13 e a Equação 2.14 descrevem formalmente um neurônio k:

$$v_k = \sum_{j=0}^{m} x_j . w_{kj} \tag{2.13}$$

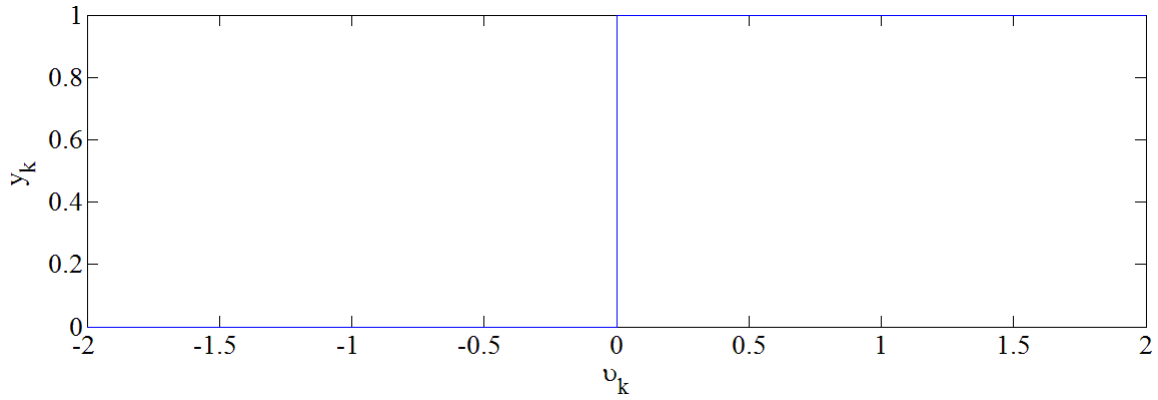
$$y_k = \varphi(v_k) \tag{2.14}$$

A função φ determina a saída do neurônio em termos de v_k (conhecido como potencial de ativação ou campo local induzido). As diversas funções de ativação podem ser classificadas como: função de limiar, função linear por partes ou função sigmóide.

Uma função de limiar descreve a propriedade tudo ou nada do modelo original de McCulloch e Pitts (1943). A Equação 2.15 define uma função de limiar binária e a Equação 2.16 a versão bipolar da primeira. O gráfico destas funções é apresentado, respectivamente, nas Figuras 11 e 12.

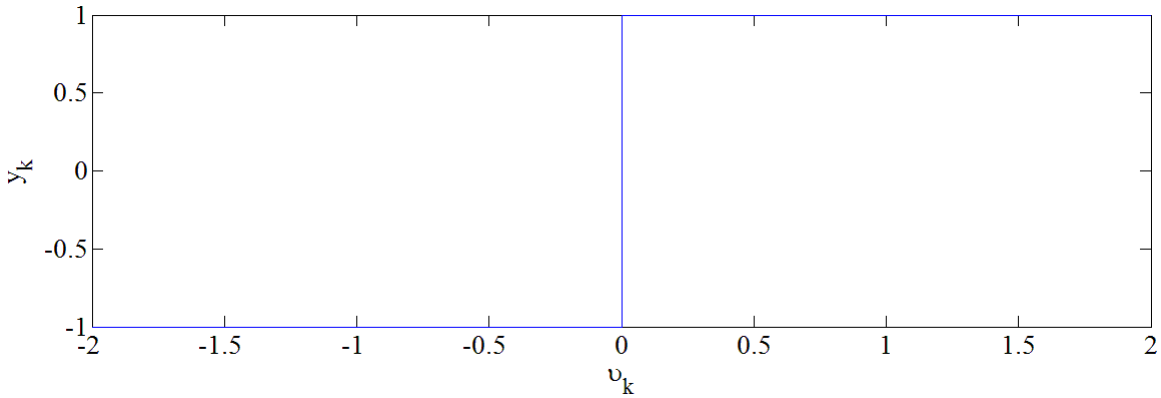
$$\varphi(v_k) = \begin{cases} 1, & \text{se} \quad v_k \geq 0 \\ 0, & \text{se} \quad v_k < 0 \end{cases} \tag{2.15}$$

Figura 11 – Função de limiar (binária)



$$\varphi(v_k) = \begin{cases} 1, & \text{se} \quad v_k \geq 0 \\ -1, & \text{se} \quad v_k < 0 \end{cases} \tag{2.16}$$

Figura 12 – Função de limiar (bipolar)

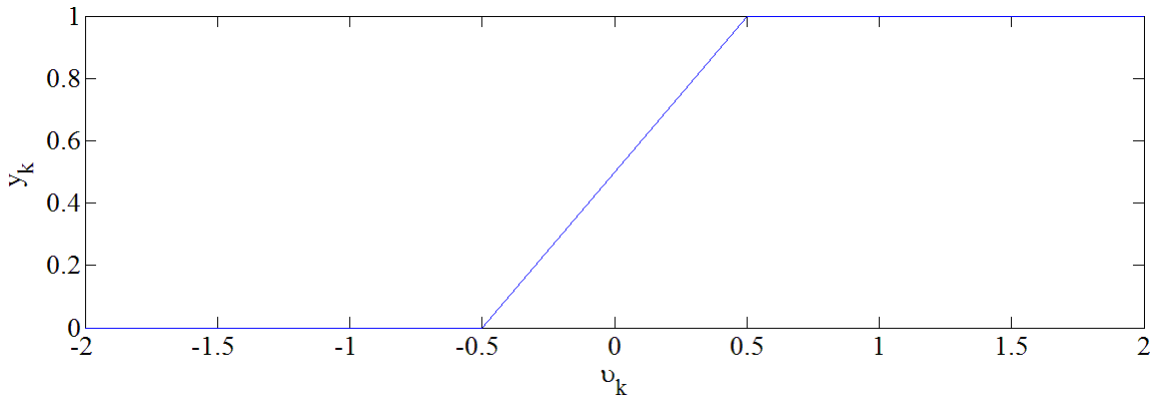


As funções mostradas anteriormente são descontínuas e limitam a saída de um neurônio a somente duas possibilidades. As funções lineares, por sua vez, são contínuas e permitem que a saída assuma qualquer valor real. Para limitar a saída na faixa desejada ($[0,1]$ ou $[-1,1]$) é utilizada a função linear por partes da Equação 2.17 ou sua versão bipolar da Equação 2.18. As

mesmas são graficamente representadas nas Figuras 13 e 14.

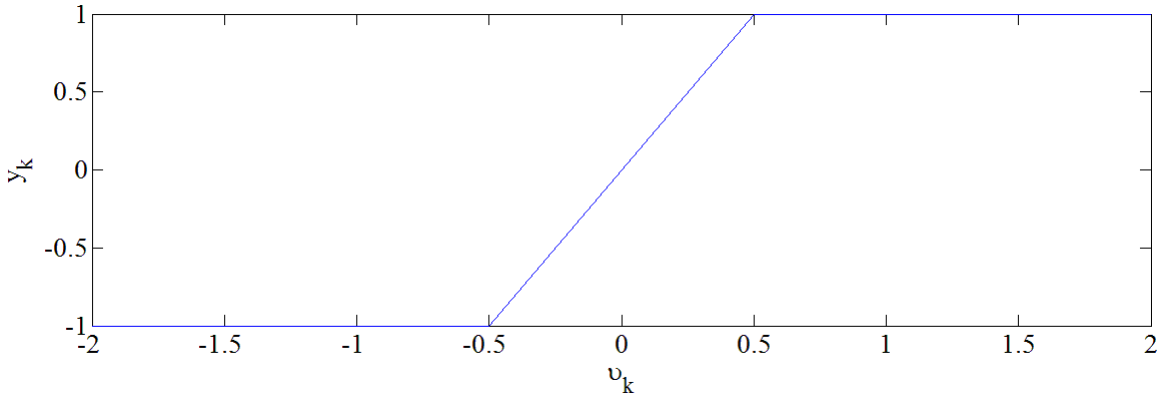
$$\varphi(\upsilon_k) = \begin{cases} 0 & , \text{ se } \upsilon_k \leq -\frac{1}{2} \\ \upsilon_k + \frac{1}{2} & , \text{ se } -\frac{1}{2} < \upsilon_k < \frac{1}{2} \\ 1 & , \text{ se } \upsilon_k \geq \frac{1}{2} \end{cases} \tag{2.17}$$

Figura 13 – Função linear por partes (binária)



$$\varphi(\upsilon_k) = \begin{cases} -1 & , \text{ se } \upsilon_k \leq -\frac{1}{2} \\ 2\upsilon_k & , \text{ se } -\frac{1}{2} < \upsilon_k < \frac{1}{2} \\ 1 & , \text{ se } \upsilon_k \geq \frac{1}{2} \end{cases} \tag{2.18}$$

Figura 14 – Função linear por partes (bipolar)



As funções sigmóides, cujo gráfico tem formato de *s*, são de longe as mais utilizadas em RNA. Estas, são estritamente crescentes e exibem um balanceamento adequado entre o comportamento linear e não-linear (HAYKIN, 2001, p. 40). A Equação 2.19 descreve a função logística que é um exemplo de função sigmóide binária, representada graficamente nas Figuras 15, 16 e 17 com *a*

$= 0,5$, $a = 1$ e $a = 2$, respectivamente.

$$\varphi(\upsilon_k) = \frac{1}{1 + e^{-a\upsilon_k}} \tag{2.19}$$

Figura 15 – Função logística com $a = 0,5$

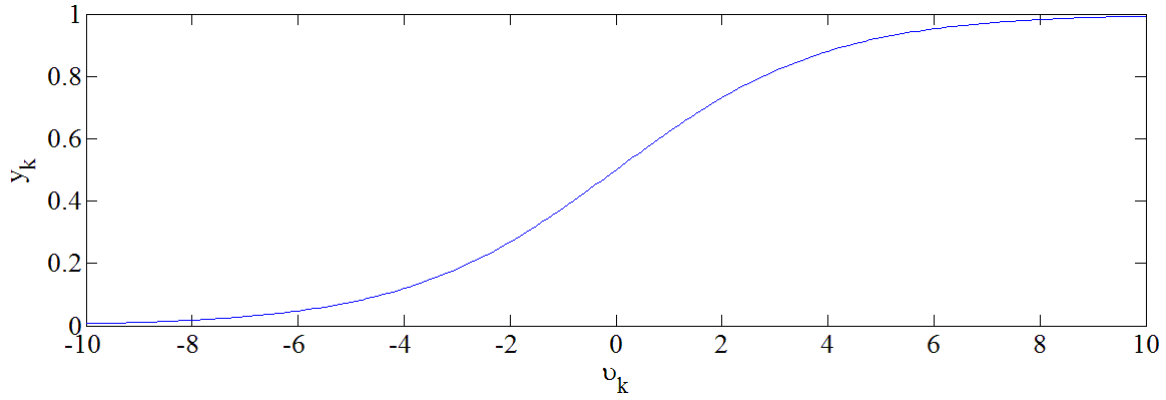


Figura 16 – Função logística com $a = 1$

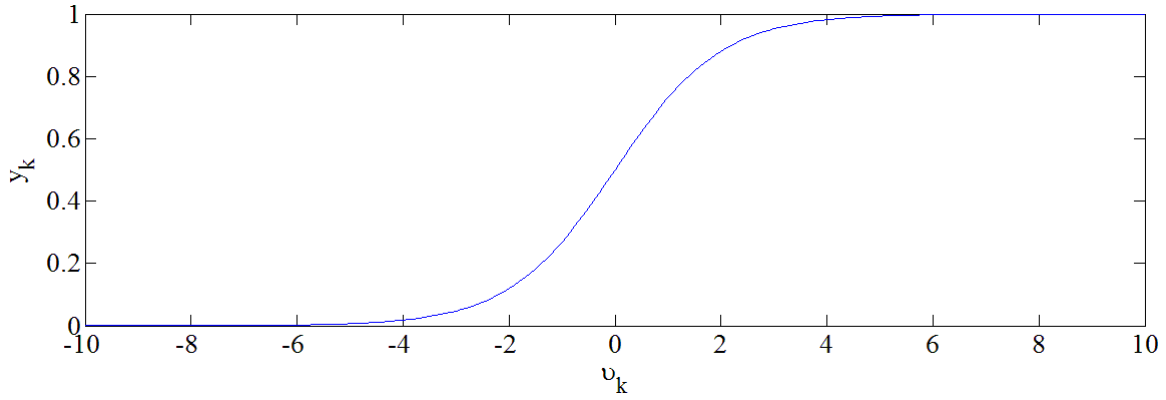
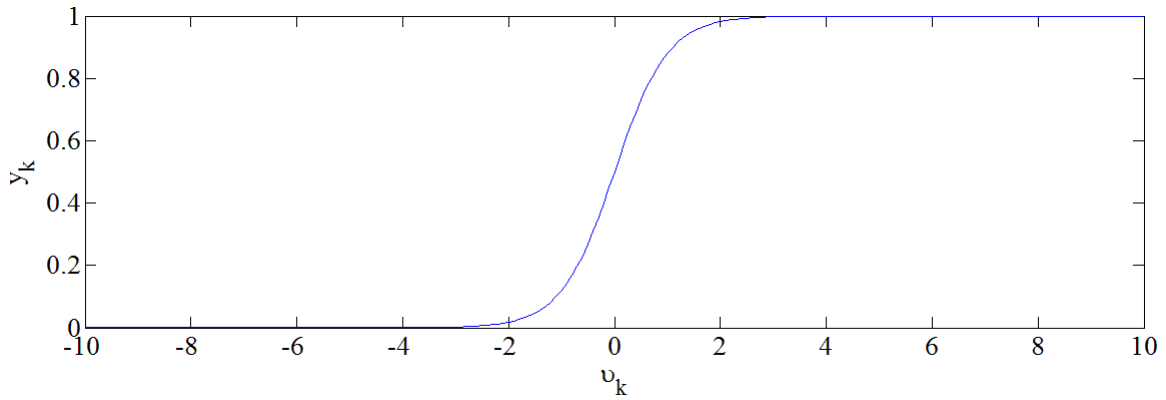


Figura 17 – Função logística com $a = 2$



A Equação 2.20 define a função tangente hiperbólica que é a forma bipolar da função logística. A função tangente hiperbólica é apresentada graficamente nas Figuras 18, 19 e 20 com

$b = 0,5$, $b = 1$ e $b = 2$, respectivamente.

$$\varphi(\upsilon_k) = tanh(b\upsilon_k) = \frac{e^{bt} - e^{-bt}}{e^{bt} + e^{-bt}} \tag{2.20}$$

Figura 18 – Função tangente hiperbólica com $b = 0,5$

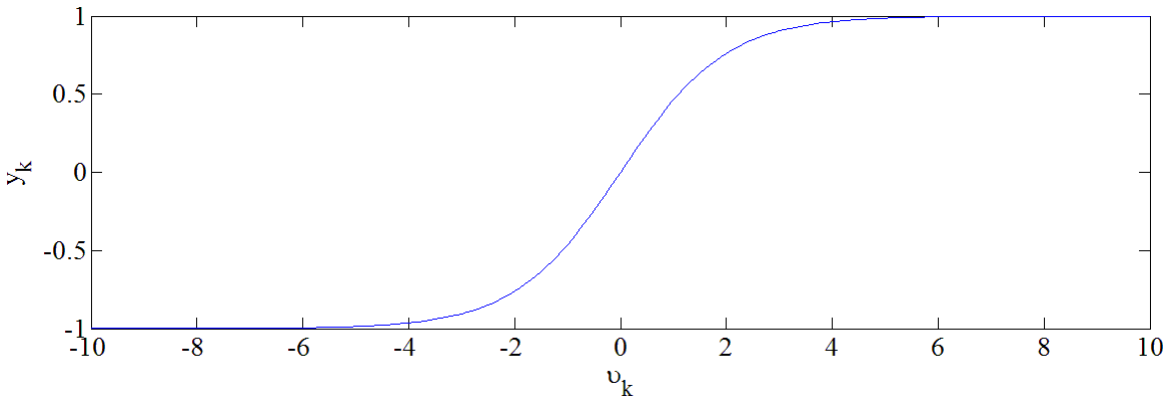


Figura 19 – Função tangente hiperbólica com $b = 1$

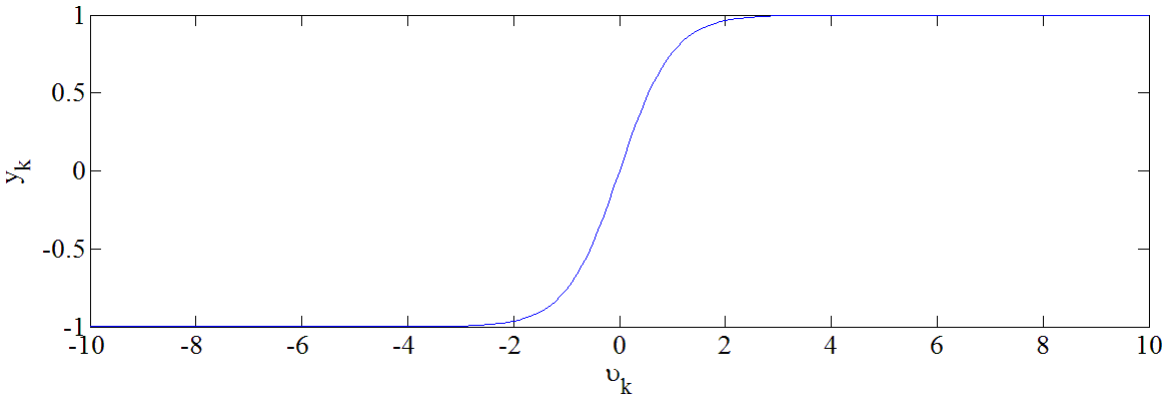
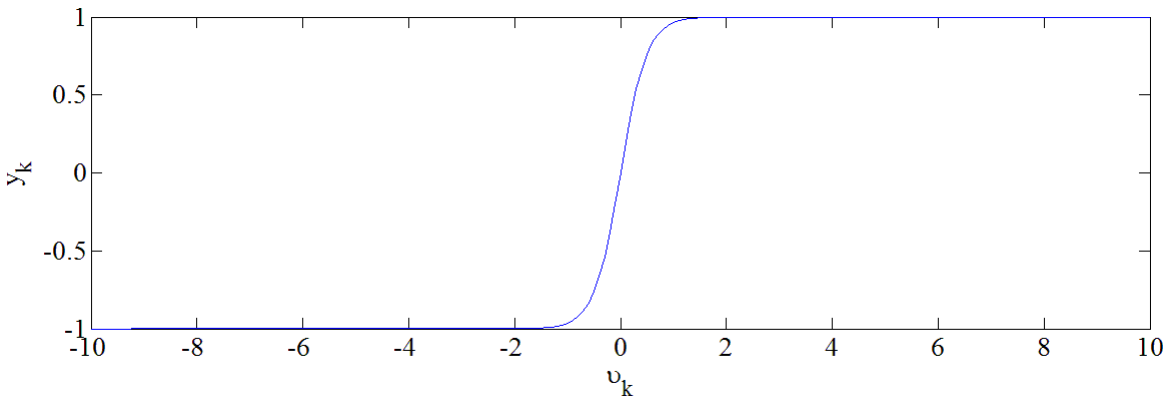


Figura 20 – Função tangente hiperbólica com $b = 2$



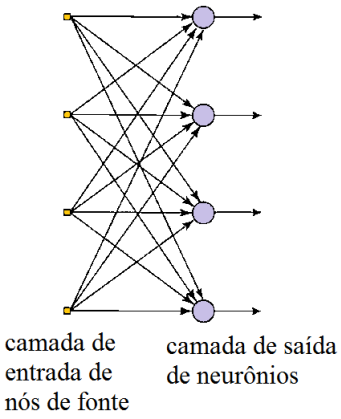
2.2.3 Arquiteturas de RNA

A arquitetura, também denominada topologia, define a disposição dos neurônios, ou seja, a estrutura da rede.

Algumas topologias de rede são organizadas em conjuntos distintos, ordenados sequencialmente e conhecidos como camadas. Redes *feed forward* ou alimentadas adiante são assim denominadas pois os neurônios em cada camada têm como entradas somente os sinais de saída da camada precedente. Estas redes podem ser *single layer* (camada única) ou *multi layer* (múltiplas camadas).

Redes *single layer* possuem uma camada de entrada de nós de fonte que projetam-se sobre uma camada de saída de neurônios. O termo camada única refere-se à camada de saída de nós computacionais (neurônios). A camada de entrada de nós de fonte não é levada em conta pois nela não é realizada computação. A Figura 21 mostra um exemplo de arquitetura *single layer feed forward*.

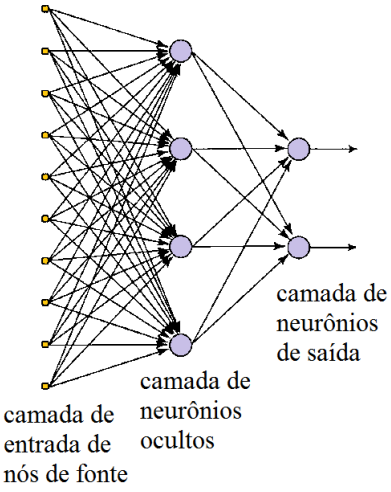
Figura 21 – Arquitetura *single layer feed forward*



Fonte: Haykin (2001, p. 47)

Redes *multi layer* além de uma camada de nós de fonte e de uma camada de saída de neurônios, possuem uma ou mais camadas intermediárias, denominadas camadas ocultas, cujos nós computacionais são chamados de neurônios ocultos. A Figura 22 ilustra uma arquitetura *multi layer feed forward* com uma camada oculta.

Figura 22 – Arquitetura *multi layer feed forward*



Fonte: Haykin (2001, p. 48)

2.2.4 Representação do conhecimento

Uma tarefa importante para uma rede neural é aprender um modelo do ambiente. As observações (medidas) do ambiente podem ser obtidas por meio de sensores. Normalmente, estas observações são inerentemente ruidosas. As observações fornecem um conjunto de informações de onde são retirados padrões (exemplos) para treinar a rede neural.

Padrões podem ser rotulados ou não-rotulados. Nos padrões rotulados, cada padrão representa um sinal de entrada e é associado a uma resposta desejada, também denominada resposta correspondente ou saída-alvo. Por outro lado, os padrões não-rotulados consistem de ocorrências dos sinais de entrada que serão agrupadas em *clusters* pela própria rede de tal modo que sinais de um mesmo *cluster* apresentem propriedades em comum. Um conjunto de padrões, rotulados ou não, representa o conhecimento acerca do ambiente de interesse que uma rede neural precisa aprender. Em uma rede neural com arquitetura específica, a representação do conhecimento é definida pelos valores assumidos pelos parâmetros livres (pesos sinápticos e biás) da rede (HAYKIN, 2001, p. 49-50).

2.2.5 Projeto de RNA

O projeto de redes neurais pode ser dividido em três etapas:

1. O primeiro passo é obter um conjunto de dados de treinamento. Um conjunto de dados de treinamento, também conhecido como amostra de treinamento é composto por um conjunto de pares entrada-saída. Cada par entrada-saída consiste de um sinal de entrada e

de uma saída-alvo.

2. Em um segundo momento é selecionada uma arquitetura apropriada e um subconjunto de dados de treinamento, este subconjunto é utilizado para treinar a rede por meio de um algoritmo de aprendizagem. Esta fase do projeto é conhecida como aprendizagem ou treinamento.

3. Por fim, o desempenho de reconhecimento da rede treinada é testado com os dados não apresentados anteriormente. Esta etapa é denominada generalização.

O projeto de uma rede neural, segundo Haykin (2001, p. 50), diferencia-se do processamento de informações clássico (classificação de padrões) principalmente pois neste último, normalmente procedemos primeiramente formulando um modelo matemático das observações do ambiente, validando o modelo com dados reais, estruturando o projeto com base neste modelo. O projeto de uma rede neural, ao contrário, é baseado diretamente nos dados do mundo real, permitindo-se que o conjunto de dados fale por si mesmo.

Para exemplificar um projeto de redes neurais, considere o problema de reconhecimento de um dígito, descrito por Haykin (2001, p. 50). Os sinais de entrada são imagens binárias, representando um dígito $(0, 1, 2, ..., 9)$. A resposta desejada é definida por uma identidade do dígito em particular cuja imagem é apresentada para a rede como sinal de entrada. Uma arquitetura é então selecionada de forma que a camada de entrada possua um número de nós igual ao número de pixeis da imagem de entrada e a camada de saída consista de 10 neurônios (um para cada dígito). Um subconjunto de treinamento é então utilizado para treinar a rede por meio do algoritmo apropriado. Finalmente, a rede é testada com os dados não apresentados anteriormente, ou seja, uma imagem de entrada é apresentada a rede, porém desta vez não lhe é fornecida a identidade do dígito que corresponde a esta imagem. O desempenho da rede é então estimado comparando-se o reconhecimento do dígito fornecido pela rede com a real identidade do dígito em questão.

2.2.6 Processo de aprendizagem

O processo de aprendizagem consiste de um conjunto de procedimentos bem definidos para adaptar os parâmetros livres da rede para que a mesma possa aprender determinada função. Diferentes algoritmos de aprendizagem diferem-se basicamente pela maneira pela qual o ajuste dos pesos é feito. Algoritmos de aprendizagem podem ser divididos em dois paradigmas principais: aprendizado supervisionado e aprendizado não-supervisionado.

O aprendizado supervisionado é o mais comum no treinamento de redes neurais. É chamado aprendizado supervisionado porque a entrada e a saída são fornecidas por um supervisor (professor) externo, ou seja, são fornecidos para a rede padrões rotulados. A rede então compara sua saída corrente (calculada) com a saída desejada e a partir do erro da resposta atual modifica os parâmetros da rede de forma que se possível caminhe para uma solução.

No aprendizado não-supervisionado, não há um supervisor para acompanhar o processo de aprendizagem. Para estes algoritmos, somente as entradas estão disponíveis para a rede, ao contrário do aprendizado supervisionado, cujo conjunto de treinamento possui pares entrada-saída, em outras palavras, são fornecidos para a rede padrões não-rotulados. A partir do momento em que a rede estabelece uma harmônia com as regularidades estatísticas das entradas, desenvolve-se nela uma habilidade de formar representações internas para codificar características das entradas e criar novas classes ou grupos automaticamente (BRAGA; LUDEMIR; CARVALHO, 2000, p. 16-18).

2.2.7 Notas históricas

Como dito anteriormente, o modelo de neurônio foi introduzido por McCulloch e Pitts (1943). Kleene (1951) mostrou que com os neurônios de McCulloch e Pitts (1943) era possível construir sistemas que comportavam-se como um computador. A grande vantagem das redes neurais, porém, está na capacidade destas modificarem seu comportamento, ajustando seus pesos no processo de aprendizagem.

O processo de aprendizagem de RNA foi primeiramente tratado por Rosenblatt (1962). Em seu livro, foi apresentado um algoritmo de aprendizagem, no qual os pesos poderiam ser alterados para realizar a computação desejada. Porém, Minsky e Papert (1969) mostraram que algumas simples computações não podiam ser feitas por uma rede neural *single layer*, como proposto por Rosenblatt (1962), e duvidaram da existência de algoritmos para treinar redes neurais *multi layer*.

Em meados dos anos 80, várias pessoas encontraram um algoritmo de aprendizagem, que ficou conhecido como *back propagation* e que poderia ajustar os pesos em redes neurais *multi layer* (VEELENTURF, 1995, p. 4).

2.2.8 Treinamento de redes *MLP* com o algoritmo de *back propagation*

O *Perceptron*, originalmente, possuia somente uma camada de neurônios. As entradas e a saída destes eram binárias assim como sua função de ativação. Este modelo, conhecido como

single layer Perceptron, ficou restrito a solução de problemas linearmente separáveis, ou seja, cuja solução pode ser obtida pela separação de duas regiões por meio de uma reta (ou um hiperplano para o caso *n*-dimensional).

Redes *Multi Layer Perceptron* (*MLP*), por sua vez, possuem uma ou mais camadas ocultas. Neste modelo, todos os neurônios são ligados aos neurônios da camada subsequente. Os neurônios ocultos (com função de transferência sigmóide) são os responsáveis por capturar a não-linearidade dos dados, superando a restrição do *SLP*.

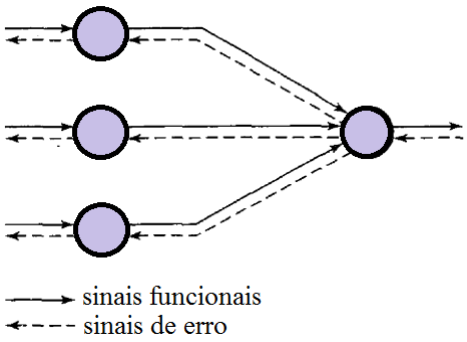
A inexistência ou desconhecimento de algoritmos para treinar redes *MLP* foi uma das causas da redução das pesquisas em RNA durante a década de 70. O surgimento do algoritmo de aprendizagem supervisionada *error back propagation*, ou simplesmente, *back propagation* foi o grande responsável pelo ressurgimento do interessem em RNA (BRAGA; LUDEMIR; CARVALHO, 2000, p. 34).

A popularização do *back propagation* se deu a partir da publicação de Rumelhart, Hinton e Williams (1986). Braga, Ludemir e Carvalho (2000, p. 59) evidenciam o fato de que este algoritmo já havia sido proposto antes, com diferentes propósitos, por diferentes pesquisadores.

De acordo com Silva (1998), as arquiteturas do tipo *MLP* constituem o modelo neural mais utilizado e conhecido. As redes *MLP* têm sido aplicadas com sucesso para resolver diversos problemas difíceis, através do seu treinamento de forma supervisionada com o algoritmo *back propagation* (HAYKIN, 2001, p. 183).

No *back propagation* assim como em qualquer processo de aprendizagem supervisionada são fornecidos padrões rotulados para a rede, cada um destes consistindo de um sinal de entrada e sua saída desejada. A aprendizagem se divide em dois passos: um passo pra frente, a propagação, e um passo pra trás, a retro-propagação. No passo pra frente, um vetor de entrada se propaga através da rede com os pesos sinápticos fixos. Durante o passo pra trás, estes pesos sinápticos são ajustados. A resposta real (produzida pela propagação) é subtraída da resposta desejada para produzir um sinal de erro. Este sinal de erro é então propagado pra trás através da rede, vindo daí o nome *error back propagation* (HAYKIN, 2001, p. 183). A Figura 23 ilustra o algoritmo de *back propagation*.

Figura 23 – Algoritmo de *back propagation*



Fonte: Haykin (2001, p. 186)

A derivação do algoritmo de *back propagation* dada a seguir foi baseada em (HAYKIN, 2001, p. 187-193):

Apresentados a rede N padrões rotulados, cada um em uma iteração (época) n. O sinal de erro na saída do neurônio k é definido por:

$$e_k(n) = d_k(n) - y_k(n) \tag{2.21}$$

e a energia total do erro por:

$$E(n) = \frac{1}{2}\sum_{k\in C} e_k^2(n) \tag{2.22}$$

onde d_k é a resposta desejada para o neurônio k e o conjunto C inclui todos os neurônios da camada de saída da rede. A Equação 2.13 e a Equação 2.14 podem ser reescritas para n iterações, conforme é dado no seguinte par de equações:

$$\upsilon_k(n) = \sum_{j=0}^{m} x_j(n) w_{kj}(n) \tag{2.23}$$

$$y_k(n) = \varphi(\upsilon_k(n)) \tag{2.24}$$

Para cada $w_{kj}(n)$ o algoritmo aplica uma correção proporcional ao fator de sensibilidade $\Delta w_{kj} = \partial E(n)/\partial w_{kj}(n)$, da seguinte forma:

$$w_{kj}(n+1) = w_{kj}(n) - \eta \frac{\partial E(n)}{\partial w_{kj(n)}} \tag{2.25}$$

onde η é o parâmetro da taxa de aprendizagem. De acordo com a regra da cadeia, temos:

$$\frac{\partial E(n)}{\partial w_{kj}(n)} = \frac{\partial E(n)}{\partial e_k(n)} \frac{\partial e_k(n)}{\partial y_k(n)} \frac{\partial y_k(n)}{\partial \upsilon_k(n)} \frac{\partial \upsilon_k(n)}{\partial w_{kj}(n)} \tag{2.26}$$

Derivando 2.21, 2.22, 2.23 e 2.24 conforme indicado na Equação 2.26 e substituindo os

resultados na mesma, obtemos:

$$\frac{\partial E(n)}{\partial w_{kj}(n)} = -e_k(n)\varphi'(\upsilon_k(n))x_j(n) \tag{2.27}$$

A Equação 2.27 possibilita o cálculo do fator de sensibilidade para um neurônio k localizado na camada de saída da rede ($k \in C$). Porém, quando o neurônio estiver em uma camada oculta ($k \notin C$), não existe uma resposta desejada ($d_k(n)$) para este, o que impossibilita o cálculo do erro ($e_k(n)$) e consequentemente o cálculo direto do fator de sensibilidade.

O fator de sensibilidade para um neurônio na penúltima camada é dado por:

$$\frac{\partial E(n)}{\partial w_{lj}(n)} = \frac{\partial E(n)}{\partial y_l(n)}\frac{\partial y_l(n)}{\partial \upsilon_l(n)}\frac{\partial \upsilon_l(n)}{\partial w_{lj}(n)} \tag{2.28}$$

onde l é um neurônio oculto na camada que antecede à camada de saída, $\upsilon_l(n)$ e $y_l(n)$ são dados pelas respectivas equações:

$$\upsilon_l(n) = \sum_{j=0}^{m} x_j(n)w_{lj}(n) \tag{2.29}$$

$$y_l(n) = \varphi(\upsilon_l(n)) \tag{2.30}$$

Diferenciando a Equação 2.22 em relação a $y_l(n)$, tem-se que:

$$\frac{\partial E(n)}{\partial y_l(n)} = \sum_{k\in C} e_k \frac{\partial e_k(n)}{\partial y_l(n)} \tag{2.31}$$

Da regra da cadeia, temos que:

$$\frac{\partial E(n)}{\partial y_l(n)} = \sum_{k\in C} e_k \frac{\partial e_k(n)}{\partial \upsilon_k(n)}\frac{\partial \upsilon_k(n)}{\partial y_l(n)} \tag{2.32}$$

Substituindo a Equação 2.24 em 2.21, obtemos:

$$e_k(n) = d_k(n) - \varphi(\upsilon_k(n)) \tag{2.33}$$

Note que para um neurônio k o campo local induzido pode ser escrito como:

$$\upsilon_k(n) = \sum_{l=0}^{m} y_l(n)w_{kl}(n) \tag{2.34}$$

Derivando a Equação 2.33 e a Equação 2.34 conforme indicado em 2.32 e substituindo os resultados na mesma, segue que:

$$\frac{\partial E(n)}{\partial y_l(n)} = -\sum_{k\in C} e_k \varphi'(\upsilon_k(n))w_{kl}(n) \tag{2.35}$$

Substituindo a Equação 2.35 na Equação 2.28:

$$\frac{\partial E(n)}{\partial w_{lj}(n)} = -\sum_{k \in C} e_k \varphi'(\upsilon_k(n)) w_{kl}(n) \frac{\partial y_l(n)}{\partial \upsilon_l(n)} \frac{\partial \upsilon_l(n)}{\partial w_{lj}(n)} \tag{2.36}$$

Por fim, derivando 2.29 e 2.30 conforme indicado em 2.36 e substituindo os resultados na mesma:

$$\frac{\partial E(n)}{\partial w_{lj}(n)} = \varphi'(\upsilon_l(n)) x_j(n) \sum_{k \in C} e_k \varphi'(\upsilon_k(n)) w_{kl}(n) \tag{2.37}$$

O algoritmo pode ser resumido como segue:

1. Os pesos são iniciados de forma aleatória.
2. Propagação: os sinais de entrada são propagados até a saída.
3. Retro-propagação: um ajuste de pesos é dado por 2.25. Este ajuste começa pela camada de saída onde o fator de sensibilidade é dado por 2.27. O ajuste segue para a penúltima camada onde o fator de sensibilidade é dado por 2.37. A Equação 2.37 será utilizada de forma recursiva caso exista mais de uma camada oculta.
4. Os passos 2 e 3 se repetem até alguma condição de parada ser satisfeita.

O ajuste dos pesos requer o conhecimento da derivada da função de ativação. Para esta derivação existir, esta função deve ser contínua. Em termos básicos, a diferenciabilidade é a única exigência que a função de ativação deve satisfazer.

Em geral, não se pode demonstrar que o algoritmo convergiu e não existem critérios bem definidos para encerrar sua operação. Em vez disso, há alguns critérios razoáveis, cada um com o seu mérito prático, que podem ser usados para encerrar o ajuste dos pesos.

Pode-se considerar que o algoritmo tenha convergido quando a norma euclidiana do vetor gradiente (derivada parcial de primeira ordem) da superfície de erro em relação ao vetor de peso alcançar um limiar suficientemente pequeno. Outro critério possível é considerar que o algoritmo tenha convergido quando a taxa absoluta de variação do erro médio quadrado por época for suficientemente pequena.

Há um outro critério de convergência útil e teoricamente fundamentado, denominado validação cruzada. Neste, após cada iteração de aprendizagem, a rede é testada pelo seu desempenho de generalização. O processo de aprendizagem é encerrado quando o desempenho de generalização for adequado, ou quando ficar aparente que o desempenho de generalização atingiu o seu máximo. Mais especificamente, podemos ver o problema da seleção da rede como a escolha, dentre um conjunto de estruturas de modelo candidatas. Primeiramente, o conjunto de dados é

dividido aleatoriamente em três conjuntos: estimação, validação e teste. A motivação é estimar os parâmetros com o conjunto de estimação e validar o modelo com o conjunto de dados de validação. Desta forma, pode-se avaliar o desempenho de vários modelos. Há, entretanto, uma possibilidade considerável de que o modelo selecionado, com os valores de parâmetros com melhor desempenho, possa acabar ajustando excessivamente o conjunto de validação. Para evitar essa possibilidade, o desempenho de generalização do modelo selecionado é medido sobre o conjunto de teste.

O uso da validação cruzada é atrativo particularmente quando tempos que projetar uma rede neural grande cujo objetivo seja uma boa generalização. Podemos, por exemplo, utilizar a validação cruzada para determinar o melhor número de neurônios ocultos e quando é melhor parar o treinamento (HAYKIN, 2001, p. 195; 199-200; 239-240).

3 METODOLOGIA

O desenvolvimento do trabalho foi orientado pelas etapas fundamentais do processamento de imagens. Para a realização dos experimentos foi utilizado um computador com processador Intel Xeon E5620, 2,4 GHz e 12 GB de memória RAM. Também foi utilizada a plataforma MATLAB R2011a (MATHWORKS, 2011b) em conjunto com a *Processing Image Toolbox* (MATHWORKS, 2011a) e a *Neural Network Toolbox* (MATHWORKS, 2011c).

3.1 Aquisição das imagens

As imagens utilizadas neste trabalho foram fornecidas por Padilha (2007). Cada imagem contém aproximadamente 50 grãos das seguintes variedades de soja: AL 83, BRS 133, BRS 184, BRS 214, CD 205, CD 206, CD 215, EMB 48, MERCEDES 70, MSOY 5826, NK 8350, RS 10 e MSOY 8000 RR. Estas variedades foram utilizadas para um estudo de caso onde o *software* desenvolvido tem por objetivo identificar a qual destas pertence um determinado grão de soja.

As amostras possuem grãos com um teor de umidade de aproximadamente 10%. Todas as imagens foram obtidas nas mesmas condições de iluminação, por uma câmera digital modelo Coolpix995, da marca Nikon, com resolução de 3,34 mega pixeis. A câmera foi acoplada a um tripé, que garantiu igual posicionamento na aquisição das imagens. Para garantir um bom contraste durante a realização das fotos, os grãos foram colocados em uma superfície plana e escura, localizada abaixo do tripé.

Por fim, a câmera foi conectada ao computador, onde as imagens foram salvas em formato JPEG. Estas imagens são representadas por três matrizes (R, G e B), com dimensões 1280 por 960 cada, os elementos destas assumem valores discretos no intervalo $[0,255]$. Estas imagens são apresentadas nas figuras a seguir:

Figura 24 – Variedades de soja utilizadas no trabalho

(a) Amostra da cultivar AL 83

(b) Amostra da cultivar BRS 133

(c) Amostra da cultivar BRS 184

(d) Amostra da cultivar BRS 214

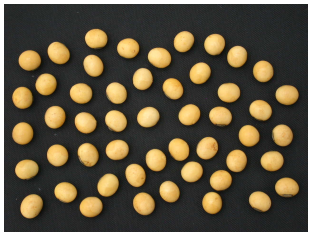
(e) Amostra da cultivar CD 205

(f) Amostra da cultivar CD 206

(g) Amostra da cultivar CD 215

(h) Amostra da cultivar EMB48

(i) Amostra da cultivar MERCEDES 70



(j) Amostra da cultivar MONSOY 5826

(k) Amostra da cultivar NK 8350

(l) Amostra da cultivar RS10

(m) Amostra da cultivar MONSOY 8000 RR

3.2 Pré-processamento

O pré-processamento modificou as imagens de forma a possibilitar uma futura distinção entre os objetos de interesse (grãos de soja) e o fundo da imagem. Para isso, foi utilizada uma máscara binária.

Consideremos $b(x,y)$ como sendo uma imagem binária ideal, onde $b(x,y) = 0$ (preto) e corresponde ao fundo da imagem ou $b(x,y) = 1$ (branco) e corresponde ao interior dos grãos. A imagem pré-processada $h(x,y) = (h_R(x,y), h_G(x,y), h_B(x,y))$ foi gerada como segue:

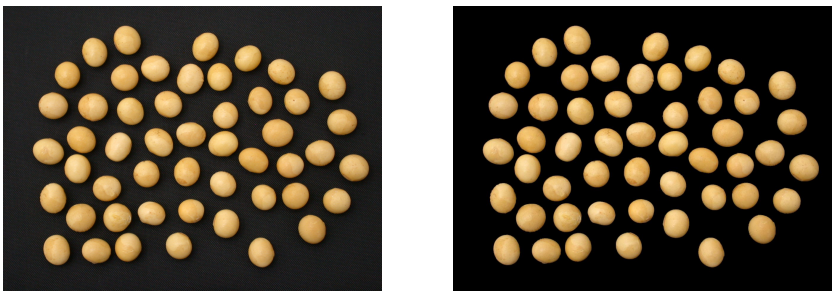
$$\begin{aligned} h_R(x,y) &= f_R(x,y) .* b(x,y) \\ h_G(x,y) &= f_G(x,y) .* b(x,y) \\ h_B(x,y) &= f_B(x,y) .* b(x,y) \end{aligned} \tag{3.1}$$

onde $f_R(x,y)$, $f_G(x,y)$ e $f_B(x,y)$ são as camadas da imagem original $f(x,y)$ e $.*$ é a multiplicação elemento a elemento entre duas matrizes.

Para verificar como funcionou este procedimento, consideremos um pixel qualquer onde $f(x,y) = (f_R(x,y), f_G(x,y), f_B(x,y))$. Se o ponto $\vec{p} = (x,y)$ estiver localizado no fundo, então $h(x,y) = (h_R(x,y), h_G(x,y), h_B(x,y)) = (f_R(x,y).0, f_G(x,y).0, f_B(x,y).0) = (0,0,0)$. Por outro lado, se o ponto estiver localizado no interior de um grão, $h(x,y) = (f_R(x,y).1, f_G(x,y).1, f_B(x,y).1) = (f_R(x,y), f_G(x,y), f_B(x,y)) = f(x,y)$. Em outras palavras, o fundo foi transformado em preto e os grãos permaneceram inalterados.

A Figura 25 apresenta o efeito da aplicação da máscara binária. Nela podemos ver a entrada $f(x,y)$ e a saída $h(x,y)$ da etapa de pré-processamento para uma amostra da cultivar MERCEDES 70.

Figura 25 – Entrada e saída da etapa de pré-processamento sem perda

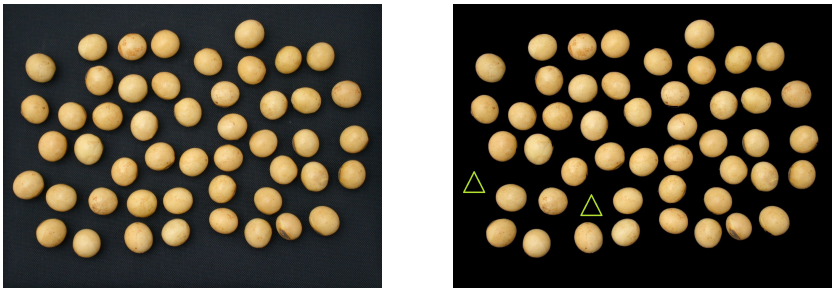


(a) Entrada (b) Saída

Em alguns casos não foi possível obter uma imagem binária ideal. Este fato implicou na

perda de grãos em algumas amostras. Note na Figura 26 que a aplicação da máscara binária na variedade MSOY 8000 RR provocou a perda de dois grãos, indicados por um triângulo verde na imagem de saída. Estes grãos desapareceram durante o processo de geração da imagem binária e resultaram em perdas na imagem pré-processada. O motivo disso será mostrado a seguir.

Figura 26 – Entrada e saída da etapa de pré-processamento com perda



(a) Entrada (b) Saída

O processo de geração da imagem binária foi dividido nas seguintes etapas:

1º Conversão da imagem original para níveis de cinza;

2º Detecção de bordas;

3º Dilatações da imagem;

4º Preenchimento dos grãos;

5º Erosão da imagem;

6º Remoção das áreas com menos de 700 pixeis brancos (ruído).

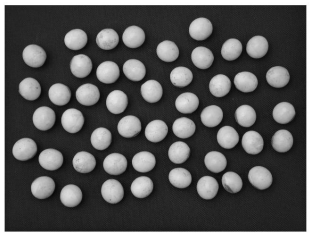
A imagem em tons de cinza $g(x,y)$ foi gerada conforme descrito na Equação 3.2, dada a seguir:

$$g(x,y) = 0,3.R(x,y) + 0,59.G(x,y) + 0,11.B(x,y) \tag{3.2}$$

onde $R(x,y)$, $G(x,y)$ e $B(x,y)$ são as camadas da imagem original $f(x,y)$. Os valores de $g(x,y)$ foram truncados para 256 níveis de cinza. A Equação 3.2 é baseada na sensibilidade em brilho do olho humano em relação as componentes *RGB* e é conhecida na literatura como padrão de conversão NTSC ou ainda Y do espaço de cor *YIQ* (GONZALES; WOODS, 2002). A Figura 27 ilustra a geração da imagem binária na variedade MSOY 8000 RR.

Figura 27 – Geração da imagem em tons de cinza

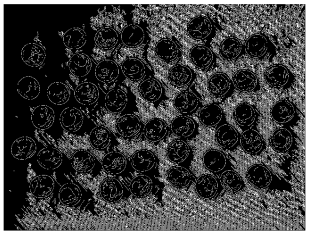
(a) *f(x,y)*



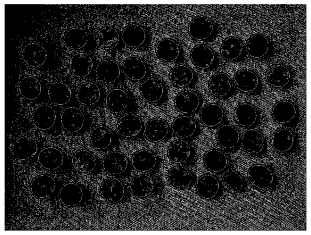
(b) *g(x,y)*

O MATLAB fornece os seguintes filtros como opções para a detecção de contornos: Canny, Log, Prewitt, Roberts, Sobel e Zero Cross. Os filtros de Canny, Log e Zero Cross se mostraram inadequados para o uso neste trabalho, conforme pode ser visto na Figura 28.

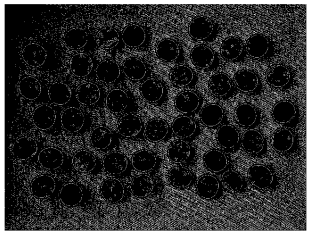
Figura 28 – Aplicação dos filtros de Canny, Log e Zero Cross na detecção de bordas



(a) Filtro de Canny



(b) Filtro Log



(c) Filtro Zero Cross

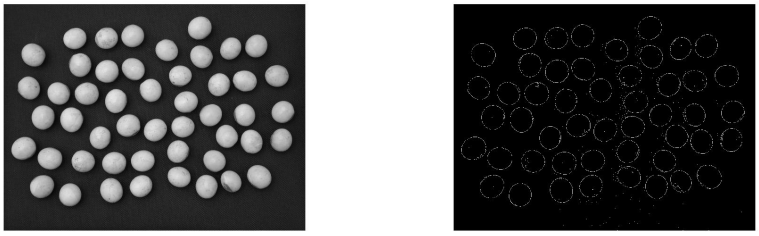
Mesmo após a dilatação da imagem, que será mostrada a seguir, alguns contornos não ficaram contínuos, independente do filtro utilizado. A escolha do filtro ideal para este trabalho se deu com base no número total de grãos ao final da segmentação. Estes dados são apresentados na Tabela 5.

Tabela 5 – Grãos segmentados por filtro

Filtro	Grãos segmentados
Prewitt	619
Roberts	393
Sobel	619

Neste momento do trabalho, verificamos que tanto o filtro de Prewitt quanto o filtro de Sobel poderiam ser selecionados. Optamos pelo filtro de Prewitt, cujo resultado é mostrado na Figura 29.

Figura 29 – Aplicação do filtro de Prewitt



(a) *g(x,y)* antes da aplicação do filtro

(b) *g(x,y)* após a aplicação do filtro (*b(x,y)*)

Conforme pode ser observado na Figura 29 (b), muitos contornos são descontínuos, o que dificultaria o preenchimento dos grãos. Foi então realizada a dilatação da imagem, tanto no sentido horizontal quanto no sentido vertical. Os elementos estruturantes utilizados para isso foram, respectivamente:

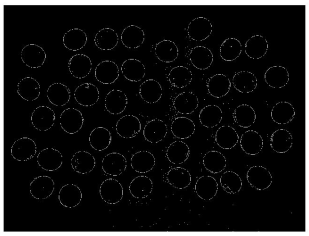
$$dilataHorizontal = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} \tag{3.3}$$

e:

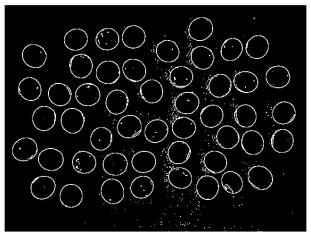
$$dilataVertical = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \tag{3.4}$$

A Figura 30 expõe o efeito da dilatação.

Figura 30 – Dilatação



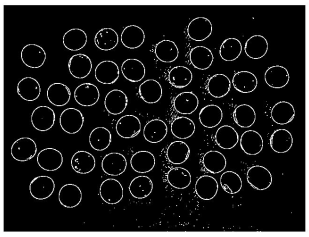
(a) *b(x,y)*



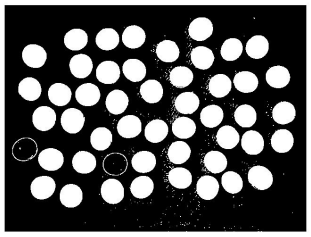
(b) *b(x,y)* dilatada

Para o preenchimento dos grãos foi utilizado o conceito de buraco. Um buraco em uma imagem binária é um conjunto de pixeis que não podem ser alcançados se formos buscando os vizinhos de mesmo valor a partir da borda. Portanto, um grão na imagem binária pode ser visto como um buraco e o valor destes pixeis, pertencentes ao buraco, podem ser alterados de forma a preencher os grãos. A Figura 31 apresenta o resultado deste preenchimento.

Figura 31 – Preenchimento dos grãos



(a) *b(x,y)* antes do preenchimento



(b) *b(x,y)* depois do preenchimento

Contudo, como pode ser observado, alguns grãos não foram preenchidos. Isto deve-se ao fato de seus contornos permanecerem descontínuos mesmo após a dilatação. Uma solução alternativa seria o uso de elementos estruturantes maiores. Porém, foi experimentalmente verificado que isto resultaria no desaparecimento de partes dos grãos, o que seria um prejuízo significativo já que os grãos são os objetos de análise no trabalho. Portanto, optou-se pela perda de grãos em algumas variedades, o que diminuiu um pouco o tamanho das amostras (quantidade de grãos).

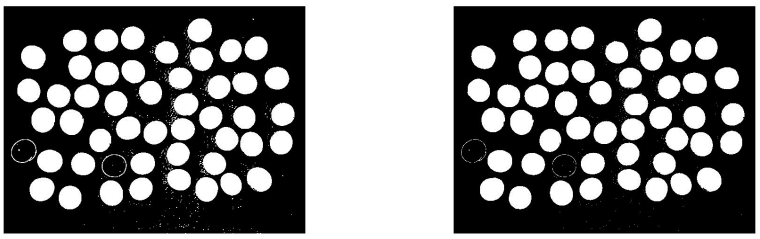
Um outro efeito da dilatação foi a expansão dos grãos. Para desfazer esta expansão foi realizada a erosão da imagem, que também reduziu a quantidade de ruídos na mesma. O elemento

estruturante utilizado para isto foi:

$$erode = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix} \tag{3.5}$$

A Figura 32 exibe o resultado da erosão.

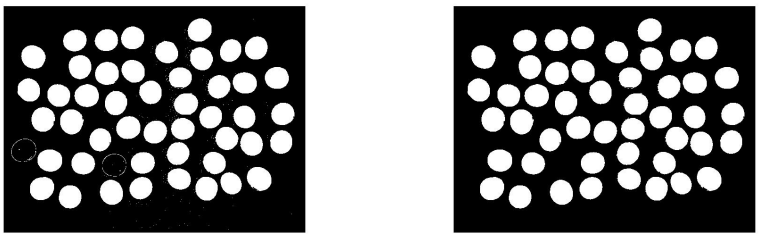
Figura 32 – Erosão



(a) *b(x,y)* antes da erosão (b) *b(x,y)* depois da erosão

Por fim, foi verificado que todos os grãos ocupavam uma área com mais de 700 pixeis brancos na imagem. Então, foram então removidas da imagem as áreas com quantidades menores que 700 pixeis brancos, conforme pode ser visto na Figura 33.

Figura 33 – Remoção das áreas com menos de 700 pixeis brancos



(a) *b(x,y)* antes da remoção (b) *b(x,y)* depois da remoção

3.3 Segmentação

Conforme mencionado anteriormente, um grão na imagem binária pode ser visto como um buraco. Para cada amostra foi realizada uma varredura na imagem binária e foram armazenadas as posições de contorno dos grãos. Para cada grão foram identificados os valores x_{min}, x_{max}, y_{min} e y_{max} do menor retângulo que o delimita. Estes valores foram utilizados para obter um grão segmentado a partir da imagem pré-processada. A Figura 34 ilustra um grão segmentado da variedade BRS 184.

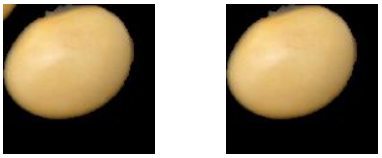
Figura 34 – Grão segmentado da variedade BRS 184

Para padronizar o tamanho das imagens segmentadas foi verificada a dimensão do maior retângulo dentre todos os grãos de todas as variedades. As imagens segmentadas foram colocadas sobre uma imagem preta de 123 por 121 (dimensão do maior retângulo). A sobreposição deu-se pelo alinhamento do canto superior esquerdo das imagens e pode ser verificada na Figura 35.

Figura 35 – Grão segmentado da cultivar *BRS 184* (tamanho padronizado)

Em particular, a escolha deste grão para ilustrar o processo de segmentação deve-se ao fato dele apresentar ruídos de outro grão na imagem segmentada. Isto decorre do uso de retângulos na segmentação da imagem. Para eliminar este tipo de ruído foram utilizados os valores x_{min}, x_{max}, y_{min} e y_{max} do grão segmentado, para obter a sub-matriz correspondente na imagem binária. Foram então eliminadas as áreas menores que 700 pixeis brancos nesta sub-matriz e a mesma foi utilizada como uma máscara binária na imagem segmentada, da mesma forma que ocorreu na etapa de pré-processamento. A Figura 36 mostra a semente segmentada antes e depois da aplicação da máscara binária.

Figura 36 – Remoção de ruídos na imagem segmentada



(a) Antes da remoção

(b) Depois da remoção

A Tabela 6 fornece a quantidade de grãos segmentados por variedade.

Tabela 6 – Quantidade de grão segmentados por variedade

Variedade	Quantidade de grãos segmentados
AL 83	50
BRS 133	44
BRS 184	47
BRS 214	48
CD 205	39
CD 206	50
CD 215	51
EMB 48	49
MERCEDES 70	50
MSOY 5826	47
NK 8350	47
RS 10	50
MSOY 8000 RR	47
TOTAL	619

3.4 Extração das características

A partir das imagens segmentadas foi gerado o conjunto de dados de treinamento da RNA, composto por pares entrada-saída.

Para que os grãos pudessem ser entendidos pela rede neural como sinais de entrada, foram extraídas características quantitativas destes. Cada imagem segmentada é representada por três matrizes (R, G e B). Estas matrizes possuem 14883 elementos (123 por 121). Os elementos das matrizes foram colocados em vetores de entrada. Portanto, um vetor de entrada possui 44649 elementos (3 camadas de 14883).

Além disso, as variedades de soja foram codificadas em vetores para que a rede pudesse interpretar as saídas desejadas. Esta codificação encontra-se na Tabela 7.

Tabela 7 – Representação das cultivares na RNA

AL 83	BRS 133	BRS 184	BRS 214	CD 205	CD 206	CD 215	EMB 48	MERCEDES 70	MSOY 5826	NK 8350	RS 10	MSOY 8000 RR
1	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0	0	0	0
0	0	0	0	0	0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	0	0	0	0	1

3.5 Reconhecimento e interpretação

Uma RNA foi utilizada para o reconhecimento e interpretação dos dados. Esta rede, com topologia *feed forward* e uma camada de neurônios ocultos empregou a função de ativação logística, com a = 1, nos neurônios ocultos e a função de transferência linear nos neurônios de saída.

Os parâmetros livres da rede foram inicializados de forma aleatória. Após inicializados, os pesos foram ajustados de acordo com o algoritmo de aprendizagem *back propagation*.

A escolha do número de neurônios ocultos ideal foi baseada na capacidade de generalização da rede, a qual foi mensurada em diversos modelos. Os modelos diferem-se entre si pelo número de neurônios ocultos. Foram analisados modelos com $5n$ neurônios ocultos, onde $n = 1, 2, ..., 20$. Em outras palavras, modelos de 5 até 100 neurônios ocultos, variando a um passo de 5. Para cada um destes modelos o conjunto de dados foi dividido de acordo com a Tabela 8.

Tabela 8 – Divisão do conjunto de dados

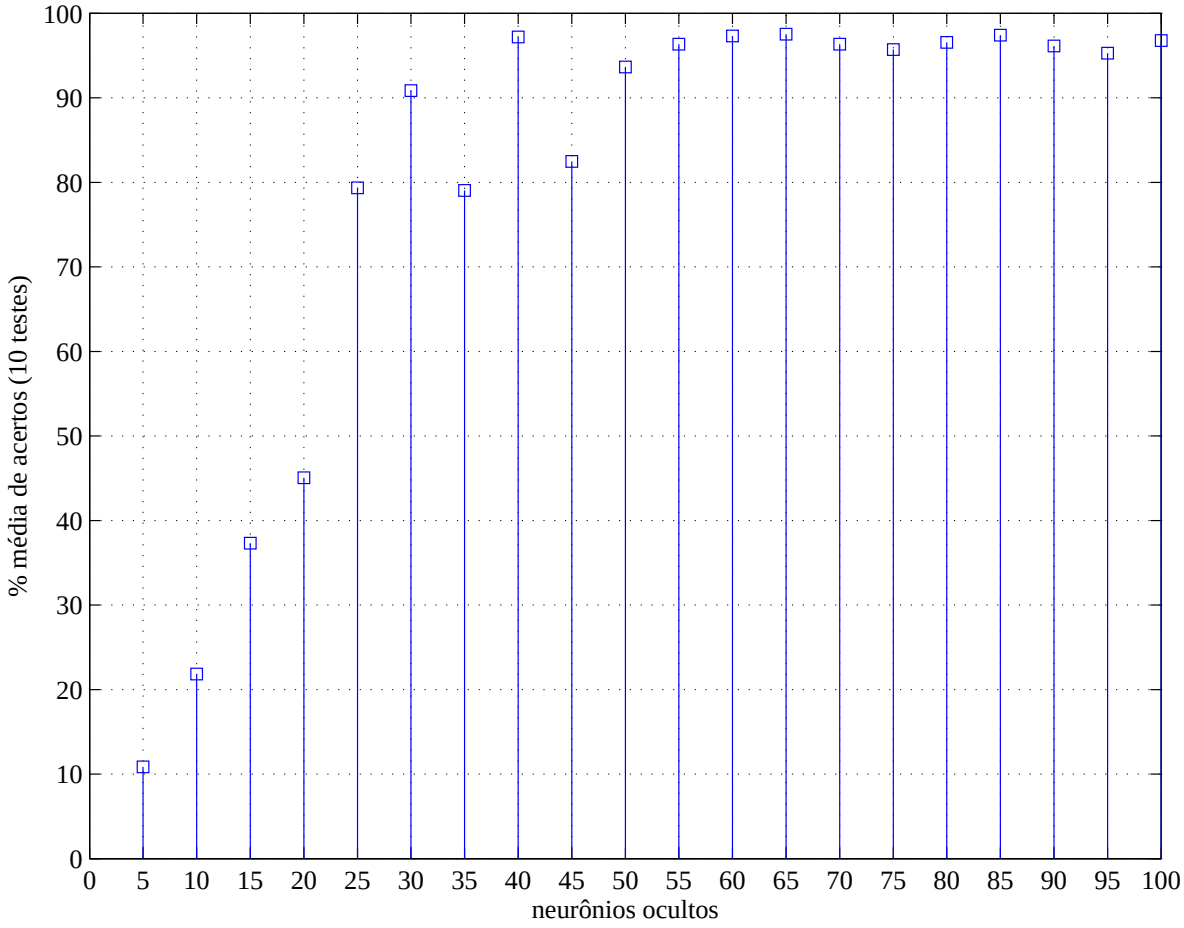
Subconjunto	% em relação ao conjunto de dados
Estimação	70
Validação	15
Teste	15

O subconjunto de estimação foi utilizado para ajustar os parâmetros livres da rede. O desempenho de generalização foi calculado época por época, com base no subconjunto de validação, até atingir um desempenho máximo. Porém, para evitar um ajuste excessivo do subconjunto de validação, o subconjunto de teste foi utilizado para determinar a capacidade de generalização do modelo. Para testar os modelos, as entradas dos padrões do subconjunto de teste foram apresentadas a rede já treinada, ou seja, com os parâmetros livres já fixados. A resposta da rede diante de uma dessas entradas foi então comparada a saída desejada do respectivo padrão, caracterizando ou não um acerto. A porcentagem de acertos foi então contabilizada. Todo este processo foi repetido 10 vezes. Em cada uma destas foi dada uma distribuição aleatória do conjunto de dados (com as proporções especificadas na Tabela 8). Uma média aritmética das 10 realizações do processo determinou o desempenho de generalização do modelo. Os resultados são mostrados em detalhes no próximo capítulo.

4 RESULTADOS E DISCUSSÕES

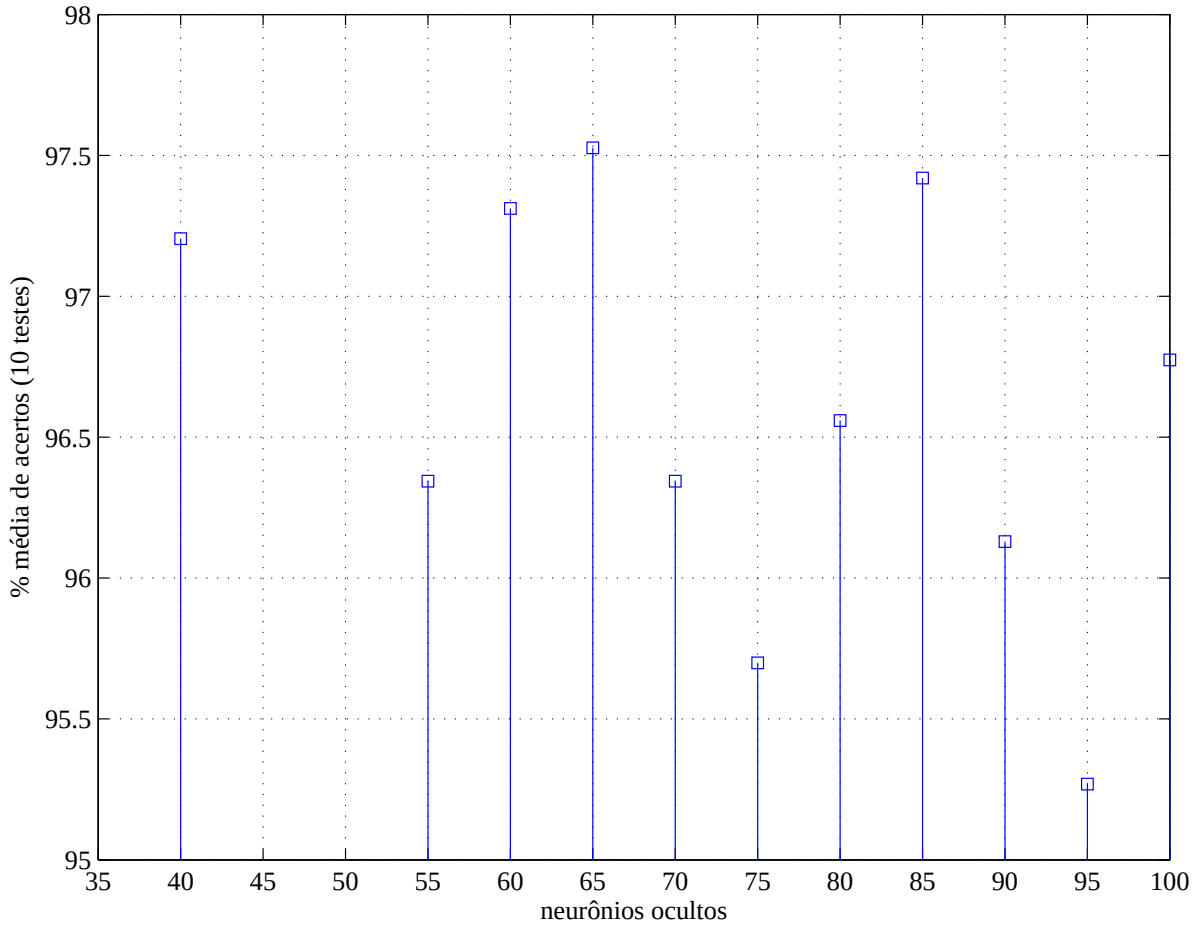
A capacidade de generalização dos modelos foi dada pela porcentagem média de acertos, tomada com base nos 10 subconjuntos de teste aos quais cada modelo foi submetido. Estes resultados são apresentados na Figura 37.

Figura 37 – Capacidade de generalização dos modelos avaliados



Diversos modelos destacaram-se com um desempenho de generalização superior a 90%, conforme pode ser observados em mais detalhes na Figura 38.

Figura 38 – Capacidade de generalização dos modelos em destaque



Com base nestes resultados, o modelo com 65 neurônios ocultos foi selecionado para uma análise mais detalhada. O processo de estimação, validação e teste foi repetido mais 10 vezes, conforme já havia sido realizado anteriormente, diferenciando-se agora por utilizar outras 10 distribuições aleatórias. O resultado deste estudo é mostrado na Tabela 9.

Tabela 9 – Distribuição aleatória /% de acertos no modelo selecionado

Distribuição	Porcentagem de acertos (aproximada)
1	95,7
2	97,8
3	96,8
4	95,7
5	95,7
6	100
7	96,8
8	95,7
9	93,5
10	89,2

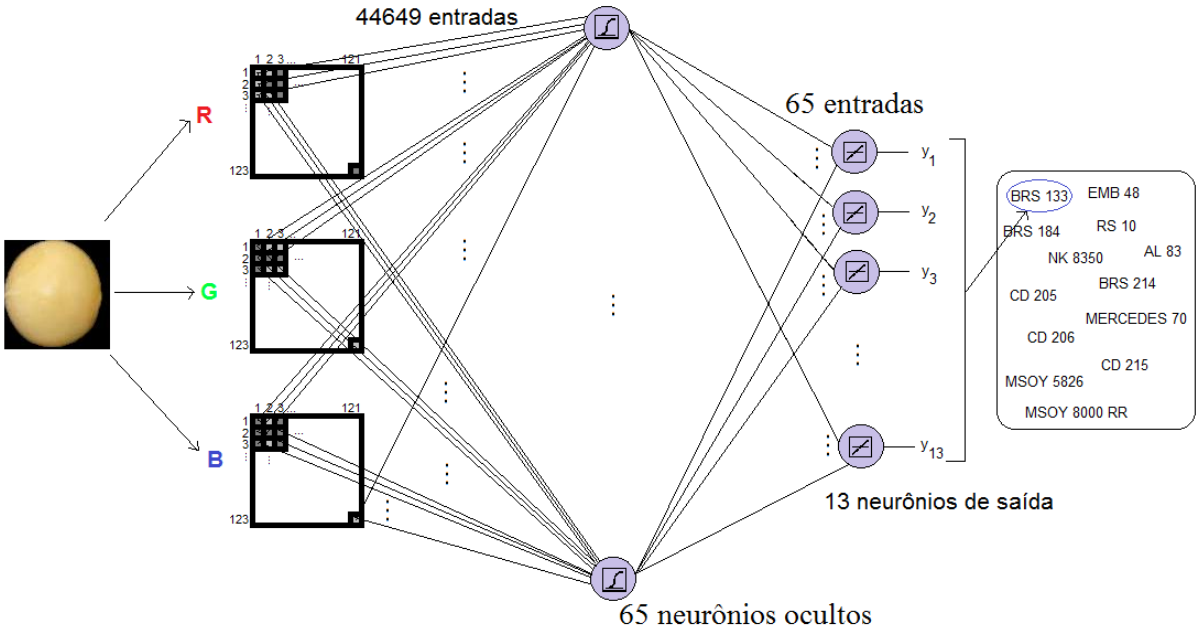
Podemos verificar que o modelo desta vez identificou em média 95,7% dos grãos corretamente, com um desvio padrão de 2,8%. Embora este resultado seja satisfatório, ainda precisamos estar conscientes da possibilidade de o fator aleatório omitir erros de treinamento. Por exemplo, suponhamos que a rede não houvesse aprendido a identificar corretamente os grãos de uma das variedades. Este fato poderia passar despercebido se os conjuntos de teste gerados de forma aleatória não contivessem grãos desta variedade. Ao realizarmos os 10 experimentos praticamente eliminamos esta possibilidade. Entretanto, para um maior esclarecimento a respeito destes conjuntos de teste podemos avaliar a Tabela 10, conhecida como matriz de confusão média, onde a soma dos elementos de uma linha qualquer indica quantos grãos em média nas distribuições aleatórias pertenciam a variedade em questão. Também é possível extrair desta tabela a porcentagem média de acertos por variedade, dividindo o elemento da diagonal principal pela soma dos elementos da linha.

Tabela 10 – Matriz de confusão do modelo selecionado

	AL 83	BRS 133	BRS 184	BRS 214	CD 205	CD 206	CD 215	EMB 48	MERCEDES 70	MSOY 5826	NK 8350	RS 10	MSOY 8000 RR
AL 83	6,9	0,1	0,1	0	0	0	0	0,1	0	0	0	0	0
BRS 133	0	5,9	0	0	0	0	0	0	0	0	0,1	0	0
BRS 184	0	0	7,7	0	0	0	0	0,2	0	0	0	0,1	0
BRS 214	0,1	0	0	6,2	0	0	0	0	0	0,1	0	0	0
CD 205	0	0,1	0	0	5,5	0	0,3	0	0	0	0	0	0,2
CD 206	0	0	0	0	0	6,5	0	0	0	0	0	0	0
CD 215	0	0	0	0,1	0,1	0	6,6	0,5	0	0,1	0	0	0
EMB 48	0	0	0	0	0	0	0,1	7,5	0	0	0	0	0
MERCEDES 70	0	0	0	0	0	0	0	0	8,1	0	0	0	0
MSOY 5826	0	0	0	0,1	0	0	0,5	0,2	0	8,4	0	0	0
NK 8350	0	0,1	0	0	0	0	0	0	0	0	8,5	0	0
RS 10	0	0	0	0	0	0,3	0	0	0	0	0,1	6,2	0
MSOY 8000 RR	0	0	0	0	0	0	0,2	0,1	0	0	0	0	5

A Figura 39 ilustra o reconhecimento de um grão da variedade BRS 133 utilizando a arquitetura selecionada, dando uma visão geral de como o *software* desenvolvido neste trabalho pode ser aplicado para identificar variedades de soja.

Figura 39 – Aplicação do modelo selecionado



5 CONCLUSÃO

Neste trabalho, foram utilizados o processamento digital de imagens e redes neurais artificiais para a identificação de variedades de soja com o objetivo de reduzir os custos relacionados a este processo. A cor dos grãos foi utilizada para o reconhecimento das variedades, fator este que se mostrou determinante para realizar de forma satisfatória este procedimento.

Os grãos de soja, utilizados como dados de entrada na rede, foram extraídos das imagens originais através do uso de máscaras binárias em conjunto com uma série de outras técnicas de processamento de imagens. Após a extração dos dados, diversas arquiteturas *feed forward* com uma camada oculta foram treinadas através do algoritmo de *back propagation*. A escolha da arquitetura ideal deu-se com base na capacidade de generalização da mesma.

Realizado o treinamento, a arquitetura selecionada obteve em média 97,5% de acertos na identificação das variedades de grãos que não haviam sido apresentados anteriormente para a rede. Estes resultados mostram que redes neurais podem se tornar uma ferramenta alternativa no processo de reconhecimento.

Em trabalhos futuros, pode-se estudar o desenvolvimento de um *hardware* específico para automatizar a tarefa do reconhecimento via imagens e a implantação deste em um ambiente adequado junto a profissionais na área de agricultura.

REFERÊNCIAS

BRAGA, A. de P.; LUDEMIR, T. B.; CARVALHO, A. C. P. de L. F. **Redes Neurais Artificiais:** teoria e aplicação. Rio de Janeiro: LTC, 2000. 262 p.

BRASIL. **Lei n. 9.456, de 25 de abril de 1997**. Institui a lei de proteção de cultivares e dá outras providências. Diário Oficial da República Federativa do Brasil, Brasília, DF, 1997. Disponível em: <http://www.planalto.gov.br/ccivil_03/leis/L9456.htm>. Acesso em: 07 nov. 2012, 20:45:30.

EMPRESA BRASILEIRA DE PESQUISA AGROPECUÁRIA. **Soja em Números**. 2011. Disponível em: <http://www.cnpso.embrapa.br/index.php?cod_pai=2&op_page=294>. Acesso em: 07 nov. 2012, 20:44:30.

GONZALES, R. C.; WOODS, R. E. **Digital Image Processing**. 2. ed. Saddle River: Prentice Hall, 2002. 793 p.

HAYKIN, S. **Redes Neurais:** princípios e prática. Porto Alegre: Bookman, 2001. 900 p.

HEBB, D. O. **The Organization of Behavior**. New York: Wiley, 1949. 378 p.

KLEENE, S. C. **Representation of Events in Nerve Nets and Finite Automata**. United States Air Force Project Rand Research Memorandum, 1951.

KOSCHAN, A.; ABIDI, M. **Digital Color Image Processing**. Hoboken: Wiley, 2008. 382 p.

MARQUES FILHO, O.; VIEIRA NETO, H. **Processamento Digital de Imagens**. Rio de Janeiro: Brasport, 1999. 331 p.

MATHWORKS. **Image Processing Toolbox**. 2011. Disponível em: <http://www.mathworks.com/products/matlab/>. Acesso em: 07 nov. 2012, 20:54:30.

MATHWORKS. **MATLAB**. 2011. Disponível em: <http://www.mathworks.com/products/matlab/>. Acesso em: 07 nov. 2012, 20:54:00.

MATHWORKS. **Neural Network Toolbox**. 2011. Disponível em: <http://www.mathworks.com/products/matlab/>. Acesso em: 07 nov. 2012, 20:55:00.

MCCULLOCH, W. S.; PITTS, W. H. **A Logical Calculus of the Ideas Immanent in Nervous Activity**. Bulletin of Mathematical Biophysics, v. 5, p. 115–133, 1943.

MILLACH, S. C. K. **Marcadores Moleculares nos Recursos Genéticos e no Melhoramento de Plantas**. EMBRAPA - Empresa Brasileira de Pesquisa Agropecuária, 2005. Disponível em: <http://www.cpatsa.embrapa.br/catalogo/livrorg/marcadormolecular.pdf>. Acesso em: 07 nov. 2012, 20:47:30.

MINISTÉRIO DA AGRICULTURA, PECUÁRIA E ABASTECIMENTO. **Soja**. 2011. Disponível em: <http://www.agricultura.gov.br/vegetal/culturas/soja>. Acesso em: 07 nov. 2012, 20:46:30.

MINSKY, M.; PAPERT, S. **Perceptrons**. Oxford: MIT Press, 1969. 292 p.

PADILHA, F. R. R. **Reconhecimento de Variedades de Soja Através do Processamento de Imagens Digitais Usando Redes Neurais Artificiais**. 2007. 94 f. Dissertação (Mestrado em Modelagem Matemática) – UNIJUI, Ijuí, 2007.

RIVAS, M. B. **Soja:** qualidade de vida e saúde com prazer e saúde. Porto Alegre: AGE, 2006. 179 p.

RIVISTA DI NEUROSCIENZE, PSICOLOGIA E SCIENZE COGNTIVE. **I Neuroni e le Cellule Gliali**. 2010. Disponível em: <http://www.neuroscienze.net/?p=1396>. Acesso em: 07 nov. 2012, 20:53:30.

RONZELLI JUNIOR, P. **Melhoramento Genético de Plantas**. Curitiba: Editora Gráfica LTDA., 1996. 219 p.

ROSENBLATT, F. **Principle of Neurodynamics**. Washington: Spartan, 1962. 616 p.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. **Error Propagation, in Parallel Distributed Processing**. STANFORD UNIVERSITY - Department of Psychology, 1986. Dis-ponível em: <http://psych.stanford.edu/~jlm/papers/PDP/Volume1/Chap8_PDP86. pdf>. Acesso em: 07 nov. 2012, 20:54:30.

SILVA, L. N. C. **Análise e Síntese de Estratégias de Aprendizado para Redes Neurais**. 1998. 248 f. Dissertação (Mestrado em Computação) – Faculdade de Engenharia Elétrica e Computação, UNICAMP, Campinas, 1998.

TREVISOLI, S. H. U. **Melhoramento Genético da Soja**. ZPM, 2007. Disponível em: <http://www.zebuparaomundo.com/zebu/index.php?option=com_content&task=view&id=363&Itemid=38>. Acesso em: 07 nov. 2012, 20:49:30.

UNITED STATES DEPARTMENT OF AGRICULTURE. **Plants Profile**. 2012. Disponível em: <http://plants.usda.gov/java/profile?symbol=glma4>. Acesso em: 07 nov. 2012, 20:48:30.

VEELENTURF, L. P. J. **Analyses and Applications of Artificial Neural Networks**. Salisbury: Prentice Hall, 1995. 259 p.