FUNDAÇÃO DE ENSINO “EURÍPIDES SOARES DA ROCHA”
CENTRO UNIVERSITARIO “EURÍPIDES DE MARÍLIA” – UNIVEM
CURSO DE CIÊNCIA DA COMPUTAÇÃO – BACHARELADO

FÁBIO BARROS TEBALDI

ESTUDO DE REDES NEURAIS ARTIFICIAS PARA VERIFICAÇÃO E CLASSIFICAÇÃO DE ASSINATURAS ATRAVÉS DE IMAGENS

MARÍLIA
2007

FABIO BARROS TEBALDI

ESTUDO DE REDES NEURAIS ARTIFICIAS PARA VERIFICAÇÃO E CLASSIFICAÇÃO DE ASSINATURAS ATRAVÉS DE IMAGENS

Trabalho de Conclusão de Curso submetido ao Centro Universitário Eurípides de Marília, mantido pela Fundação de Ensino Eurípides Soares da Rocha, para obtenção do Título de Bacharel em Ciência da Computação. (Área de Concentração: Redes Neurais Artificiais).

Orientador:
Prof. Dr. José Celso Rocha

MARÍLIA
2007

CENTRO UNIVERSITÁRIO EURÍPIDES DE MARÍLIA
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

TRABALHO DE CONCLUSÃO DE CURSO – AVALIAÇÃO FINAL

Fabio Barros Tebaldi

ESTUDO DE REDES NEURAIS ARTIFICIAIS PARA VERIFICAÇÃO E CLASSIFICAÇÃO DE ASSINATURAS ATRAVÉS DE IMAGENS

Banca examinadora da monografia apresentada ao Curso de Bacharelado em Ciência da Computação do UNIVEM/F.E.E.S.R., para obtenção do Título de Bacharel em Ciência da Computação.

Nota: 8,5 (Oito e Meio)

Orientador: José Celso Rocha

1°. Examinador: Ildeberto de Gênova Bugatti

2°. Examinador: Rodolfo Barros Chiaramonte

Marília, 22 de novembro de 2007.

TEBALDI, Fábio Barros e. Estudo de redes neurais artificiais para verificação e classificação de assinaturas através de imagens. 2007. Monografia (Bacharelado em Ciência da Computação) – Centro Universitário Eurípides de Marília, Fundação de Ensino Eurípides Soares da Rocha, Marília, 2007.

RESUMO

Este trabalho apresenta o estudo de redes neurais artificiais usada na verificação e classificação de assinaturas através de imagens. A estrutura do trabalho é dividida em duas partes, na primeira fase é feito um estudo sobre redes neurais artificiais, levantando seu aspecto histórico, teórico e sua arquitetura, além de métodos utilizados para classificar uma assinatura. A segunda fase está dirigida à implementação do projeto. Foi implementado uma rede neural do tipo MLP treinada com o algoritmo *R-prop* para verificação e classificação de assinaturas, o sistema é capaz de identificar o usuário pertencente à assinatura apresentada à rede neural.

Palavras-chave: Redes Neurais Artificiais; Inteligência Artificial; Assinaturas.

TEBALDI, Fábio Barros e. Estudo de redes neurais artificiais para verificação de assinaturas através de imagens. 2007. Monografia (Bacharelado em Ciência da Computação) – Centro Universitário Eurípides de Marília, Fundação de Ensino Eurípides Soares da Rocha, Marília, 2007.

ABSTRACT

This work presents the study of artificial neural networks used in the verification and classification of signatures through images. The structure of the work is divided in two parts, in the first phase is made a study on artificial neural networks, raising its historical, theory aspects and its architecture, beyond used methods to classify a signature. The second phase is directed to the implementation of the project. Was be implemented a MLP type neural network trained with the R-prop algorithm for verification and classification of signatures, the system is able to identify the signature user presented to the neural network.

Keywords: Artificial Neural Networks; Artificial Intelligence; Signatures.

LISTA DE FIGURAS

SUMÁRIO

Capítulo 1 - INTRODUÇÃO

Segurança é de suma importância para humanidade, meios para garantir a integridade de informações sempre foram objetivos de estudos e pesquisas realizadas por vários cientistas. Existem vários métodos para controlar e diferenciar uma pessoa de outra. Métodos tradicionais de controle de acesso através de senhas e documentações particulares pode ser uma forma de diferenciar uma pessoa de outra, porém não oferecem a segurança necessária para garantir a integridade no controle de acesso já que podem ser facilmente copiadas. Outro método e mais eficaz para garantir essa integridade, é através de componentes biológicos que são únicos entre cada pessoa, como por exemplo: impressões digitais, assinaturas, voz, íris entre outros.

Neste projeto será abordado o uso de assinaturas com a finalidade de identificar um usuário do sistema através de sua assinatura. Para tal finalidade será necessária a criação de um método que faça essa verificação. Atualmente uma das formas que vem apresentando melhores resultados para questões de classificação e reconhecimento de padrões são as redes neurais artificiais.

Redes neurais artificiais são modelos de computação inspiradas no funcionamento do cérebro humano. Uma rede neural é capaz de aprender e generalizar uma determinada função através de um padrão de treinamento, o que as torna uma alternativa muito interessante para a função de verificação de assinaturas.

O objetivo deste projeto é implementar uma rede neural artificial para classificar uma assinatura de forma única. Para isso é necessário ser feito uma série de estudos sobre o funcionamento de uma rede neural, como elas se comportam e como são construídas, além de estudos sobre métodos para classificar uma assinatura. Outro ponto importante para o desenvolvimento do projeto é a escolha do software a ser utilizado para implementação. Para

a construção da rede neural será utilizado o software Matlab e seus *toolboxes* de redes neurais e processamento de imagens.

Nos capítulos subseqüentes será mostrado o estudo feito sobre redes neurais artificiais e os principais tipos de arquiteturas utilizadas, métodos para classificar uma assinatura de forma única, além da implementação de uma rede neural capaz de classificar padrões utilizando o software Matlab.

Capítulo 2 - REDES NEURAIS ARTIFICIAIS

O cérebro humano é o dispositivo mais complexo conhecido pelo Homem, a capacidade de pensar, memorizar e resolver problemas tem levado vários cientistas a tentar criar modelos computacionais que visam representar a funcionalidade do cérebro humano. Um desses modelos resultou na criação das Redes Neurais Artificiais (RNA).

> RNAs são sistemas paralelos distribuídos compostos por unidades de processamento simples (nodos) que calculam determinadas funções matemáticas (normalmente não-lineares). Tais unidades são dispostas em uma ou mais camadas e interligadas por um grande numero de conexões, geralmente unidirecionais. Na maioria dos modelos estas conexões estão associadas a pesos, os quais armazenam o conhecimento representado no modelo e servem para ponderar a entrada recebida por cada neurônio da rede. O funcionamento destas redes é inspirado em uma estrutura física concebida pela natureza: o cérebro humano.(Braga, Ludermir & Carvalho - Redes Neurais Artificiais Teoria e aplicações, 2000).

As redes neurais artificiais derivam do conceito de inteligência artificial conexionista, onde se busca construir um sistema capaz de exibir um comportamento inteligente, uma metáfora cerebral. De acordo com FOGEL (1997), inteligência pode ser definida como a capacidade de um sistema adaptar seu comportamento para atingir seus objetivos em uma variedade de ambientes.

O objetivo da criação de sistemas baseados em inteligência artificial é desenvolver sistemas para realização de tarefas que geralmente são realizadas melhores por humanos do que por maquinas, ou não possuem uma solução algorítmica viável pela computação convencional.

Diferente dos sistemas computacionais tradicionais, as redes neurais não são programadas ou executam operações lógicas. As RNAs são treinadas por meio de padrões de treinamento que são representados por vetores originários de fontes como: imagens, sinais de voz e dados financeiros. Devido à forma de treinamento e aprendizagem as redes neurais são capazes de generalizar, reconhecendo a informação aprendida mesmo que esteja incompleta, não necessita de um algoritmo explicito nem da descrição do problema, baseada na

adaptabilidade, além de possuir um grande paralelismo natural devido a sua arquitetura, o que gera rapidez no processamento criando a possibilidade de um desempenho superior aos modelos convencionais.

A capacidade de uma rede neural depende, principalmente, da sua estrutura paralela distribuída e de sua habilidade de aprender e, como conseqüência, generalizar. Algumas das características importantes das redes neurais são:

- Tolerância à falhas, que permite que a rede continue a apresentar resultados aceitáveis, no caso de falha de algum neurônio. A informação contida na rede está distribuída por todos os seus elementos, possibilitando que, mesmo que parte da rede seja destruída, a informação esteja contida nos elementos restantes e possa ser recuperada.
- Generalização, que possibilita à rede obter saídas adequadas como resposta a dados de entrada desconhecidos, ou seja, não pertencentes ao conjunto de treinamento.
- Capacidade de aprendizagem, processo que envolve a modificação dos pesos sinápticos de uma rede através da aplicação de um conjunto de pares de treinamento. O treinamento é repetido até que a rede atinja um nível em que não haja mudanças significativas nos pesos.
- Habilidade de aproximação - Dada à capacidade de aprendizado, a rede tem a possibilidade de encontrar qualquer mapeamento entrada/saída, e, desde que os dados sejam representativos do processo que se esteja tratando e desde que sejam adequadamente escolhidos a arquitetura de rede e o seu algoritmo de treinamento, as redes são capazes de aproximar funções contínuas de ordem qualquer.

2.1. Histórico

BRAGA, LUDERMIR & CARVALHO (2000) comentam que o primeiro modelo artificial de um neurônio biológico foi desenvolvido por Warren McCulloch e Walter Pitts em 1943. O trabalho de McCulloch e Pitts se concentra em descrever um modelo artificial de um neurônio e apresentar suas capacidades computacionais que em apresentar técnicas de aprendizado.

Alguns anos depois o aprendizado de redes biológicas e artificiais veio a ser objeto de estudo. Em 1949 Donald Hebb mostrou como a plasticidade da aprendizagem de redes neurais é conseguida através da variação dos pesos de entrada dos nodos. A regra de Hebb propõe uma teoria para explicar o aprendizado em nodos biológicos baseada no reforço das ligações sinápticas entre nodos excitados. Mais tarde surge a regra de Widrow-Hoff, também conhecida por *regra delta*. A regra delta ainda hoje é bastante utilizada, é baseada no método do gradiente para minimização do erro na saída de um neurônio com resposta linear.

Em 1958, Frank Rosenblatt cria o modelo *perceptron* com a finalidade de projetar uma RNA que fosse capaz de fazer descobertas sem a necessidade de regras. Rosenblatt descreveu uma topologia de RNA, estruturas de ligação entre os nodos e propôs um algoritmo para treinar a rede. O perceptron simples comporta-se como um classificador de padrões, dividindo o espaço de entrada em regiões distintas para cada uma das classes existentes.

Em 1969 Minsky e Papert lançam o livro *Perceptrons* onde apontam os problemas do modelo Perceptron, como a falha em resolver problemas que não são linearmente separáveis como o caso da função boolena *XOR* ou *ou-exlusivo*.

A década de 70 (1969-1982) foi considerada como infrutífera para área, causando grande desinteresse em relação a pesquisas sobre redes neurais artificiais devido aos argumentos de Minsky e Papert.

Em 1982, John Hopfield publicou um artigo que chamou a atenção para as propriedades associativas das RNAs. O grande feito de Hopfield foi mostrar a relação entre redes recorrentes auto-associativas e sistemas físicos. Alguns anos mais tarde, o algoritmo de treinamento *back-progation* mostrou que as RNAs de múltiplas camadas são capazes de resolver "problemas difíceis de aprender" como problemas que não são linearmente separáveis.

A tabela abaixo descreve o histórico das RNAs de acordo com sua cronologia:

RESUMO DA EVOLUÇÃO DAS REDES NEURAIS ARTIFICIAIS	
1943	Psychon – McCulloch & Pits (Exitação/ Inibição – Sem aprendizado)
1949	Regra de Hebb – D.O. Hebb (Aprendizado/ Adaptação)
1960	Adaline – Widrow & Hoff (Valores discretos/ binários/ Regra Delta)
1962	Perceptron – Frank Rosenblatt (Valores contínuos) Madaline – Bernard Widrow (Combinando Adalines manualmente)
1969	Problemas do XOR – Minsky & Papert (Livro "Perceptrons")
1970-1980	Década infrutífera para área.
1982	Modelo de Hopfield (Redes recorrentes – Memórias Auto-Associativas) Modelo de Kohonen – SOFM (Redes recorrentes – Clustering)
1983	Modelo ART – Carpenter & Grossberg (Protótipos não-supervisionados)
1986	MLP Back-Propagation – Rumelhart, Hinton & Willians (Multi-nível)
1980-2000	Aplicações em jogos, robótica, visão, reconhecimento de imagens e padrões, previsões temporais. Revisão de conceitos e limitações.

Tabela 1 – Cronograma das redes neurais artificiais.

2.2. Neurônio Biológico

O cérebro humano possui cerca de 10 bilhões de neurônios, que são responsáveis pelo que se chama de emoção, pensamento, percepção e cognição, além de funções sensoriais e motoras. Sua rede de nodos tem a capacidade de reconhecer padrões, armazenar conhecimento e interpretar observações. Em um humano adulto a massa do cérebro é equivalente a 2% do seu peso, onde pode variar entre 1 e 2 Kg. O córtex cerebral é o que diferencia os humanos das outras espécies, o córtex humano possui cerca de 2000cm2 e 3x10 elevado à 10 neurônios.

> Apesar do estudo continuo, o funcionamento das redes biológicas ainda não foi totalmente desvendado pelo homem. Não se sabe ao certo a forma como as funções cerebrais são realizadas. O que se tem até o momento são modelos, os quais são atualizados a cada nova descoberta. No entanto, a estrutura fisiológica básica destas redes de nodos naturais é conhecida, e é exatamente nesta estrutura fisiológica que se baseiam as RNAs.(Braga, Ludermir & Carvalho - Redes Neurais Artificiais Teoria e aplicações, 2000).

A célula é composta basicamente de água, eletrólitos, proteínas, lipídios e carboidratos. São divididas em núcleo e citoplasma. O núcleo controla as reações químicas e a reprodução e o citoplasma é onde as organelas estão dispersas.

O sistema nervoso controla as reações rápidas do corpo, funções motoras e sensoras. Recebe informações dos sensores combinando essas informações com informações armazenadas para produzir uma resposta. É composto pela medula da coluna vertebral e por componentes do cérebro como mesencéfalo, hipotálamo, tálamo, cerebelo e córtex.

A figura 1 exemplifica as principais regiões que compõem o cérebro humano.

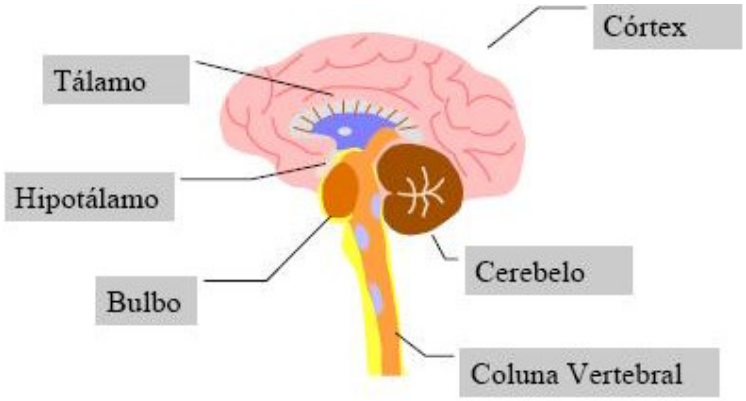

Figura 1 – O Cérebro Humano.[1]

O neurônio possui um corpo que é chamado de *soma* e medem apenas alguns milésimos de milímetros, é o centro dos processos metabólicos da célula nervosa, a partir da qual projetam-se filamentos, os dendritos e o axônio. Os Dendritos podem ter um volume maior do que o próprio corpo celular. São dispositivos de entrada e tem a função de receber os impulsos nervosos de outros neurônios e conduzi-los ate o corpo celular. O Axônio é um dispositivo de saída que transmite o sinal do corpo celular para as extremidades que estão conectadas com dendritos de outros neurônios através das s*inapses*, que é o ponto de contato entre a terminação do axônio de um neurônio e o dendrito do outro. Pelas sinapses os nodos se unem funcionalmente permitindo a propagação dos impulsos nervosos de uma célula à outra, formando redes neurais. As sinapses controlam o fluxo da informação e possuem um efeito variável, o que dá ao neurônio a capacidade de adaptação, podem ser excitatórias onde permitem a passagem da informação entre os neurônios ou inibitórias que bloqueiam a atividade da célula.

A figura 2 faz referência à estrutura do cérebro descrita no parágrafo acima.

[1] Adaptado de Prof. Dr. Mauro Roisenberg – Redes Neurais.

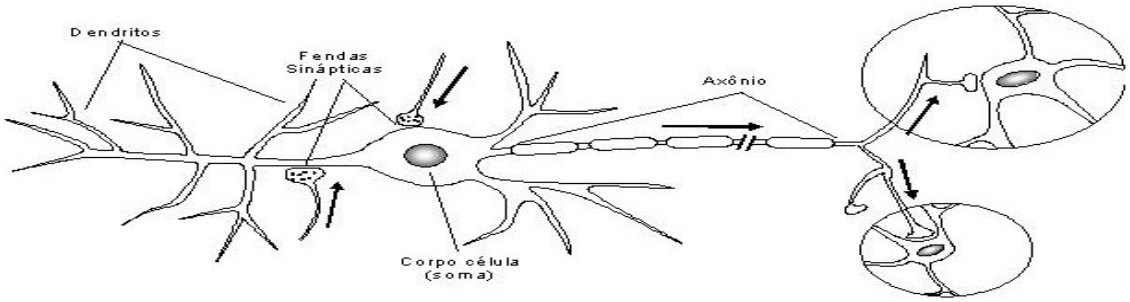

Figura 2 - Representação de um Neurônio Biológico.[2]

2.3. Neurônio Artificial

Redes neurais artificiais se inspiram nas redes biológicas e procuram imitar a arquitetura do cérebro humano para atingir o objetivo de surgir um comportamento inteligente da estrutura criada. Formam um dos paradigmas da inteligência artificial, que é conhecido como conexionista.

O cérebro como um sistema computacional possui neurônios onde sua atividade é um processo de tudo ou nada, um número fixo de entradas devem ser estimulados durante um período para estimular um neurônio, a atividade de qualquer sinapse inibitória evita que o neurônio seja excitado e a estrutura das interconexões não mudam com o tempo.

BRAGA, LUDERMIR & CARVALHO (2000) explicam que o modelo de neurônio proposto por McCulloch e Pitts (conhecido como modelo MCP) era uma simplificação do que se sabia a respeito do neurônio biológico. Seu modelo matemático usa *n* terminais de entrada *x1,x2,...,xn* para representar os dendritos e um terminal de saída *y* para representar o axônio. As sinapses eram representadas por pesos acoplados aos terminais de entrada podendo ser positivos ou negativos e são representados por *wn*.

A ativação do neurônio MCP é feita por meio de uma “função de ativação”, que ativa ou não a saída dependendo do valor da soma ponderada das entradas. A função de ativação do

[1] Figura Disponível em: http://lsin.unisantos.br/lvcon/web/imagens/temas/2/id405.jpg.

nodo MPC é dada por:

$$\sum_{i=1}^{n} piwi \geq \theta$$

Onde *pi* são as entradas para o neurônio, wi os pesos da conexão e θ é o valor de limiar que deve ser atingido para o neurônio ser ativado. A figura 3 exemplifica o modelo de neurônio artificial MCP.

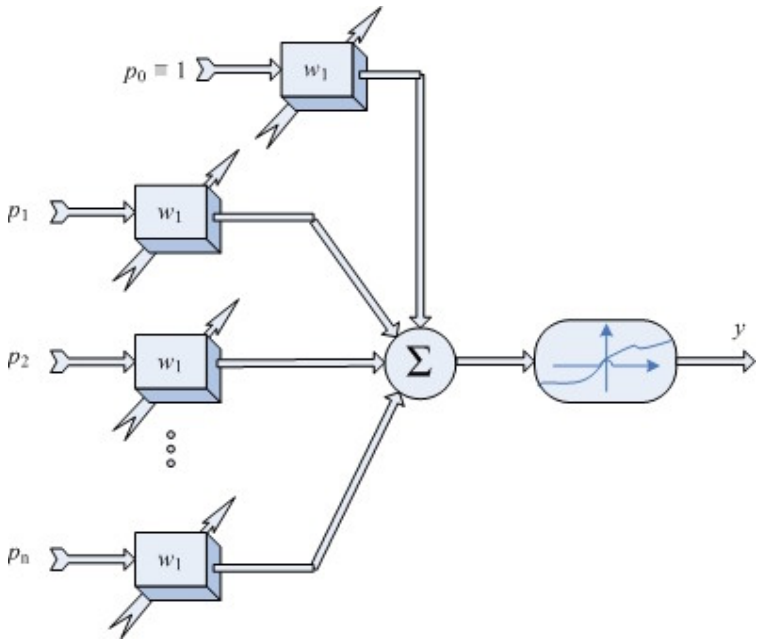

Figura 3 - Neurônio Artificial MCP.[3]

Segundo AZEVEDO, BRASIL & OLIVEIRA (2000) uma RNA é um sistema composto por vários neurônios que estão ligados por conexões conhecidas como sinápticas. Alguns desses neurônios recebem excitações do exterior, são conhecidos como neurônios de entrada e correspondem aos neurônios responsáveis pelos nossos sentidos. Outros neurônios usam suas respostas para alterar o mundo exterior e são conhecidos como neurônios de saída, correspondendo aos neurônios biológicos que excitam nossos músculos. Além dos neurônios de entrada e saída, existem os neurônios intermediários que são de suma importância no aprendizado e são conhecidos como *hidden layer* ou camada oculta.

[3] Figura disponível em: http://www.ene.unb.br/~adolfo/ISI/Neur%F4nio%20Artificial.jpg.

> As redes neurais são modelos “inspirados biologicamente” e estão muito longe de representar os neurônios biológicos naturais. Na verdade, esta inspiração freqüentemente é muito limitada pois as redes neurais naturais não são circuitos digitais, não podem ter excitação negativa, não são homogêneas, não tem relógio de sincronização, os neurônios e sinapses não podem ter dois valores e os circuitos celebrais não são capazes de executar cálculos recursivos como os modelos artificiais.(Jain, Mao, Mochiuddin).

As entradas de um neurônio podem ser as saídas de outros neurônios, entradas externas, ou qualquer combinação desses elementos. A combinação das entradas é conhecida como “net”, que é a soma das entradas multiplicadas por seus pesos sinápticos. A conexão sinaptica é excitatória quando o peso conectado ao neurônio é maior que zero ($wij>0$) ou inibitória quando o peso é menor que zero ($wij<0$). Depois o valor da ativação do neurônio é atualizado através da função de ativação e o valor da saída do neurônio é produzido através da função de saída. Quando o neurônio é dinâmico ou possui memória, os estados futuros do neurônio são afetados pelo estado atual do neurônio e pelo valor do “net” de entrada. Neurônios estáticos são aqueles em que a função é constante, onde o estado atual é igual aos estados anteriores.

O corpo faz uma soma ponderada do produto dos pesos da entrada e uma função de transferência é aplicada sobre a função de ativação para gerar a saída. A função de ativação tem de decidir o que fazer com o valor resultante do somatório das entradas ponderadas, em modelos mais complexos, a função de ativação pode usar um valor de saída estipulado como entrada para o próprio neurônio (realimentação). A função de transferência compara o valor somado com um valor estipulado conhecido como valor limiar, que quando atingido é passado adiante por via da saída.

2.4. Aprendizado

Redes neurais artificiais caracterizam-se pelo aprendizado por meio de exemplos. No processo de aprendizado a rede é estimulada pela apresentação de um conjunto de dados do meio externo. O algoritmo de aprendizado muda os parâmetros da rede neural o que influência no seu comportamento melhorando o desempenho.

> Aprendizagem é o processo pelo qual os parâmetros de uma rede neural são ajustados através de uma forma continuada de estimulo pelo ambiente no qual a rede esta operando, sendo o tipo especifico de aprendizagem realizada definida pela maneira particular como ocorrem os ajustes realizados nos parâmetros.(Braga, Ludermir & Carvalho - Redes Neurais Artificiais Teoria e aplicações, 2000, pg.15).

A base para todas as regras de aprendizagem derivam do aprendizado de Hebb criado em 1949. Segundo a regra de Hebb quando um axônio de um neurônio A esta próximo o suficiente para excitar um neurônio B, e persistentemente toma parte do disparo de B, então ocorre um processo de mudança metabólica em uma das duas células, de forma que a eficiência de A em contribuir para o disparo de B é aumentado. Ou seja, se duas unidades são ativadas simultaneamente, suas interconexões tendem a se fortalecer.

De acordo com AZEVEDO, BRASIL & OLIVEIRA (2000) o primeiro passo é estabelecer um conjunto de pesos para suas conexões, ativar um conjunto de unidades que representem um padrão de entrada e observar o padrão em que a rede converge e se estabiliza. Se o padrão final não corresponder a saída desejada para aquele padrão é necessário ajustar os pesos e ativar novamente o padrão de entrada, até que o erro mínimo estipulado seja atingido ou a rede neural atinja um número máximo de ciclos.

KOVÁCS (1996) diz que a mudança do processamento ou da estrutura de conhecimento de uma rede neural envolve a alteração do seu padrão de interconexão, o que pode ser feito através do desenvolvimento de novas conexões, perda de conexões na rede ou

modificando os pesos das conexões existentes.

BRAGA, LUDERMIR & Carvalho (2000) ensinam que o principal objetivo da aprendizagem em redes neurais é a obtenção de modelos capazes de fazer generalização através do conjunto de dados. Em redes projetadas para aproximação, predição e classificação o conjunto de treinamento é composto por pares de entrada e saída, tendo uma saída esperada para um valor de entrada. Os pesos são alterados para fim de diminuir o erro a cada iteração, no entanto apenas a minimização do erro pode não gerar resultados satisfatórios. Portanto o objetivo de algoritmos de treinamento é aproximar funções geradoras dos dados e não apenas minimizar o erro do conjunto de treinamento.

Existem dois paradigmas principais para o treinamento de redes neurais, que são: aprendizado supervisionado e aprendizado não-supervisionado.

O aprendizado supervisionado, que está referenciado na figura 4, é o treinamento mais comum das RNAs. No aprendizado supervisionado a entrada e a saída desejada são fornecidas por um professor externo, com o objetivo de ajustar os parâmetros da rede para conseguir uma ligação entre a entrada e a saída desejada conhecida. Utiliza-se a adaptação por correção de erros para minimizar a diferença entre a soma ponderada das entradas pelos pesos, comparada à saída desejada. Os Algoritmos de aprendizado supervisionado mais conhecidos são a *regra delta* e o algoritmo *backpropagation* que é usado em redes de múltiplas camadas.

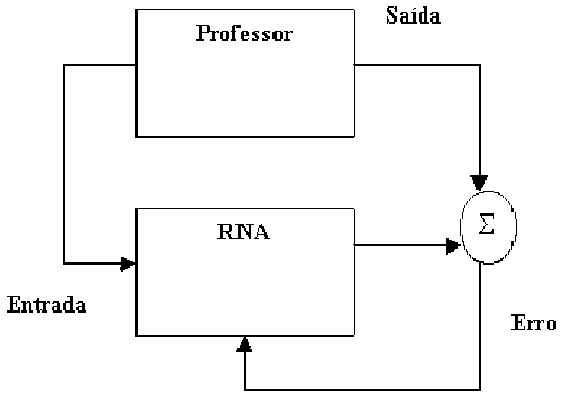

Figura 4 - Aprendizado supervisionado.[4]

Diferente do aprendizado supervisionado, o aprendizado não-supervisionado, que está referenciado na figura 5, não necessita de um supervisor para acompanhar o processo de aprendizado. No aprendizado não-supervisionado é apresentada à rede somente os valores de entrada e não pares de entrada e saída além de nenhuma medida de erro a ser utilizada para realimentar a rede. Esse tipo de aprendizado só é possível se existir redundância nos dados de entrada, senão seria impossível encontrar quaisquer padrões dos dados de entrada.

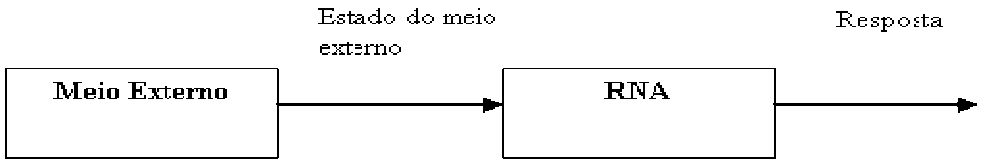

Figura 5– Aprendizado não-supervisionado. [5]

Resumindo, redes neurais aprendem a reconhecer seu ambiente e com isso melhoram seu desempenho. O treinamento é feito por meio de ajustes nos pesos e o aprendizado é concretizado quando a rede atinge uma solução generalizada para a classe de problema apresentado. O conhecimento que uma rede neural adquire no aprendizado não é armazenada em um endereço de memória especifico, mas sim distribuído por toda a rede. Para o

[4] **Fonte:** (Redes Neurais Teoria e Aplicações, 2000, pg.17).

[5] **Fonte:** (Redes Neurais Teoria e aplicação, 2000, pg.19).

treinamento são usados algoritmos de aprendizagem, que fazem os ajustes dos pesos das suas conexões de acordo com os padrões apresentados.

Outra forma de aprendizado é o aprendizado competitivo, que é utilizado em redes neurais do tipo Counterpropagation e mapas auto-organizáveis de Kohonen. No aprendizado competitivo, os neurônios são inibidos por outros neurônios de modo que a competição entre eles resulte em apenas um neurônio acabar excitado. Ao contrario das redes neurais convencionais onde vários neurônios de saída podem estar ativados ao mesmo tempo, no aprendizado competitivo apenas um neurônio de saída esta ativo a cada vez. Entradas que possuem as mesmas semelhanças tendem a excitar o mesmo neurônio de saída.

2.5. Função de Ativação

O neurônio artificial é um elemento de processamento cuja saída é calculada multiplicando o vetor de entrada pelo vetor de pesos, o resultado é um escalar, que é a combinação linear das entradas, a qual se aplica uma função de ativação.

A função de ativação é diferenciada para cada tipo de arquitetura de rede neural. Redes de camada única como o perceptron, utilizam função “degrau”, enquanto que arquiteturas de múltiplas camadas geralmente utilizam funções do tipo sigmóide.

As funções do tipo degrau e rampa (mostradas nas figuras 6 e 7 respectivamente), comumente são utilizadas para a ativação de modelos perceptron, são funções lineares restringidas. A função rampa produz valores constantes numa faixa [-y, +y]. A função degrau produz valores de saída +y para valores de x maiores que zero, e valores de saída –y para valores de x menores que zero.

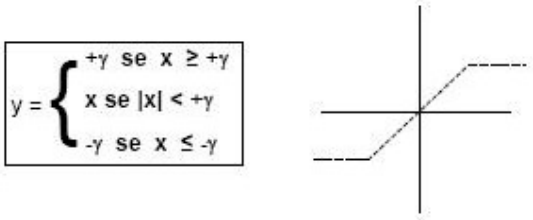

Figura 6 – Gráfico: Função Rampa.

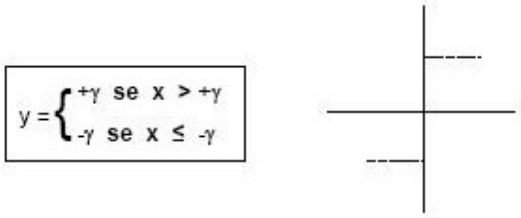

Figura 7 – Gráfico: Função Degrau.

Outra função comumente utilizada principalmente para neurônios responsáveis pela saída da rede neural, ou seja, que se encontram na camada de saída é a função linear, que está demonstrada na figura 8. Em uma função de ativação do tipo linear, sua saída é exatamente a mesma da sua entrada. Seu calculo é dado pela expressão (1):

$$f(x) = x \quad \textbf{(1)}$$

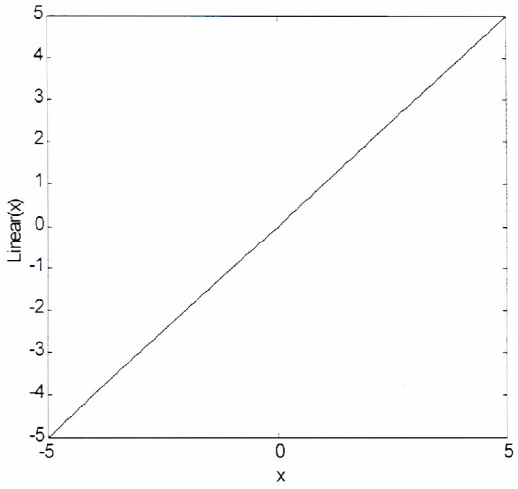

Figura 8 – Gráfico: Função Linear.

Existem vários tipos de funções de ativação não linear. As mais comuns são as funções logística (sigmoide) e tangente hiperbólica. A faixa de saída da função logística é no intervalo [0, 1], e para tangente hiperbólica é no intervalo [-1, 1]. O calculo da função logística sigmóide é dado pela expressão (2) e seu gráfico exemplificado na figura 9.

$$f(x) = \log istica(x) = \frac{1}{1 + \exp(-\alpha x)} \qquad \textbf{(2)}$$

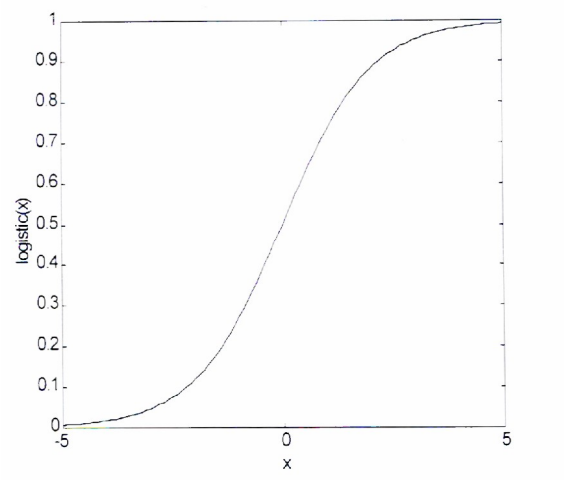

Figura 9 – Gráfico: Função Logística Sigmóide.

O calculo da função tangente hiperbólica é dado pela expressão (3) e seu gráfico é mostrado na figura 10.

$$f(x) = \tanh(x) = \frac{\exp(x) - \exp(-x)}{\exp(x) + \exp(-x)} \qquad \textbf{(3)}$$

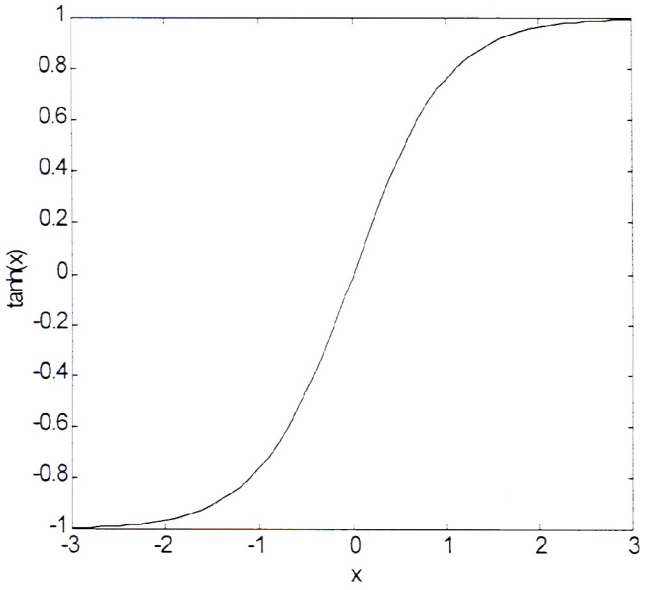

Figura 10 – Gráfico: Função Tangente Hiperbólica.

A função de base radial é outra função de ativação que se destaca principalmente em redes neurais do tipo *Radial Basis*. Uma função de base radial se caracteriza por apresentar uma resposta que decresce ou cresce com a distancia a um ponto central.

O calculo da função de base radial é dado pela expressão (4) e seu gráfico exemplificado na figura 11.

$$h_j(\mathbf{x}) = \exp\left(-\left(\mathbf{x}-\mathbf{c}_j\right)^T \Sigma_j^{-1}\left(\mathbf{x}-\mathbf{c}_j\right)\right) \quad \textbf{(4)}$$

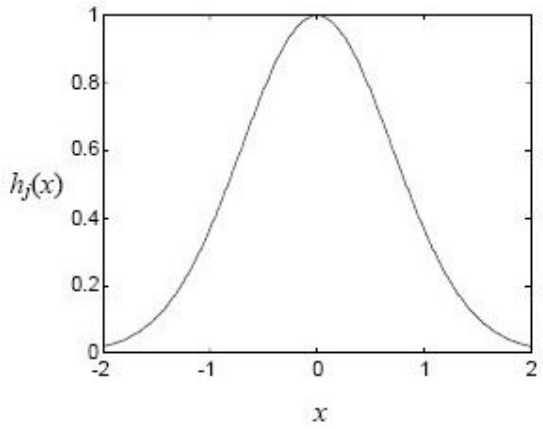

Figura 11 - Gráfico: Função Base Radial.

Capítulo 3 - REDES NEURAIS ARTIFICIAIS – ARQUITETURAS

De acordo com JAIN, MAO & MOCHIUDDIN (2000), RNAs podem ser vistas como grafos direcionados ponderados onde os neurônios artificiais são os nós e as conexões entre as entradas e saída dos neurônios são as arestas com pesos. Diferentes conectividades levam a redes neurais com comportamentos diferentes e arquiteturas diferentes necessitam de algoritmos de treinamento distintos.

A arquitetura de redes neurais pode ser agrupada em duas categorias que são: redes diretas e redes recorrentes. Redes diretas ou *feed-forward networks*, exemplificadas na figura 12, podem ser consideradas como redes estáticas, pois geram apenas valores de saída. Sua resposta a um padrão de entrada é independente do estado anterior da rede, são redes cujos grafos de interconexões não possuem ciclos e comumente são usadas em camadas. Possuem uma primeira camada, conhecida como camada de entrada, onde os neurônios recebem sinais de excitação e uma ultima camada conhecida como camada de saída onde os neurônios tem sua saída como saída da rede.

Redes recorrentes ou *feedback networks*, exemplificadas na figura 13, são sistemas dinâmicos, pois para cada padrão de entrada apresentado os neurônios de saída são computados. As entradas de cada neurônio são modificadas, fazendo com que a rede entre em um novo estado. São redes em que o grafo de conectividade possui pelo menos um ciclo.

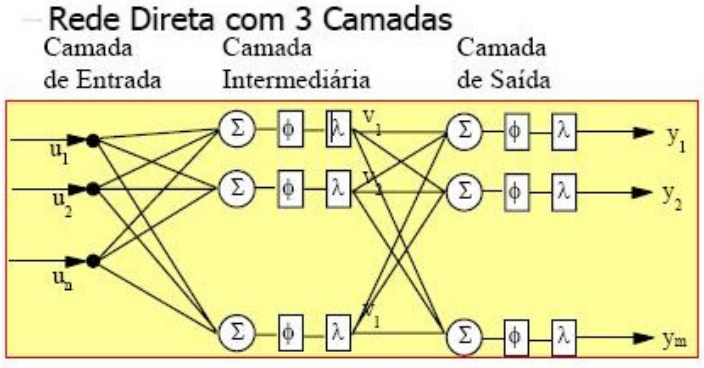

Figura 12– Rede Direta ou *FeedForward*.

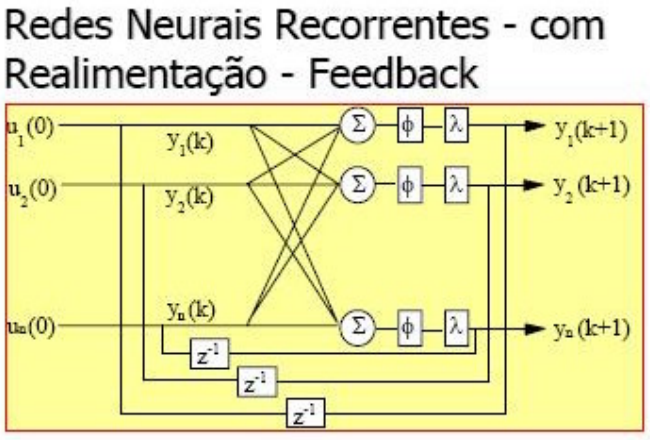

Figura 13 - Rede Recorrente ou *FeedBack*.

A arquitetura de uma rede neural influencia diretamente o seu comportamento perante o padrão de características apresentado a ela. Cada tipo de arquitetura é otimizada para determinada classe de problemas, que podem variar desde o reconhecimento de padrões, aproximação de funções, reconhecimento e sons, entre outros.

3.1. Portas de Limiar

De acordo com BRAGA, LUDERMIR & CARVALHO (2000) portas do tipo limiar ou *threshold gates* comparam a soma ponderada das entradas com um valor limiar ou *threshold*. Se a soma exceder o valor de limiar, a saída é ativada, caso contrário permanece desativado. Podem ser dividas em linear, quadrática e polinomial, diferindo-se pela complexidade com que seus pesos são calculados. Quanto mais complexos os termos associados aos pesos, maior flexibilidade a porta ganha na solução do problema de mapeamento.

Portas de limiar lineares se restringem a resolver apenas problemas que sejam linearmente separáveis, ou seja, problemas que podem ser resolvidos pela separação de duas

regiões por meio de uma reta. Apesar da limitação, portas de limiar linear são melhores do que portas lógicas convencionais, pois com uma mesma porta de limiar linear é possível implementar qualquer função *e, ou, não-e, não-ou* apenas mudando os parâmetros da porta. Porém para implementação de funções não linearmente separáveis como a função booleana *ou-exclusivo* são necessárias pelo menos duas camadas de portas limiares lineares.

Portas de limiar quadráticas tem um maior poder computacional, pois possuem um numero maior de parâmetros livres ajustáveis. Portas de limiar quadráticas possuem termos quadráticos em cada uma das variáveis de entrada, além de termos com produtos cruzados entre cada uma delas onde os pesos de entrada definem a importância de cada um desses termos na definição da superfície de decisão.

> Adicionando mais parâmetros livres gera mais flexibilidade as soluções através de uma porta de limiar. Portas que possuem termos mais complexos para os pesos do que as portas quadráticas são conhecidas com portas de limiar polinomiais que são mapeadores universais de funções booleanas de *n* variáveis.(Braga, Ludermir & Carvalho – Redes Neurais Artificiais Teoria e Aplicações, 2000, pg.34).

Uma rede com um número reduzido de parâmetros pode não possuir a flexibilidade necessária para solução de um problema, e uma rede com um numero muito grande de parâmetros pode ser flexível demais, gerando estimativas indesejáveis para pontos fora do conjunto de treinamento. Isto é conhecido como o dilema entre a polarização e variância.

3.2. Arquitetura dos modelos Perceptron, Adaline e Madaline

BRAGA, LUDERMIR & CARVALHO (2000) nos ensinam que o modelo Perceptron foi introduzido por Frank Rosenblatt em 1958. É composto por nodos MCP[6] e por uma regra de aprendizagem. Rosenblatt ainda mostra que um nodo MCP treinado com o

[6] Modelo de neurônio criado por McCulloch e Pitts.

algoritmo de aprendizado do Perceptron sempre converge se o problema for linearmente separável.

O Perceptron original, ilustrado na figura 14, conhecido como Perceptron de uma única camada é composto por unidades de entrada que são unidades sensoras, um nível intermediário formado por unidades de associação com pesos fixos que são definidos antes do treinamento, e um nível de saída, formados por unidades de resposta tendo somente o nível de saída com propriedades adaptativas. Os pesos das conexões de entrada são multiplicados pelos valores das entradas e então somados e submetidos a uma função de transferência para gerar a saída do neurônio.

JAIN, MAO & MOCHIUDDIN (2000) mostram que o aprendizado do perceptron é dado pelo seguinte algoritmo:

1. Inicializar os pesos e o limiar a pequenos números aleatórios.

2. Apresentar um vetor dos padrões $(x_1, x_2, \ldots, x_n)^t$ e avaliar a saída do neurônio.

3. Atualizar os pesos de acordo com a expressão (5) :

$$w_i(t + 1) = w_i(t) + \eta (d - y) x_i \quad \textbf{(5)}$$

Onde *d* é a saída desejada, *xi* são as entradas, *wi* são os pesos, y é a saída obtida, *t* é o número da iteração e η ($0.0 < \eta < 1.0$) é a taxa de aprendizado.

O calculo da função de ativação do perceptron é dado pela expressão (6):

$$\sum_{i=1}^{n} x_i w_i \geq \theta \quad \textbf{(6)}$$

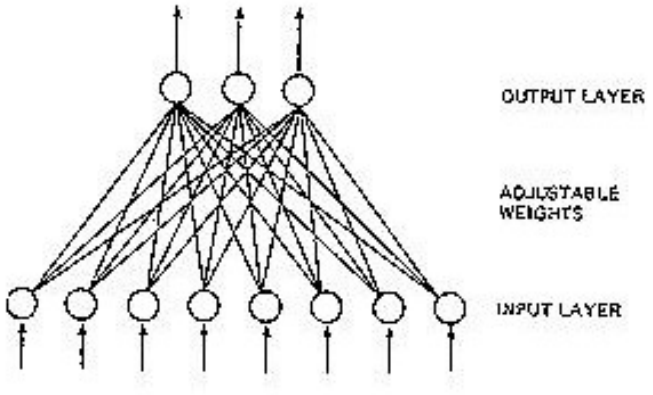

Figura 14 - Arquitetura do modelo Perceptron.

Segundo BRAGA, LUDERMIR & CARVALHO (2000) o modelo ADALINE foi criado por Widrow e Hoff, seu algoritmo de treinamento conhecido como regra delta foi de suma importância para a criação do algoritmo *back-progagation* que é um dos mais utilizados atualmente para treinar redes perceptron de múltiplas camadas. O algoritmo consiste em minimizar o erro médio quadrático entre a saída desejada e a saída apresentada por uma rede. Como o Perceptron, a base do modelo ADALINE é processar elementos através da soma ponderada e comparar a saída a um valor de limiar, porém também está restrito a solução de problemas que sejam apenas linearmente separáveis.

De acordo com AZEVEDO, BRASIL & OLIVEIRA (2000) a principal diferença entre o modelo ADALINE e o Perceptron é a maneira como a saída do sistema é usada na regra de aprendizagem. A regra de aprendizado do Perceptron utiliza a função de saída para o aprendizado enquanto que a regra de aprendizado do ADALINE utiliza a função de entrada no aprendizado antes da saída ser mapeada pela função de saída.

Osório (2001) mostra que a regra delta pode ser definida pela equação (7):

$$\text{Peso\_Novo}(i) = \text{Peso\_Antigo}(i) + \frac{\beta * \text{Erro}(i) * \text{Entrada}(i)}{|\ \text{Entrada}(i)\ |} \quad \textbf{(7)}$$

Onde *Erro(i)* é o erro estimado na saída do neurônio, *Peso_Novo(i)* é o peso da entrada i de um neurônio após a adaptação, *Peso_Antigo(i)* é o peso da entrada i de um neurônio antes da adaptação, *Entrada(i)* é o valor da entrada do neurônio e β é o fator de ajuste aplicado aos pesos (valor entre 0 e 1).

O modelo MADALINE ou *Many Adaline* é uma extensão do modelo ADALINE criada por Widrow. É uma possível solução para o problema de aprendizado para funções não lineares. A arquitetura Madaline consiste de um sistema que apresenta uma camada de unidades ADALINE que são conectadas a uma única ADALINE de saída. Não existem pesos de ajustes nesse modelo, a saída é estipulada de acordo com a maioria das respostas obtidas pelas Adaline internas (Se a maioria dos ADALINE obtiver um resultado +1, a saída será +1).

Quando Redes Neurais Artificiais de uma só camada são utilizadas os padrões de treinamento apresentados à entrada são mapeados diretamente em um conjunto de padrões de saída da rede, ou seja, não é possível a formação de uma representação interna. Neste caso, a codificação proveniente do mundo exterior deve ser suficiente para implementar esse mapeamento.

Tal restrição implica que padrões de entrada similares resultem em padrões de saída similares, o que leva o sistema à incapacidade de aprender importantes mapeamentos. Como resultado, padrões de entrada com estruturas similares, fornecidos do mundo externo, que levem a saídas diferentes não são possíveis de serem mapeados por redes sem representações internas, isto é, sem camadas intermediárias. Um exemplo clássico deste caso é a função ou-exclusivo (XOR).

Minsky e Papert analisaram matematicamente o Perceptron e demonstraram que redes de uma camada não são capazes de solucionar problemas que não sejam linearmente separáveis. Como não acreditavam na possibilidade de se construir um método de treinamento

para redes com mais de uma camada, eles concluíram que as redes neurais seriam sempre suscetíveis a essa limitação.

3.3. Modelo Hopfield (Memórias Associativas)

Em 1982 Hopfield publicou um artigo que influenciou vários pesquisadores, chamando a atenção para as propriedades associativas de uma classe de Redes Neurais. A análise é baseada na definição de “energia” da rede é uma prova de que a rede opera minimizando esta energia quando evolui para padrões estáveis de operação.

Hopfield demonstrou o uso de sua rede conhecida como Rede de Hopfield, provando o poder computacional que uma rede neural pode ter. Hopfield provou que é possível resolver problemas de otimização, como o problema de memória associativa através de sua rede neural.

De acordo com AZEVEDO, BRASIL & OLIVEIRA (2000) a rede binária de Hopfield, mostrado na figura 15, tem uma única camada de neurônios, onde cada uma delas tem um estado (vetor de 1s e 0s) binário de dois valores possíveis. É uma topologia que é totalmente interconectada, com isso a rede torna-se recursiva, pois as saídas de cada unidade tornam-se as entradas de outras. Essa inter-conectividade, permite a rede convergir (atingir um estado estável) na ausência de entradas externas.

Possui uma natureza de operação assíncrona, a cada instante de tempo, cada neurônio tem seu estado de ativação avaliado independente dos outros neurônios.

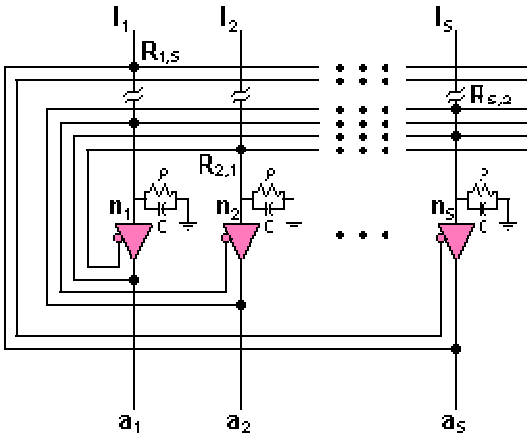

Figura 15 - Modelo de Arquitetura Hopfield.[7]

3.4. Modelo SOM (*Self-Organizing Maps*)

Mapas auto-organizáveis são redes neurais baseadas na metáfora biológica do córtex cerebral, que são regiões especializadas, um exemplo é o mapa somatotópico que controla partes corpo humano. A estrutura básica do corpo é refletida na organização do córtex nesta região.

Mapas de Kohonen ou SOM (*Self-Organizing Maps*) são redes neurais treinadas por um padrão de treinamento não-supervisionado competitivo, isto é não é apresentada uma saída desejada para a rede. É composta por duas camadas, sendo uma de entrada que lê os vetores apresentados à rede e uma camada de saída que é composta de uma matriz MxM de neurônios que geram uma resposta ao padrão apresentado. São redes utilizadas principalmente para reconhecimento de padrões quando as classes a que devem pertencer os elementos a serem tratados não são conhecidas.

Segundo AZEVEDO, BRASIL & OLIVEIRA (2000) a regra de aprendizado competitivo utilizado pelos mapas de Kohonen fazem com que as unidades disputem o direito de resposta a um conjunto de entradas apresentadas a rede, em que o neurônio com maior

[7] Figura disponível em: http://ohm.utp.edu.co/neuronales/Capitulo2/Recurrentes/ImagesR/Fig261.gif.

atividade é o único a participar do processo de aprendizado. A arquitetura de aprendizado competitivo consiste em um conjunto de camadas interligadas através de conexões excitatórias e inibitórias entre unidades da mesma camada. Cada unidade em uma camada recebe a entrada de cada unidade da camada imediatamente anterior, e envia um sinal para a camada superior a ela.

3.5. Modelo ART (*Adaptative Resonance Theory*)

Redes neurais baseadas na metáfora biológica referente a memória humana, onde pode-se adicionar uma nova informação sem esquecer as informações previamente armazenadas. Geralmente uma rede neural codifica a informação através do ajuste dos pesos de suas conexões, quando um novo padrão é apresentado a rede é necessário calcular novamente o valor dos pesos, ou seja, a rede perde toda a informação quando um novo padrão é apresentado a ela.

De acordo com AZEVEDO, BRASIL & OLIVEIRA (2000) o modelo ART foi projetado para solucionar os problemas de instabilidade de redes do tipo *forward*. Para tal a rede ART incorpora um modelo de aprendizado competitivo dentro de uma estrutura de controle auto-organizada, onde o reconhecimento e aprendizado são autônomos e continuam estáveis em resposta a uma seqüência de padrões de entrada.

O modelo ART (*Adaptative Resonance Theory*), ilustrado na figura 16, foi criado por Grossberg em 1978, e tem a característica de aprender incrementalmente, isto é, a rede pode aprender novas informações sem que as informações aprendidas anteriormente sejam perdidas. Geralmente esse tipo de rede é utilizado para o reconhecimento de padrões em que seres humanos não são muito eficientes como: sinais de radar e impressões de registros sonoros, entre outros. Derivando da rede ART, foram criadas as redes ART1 (entradas

binárias), ART2 (entradas analógicas) e ART3 (hierarquia multilateral), onde suas peculiaridades estão nos seus padrões de entrada.

Redes ART adquirem seu aprendizado, controle da estabilidade e memória dos padrões aprendidos através da ressonância. Ressonância se da pelo conceito de que uma vibração de pequena amplitude em uma freqüência apropriada causa uma vibração de grande amplitude.

Em uma rede ART, a informação, na forma de saída dos neurônios reverbera para frente e para trás entre as camadas. Se um padrão apropriado se desenvolve no interior do sistema, aparece uma oscilação estável, o que é o equivalente ao conceito de ressonância para a rede neural. Nesse período de ressonância que a rede neural pode aprender.

Um estado ressonante é conseguido de duas maneiras:

- Se a rede já havia aprendido anteriormente a reconhecer um vetor de entrada, então o estado ressonante será rapidamente atingido quando o vetor for reapresentado. Durante a ressonância, o processo de adaptação irá reforçar a memorização do padrão armazenado.

- Se o vetor de entrada não for imediatamente reconhecido, a rede irá rapidamente procurar entre os padrões já armazenados por um “match”. Se não ocorrer um “match” a rede irá entrar em um estado ressonante durante o qual o novo padrão será armazenado pela primeira vez.

Assim a rede consegue responder rapidamente a padrões já memorizados, e continua capaz de aprender um novo padrão.

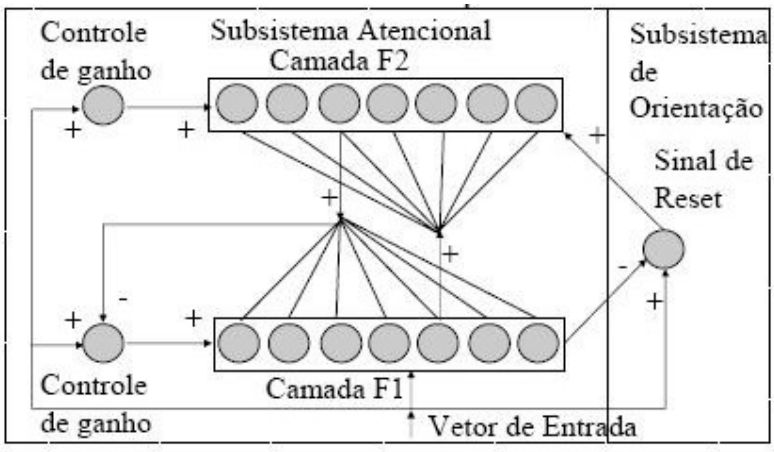

Figura 16 - Modelo ART.

3.6. Modelo RBF (*Radial Basis Function*)

A rede de base radial (RBF) foca o projeto de uma rede neural como um problema de ajuste de curva (aproximação). O aprendizado para este tipo de rede é o equivalente a encontrar uma superfície, em um espaço multidimensional, que forneça o melhor ajuste para os dados de treinamento. As unidades ocultas fornecem um conjunto de funções que constituem uma base arbitraria para o padrão de entrada. Quando esses padrões são expandidos sobre o espaço oculto essas funções são chamadas de base radial.

As principais diferenças desse modelo para o modelo Perceptron de múltiplas camadas é que redes RBF possuem uma camada oculta única tendo sua função de ativação do tipo base radial e os neurônios de saída são sempre lineares. As redes *RBF* são redes de aprendizado local, de modo que é possível chegar a uma boa aproximação desde que um número suficiente de dados para treinamento seja fornecido na região de interesse. Em contraste, perceptrons multicamadas são redes de aprendizado “global” (em virtude da natureza das funções de ativação) que fazem aproximações de efeito global, em regiões compactas do espaço de aproximação.

Redes RBF são constituídas de três camadas: entrada, oculta e saída. A camada de entrada é constituída por unidades sensoriais e conectam a rede ao seu ambiente. A camada oculta, que é única para a rede, aplica uma transformação não linear do espaço de entrada para o espaço oculto. A camada de saída é linear e fornece a resposta da rede para o padrão apresentado a camada de entrada.

Dado um número suficiente de neurônios com função de base radial, qualquer função contínua definida numa região compacta pode ser devidamente aproximada usando uma rede *RBF* (PARK & SANDBERG, 1991).

3.7. Modelo Perceptron Multi-Camadas - MLP (*Multy-Layer Perceptron)*

Devido ao problema das arquiteturas de redes com única camada apontado por Minsky e Papert, em que era apenas possível resolver funções boolenas linearmente separáveis foi necessário desenvolver uma nova arquitetura para suprir essa necessidade. A solução foi apontada por Hopfield em 1982 com o desenvolvimento da arquitetura Perceptron Multi-Camadas. Teoricamente uma rede neural com duas camadas intermediarias é capaz de implementar qualquer função, seja linearmente separável ou não.

Uma única camada intermediaria é suficiente para aproximar qualquer função continua, e duas camadas intermediarias é suficiente aproximar qualquer função matemática. Porém o uso de um grande número de camadas intermediarias não é interessante, pois cada vez que o erro medido é propagado para a camada anterior se torna menos preciso, sendo a camada de saída a única camada da rede neural que tem uma noção precisa do erro.

BRAGA, LUDERMIR & CARVALHO (2000) explicam que o número de nodos nas camadas intermediárias depende de vários fatores como o número de exemplos de treinamento, a quantidade de ruído presente nos exemplos, a complexidade da função a ser

aprendida e a distribuição estatística dos dados de treinamento. Logo não se pode generalizar a quantidade de nodos intermediários que uma rede neural MLP deve possuir.

Há um problema no treinamento de redes neurais, conhecido como *bias and variance dilemma* ou o dilema da rigidez-flexibilidade, em que uma rede não pode conter um número muito pequeno de unidade intermediarias, pois a rede pode gastar um tempo excessivo para encontrar uma representação ótima (*underfitting),* nem camadas intermediarias demais pois pode fazer com que a rede memorize padrões de treinamento não vistos durante o treinamento (*overfitting*). O controle desse problema é obtido através do dimensionamento da rede neural.

Segundo BRAGA, LUDERMIR & CARVALHO (2000) uma das formas para evitar o problema de *overfitting* é estimar o erro de generalização durante o treinamento. Para isso o conjunto de dados é dividido em conjunto de treinamento para modificação dos pesos e conjunto de validação para estimar a capacidade de generalização. O treinamento deve ser interrompido quando o erro do conjunto de validação começar a subir fazendo com que a rede degrade seu processo de generalização. Essa é uma solução valida apenas para redes neurais onde o conjunto de treinamento não seja muito grande, pois os dados do conjunto de validação não podem ser utilizados para treinamento.

Outra forma de controlar o problema de *overfitting* é por meio de uma técnica chamada *pruning*, onde os pesos e nodos irrelevantes para o funcionamento da rede são retirados.

A figura 17 demonstra uma rede MLP.

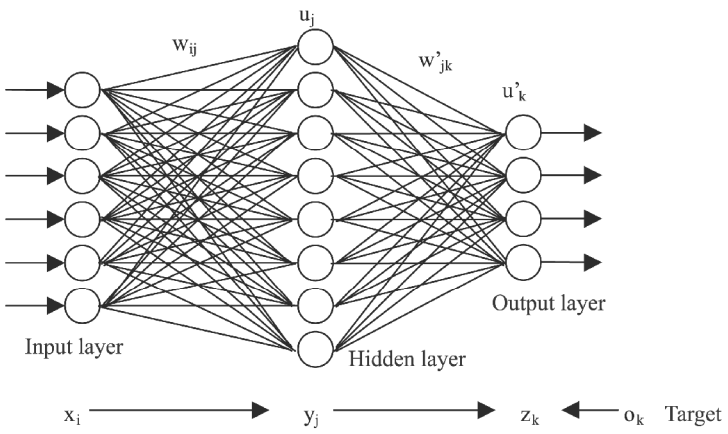

Figura 17 – Arquitetura do modelo Perceptron Multi-Camadas (MLP). [8]

3.8. Treinamento de redes MLP – Algoritmo *back-propagation*

Embora existam diversos algoritmos de treinamento de redes disponíveis, o algoritmo backpropagation tem apresentado excelentes resultados na análise e na resolução de problemas de confiabilidade estrutural (Saraiva, 1997).

O algoritmo backpropagation é o principal algoritmo de treinamento de rede utilizado e é, freqüentemente, usado como o algoritmo de aprendizado em redes neurais estruturadas em camadas, devido à sua eficiência (Rumelhart et al., apud Hirose et al., 1991).

De acordo com BRAGA, LUDERMIR & CARVALHO (2000) existem vários algoritmos para treinar uma rede MLP, sendo estes divididos em dois grupos: estáticos e dinâmicos. Algoritmos estáticos não alteram a estrutura da rede neural, apenas variam valor dos pesos, enquanto algoritmos dinâmicos podem reduzir ou aumentar o número de camadas, conexões e o número de nodos nas camadas intermediarias.

O principal algoritmo utilizado para treinar redes de múltiplas camadas é conhecido como *back-propagation* ou regra delta generalizada. O algoritmo *back-propagation* é um

[8] Figura disponível em: http://www.emeraldinsight.com/fig/0680140209001.png.

algoritmo supervisionado que aprende por meio de exemplos e utiliza um método de gradiente descendente para correção de erro. O algoritmo consiste de duas fases, conhecidas como *feed-forward* e *feed-backward*. Na fase *forward* um padrão é apresentado às unidades da camada de entrada, as camadas intermediarias calculam a resposta que é obtida na camada de saída. O valor de saída produzido pelos nodos da ultima camada são comparados com o valor da saída desejada e o erro é calculado. Na fase *backward* o erro calculado é propagado a partir da camada de saída até a camada de entrada, e os pesos das unidades das camadas intermediárias vão sendo atualizados de acordo com a regra delta generalizada. Depois de treinada e com o erro em um valor próximo do valor desejado, a rede pode ser usada para aprender novos padrões. Para isso a rede é utilizada apenas no modo *feed-forward* sem necessidade de retro-propagação.

O algoritmo *back-propagation* procura minimizar o erro ajustando os pesos e limiares utilizando um método de gradiente descendente, ou seja, o algoritmo executa um mapeamento de entrada e saída por meio da minimização de uma função de custo, esta que por sua vez é minimizada através de ajustes iterativos nos pesos de acordo com o erro quadrático acumulado. Porem não é muito raro o algoritmo convergir para mínimos locais que são pontos na superfície do erro que apresentam uma solução estável, embora não seja a saída correta.

> O gradiente de uma função esta na direção e sentido em que a função tem taxa de variação máxima. Isto garante que a rede caminha na superfície na direção que vai reduzir mais o erro obtido. Para superfícies simples, este método certamente encontra a solução com erro mínimo. Para superfícies mais complexas, esta garantia não mais existe, podendo levar o algoritmo a convergir para mínimos locais. O algoritmo *back-propagation* fornece uma aproximação da trajetória no espaço de pesos calculado pelo método do gradiente descendente. Estes pontos ou áreas podem incluir platôs, mínimos locais ou arestas. (Braga, Ludermir & Carvalho - Redes Neurais Artificiais Teoria e aplicações, 2000, pg.67).

BRAGA, LUDERMIR & CARVALHO (2000) mostram o algoritmo de aprendizado *back-propagation* descrito nos seguintes passos:

1. Inicializar pesos e parâmetros.
2. Repetir até o erro ser mínimo ou até a realização de um dado número de ciclos:

   2.1. Para cada padrão de treinamento X

   2.2. Definir a saída da rede através da fase *forward*.

   2.3. Comparar saídas produzidas com as saídas desejadas.

   2.4. Atualizar pesos dos nodos através da fase *backward*.

Através do conjunto de treinamento, mostrado na expressão (8), composto por um vetor x com n entradas e um vetor de saída exato y_e:

$$Y = \{x_i, y_{ei}\}\ i = 1, ..., n \quad \textbf{(8)}$$

Pode-se definir o erro quadrático médio na expressão (9):

$$E_R = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - y_{ei}\right)^2 \qquad (9)$$

Onde y_i é a saída fornecida pela rede e y_{ei} é o valor exato correspondente à saída da rede.

A expressão (9) pode ser reescrita em função dos pesos, como apresentado na expressao (10):

$$E_R = \frac{1}{n}\sum_{i=1}^{n}\left(f(\mathbf{W}^T.x) - y_{ei}\right)^2 \qquad (10)$$

3.9. Métodos para acelerar o algorimo *back-propagation*

O algoritmo *back-propagation* pode apresentar uma serie de dificuldades como: o tempo de treinamento ser longo sendo preciso milhares de ciclos para chegar a níveis de erro aceitáveis. Redes neurais podem ser consideradas como caixas pretas, pois quase não se sabe como a rede atinge determinado resultado.

Segundo BRAGA, LUDERMIR & CARVALHO (2000) para contornar os problemas de mínimos locais e o tempo de treinamento do algoritmo *back-propagation* pode-se utilizar técnicas como: utilizar taxa de aprendizado decrescente; adicionar nós intermediários; utilizar um termo *momentum*; adicionar ruído aos dados. A adição de um termo *momentum* é uma das mais utilizadas por acelerar o processo de treinamento e evitar mínimos locais, alem de ser simples e efetiva.

O calculo para o ajuste dos pesos usando o termo *momentum* é dado por:

A minimização do erro quadrático pode ser obtida utilizando o processo conhecido por regra delta. De acordo com essa regra, sendo **w**(k) um ponto sobre a superfície de erro, o ajuste a ser aplicado a esse ponto é expresso em (11):

$$\Delta \mathbf{W}(\mathrm{k}) = \eta \frac{\partial \mathrm{E}_{\mathrm{R}}(\mathbf{W})}{\partial \mathbf{W}} \qquad (11) \quad \textbf{(7)}$$

Onde η é uma constante positiva denominada taxa de aprendizado. Efetuado o processo de ajuste, o valor atualizado do peso é descrito em (12):

$$W(K+1) = W(k) - DW(k) \quad \textbf{(12)}$$

É de particular interesse a determinação do parâmetro η, que é diretamente responsável pela rapidez do processo de aprendizado. O algoritmo backpropagation provê uma aproximação da trajetória de movimento sobre a superfície de erro, a qual, a cada ponto da superfície, segue a direção do ponto mais íngreme em busca do ponto de mínimo global. Quanto menor for a taxa de aprendizado, menores vão ser as correções a serem aplicadas aos pesos entre cada iteração, ocasionando um processo de convergência lento. Caso contrário, se o valor desse parâmetro for alto, pode-se obter uma aceleração no processo de convergência, mas pode-se tornar o algoritmo instável pela oscilação em torno de um ponto de mínimo local.

Uma forma simples de garantir a estabilidade e acelerar a convergência é a utilização da regra delta acrescida do fator de momento. Essa é representada na expressão (13):

$$\Delta \mathbf{W}(k) = \beta \Delta \mathbf{W}(k-1) + \eta \frac{\partial E_R(\mathbf{W})}{\partial \mathbf{W}} \qquad \textbf{(13)}$$

Onde β é denominado constante de momento e possui a variação $0 < \beta < 1$. O efeito dessa constante é aumentar a velocidade na direção do ponto de mínimo. O que se deseja com a inserção do termo de momento é a redução no tempo de treinamento, a melhora na estabilidade do processo e, com isso, aumentar a possibilidade de encontrar o mínimo global.

Outro método para acelerar o desempenho do algoritmo *back-propagation* é o algoritmo Marquardt de Levenberg. Esse algoritmo é considerado o método mais rápido para treinamento de redes *feedforward backpropagation*, que possui uma quantidade moderada de pesos sinápticos. Ele se baseia, para a aceleração do treinamento, na determinação das derivadas de segunda ordem do erro quadrático em relação aos pesos, diferindo do algoritmo backpropagation tradicional que considera as derivadas de primeira ordem.

O algoritmo de Levenberg - Marquardt se baseia no método de otimização de Newton, que faz uso da matriz Hessiana H . No método de Levenberg - Marquardt se faz uma aproximação para essa matriz, mostrada em (14), determinada em função da matriz Jacobiana, que contém as primeiras derivadas dos pesos em função dos pesos sinápticos, expressa em (15):

$$\mathbf{H} = \frac{\partial^2 E_R(\mathbf{W})}{\partial \mathbf{W}^2} \qquad (14)$$

$$\mathbf{J} = \frac{\partial e(\mathbf{W})}{\partial \mathbf{W}} \qquad (15)$$

Onde e(**W**) é definido conforme a expressão (16):

$$e(\mathbf{W}) = \sum_{i=1}^{n} (y_i - y_{ei}) \qquad (16)$$

A determinação da matriz Jacobiana é muito mais simples que a determinação da matriz Hessiana. Como, para uma rede neural, a performance de treinamento é expressa em função da soma dos erros quadráticos, a matriz Hessiana pode ser expressa pela expressão (17):

$$H = J^T(W) \cdot J(W) \qquad \mathbf{(17)}$$

O método de Newton atualiza os pesos segundo (18):

$$W(k+1) = W(k) - H^{-1} \cdot g_k \qquad \mathbf{(18)}$$

Onde g_k pode ser escrito conforme (19):

$$g_k = 2J^T(W).e(W) \qquad \mathbf{(19)}$$

O algoritmo de Levenberg - Marquardt procede a atualização dos pesos baseado na mesma expressão do método de Newton (18), realizando as modificações para a determinação da matriz Hessiana, mostrada em (20):

$$W(k+1) = W(k) - [J^T(W).J(W)+\mu_k I]^{-1}.J^T(W).e(W) \qquad \textbf{(20)}$$

Onde: I é a matriz identidade e μ_K é a constante do método de Levenberg - Marquardt.

O parâmetro μ_K funciona como um fator de estabilização do treinamento, ajustando a aproximação de forma a utilizar a rápida convergência do método de Newton e evitando passos muito grandes que possam levar a um erro de convergência.

Esse método apresenta convergência em menos iterações, mas requer mais cálculos por iteração devido ao cálculo de matrizes inversas. Apesar do grande esforço computacional, ele segue sendo o algoritmo de treinamento mais rápido para redes neurais, quando se trabalha com um número moderado de parâmetros na rede. Se esse número é elevado, a utilização desse algoritmo é pouco prática devido a exigência computacional elevada.

3.10. Variações do algorimo *back-propagation*

De acordo com BRAGA, LUDEMIR & CARVALHO (2000) devido aos problemas de tempo de aprendizado, desde o seu surgimento o algoritmo *back-propagation* vem sendo realizadas varias alterações no algoritmo visando à melhoria do tempo de treinamento e classificações de padrões. Os algoritmos mais utilizados entre essas alterações são: os algoritmos *Quickprop* e *Rprop*.

A principal diferença entre o algoritmo *Quickprop* e o algoritmo *back-propagation* padrão, é que para cada peso são usadas as inclinações anterior e atual do erro. O algoritmo *Quickprop* tem a vantagem de convergir rapidamente porem essa convergência muito rápida pode levar ao estouro da capacidade de representação numérica utilizada.

O algoritmo *Rprop* procura eliminar a influencia negativa da derivada parcial no ajuste dos pesos utilizando apenas o sinal da derivada ao invés do seu valor. Se a derivada parcial trocar de sinal, indicando que o algoritmo atingiu o mínimo local, o fator de ajuste é diminuído. Se a derivada manter o sinal, o fator de ajuste é ligeiramente aumentado a fim de aumentar a convergência.

3.11. Arquiterura do modelo CasCor (*Cascade Correlation*)

A arquitetura CasCor é um modelo de rede neural construtiva e treinada com o algoritmo *Cascade Correlation* e técnicas de *quickprop* para acelerar o processo de aprendizagem. Uma rede construtiva otimiza sua topologia, podendo inserir ou remover nodos durante o processo de treinamento, apresentando as seguintes vantagens: não precisar estipular o numero de unidades que a rede deve possuir, e aumento da velocidade no tempo de treinamento.

Segundo BRAGA, LUDERMIR & CARVALHO (2000) a rede C*ascade Correlation* foi criado por Fahlman com o objetivo de superar as limitações da rede MLP treinada com o algoritmo *back-propagation*. Uma rede CasCor inicialmente possui apenas as camadas de entrada e saída, e o algoritmo *cascade correlation* insere novas camadas intermediarias durante o processo de treinamento. Cada nova camada inserida na rede tem suas entradas conectadas aos nodos de entrada e nas camadas intermediarias, e sua saída conectada ao nodo

de saída. Os pesos dos neurônios vão sendo ajustados de forma a minimizar ao máximo o erro, quando não for mais possível descer na curva de erro, um conjunto de unidades candidatas são alocadas e treinadas a parte até encontrar a unidade com maior correlação com o erro gerado pelo neurônio de saída. A nova unidade é inserida na rede e seus pesos são "congelados" para não serem mais ajustados, depois disto o neurônio de saída é novamente treinado, e repete-se os passos para inserção de uma nova camada. Diferente da arquitetura de múltiplas camadas tradicional, onde as camadas ocultas são alinhadas lado a lado, na arquitetura CasCor as camadas ocultas são alinhadas em cascata.

A figura 18 demonstra o funcionamento básico de uma rede CasCor.

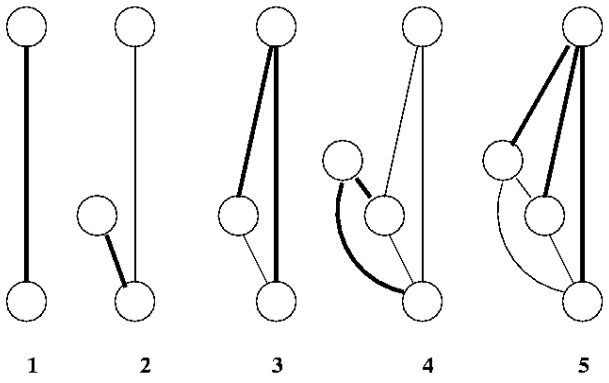

Figura 18 – Arquitetura do modelo Cascade Correlation. [9]

[9] Figura disponível em: http://www.willamette.edu/~gorr/classes/cs449/figs/cascor.gif.

Capítulo 4 - RECONHECIMENTO DE ASSINATURAS

Quando escrevemos nossos movimentos são tão automatizados que nossa mão se move duas vezes mais rápido do que podemos controlar conscientemente. Com isso cada pessoa tem sua peculiaridade na sua forma de escrita, o que diferencia uma assinatura de outra e o que dificulta tentativas de fraudes, pois perdem o dinamismo com que a assinatura original é feita. A identificação por meio de assinaturas é amplamente utilizada hoje em dia por sua simplicidade, confiabilidade e baixo custo.

A ciência que estuda os traços biológicos que são únicos para cada individuo é conhecida como biometria. Biometria é o uso de características biológicas em mecanismos de identificação, por exemplo: a voz, a íris, a retina, impressão digital, assinatura entre outros.

Um sistema de identificação de assinaturas é um sistema que recebe uma assinatura e informa o nome do usuário que a gerou, dentre os diversos usuários cadastrados em uma base de dados de assinaturas. Em sistemas deste tipo o sistema deve informar qual o usuário que tem a assinatura mais semelhante a que está sendo analisada.

4.1. Etapas para identificação de uma assinatura

O processo de classificação de uma assinatura engloba varias etapas de acordo com o modelo a ser implementado, mas para todo processo de identificação é necessário passar por etapas principais que são: obtenção e armazenamento da assinatura, pré-processamento e processamento da assinatura.

Na etapa de obtenção e armazenamento, é coletada a assinatura através de um dispositivo de *hardware* como um *scanner* ou câmera digital, e armazenada de forma

estruturada em um arquivo ou tabela de banco de dados para que possa ser acessada posteriormente.

A etapa de pré-processamento prepara a assinatura, fazendo ajustes de posicionamento, ajustes de escala para minimizar as variações de tamanho que podem ocorrer entre diversas assinaturas de uma mesma pessoa, a utilização de algumas técnicas de processamento de imagens, que estão descritas no item 4.2, para contornar eventuais defeitos na imagem adquirida, além da extração dos atributos da assinatura, que servem para diferenciar uma assinatura das demais.

A extração de atributos é necessária para garantir a integridade da assinatura, pois pode ocorrer uma pequena variabilidade entre assinaturas de uma mesma pessoa, fazendo-se necessário obter atributos que sejam menos variáveis dentro de uma assinatura de uma mesma pessoa e que diferencie em relação a uma assinatura de uma outra pessoa.

Os atributos comumente extraídos para identificar uma assinatura são classificados como dinâmicos e estáticos. Atributos conhecidos como atributos dinâmicos são melhores utilizados para verificação *On-line* da assinatura. Pode-se exemplificar atributos dinâmicos como sendo: a inclinação da assinatura, o espaçamento entre os caracteres, a espessura do traço em cada parte da assinatura, o alinhamento da assinatura em relação à linha de base, a relação de proporcionalidade gráfica, a velocidade da escrita, a força aplicada, o número de laços fechados, a quantidade de vezes que a caneta foi levantada da superfície entre outros fatores.

São considerados atributos estáticos: a quantidade de pixels pertencentes a determinada região da imagem, este método também é conhecido como densidade da assinatura com informações de grade. Outro método que pode ser utilizado é a aplicação de uma função estatística conhecida como PDF (função de densidade probabilística).

Para o projeto proposto será implementado um sistema de verificação *Off-line* onde será necessário fazer a extração de atributos estáticos.

Na etapa de processamento é feita a classificação da assinatura, ou seja, determina se a assinatura pertence ou não ao usuário. Esta etapa tem duas fases distintas, primeiro é necessário que ocorra o aprendizado, onde uma base de dados é apresentada ao sistema para que ele possa aprender a classificar adequadamente. A segunda fase é a fase de reconhecimento, onde uma assinatura é submetida ao sistema e este por sua vez retorna o nome do usuário que é dono da assinatura informada.

4.2. Processamento de Imagens

Processamento de imagens digitais são técnicas voltadas para a análise de dados multidimensionais, adquiridos por diversos tipos de sensores, ou seja, é a manipulação de uma imagem por computador de modo que a entrada e a saída do processo são imagens. É utilizado para melhorar o aspecto visual de certas feições estruturais para o analista humano e para fornecer outros subsídios para a sua interpretação, inclusive gerando produtos que possam ser posteriormente submetidos a outros processamentos.

De acordo com CHELLAPPA (1992) a imagem em que a assinatura esta contida é de suma importância para verificação, e técnicas de processamento de imagens podem melhorar o resultado obtido no treinamento da rede neural.

O objetivo dos métodos de realce de imagens é processar uma imagem visando à melhoria do desempenho do sistema em que essa imagem é aplicada. As técnicas de realce podem ser baseadas em processamento ponto-a-ponto, que modifica o nível de cinza de um

pixel independente dos *pixels* vizinhos, ou pelo processo de filtragem onde o novo valor depende dos *pixels* vizinhos.

Técnicas de calculo da média, limiarização, aumento de contraste e detecção de bordas, podem ser interessantes quando aplicadas no reconhecimento de imagens através de redes neurais, pois podem melhorar significativamente seu desempenho.

A técnica do filtro da média é aplicada para tentar eliminar eventuais ruídos na imagem da assinatura. O filtro de média faz a média dos valores de cinza dos pixels de uma vizinhança.

O aumento de contraste é uma técnica de realce utilizado com o objetivo de destacar somente o traçado da assinatura aplicando um nível de luminosidade a cada *pixel* da imagem e é normalmente utilizada como uma etapa de pré-processamento para sistemas de reconhecimento de padrões. O contraste entre dois objetos pode ser definido como a razão entre os seus níveis de cinza médios, a manipulação do contraste consiste numa transferência radiométrica em cada pixel, com o objetivo de aumentar a discriminação visual entre os objetos presentes na imagem. Realiza-se a operação ponto a ponto, independentemente da vizinhança.

O método de limiarização é uma transformada que transforma uma imagem em níveis de cinza em uma imagem binária, alocando o valor "1" (branco) se o nível de cinza do pixel é maior que o limiar T, $0 < T < L\text{-}1$, ou "0" (preto) em caso contrário. Este método diminui a complexidade do processamento, pois ignora os valores de cor irrelevantes na análise da assinatura.

Uma borda consiste em variação de gradientes, entre o objeto e o seu fundo. A técnica para detectar bordas é utilizada para fins de destacar as linhas da assinatura, e somente

elas. Há diversos algoritmos que foram desenvolvidos com a finalidade de detecção de bordas, dentre eles podemos destacar o algoritmo de Sobel, Roberts e Canny.

Foram testados os diversos tipos de métodos para detectar as bordas, o método que apresentou melhor resultados, ou seja, o que conseguiu identificar um número maior de bordas relevantes foi o método de Sobel, por isso foi o método escolhido para o projeto.

O filtro Sobel calcula o gradiente da intensidade da imagem em cada ponto, dando a direção da maior variação de claro para escuro e a quantidade de variação nessa direção. Com isto consegue-se estimar a presença de uma transição claro-escuro e sua orientação. Como as variações claro-escuro intensas correspondem a fronteiras bem definidas entre objectos, consegue-se fazer a detecção de bordas.

$$\mathbf{G_x} = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * \mathbf{A} \quad \text{e} \quad \mathbf{G_y} = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * \mathbf{A}$$

Figura 19 - Template Sobel.[10]

Template é uma máscara utilizada para a realização de operações na vizinhança de um pixel. Os elementos da vizinhança e o pixel central são multiplicados pelos valores indicados nas posições correspondentes da matriz.

Depois desses métodos podemos extrair os dados da imagem para serem usados como entradas para a rede neural. Essas etapas de pré-processamento são importantes para o reconhecimento de imagens, pois qualquer ruído ou distorção pode levar a rede neural a classificar uma imagem erroneamente.

[10] Fonte: http://pt.wikipedia.org/wiki/Filtro_Sobel

Nos quatro primeiros capítulos foram mostrados estudos feitos sobre redes neurais artificiais, expondo suas principais características e arquiteturas, além de alguns métodos para o reconhecimento de assinaturas e processamento de imagens.

Os próximos passos para a concretização do projeto será o estudo do funcionamento do software Matlab e o uso de seus *toolboxes* para implementar uma rede neural para reconhecer uma determinada assinatura apresentada como entrada, e classificá-la entre outras assinaturas. Serão feitos estudos para a escolha da arquitetura para a rede neural, além da escolha da melhor interface entre o usuário e o Matlab.

Capítulo 5 - DESENVOLVIMENTO DO PROTÓTIPO

Neste capítulo será mostrado as etapas desenvolvidas para a criação do protótipo de rede neural para identificação de usuários através da assinatura correspondente ao mesmo.

5.1. Requisitos do problema a ser trabalhado

O sistema proposto a ser desenvolvido deve atender aos seguintes requisitos:

a) Criação de uma rede neural.

b) Treinamento de uma rede neural.

c) Validação da rede neural através de sua generalização.

d) Salvamento dos valores dos pesos sinápticos após o treinamento.

e) Carregamento do valor dos pesos sinápticos para continuação de um treinamento.

f) Execução da rede neural.

g) Criação de uma interface gráfica entre o usuário e o Matlab.

5.2. Visão geral do sistema

Foram adquiridas cinco assinaturas de doze pessoas distintas, totalizando uma base de dados com 60 assinaturas. Como o projeto proposto é de uma rede neural capaz de fazer a identificação de usuários através das assinaturas dos mesmos, um número muito grande de pessoas distintas resultaria em uma rede neural com dimensões elevadas. Por essa razão optou-se por uma base de dados relativamente pequena para a prática real física da rede e avaliação do seu desempenho.

O protótipo deve permitir que o projetista treine a rede neural passando como entrada para a rede um vetor de características da imagem da assinatura. Este vetor de características é gerado após o pré-processamento da imagem, que se faz necessário devido à baixa qualidade das imagens adquiridas.

Nesta fase de pré-processamento são recortadas as áreas da imagem que não fazem parte da assinatura, conseguindo com isso um melhor ajuste de posição. Depois de delimitar a área de interesse da imagem as imagens são padronizadas para um tamanho de 48x128 pixels. O próximo passo é binarizar (digitalizar) o valor dessas imagens para posteriormente extrair suas bordas, o que facilita no processo de extração do vetor de características para ser usado no treinamento da rede neural.

Depois do pré-processamento o vetor de características é gerado através do método da densidade de pontos pertencentes à assinatura com informação de grades. Esse método consiste em dividir a imagem em blocos numa proporção de 12x16 e contabilizar os pixels pertencentes à assinatura em cada bloco, gerando assim uma matriz resultante de 4x8. Esta matriz é transformada em um vetor de 32 posições que corresponde à entrada da rede neural.

O treinamento da rede neural é feito com as assinaturas coletadas dos usuários, sendo que as mesmas foram divididas nas proporções de 60% do numero total para treinamento da rede e 40% para a realização de testes de generalização.

Após ser concretizado o treinamento da rede, seu conhecimento (vetores de pesos) devem ser armazenados em um arquivo e a rede neural exportada para realização dos testes de generalização.

Para a utilização da rede neural foi construída uma interface gráfica de fácil manuseio, que realiza todas as operações listadas acima de uma forma bem simples.

5.3. Pré-processamento das imagens

As assinaturas foram coletadas através de uma câmera digital e repassadas para o computador para a realização das operações necessárias. Este método de captura é relativamente simples de se realizar, porém as imagens obtidas não são de muita qualidade e vários fatores como luminosidade e foco afetam na imagem resultante.

Para padronizar as assinaturas obtidas foram realizados diversos métodos de pré-processamento. O primeiro passo foi destacar da imagem apenas a região pertencente à assinatura, recortando da imagem as partes irrelevantes.

Esta tarefa foi realizada através de uma ferramenta bem simples que é o Paint. A figura 20 ilustra a comparação da imagem original e o recorte da área de interesse correspondente.

Figura 20 - Área de interesse da assinatura.

Após o recorte da área de interesse, as imagens foram redimensionas para um tamanho padrão de 48x128 pixels. O redimensionamento e as operações de processamento seguintes foram realizados com o auxilio do Matlab e de sua toolbox de processamento de imagens.

Depois da padronização das imagens é realizada uma operação de binarização. Esta operação consiste em transformar os valores da imagem para 0 e 1, sendo que valores abaixo de um determinado limiar são trocados por zero e valores acima por um, o limiar escolhido

para a realização da operação de binarização foi de 0,3 por ter apresentado melhores resultados para a conversão, destacando os principais pontos da assinatura.

A figura 21 mostra a comparação da imagem padronizada e a aplicação da binarização.

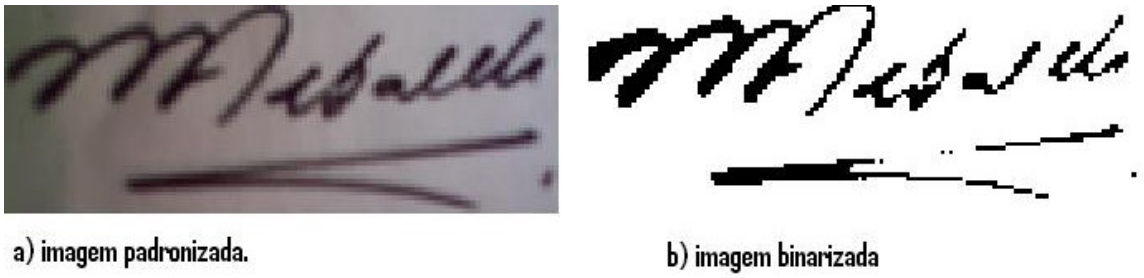

Figura 21 – Binarização da Imagem**.**

Após a binarização da imagem é aplicado o método de Sobel discutido nos capítulos anteriores para detectar as bordas da imagem. Através desse método consegue-se segmentar a imagem e os valores pertencentes ao contorno da imagem passam a ter o valor 1 e o restante da imagem o valor 0. Este método ajuda na extração de características da assinatura, pois será feito a contagem do numero de pixels pertencentes a assinatura.

A figura 22 ilustra o método de detecção de bordas de Sobel.

Figura 22 - Detecção das Bordas da Assinatura.

5.4. Extração de características pertencentes à assinatura

É necessário extrair as características da assinatura para que a rede neural possa classificá-la corretamente. Inicialmente foi construída uma rede neural tendo como entrada o numero de pixels da imagem, porém este método não se mostrou eficaz, pois não foi possível conseguir um nível de generalização aceitável, além de ser preciso um numero de entradas para a rede neural muito grande, o que dificulta no tempo de treinamento e no processo de generalização.

O método mais eficaz encontrado para a extração das características da assinatura foi a extração de características através da densidade da assinatura através de informação de grades. É um método comumente utilizado para a extração de características de imagens, podemos citar os trabalhos de HEINEN (2002) que utilizou uma adaptação de informações de grade baseado nos trabalhos de BALTZAKIS e PAPAMARKOS, ambos para reconhecimento de assinaturas.

Este método consiste em dividir a imagem da assinatura em blocos, e para cada bloco é contabilizado o numero de pixels pertencentes à assinatura. Como resultado dessa operação é gerado uma nova matriz com cada elemento contendo o numero de pixels contabilizados pelo processo anterior.

É necessário estimar um numero apropriado de blocos a serem divididos, pois um número muito grande de blocos deixa vários espaços vazios o que não é proveitoso para a rede neural, e blocos muito pequenos impedem a rede diferenciar uma assinatura de outra.

Foram testados varias divisões para esse numero de blocos e o que apresentou melhores resultados foram blocos com proporções de 12x16 levando em consideração as imagens padronizadas em 48x128 pixels. Isso significa que a imagem de proporções de 48x128 será dividida por 12x16, gerando uma matriz resultante com dimensões de 4x8

(quatro linhas e oito colunas). Esta matriz é finalmente transformada em um vetor de 32 posições que será utilizado como entrada para a rede neural.

A figura 23 mostra um exemplo de como os blocos são utilizados.

Figura 23 - Imagem de tamanho 48x128 dividida em blocos de 4x8.

5.5. Estrutura da rede neural

A estrutura da rede neural está diretamente associada ao seu comportamento, determinar sua topologia é umas das principais tarefas a serem realizadas. Com base em pesquisas feitas e vários testes realizados pode-se tirar algumas conclusões sobre a topologia da rede neural.

Primeiramente é necessário determinar o numero de neurônios que constituirão a camada de entrada e a camada de saída. Um número elevado de neurônios na camada de entrada não é interessante, pois o tempo de treinamento torna-se demasiadamente lento, além de resultar em redundância de informação que pode atrapalhar a rede na classificação da assinatura. Baseando-se nesse principio optou-se por utilizar o mínimo de camadas de entrada com o máximo de informação disponível através da extração de características. O vetor de características está diretamente associado ao número de entradas para a rede neural, o que corresponde a um total de 32 neurônios na camada de entrada.

A camada de saída corresponde às saídas desejadas para o vetor de entrada apresentado a rede. O cálculo do número de neurônios na camada de saída foi baseado no número de classes para os padrões de entrada, sendo que para cada usuário foi estipulada uma

classe. O projeto conta com 12 classes de usuário distintas, por tanto o número de neurônios na camada de saída é igual a doze.

Após determinado o numero das camadas de entrada e saída, é necessário estipular o número de camadas ocultas para a rede neural. Sabe-se que um número elevado de camadas ocultas não é interessante, pois para cada camada oculta o cálculo do erro gerado é menos preciso, sendo que a única camada que sabe o valor absoluto do erro é a camada de saída, para cada camada oculta é feito uma estimativa do erro. Por isso foi escolhida apenas uma camada oculta para a rede desenvolvida.

O passo seguinte é estipular o número de neurônios para a camada oculta. Um número elevado de neurônios nessa camada faz com que a rede neural perca sua capacidade de generalização, este problema é conhecido como *overffiting*. Foram feitos vários testes com diferentes números de neurônios na camada oculta tentando obter o melhor resultado com o mínimo de neurônios possíveis nesta camada. Inicio-se com 10 neurônios para verificar o comportamento da rede, porém os resultados não foram aceitáveis, devido a esse motivo foram realizados incrementos em relação a esse número até atingir um resultado satisfatório. O número de neurônios na camada que apresentou melhor resultado foi com 14 neurônios.

A figura 24 ilustra a topologia utilizada para este projeto.

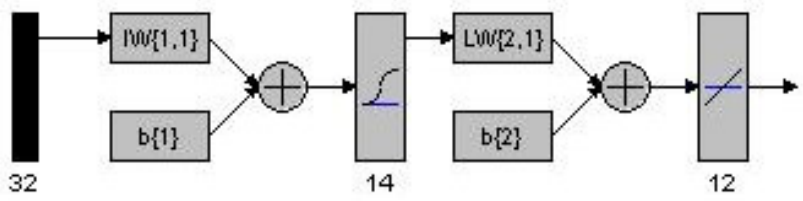

Figura 24 - Topologia Utilizada.

5.6. Parâmetros de aprendizagem

Os parâmetros de treino são tão importantes quanto a definição da quantidade de neurônios presentes em cada camada. Os principais parâmetros a serem considerados são o número de ciclos que os padrões devem ser apresentados para o treino, a taxa de aprendizagem da rede neural e o algoritmo de treinamento a ser utilizado.

O algoritmo de treinamento utilizado foi o R-Prop (*Resilient Propagation*). Este algoritmo é uma variação do Back-Propagation tradicional onde é utilizado apenas o sinal da derivada para o ajuste dos pesos e não seu valor. Foram testados vários algoritmos de treinamento como o próprio backpropagation tradicional, backpropation com taxa de aprendizado adaptativa, backpropagation de Levenberg e R-Prop. O algoritmo R-Prop mostrou-se o melhor algoritmo, pois o tempo de treinamento é rápido e alcançou níveis de generalização satisfatórios.

A forma de treinamento utilizado foi através do método por *batch*. Neste método de treinamento os pesos das conexões são modificados apenas após todos os padrões de entradas serem apresentados à rede neural. O número de ciclos utilizados foi de 5000 épocas com erro mínimo de 0.0001, calculado através do erro médio quadrático.

Outro parâmetro a ser ajustado é taxa de aprendizagem e o valor de momentum da rede neural, sendo recomendado por muitos pesquisadores à utilização do valor 0.1 e 0.3 respectivamente. A taxa de aprendizado e o momentum são responsáveis pela velocidade de aprendizado para a rede e um valor elevado para estes parâmetros pode resultar na má generalização da rede neural.

5.7. Interface gráfica

Foi construída uma interface para melhor uso do sistema, a interface gráfica é composta de elementos simples como *push buttons* e *text boxes*. É disponibilizado para o usuário as opções de treinar a rede neural, verificar sua topologia, simulação, escolha da assinatura e verificação dos passos de processamento.

As figuras 25 e 26 mostram as principais funcionalidades do sistema através da interface gráfica projetada.

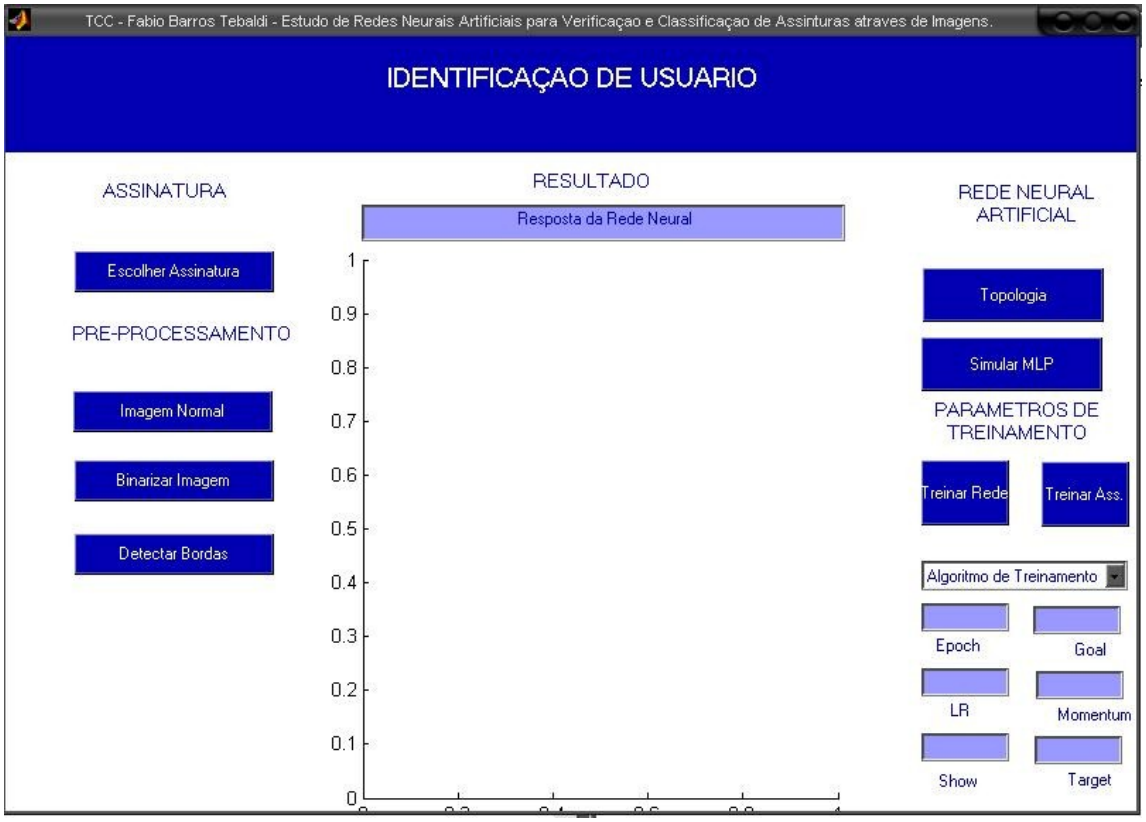

Figura 25 - Tela inicial da interface.

A tela principal do sistema onde é possível escolher as seguintes funcionalidades: Escolher Assinatura, Imagem Normal, Binarizar Imagem, Detectar Bordas, Simular MLP, Topologia, Treinar Rede e Treinar Assinatura.

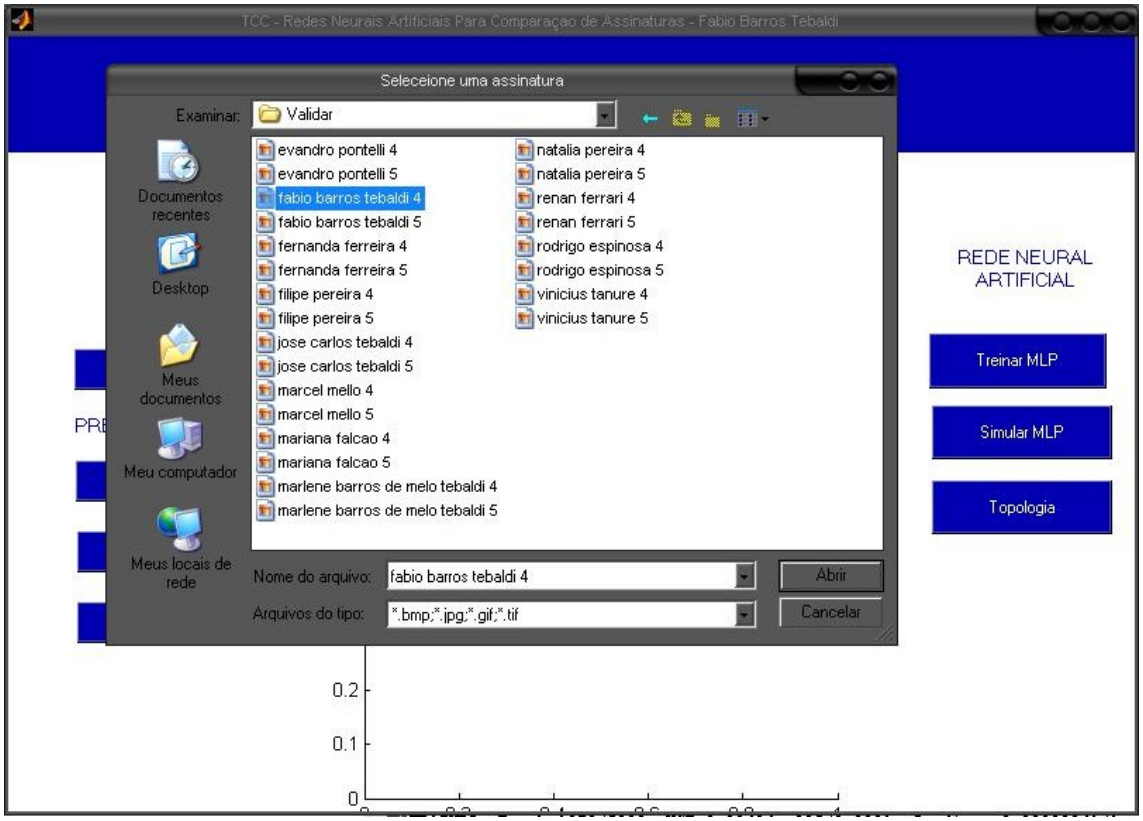

Figura 26 - Escolher Assinatura.

A função Escolher Assinatura mostrada na figura 26 abre uma janela onde da o direito ao usuário de escolher uma assinatura através do diretório especificado. Após a escolha da assinatura a imagem da mesma é mostrada na tela central do sistema, utilizando-se da função Imagem Normal.

As funções Binarizar Imagem e Detectar Bordas fazem as operacionalidades de mesmo nome e são mostradas nas figuras 27, 28 e 29 respectivamente.

A função Treinar Rede, mostrado na figura 30, faz o treinamento da rede neural usando 60% do conjunto de assinaturas pertencentes à base de dados. A função Treinar Assinatura, permite ao usuário o treinamento de uma única assinatura, permitindo a escolha dos padrões de treinamento como o número de épocas, a taxa de aprendizado, o valor de momentum e erro mínimo tolerável. Também é possível que o usuário escolha o algoritmo de treinamento a ser usado pela rede neural.

Um gráfico de treinamento será mostrado, ilustrando a curva de erro e quando a rede finalmente convergir para o erro mínimo tolerável (*goal)* o treinamento é encerrado. Exemplificado na figura 31.

A função Simular, demonstrada na figura 32, gera a resposta da rede neural relativa à assinatura escolhida, ou seja, retorna o nome de quem à assinatura pertence.

A função Topologia mostra a topologia da rede neural mostrada na Figura 24.

Figura 27 - Imagem Normal.

Figura 28 - Binarizar Imagem.

Figura 29 – Detectar Bordas.

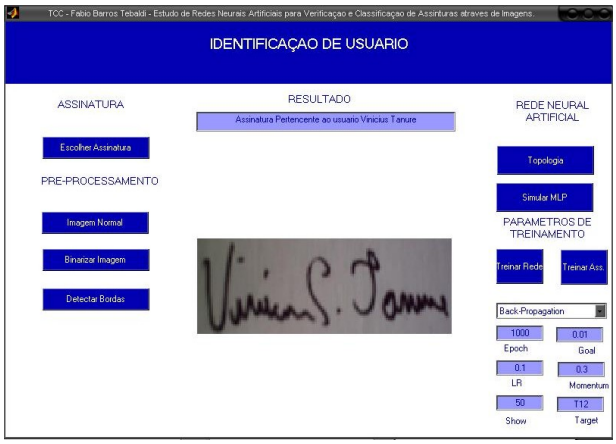

Figura 30 – Treinamento.

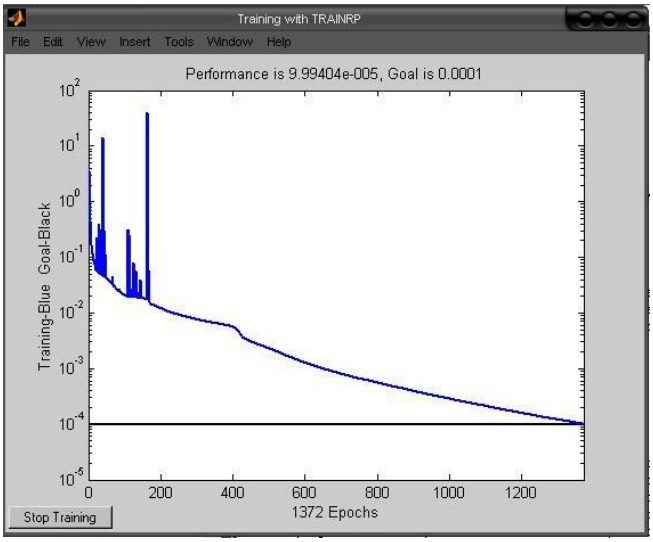

Figura 31 – Gráfico da Curva de Erro.

Figura 32 – Simular MLP.

5.8. Resultados obtidos

Ao longo do projeto foi necessário o desenvolvimento de várias redes neurais com diferentes topologias e padrões de entrada até alcançar um resultado de generalização satisfatório.

Inicialmente foram criadas duas rede neurais que se complementassem, uma rede que gerasse uma resposta visual ao padrão de entrada apresentado e uma rede que gerasse uma resposta numérica ao mesmo padrão, complementando a resposta da primeira rede neural. Estas redes realizavam a comparação entre duas assinaturas para formarem suas saídas, tendo como entrada os pixels de cada assinatura.

A topologia utilizada para rede neural responsável pela resposta visual foi uma rede do tipo MLP com 100 neurônios na camada de entrada, 2 camadas ocultas com 300 e 100 neurônios respectivamente e 50 neurônios na camada de saída treinada com o algoritmo backpropagation com taxa de aprendizado adaptativa. Para a resposta numérica ao padrão apresentado foi desenvolvida uma rede neural do tipo RBF, que calculava o valor da correlação entre os padrões de entrada. A topologia da rede contava com 100 neurônios na camada de entrada, 1 neurônio na camada intermediaria e 1 neurônio na camada de saída.

O problema encontrado para estas redes neurais foi sua capacidade de generalizar, tendo em vista que seu comportamento apenas atingia graus aceitáveis de generalização quando as imagens comparadas fossem extremamente semelhantes. Essas redes não foram capazes de extrair as características gerais das assinaturas, por isso foi necessária à construção de um outro modelo de rede neural, além da necessidade de um tempo muito longo para a realização do treinamento, demorando no total de doze horas para a realização do mesmo.

A figura 33 exemplifica o funcionamento dessas redes, para melhor entendimento: as duas primeiras imagens são as assinaturas escolhidas pelo usuário e a terceira imagem é a resposta gerada pela rede neural.

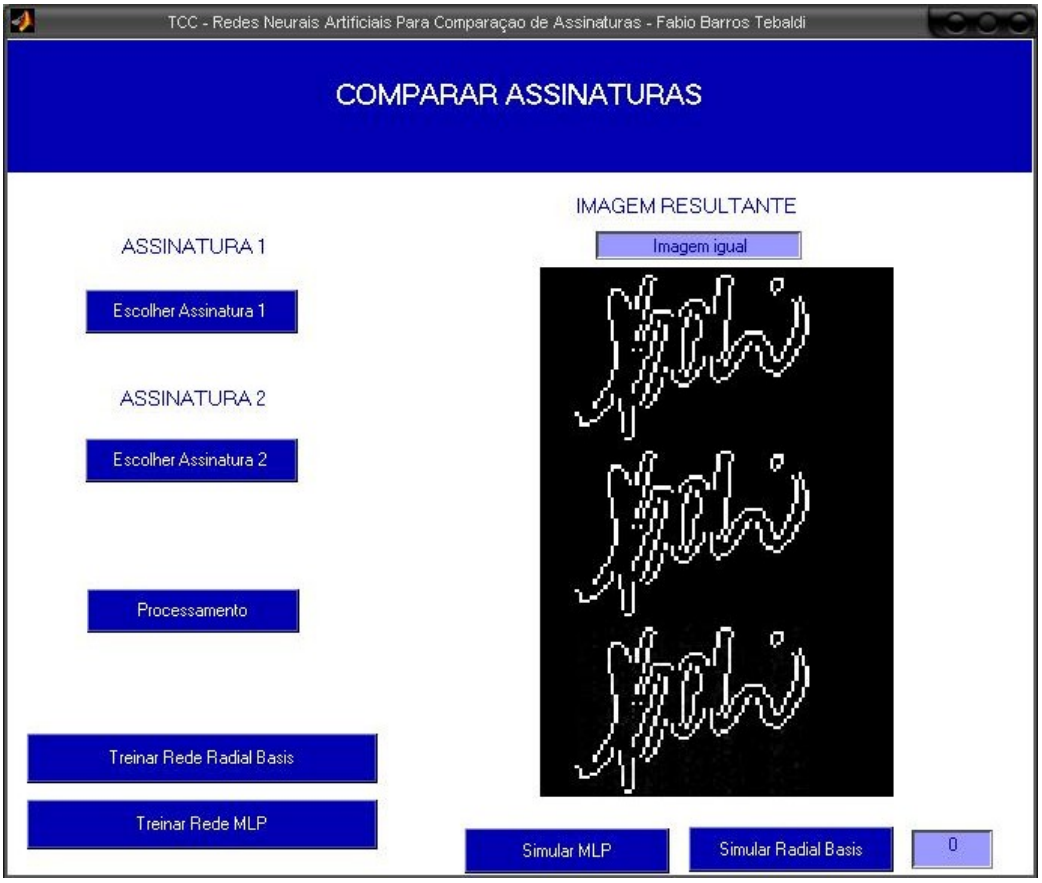

Figura 33 – Protótipo da Primeira Rede Neural Construída.

Optou-se pela construção de um protótipo que fosse capaz de fazer a identificação de usuários através de suas respectivas assinaturas ao invés da mera comparação visual entre as assinaturas. Para tal foi desenvolvido inicialmente um protótipo de rede neural que tinha como entrada a imagem da assinatura de proporções 420x720 dividida em blocos de 7x12 com o numero de pixels pertencentes à assinatura contabilizados em cada bloco. A topologia utilizada foi de uma rede neural MLP com 84 neurônios na camada de entrada, 1 camada oculta com 168 neurônios e a camada de saída com 12 neurônios.

O treinamento dessa rede foi realizado com o algoritmo R-Prop e dividindo o conjunto de assinaturas da base de dados em 80% (48 assinaturas) para treinamento e 20% (12 assinaturas) para testes de generalização. Isto é como a base de dados é composta por doze usuários com cinco assinaturas distintas para cada usuário, foram utilizadas quatro assinaturas

de cada usuário para o treinamento e deixado uma assinatura de cada usuário para realização dos testes.

Esse protótipo de rede neural conseguiu uma taxa de 100% de aprendizagem e 50% nos testes de generalização. Entres as sessenta assinaturas disponíveis foram classificadas erroneamente seis assinaturas, dando ao sistema uma média de 90% de desempenho e está demonstrado na Tabela 2.

Para realização de novos testes, com um número reduzido de divisões, foi construído uma rede neural que tinha como entrada a imagem da assinatura de proporções 48x128 dividida em blocos de 4x8 com o numero de pixels pertencentes à assinatura contabilizados em cada bloco. A topologia utilizada foi de uma rede neural MLP com 32 neurônios na camada de entrada, 1 camada oculta com 16 neurônios e a camada de saída com 12 neurônios.

O treinamento dessa rede foi realizado com o algoritmo R-Prop e dividindo o conjunto de assinaturas da base de dados em 80% (48 assinaturas) para treinamento e 20% (12 assinaturas) para testes de generalização. Esse protótipo de rede neural conseguiu uma taxa de 100% de aprendizagem e 59% nos testes de generalização. Entres as sessenta assinaturas disponíveis foram classificadas erroneamente cinco assinaturas, dando ao sistema uma média de 92% de desempenho e está demonstrado na Tabela 3.

Com base no ultimo protótipo desenvolvido foi implementando o modelo final da rede neural utilizando os mesmos princípios do protótipo anterior. A camada de entrada recebe uma imagem de tamanho 48x128 dividida em blocos de 4x8 com o numero de pixels pertencentes à assinatura contabilizados em cada bloco.

A topologia utilizada foi a construção de uma rede neural MLP com 32 neurônios na camada de entrada, 1 camada oculta com 14 neurônios e 12 neurônios na camada de saída.

O treinamento dessa rede foi realizado com o algoritmo R-Prop e dividindo o conjunto de assinaturas da base de dados em 60% para treinamento e 40% para testes de

generalização. Isto é como a base de dados é composta por doze usuários com cinco assinaturas distintas para cada usuário, foram utilizadas três assinaturas de cada usuário para o treinamento (36 assinaturas) e deixado duas assinatura de cada usuário para realização dos testes (24 assinaturas).

Esse protótipo de rede neural conseguiu uma taxa de 100% de aprendizagem e 71% nos testes de generalização. Entres as sessenta assinaturas disponíveis foram classificadas erroneamente sete assinaturas, dando ao sistema uma média de 88% de desempenho e está demonstrado na Tabela 4.

Comparado ao protótipo anterior, os índices de generalização deste protótipo mostrou-se superior e seu índice de desempenho é semelhante. Devido aos níveis de generalização conseguidos com essa rede neural, ela foi adotada como modelo final do projeto.

As figuras 34, 35 e 36 demonstram a comparação entre os algoritmos de treinamento testados. É possível visualizar que o algoritmo R-Prop demonstra uma melhor convergência em relação aos algoritmos Backpropagation tradicional, que não foi capaz de convergir e o algoritmo de Levenberg, que convergiu rápido demais.

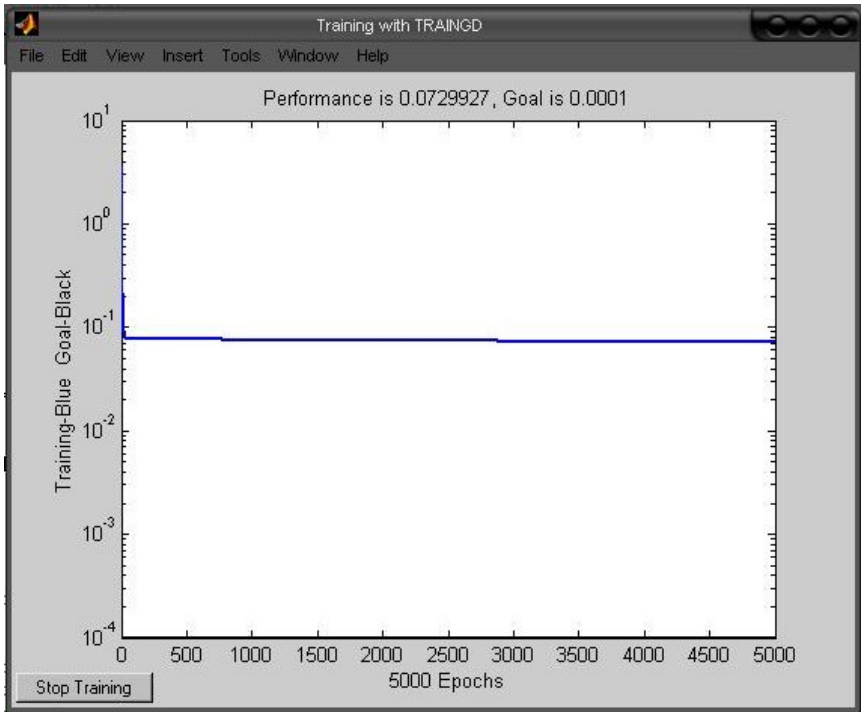

Figura 34 – Gráfico de treinamento Backpropagation.

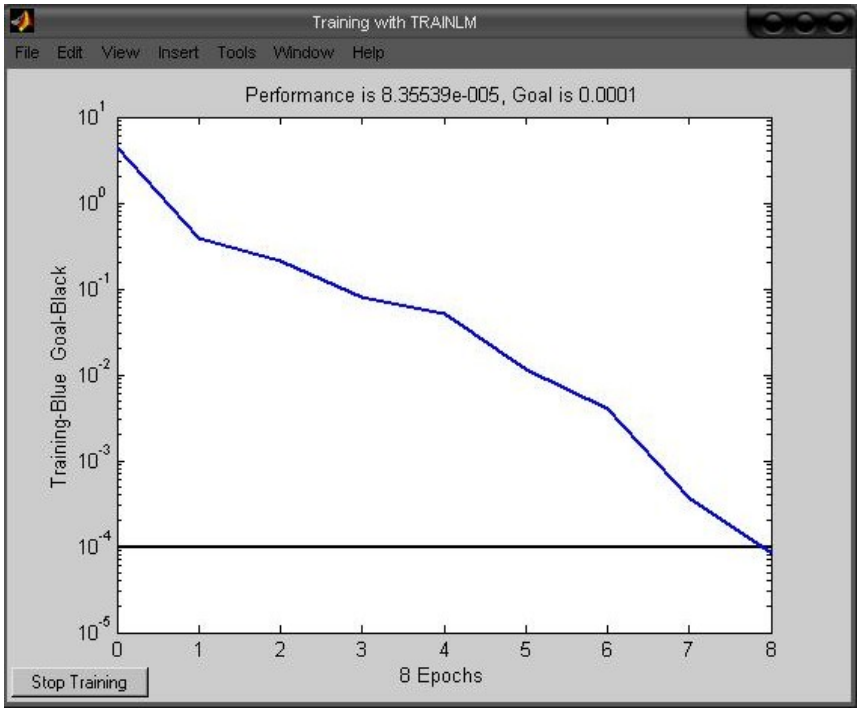

Figura 35 - Gráfico de treinamento Levenberg.

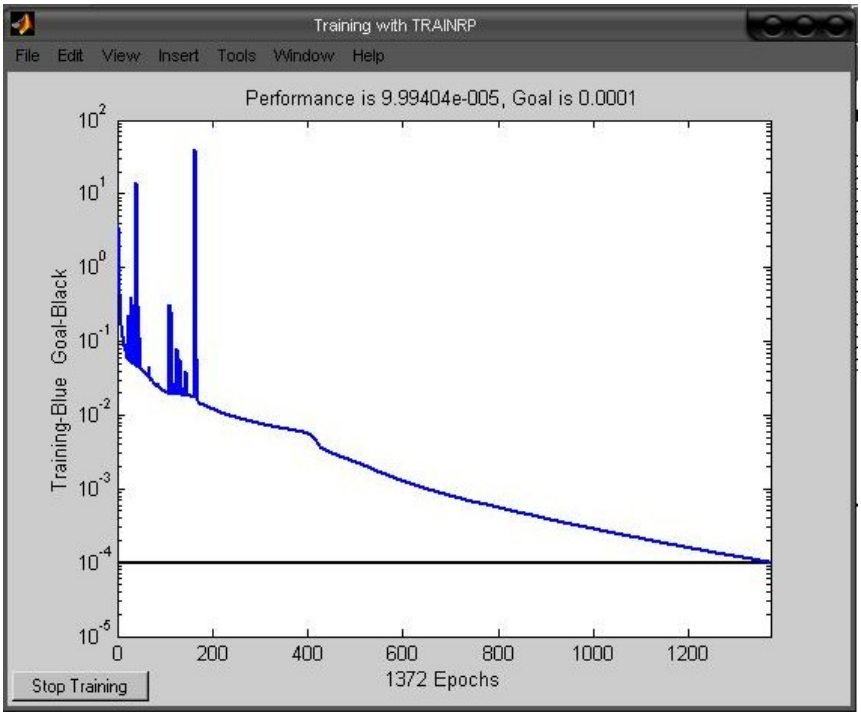

Figura 36 - Gráfico de treinamento R-prop.

As tabelas 2, 3 e 4 demonstram as comparações realizadas entre as redes neurais construídas que obtiveram melhores resultados.

REDE NEURAL 1	
NEURÔNIOS CAMADA ENTRADA	84
NEURÔNIOS CAMADA OCULTA	168
NEURÔNIOS CAMADA SAÍDA	12
RESULUÇÃO DA IMAGEM	420x720
DIVISÃO DE BLOCOS	7x12
ASSINATURA TREINAMENTO	48
ASSINATURA TESTES	12
APRENDIZAGEM %	100
GENERALIZAÇÃO %	50
DESEMPENHO TOTAL %	90

Tabela 2 – Resultados da primeira rede neural estável construída.

REDE NEURAL 2	
NEURÔNIOS CAMADA ENTRADA	32
NEURÔNIOS CAMADA OCULTA	16
NEURÔNIOS CAMADA SAÍDA	12
RESULUÇÃO DA IMAGEM	48x128
DIVISÃO DE BLOCOS	4x8
ASSINATURA TREINAMENTO	48
ASSINATURA TESTES	12
APRENDIZAGEM %	100
GENERALIZAÇÃO %	59
DESEMPENHO TOTAL %	92

Tabela 3 - Resultados da segunda rede neural estável construída.

REDE NEURAL FINAL	
NEURÔNIOS CAMADA ENTRADA	32
NEURÔNIOS CAMADA OCULTA	14
NEURÔNIOS CAMADA SAÍDA	12
RESULUÇÃO DA IMAGEM	48x128
DIVISÃO DE BLOCOS	4x8
ASSINATURA TREINAMENTO	36
ASSINATURA TESTES	24
APRENDIZAGEM %	100
GENERALIZAÇÃO %	71
DESEMPENHO TOTAL %	88

Tabela 4 - Resultados do modelo final construído.

Capítulo 6 - CONCLUSÕES

Neste capitulo serão apresentadas as conclusões referentes a todo o processo envolvido tanto no estudo, como no desenvolvimento deste protótipo.

6.1. Objetivos do trabalho

A proposta inicial do trabalho era de criar um protótipo que através de Redes Neurais Artificiais pudesse realizar a identificação de assinaturas e classifica-las de acordo com o usuário pertencente à mesma. Este requisito foi atingido com sucesso apesar do sistema obter uma margem de erro de 12%.

Outro requisito importante era a criação de um meio de interação entre o usuário e o sistema desenvolvido de uma forma simples através de uma interface gráfica amigável. Este requisito foi totalmente atendido.

6.2. Ferramentas utilizadas

A principal ferramenta utilizada foi a ferramenta Matlab. O Matlab demonstrou-se uma ferramenta de fácil uso com uma linguagem de programação de fácil acesso. O Matlab é composto por varias rotinas e funções pré-desenvolvidas conhecidas como *toolboxes*. Estes *toolboxes* foram de imensa ajuda para o desenvolvimento do projeto, principalmente para o desenvolvimento de funções complicadas como o treinamento da rede neural e para algoritmos de processamento de imagens.

Outra ferramenta utilizada foi o conhecido Paint, devido a complexidade da construção de algoritmo eficaz para posicionamento e delimitação da área de interesse das imagens, fez-se necessário que o projetista realiza-se essas operações manualmente.

6.3. Protótipo

O protótipo desenvolvido possui as seguintes vantagens:

a) Treinamento de uma rede neural de forma que seja capaz de aprender todos os padrões apresentados a ela.
b) Realizar a identificação de usuários através de suas assinaturas no formato de imagens.
c) Capacidade de generalizar informações que não foram utilizadas no treinamento da rede.
d) Realizar algoritmos de processamento de imagens como binarização, detecção de bordas e operação em blocos de forma simplificada.
e) Um ambiente gráfico de fácil utilização.

O protótipo desenvolvido possui as seguintes desvantagens:

a) Não implementação de um algoritmo capaz de efetuar o posicionamento e delimitação da área de interesse da imagem de forma automática.
b) Necessidade de re-treinamento da rede neural e ajustes de topologia quando um novo usuário é inserido no sistema.
c) A impossibilidade da extração de atributos dinâmicos referentes à assinatura.

6.4. Conclusões finais

O uso de redes neurais para resolução de problemas de reconhecimento de padrões mostrou-se bastante eficaz e interessante. A capacidade de generalizar informações não vistas através de um treinamento correto mostra a capacidade das redes neurais de solucionar problemas de classificação de forma satisfatória.

A maior dificuldade encontrada neste projeto foi extrair características capazes de diferenciar uma assinatura de outra devido à baixa qualidade das imagens adquiridas, ficando uma limitação na parte de extração de características pertencentes à assinatura.

Um dos problemas encontrados em relação às redes neurais artificiais foi a definição de sua topologia, pois não existem regras definidas para a criação de uma rede neural, sendo necessário certa experiência na área para a otimização de uma topologia. Não é possível saber como a rede neural conseguiu aprender ou onde se encontra o problema do não aprendizado, para conseguir um resultado satisfatório no uso de redes neurais é necessário a criação de vários modelos até conseguir a otimização de sua estrutura.

O estudo de redes neurais foi de grande interesse e bastante enriquecedor para o projetista, sendo bastante envolvente e é um assunto que ainda tem muito a ser explorado.

Chegando ao final do projeto, pode-se dizer que o mesmo teve seus objetivos cumpridos uma vez que o papel dos cientistas são pesquisar, testar e provar se coisas funcionam ou não, gerando a possibilidade de deixar material de pesquisa a novos alunos. Por parte do projetista, espera-se dar continuidade ao trabalho desenvolvido e se aprofundar ainda mais nos estudos sobre redes neurais.

6.5. Extensões do projeto

Seguem algumas idéias para complementar o trabalho apresentado:

a) Possível criação de um sistema de identificação de usuários utilizando atributos dinâmicos para a extração das características da assinatura.

b) Utilizar uma base de dados maior para verificação do comportamento da rede neural.

c) Desenvolvimento de um algoritmo para ajustes de posicionamento de forma automática.

d) Unificar este trabalho com técnicas de autenticação de assinaturas para verificação de fraudes.

e) Realização de testes em diferentes arquiteturas de RNAs e estudo do comportamento com redes de petri coloridas.

REFERÊNCIAS

ABAS, Rasha. **Backpropagation Networks prototype for off-line signature verification**. Minor thesis, RMIT, Department of Computer Science, Melbourne, March 1994.

ALVES, Cristiano Araújo Maciel Alves. **Uma ferramenta de extração de regras de redes neurais.** Rio de Janeiro, 2001.

AZEVEDO, Fernando Mendes; BRASIL, Lourdes Mattos; OLIVEIRA, Roberto Célio Limão. **Redes neurais com aplicações em controle e em sistemas especialistas**. Visual Books, 2000.

BAUCHSPIESS, Adolfo, et al. **Servocontrole não linear auto-sintonizado por redes neurais de base radial.** UnB - Brasília.

BITTENCOURT, João Ricardo. **Processamento de imagens inteligente usando redes neurais artificiais.** São Leopoldo, 2000.

BRAGA, Antônio; LUDERMIR, Teresa; CARVALHO, André. **Redes neurais artificiais: teoria e aplicações**. LTC, 2000.

DROUHARD, J. P., SABOURIN, R., GODBOUT, M. **A neural network approach to off-line signature verification using directional PDF**. Pattern Recognition, V29, N3. pp-415-424. Elsevier Science Ltd., 1996.

FAUSETT, Laurene. **Fundamentals of neural networks: architectures, algorithms and applications**. Prentice Hall, 1994.

FERREIRA, Helder Filipe Patrício Cabral; PEREIRA, Paulo César Carvalho. **Software para interpolação espacial de imagens.** FEUP, 2002.

FILHO, Adhemar M. Valle, et al. **Uma visão geral sobre rede neural com arquitetura art2.** UFSC, Santa Catarina.

FOGEL, A. Michalewicz, et al. **Handbook of Evolutionary Computation**. Oxford University Press, 1997.

HANSELMAN, Duane; LITTLEFIELD, Bruce. **Matlab versão do estudante guia do usuário**. Makron Books, 1997.

HEINEN, Milton Roberto. **Autenticação on-line de assinaturas utilizando redes neurais**. São Leopoldo, 2002.

JAIN, Anil; MAO, Jianchang; MOCHIUDDIN, K. M. **As redes neurais artificiais: arquitetura e aprendizado.** 2000.

KARRER, Daniel, et al. **Redes neurais artificiais: conceição e aplicações.**

LAGO, André Elder da Rocha. **SiRA – sistema de reconhecimento de assinaturas**. Uruguaiana, 2005.

MATIAS, Caio Rafael Silva. **Protótipo de um sistema de identificação do(s) delta(s) e núcleo em impressões digitais utilizando redes neurais artificiais**. Blumenau, 2004.

NETO, Hugo Vieira. **Reconhecimento automático de impressões digitais utilizando *wavelets* e redes neuronais artificiais.** CEFET-PR, Curitiba, 1998.

OSÓRIO, Fernando. **Redes neurais artificiais: histórico, modelos e aprendizagem neural**. Unisinos, 2001/2002.

OSÓRIO, Fernando. **Redes neurais artificiais: simulação: base de dados, codificação, parâmetros; aprendizado: generalização, avaliação e resultados**. Unisinos, 2001/2002.

OSÓRIO, Fernando. **Redes neurais artificiais: MLP: modelos avançados – acelerando o back-propagation; modelos: rprop, quickprop, cascade-correlation**. Unisinos, 2001/2002.

SILVA, Eugenio; THOMÉ, Antonio C. G. **Time de redes neurais aplicado ao problema do reconhecimento de caracteres manuscritos.** UFRJ, Rio de Janeiro, 2002

SILVA, Eugenio, et al. **Extração de características para o reconhecimento de letras manuscritas.** UFRJ - Rio de Janeiro.

SILVA, Luis Carlos Ferreira. **Inteligência computacional para predições de produção de reservatórios de petróleo.** Rio de Janeiro, 2006.

VIEIRA, José Manuel Neto. **Matlab num instante.** Aveiro, 2004.

ZUBEN, Fernando J. Von; ATTUX, Romis R. F. **Redes neurais com funções de ativação de base radial.**

.

ANEXO 1 – Código Fonte

O sistema foi totalmente implementado utilizando a linguagem Matlab e seus toolboxes de processamento de imagens e redes neurais. Para a interface gráfica construída utilizou-se a ferramenta GUI do próprio Matlab.

Segue abaixo os códigos fonte de todos os programas desenvolvidos no Matlab. Os programas estão divididos por níveis, sendo o programa da interface gráfica o de nível mais alto, pois faz a chamada de todos os outros sub-programas (scripts) e sub-programas como processamento de imagens sendo de nível mais baixo.

Código fonte correspondente ao programa Interface, que faz a chamada de todos os sub-programas responsáveis pelo funcionamento da rede neural, além da criação do formato visual do sistema:

```
function varargout = interface(varargin)
if nargin == 0  % LAUNCH GUI
        fig = openfig(mfilename,'reuse');
        % Generate a structure of handles to pass to callbacks, and store it.
        handles = guihandles(fig);
        guidata(fig, handles);
        if nargout > 0
                varargout{1} = fig;
        end


elseif ischar(varargin{1}) % INVOKE NAMED SUBFUNCTION OR CALLBACK
        try
                [varargout{1:nargout}] = feval(varargin{:}); % FEVAL switchyard
        catch
                disp(lasterr);
        end
```

```
end


% --------------------------------------------------------------------
function varargout = TreinarR_Callback(h, eventdata, handles, varargin)
% Stub for Callback of the uicontrol handles.pushbutton1.
rede_neural;

% --------------------------------------------------------------------
function varargout = Treinar_Callback(h, eventdata, handles, varargin)
% Stub for Callback of the uicontrol handles.pushbutton1.
load ass;

P = [ass];
T = get(handles.target,'String');

T01 =  [1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0];
T02 =  [0; 1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0];
T03 =  [0; 0; 1; 0; 0; 0; 0; 0; 0; 0; 0; 0];
T04 =  [0; 0; 0; 1; 0; 0; 0; 0; 0; 0; 0; 0];
T05 =  [0; 0; 0; 0; 1; 0; 0; 0; 0; 0; 0; 0];
T06 =  [0; 0; 0; 0; 0; 1; 0; 0; 0; 0; 0; 0];
T07 =  [0; 0; 0; 0; 0; 0; 1; 0; 0; 0; 0; 0];
T08 =  [0; 0; 0; 0; 0; 0; 0; 1; 0; 0; 0; 0];
T09 =  [0; 0; 0; 0; 0; 0; 0; 0; 1; 0; 0; 0];
T10 = [0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 0; 0];
T11 = [0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 0];
T12 = [0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1];

epoch = get(handles.epoch,'String');
epoch = str2num(epoch);

goal = get(handles.goal,'String');
goal = str2num(goal);

lr = get(handles.lr,'String');
lr = str2num(lr);

mc = get(handles.mc,'String');
mc = str2num(mc);

show = get(handles.show,'String');
show = str2num(show);

algoritmo = get(handles.alg,'Value');

if algoritmo == 2
net = newff([minmax(P)],[1 12],{'logsig' 'purelin'},'traingd','learngd','mse');
```

```
elseif algoritmo == 3
net = newff([minmax(P)],[1 12],{'logsig' 'purelin'},'traingdm','learngd','mse');
elseif algoritmo == 4
net = newff([minmax(P)],[1 12],{'logsig' 'purelin'},'traingda','learngd','mse');
elseif algoritmo == 5
net = newff([minmax(P)],[1 12],{'logsig' 'purelin'},'traingdx','learngd','mse');
elseif algoritmo == 6
net = newff([minmax(P)],[1 12],{'logsig' 'purelin'},'trainlm','learngd','mse');
else
net = newff([minmax(P)],[1 12],{'logsig' 'purelin'},'trainrp','learngd','mse');
end



net.trainParam.epochs = epoch;
net.trainParam.goal = goal;
net.trainParam.lr = lr;
net.trainParam.mc = mc;
net.trainParam.show = show;


if T == 'T01'
    net = train(net,P,T1);
elseif T == 'T02'
    net = train(net,P,T2);
elseif T == 'T03'
    net = train(net,P,T3);
elseif T == 'T04'
    net = train(net,P,T4);
elseif T == 'T05'
    net = train(net,P,T5);
elseif T == 'T06'
    net = train(net,P,T6);
elseif T == 'T07'
    net = train(net,P,T7);
elseif T == 'T08'
    net = train(net,P,T8);
elseif T == 'T09'
    net = train(net,P,T9);
elseif T == 'T10'
    net = train(net,P,T10);
elseif T == 'T11'
   net = train(net,P,T11);
elseif T == 'T12'
    net = train(net,P,T12);
end

Y = sim(net,ass);
resultado;
% --------------------------------------------------------------------
```

```
function varargout = top_Callback(h, eventdata, handles, varargin)
% Stub for Callback of the uicontrol handles.pushbutton1.

N = imread('_rede_neural_.jpg');
imshow(N);
% --------------------------------------------------------------------
function varargout = Simular_Callback(h, eventdata, handles, varargin)
% Stub for Callback of the uicontrol handles.pushbutton2.
load net;
load ass;

Y = sim(net,ass);

resultado;


% --------------------------------------------------------------------


function varargout = ass1_Callback(h, eventdata, handles, varargin)
% Stub for Callback of the uicontrol handles.ass1.
%Leitura da imagem pelo diretorio
[filename, pathname] = uigetfile('*.bmp;*.jpg;*.gif;*.tif', 'Seleceione uma assinatura');
img_path = sprintf('%s%s', pathname, filename);
ass = imread(img_path);

ass = imresize(ass,[48 128]);
imshow(ass);
normal = ass;

ass = im2bw(ass,0.3);
binario = ass;


ass = edge(ass);
ass = double(ass);
borda = ass;


i = inline('sum(x(:))');
ass = blkproc(ass,[12 16],i);
ass = ass(:);

save ass ass normal binario borda;

% --------------------------------------------------------------------
function varargout = normal_Callback(h, eventdata, handles, varargin)
% Stub for Callback of the uicontrol handles.edit1.
load ass;
imshow(normal);
```

```
% --------------------------------------------------------------------
function varargout = binarizar_Callback(h, eventdata, handles, varargin)
% Stub for Callback of the uicontrol handles.edit1.
load ass;
imshow(binario);

% --------------------------------------------------------------------
function varargout = bordas_Callback(h, eventdata, handles, varargin)
% Stub for Callback of the uicontrol handles.edit1.
load ass;
imshow(borda);

% --------------------------------------------------------------------
function varargout = resultado_Callback(h, eventdata, handles, varargin)
% Stub for Callback of the uicontrol handles.edit1.
```

O nível abaixo do nível de interface é o programa Rede_Neural, que é responsável pela criação e treinamento da rede neural desenvolvida. Esse programa faz a chamada do sub-programa Processamento para o uso das assinaturas como entrada para a rede neural. Também é tarefa deste programa a criação de um arquivo ajustar_pesos e net, que são os valores dos pesos e a própria rede neural. Seu código fonte é mostrado abaixo:

```
%PRE-PROCESSAMENTO
%Primeiro e chamado o sub-programa responsavel pelo processamento das imagens.

processamento;

%ENTRADAS E SAIDAS DESEJADAS

% Saidas desejadas para cada classe.
T1 =  [1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0];
T2 =  [0; 1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0];
T3 =  [0; 0; 1; 0; 0; 0; 0; 0; 0; 0; 0; 0];
T4 =  [0; 0; 0; 1; 0; 0; 0; 0; 0; 0; 0; 0];
T5 =  [0; 0; 0; 0; 1; 0; 0; 0; 0; 0; 0; 0];
T6 =  [0; 0; 0; 0; 0; 1; 0; 0; 0; 0; 0; 0];
T7 =  [0; 0; 0; 0; 0; 0; 1; 0; 0; 0; 0; 0];
T8 =  [0; 0; 0; 0; 0; 0; 0; 1; 0; 0; 0; 0];
T9 =  [0; 0; 0; 0; 0; 0; 0; 0; 1; 0; 0; 0];
T10 = [0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 0; 0];
T11 = [0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1; 0];
T12 = [0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1];
```

```
% vetores de entrada: conjunto de vetores das assinaturas
P = [[ass1] [ass2] [ass3] [ass4] [ass5] [ass6] [ass7] [ass8] [ass9] [ass10] [ass11] [ass12] [ass13] [ass14] [ass15] [ass16] [ass17] [ass18] [ass19] [ass20] [ass21] [ass22] [ass23] [ass24] [ass25] [ass26] [ass27] [ass28] [ass29] [ass30] [ass31] [ass32] [ass33] [ass34] [ass35] [ass36]];

% targets para P
T = [[T1] [T1] [T1] [T2] [T2] [T2] [T3] [T3] [T3] [T4] [T4] [T4] [T5] [T5] [T5] [T6] [T6] [T6] [T7] [T7] [T7] [T8] [T8] [T8] [T9] [T9] [T9] [T10] [T10] [T10] [T11] [T11] [T11] [T12] [T12] [T12]];



% CRIACAO DA REDE NEURAL.

% A funçao da rede e receber como entrada a imagem processada da assinatura e como resposta classifica-la de acordo com sua classe.


% Topologia da Rede: 36 neuronios na camada de entrada tendo os dados da imagem processada como entrada.
% 1 camada intermediaria para melhor estimativa do erro, com 14 neuronios para melhor generalizacao.
% 12 neuronios na saida correspondente ao numero de classes de usuario.
% Algoritmos de Treinamento: R-Prop.
% Erro calculado atraves do erro quadrico medio.

net = newff([minmax(P)],[14 12],{'logsig' 'purelin'},'trainrp','learngd','mse');

%Parametros de treinamento
net.trainParam.epochs = 5000;
net.trainParam.goal = 0.0001;
net.trainParam.lr = 0.1;
net.trainParam.mc = 0.3;
net.trainParam.show = 100;
net.trainParam.mu = 1;
net.trainParam.mu_dec = 0.8;
net.trainParam.mu_inc = 1.5;


%TREINAMENTO DA REDE:

%Carrega os valores dos pesos e bias do treinamento da rede.
load ajustar_pesos;
net.iw = pesosI;
net.lw = pesosL;
net.b = bias;

%A rede e treinada por batch (pesos ajustados apos todos os padroes serem apresentados)
```

```
%O algoritmo de R-prop e uma variacao do backpropagation que acelera o tempo de treinamento usando apenas o sinal da derivada.

net = train(net,P,T);


%ARMAZENAMENTO DO APRENDIZADO DA REDE

%Os pesos e bias da rede apos treinada, sao armazenados em variaveis e exportadas para um arquivo chamado ajustar_pesos
pesosI = net.iw;
pesosL = net.lw;
bias = net.b;

save net net;

save ajustar_pesos pesosI pesosL bias;
```

O sub-programa Processamento realiza o pré-processamento necessário em cada imagem. Partes do código fonte é mostrado abaixo exemplificando para dois usuários:

```
%O Pre-processamento esta divido nestas etapas:
% 1 – redimensionar a imagem para 48x128
% 2 – binarizar imagem para padronizar os valores em 0 e 1
% 3 – detectar bordas da imagem
% 4 – dividir a imagem em blocos na     roporção de 12x16
% 5 – somar numero de elementos correspondentes a assinatura em cada bloco

ass = imread('evandro pontelli 1.jpg');
ass = imresize(ass,[48 128]);

ass = im2bw(ass,0.3);

ass = edge(ass);
ass = double(ass);

i = inline('sum(x(:)');
ass = blkproc(ass,[12 16],i);
ass = ass(:);

ass1 = ass;
%-------------------------
ass = imread('evandro pontelli 2.jpg');
ass = imresize(ass,[48 128]);

ass = im2bw(ass,0.3);
```

```
ass = edge(ass);
ass = double(ass);

i = inline('sum(x(:)');
ass = blkproc(ass,[12 16],i);
ass = ass(:) ;

ass2 = ass;
%-------------------------
ass = imread('evandro pontelli 3.jpg');
ass = imresize(ass,[48 128]);

ass = im2bw(ass,0.3);

ass = edge(ass);
ass = double(ass);

i = inline('sum(x(:)');
ass = blkproc(ass,[12 16],i);
ass = ass(:);

ass3 = ass;
%-------------------------
%-------------------------
ass = imread('fabio barros tebaldi 1.jpg');
ass = imresize(ass,[48 128]);

ass = im2bw(ass,0.3);

ass = edge(ass);
ass = double(ass);

i = inline('sum(x(:)');
ass = blkproc(ass,[12 16],i);
ass = ass(:);

ass4 = ass;
%-------------------------
ass = imread('fabio barros tebaldi 2.jpg');
ass = imresize(ass,[48 128]);

ass = im2bw(ass,0.3);

ass = edge(ass);
ass = double(ass);

i = inline('sum(x(:)');
ass = blkproc(ass,[12 16],i);
ass = ass(:);
```

```
ass5 = ass;
%-------------------------
ass = imread('fabio barros tebaldi 3.jpg');
ass = imresize(ass,[48 128]);

ass = im2bw(ass,0.3);

ass = edge(ass);
ass = double(ass);

i = inline('sum(x(:)');
ass = blkproc(ass,[12 16],i);
ass = ass(:);

ass6 = ass;
%-------------------------
```

O ultimo nível do sistema caracteriza-se pelo sub-programa chamado Resultado. Este sub-programa faz a analise da resposta gerada pela rede neural e a transforma em uma String de forma a caracterizar a quem pertence a assinatura usada como entrada para a rede. Seu código fonte é disponibilizado abaixo:

```
m = 0;
maior = 0;
i =1;

while i < 13
   if Y(i) > m
      m = Y(i);
      maior = i;
   end
   i = i+1;
end


if maior == 1
   msg = 'Assinatura Pertence ao usuario(a) Evandro Pontelli';
   set(handles.resultado,'String',msg);
elseif maior == 2
   msg = 'Assinatura Pertence ao usuario(a) Fabio Barros Tebaldi';
   set(handles.resultado,'String',msg);
elseif maior == 3
   msg = 'Assinatura Pertence ao usuario(a) Fernanda Ferreira';
```

```
    set(handles.resultado,'String',msg);
elseif maior == 4
    msg = 'Assinatura Pertence ao usuario(a) Filipe Pereira';
    set(handles.resultado,'String',msg);
elseif maior == 5
    msg = 'Assinatura Pertence ao usuario(a) Jose Carlos Tebaldi';
    set(handles.resultado,'String',msg);
elseif maior == 6
    msg = 'Assinatura Pertence ao usuario(a) Marcel Mello';
    set(handles.resultado,'String',msg);
elseif maior == 7
    msg = 'Assinatura Pertence ao usuario(a) Mariana Falcao';
    set(handles.resultado,'String',msg);
elseif maior == 8
    msg = 'Assinatura Pertence ao usuario(a) Marlene Barros de Melo Tebaldi';
    set(handles.resultado,'String',msg);
elseif maior == 9
    msg = 'Assinatura Pertence ao usuario(a) Natalia Pereira';
    set(handles.resultado,'String',msg);
elseif maior == 10
    msg = 'Assinatura Pertence ao usuario(a) Renan Ferrari';
    set(handles.resultado,'String',msg);
elseif maior == 11
    msg = 'Assinatura Pertence ao usuario(a) Rodrigo Espinosa';
    set(handles.resultado,'String',msg);
else
    msg = 'Assinatura Pertence ao usuario(a) Vinicius Tanure';
    set(handles.resultado,'String',msg);
end
```