FUNDAÇÃO DE ENSINO “EURÍPIDES SOARES DA ROCHA”
Mantenedora do Centro Universitário Eurípides de Marília – UNIVEM
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

MARCOS ROGÉRIO FERRARA

UTILIZAÇÃO DE REDES NEURAIS ARTIFICIAIS PARA A IDENTIFICAÇÃO DO DIABETES MELLITUS

MARÍLIA
2005

MARCOS ROGÉRIO FERRARA

UTILIZAÇÃO DE REDES NEURAIS ARTIFICIAIS PARA A IDENTIFICAÇÃO DO DIABETES MELLITUS

Monografia apresentada ao Curso de Ciência da Computação, da Fundação de Ensino “Eurípides Soares da Rocha”, Mantenedora do Centro Universitário Eurípides de Marília – UNIVEM, como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação.

Orientador:
Prof. Dr. José Celso Rocha.

MARÍLIA
2005

FERRARA, Marcos Rogério
Utilização de Redes Neurais Artificiais para a identificação do Diabetes Mellitus./ Marcos Rogério Ferrara; orientador: José Celso Rocha. Marília, SP: [s.n.], 2005.

MONOGRAFIA (GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO)- CURSO DE CIÊNCIA DA COMPUTAÇÃO, FUNDAÇÃO DE ENSINO "EURÍPIDES SOARES DA ROCHA", MANTENEDORA DO CENTRO UNIVERSITÁRIO EURÍPIDES DE MARÍLIA –UNIVEM.

1 REDES NEURAIS ARTIFICIAIS, 2 DIABETES MELLITUS, 3 MATLAB, 4 TOOLBOX DE REDES NEURAIS, 5 APRENDIZADO.

CDD: 006.32

MARCOS ROGÉRIO FERRARA
RA Nº 300901

UTILIZAÇÃO DE REDES NEURAIS ARTIFICIAIS PARA A IDENTIFICAÇÃO DO DIABETES MELLITUS

BANCA EXAMINADORA DA MONOGRAFIA PARA
OBTENÇÃO DO GRAU DE BACHAREL EM CIÊNCIA DA COMPUTAÇÃO

CONCEITO FINAL: _____(_____)

ORIENTADOR: _____
Prof. Dr. José Celso Rocha

1º EXAMINADOR: _____

2º EXAMINADOR: _____

Marília, _____de _____de 2005

FERRARA, Marcos Rogério Ferrara. Utilização de Redes Neurais Artificiais para a identificação do Diabetes Mellitus. Monografia (Bacharelado em Ciência da Computação) – Curso de Ciência da Computação, Fundação de Ensino "Eurípides Soares da Rocha", Mantenedora do Centro Universitário Eurípides de Marília – UNIVEM.

RESUMO

ESTE TRABALHO APRESENTA UMA INTRODUÇÃO ÀS REDES NEURAIS ARTIFICIAIS COMO UMA ALTERNATIVA PARA A IDENTIFICAÇÃO DO DIABETES MELLITUS ATRAVÉS DE EXAMES REALIZADOS EM PACIENTES DA ALDEIA INDÍGENA DE PIMA. INICIALMENTE É FEITA UMA RÁPIDA REVISÃO BIBLIOGRÁFICA SOBRE REDES NEURAIS ARTIFICIAIS, ONDE SÃO APRESENTADOS OS MODELOS DE NEURÔNIO, PERCEPTRON, AS REDES DE RETROPROPAGAÇÃO BEM COMO UM RESUMO DE SEU ALGORITMO. EM SEGUIDA É APRESENTANDO O SOFTWARE MATLAB QUE SERÁ UTILIZADO NA IMPLEMENTAÇÃO DA APLICAÇÃO, NESTE TÓPICO SERÁ DESCRITO COMO UTILIZAR SEU TOOLBOX DE REDES NEURAIS ARTIFICIAIS. A PARTIR DESTAS CONSIDERAÇÕES, ESTE TRABALHO TERÁ COMO FINALIDADE O TREINAMENTO DE UMA REDE NEURAL ARTIFICIAL PARA O RECONHECIMENTO DE EXAMES PARA DIAGNOSTICAR A PRESENÇA DO DIABETES EM PACIENTES. O MÉTODO DE SE IDENTIFICAR A DOENÇA, LOCAL ONDE EFETUADO A PESQUISA E A DESCRIÇÃO DE CADA ITEM TAMBÉM SÃO TRATADOS. COM BASE NESTAS CONSIDERAÇÕES FOI TREINADA UMA REDE NEURAL ARTIFICIAL E APLICADO SOBRE ELA TESTES DE EFICIÊNCIA E OPERACIONALIDADE PARA CHEGAR AO PONTO DE SEU MAIOR DESEMPENHO.

Palavras-chave: Redes Neurais Artificiais, Diabetes Mellitus, MatLab, TOOLBOX de Redes Neurais, Aprendizado

FERRARA, Marcos Rogério Ferrara. Utilização de Redes Neurais Artificiais para a identificação do Diabetes Mellitus. Monografia (Bacharelado em Ciência da Computação) – Curso de Ciência da Computação, Fundação de Ensino "Eurípides Soares da Rocha", Mantenedora do Centro Universitário Eurípides de Marília – UNIVEM.

ABSTRACT

THIS WORK PRESENTS AN INTRODUCTION TO NEURAL ARTIFICIAL NETWORKS AS AN ALTERNATIVE TO THE IDENTIFICATION OF DIABETES MELLITUS THROUGH EXAMINATIONS CARRIED IN PATIENTS OF THE PIMA´S ABORIGINAL VILLAGE. INITIALLY A FAST BIBLIOGRAPHICAL REVISION ON NEURAL ARTIFICIAL NETWORKS IS MADE, WHERE THE NEURON MODELS ARE PRESENTED, PERCEPTRON, THE BACKPROPAGATION NETWORKS AS WELL AS A SUMMARY OF ITS ALGORITHM. AFTER THAT IT IS PRESENTING THE MATLAB SOFTWARE THAT WILL BE USED IN THE IMPLEMENTATION OF THE APPLICATION, IN THIS TOPIC WILL BE DESCRIBED AS TO USE ITS TOOLBOX OF NEURAL ARTIFICIAL NETWORKS. TO LEAVE OF THESE CONSIDERATIONS, THIS WORK WILL HAVE AS PURPOSE THE TRAINING OF A NEURAL ARTIFICIAL NETWORK FOR THE RECOGNITION OF EXAMINATIONS TO DIAGNOSIS THE PRESENCE OF DIABETES IN PATIENTS. THE METHOD OF IDENTIFYING THE ILLNESS, PLACE WHERE EFFECTED THE RESEARCH AND THE DESCRIPTION OF EACH ITEM ARE TREATED TOO. WITH BASE IN THESE CONSIDERATIONS A NEURAL ARTIFICIAL NETWORK AND APPLIED ON IT WAS TRAINED TESTS OF EFFICIENCY AND OPERATIONALIZATION TO ARRIVE AT THE POINT OF ITS BIGGER PERFORMANCE.

Keywords: Neural Artificial Networks, Diabetes Mellitus, MatLab, TOOLBOX of Neural Artificial Networks, Learning

SUMÁRIO

LISTA DE FIGURAS

INTRODUÇÃO

O estudo das Redes Neurais Artificiais é algo fascinante e esse fascínio aumenta à medida que se tem mais conhecimento sobre o assunto. Trata-se de um conceito de extrema importância na computação, responsável pela solução de muitos problemas complexos.

A solução de problemas através de Redes Neurais Artificiais é bastante interessante, tanto pela forma de como estes problemas são representados internamente pela rede como também pelos resultados que gera, pois podem chegar a apresentar um desempenho superior ao dos modelos tradicionais. Em Redes Neurais Artificiais, o procedimento usual na solução de problemas passa inicialmente por uma fase de *aprendizagem*, em que um conjunto de exemplos padrões é apresentado para a rede, que extrai automaticamente dos mesmos as características necessárias para representar a informação fornecida. Essas características são utilizadas posteriormente para gerar respostas a problemas com características similares às dos exemplos.

As possibilidades de *aprender* através de exemplos e de *generalizar* a informação aprendida são, sem dúvida, os atrativos principais na solução de problemas através de RNAs. A generalização, que está associada à capacidade da rede aprender através de um conjunto reduzido de exemplos e posteriormente dar respostas coerentes para dados não conhecidos, é uma demonstração de que a capacidade das Redes Neurais Artificiais vai muito além do que simplesmente mapear relações de entrada e saída. As Redes Neurais Artificiais são capazes de extrair informações não apresentadas de forma explícita, através dos exemplos e assimilar estas entradas a uma resposta plausível.

No entanto, mesmo sendo razoavelmente conhecida, somente nesta última década as Redes Neurais Artificiais passaram a ser utilizada com mais freqüência no auxilio de soluções

em várias áreas do conhecimento. De acordo com os resultados encontrados na literatura, esta técnica parece ter se mostrado adequada também para aplicações na área médica.

Os modelos construídos a partir de Redes Neurais Artificiais, no entanto, são em geral obscuros para o usuário final. Além de não ser fácil entender o processo de aprendizado, em que os dados de entrada tornam-se os resultados de saída, pois os "modelos" em geral dependem do *software* usado no processo de treinamento para produzir estimativas e simulações. Com os procedimentos propostos neste trabalho, espera-se poder reconhecer um resultado de exames de *diabetes mellitus* com um modelo construído a partir de uma Rede Neural Artificial treinada para reconhecer todos os dados contidos nos exames à ela fornecida.

Neste trabalho, como um exemplo do potencial desta integração de RNA com outros recursos computacionais, serão efetuado dentro do software matemático MatLab a fundamentação dos métodos que constituirão em uma Rede Neural Artificial treinada para reconhecer se determinado exame, de uma base de dados, de pacientes apresentarão ou não a presença do diabetes no organismo.

Objetivos do Trabalho

- Desenvolver um procedimento capaz de permitir que uma Rede Neural Artificial desenvolvida no MatLab seja treinada para reconhecer dados de exames de pacientes da Aldeia de Pima, nos Estados Unidos, apresentados em uma base de dados fornecida por um laboratório norte-americano.
- Conduzir algumas análises a partir da alteração dos valores das variáveis de entrada dos modelos considerados, verificando os impactos nos valores de saída estimados.
- Aplicar o modelo estruturado para que futuramente possam ser apresentados novos padrões de entrada que serão reconhecidos como dados de entrada de uma Rede

Neural Artificial já com seu aprendizado definido, como os exames já coletados anteriormente, para saber se estes novos pacientes também são portadores ou não do diabetes mellitus.

Estrutura do Trabalho

Para atingir os objetivos propostos para o trabalho, dois eixos teóricos são fundamentais: um que trata da principal ferramenta explorada na pesquisa, que são as Redes Neurais Artificiais, e outro que se refere à base conceitual do modelo que será utilizado para aplicação dos procedimentos propostos. À luz destas considerações, este trabalho é composto por seis capítulos. O segundo capítulo é a primeira parte da revisão bibliográfica, apresentando uma breve introdução às Redes Neurais Artificiais e seu método mais conhecido de aprendizagem, o algoritmo *backpropagation* o qual será utilizado no treinamento apresentada na rede, dentre outros modelos citados também. O terceiro capítulo apresenta, por sua vez, alguns conceitos de como se utilizar o software MatLab, utilizado na implementação da aplicação para constituir os conceitos básicos para a compreensão do modelo explorado nesta pesquisa.

No quarto capítulo é fundamentado o método empregado para o desenvolvimento do trabalho: a preparação do material básico para dar início à aplicação em questão e a definição dos procedimentos desenvolvidos na seqüência, tais como o método de se identificar todos os atributos contidos na base de dados, como se chegar aos fatores relevantes ao diagnostico, e como foi o método utilizado para se conseguir a coleta de um número razoável de padrões de treinamento para conseguir um aprendizado satisfatório em uma Rede Neural Artificial.

Já no quinto capitulo é mostrado realmente como será feita a implementação da Rede Neural Artificial para o diagnostico de *diabetes mellitus* presente em pacientes mulheres da

aldeia de Pima. Implementada a Rede Neural Artificial será demonstrado a operacionalidade e alguns resultados feito sobre o protótipo desenvolvido.

O sexto capítulo contém as conclusões mais importantes tirada diante de um projeto que envolveu desde a fundamentação teórica em Redes Neurais Artificiais à implementação de um protótipo onde demonstra sua funcionalidade.

Redes Neurais Artificiais

Este capítulo apresenta uma breve introdução sobre os conceitos básicos e um histórico das Redes Neurais Artificiais. Fala-se ainda sobre o neurônio artificial, sobre as diferentes funções de ativação e as diversas topologias das Redes Neurais Artificiais, além dos processos de treinamento. Também se apresenta neste capítulo as definições e noções gerais das redes *Multilayer Perceptron* (MLP), bem como o algoritmo de aprendizado utilizado por estas.

Conceitos Básicos

As Redes Neurais Artificiais (RNAs), caracteriza-se por uma terminologia genérica que abrange uma grande quantidade de arquiteturas e paradigmas, têm como objetivo compreender o funcionamento do cérebro humano e de alguma forma procurar reproduzi-lo. Elas são compostas por elementos de processamento, denominados neurônios.

As RNAs podem ser definidas como sistemas paralelos distribuídos, compostos por unidades de processamento simples (nós) que calculam determinadas funções matemáticas (normalmente não-lineares). Essas unidades geralmente são conectadas por canais de comunicação que estão associados a determinado peso. As unidades fazem operações apenas sobre seus dados locais, que são entradas recebidas pelas suas conexões. O comportamento "inteligente" de uma Rede Neural Artificial vem das interações entre as unidades de processamento da rede. As Redes Neurais Artificiais podem apresentar uma ou mais camadas intermediárias ou escondidas de neurônios. No tipo de rede denominada MLP (*Multilayer Perceptron*) pode-se, por exemplo, implementar qualquer função contínua em uma rede com uma camada intermediária.

Já a utilização de duas camadas intermediárias permite a aproximação de qualquer função. Do ponto de vista prático, as RNAs tem como vantagem o fato de não necessitarem de conhecimentos de especialistas para tomar decisões; elas se baseiam unicamente nos exemplos que lhes são fornecidos: não é necessário informar porque tal situação resultou em tal decisão no passado, ou porque tal decisão resultou em tal conseqüência.

As RNAs podem ser utilizadas na solução de uma grande quantidade de problemas encontrados nas mais diversas áreas de aplicação: classificação, diagnóstico, análise de sinais e de imagens, otimização e controle. As redes têm se mostrado de fácil implementação e robustas no tratamento de dados com ruídos. São eficientes, particularmente, na resolução de problemas em que não se tem formulação analítica, não se tem conhecimento explícito acessível, os dados estão contaminados por ruídos e/ou o próprio problema modifica-se com o passar do tempo (OLIVEIRA, 2000).

Dependendo do tipo de problema ao qual são submetidas, as RNAs tem apresentado um desempenho considerado superior aos métodos estatísticos utilizados para o mesmo fim.

Histórico das Redes Neurais Artificiais

A história das Redes Neurais Artificiais é relativamente recente. Conta com pouco mais de meio século, se considerarmos como pioneiros os trabalhos dos neurofisiologistas Donald Hebb e Karl Lashley, por volta de 1940, quando as primeiras simulações foram feitas com papel e lápis (LOESCH & SARI, 1996). No artigo clássico intitulado *A logical calculus of the ideas immanent in nervous activity* (Um cálculo lógico das idéias imanentes na atividade nervosa), publicado em 1943, McCulloch e Pitts propuseram um modelo simplificado de neurônios biológicos. O modelo baseia-se no fato de que, em dado instante de tempo, o neurônio ou está ativo ou está inativo (LOESCH & SARI, 1996). O trabalho de

McCulloch e Pitts propiciou uma rápida disseminação do entendimento de possíveis modelos neurais, atraindo e influenciando muitos pesquisadores famosos.

Rossenblatt, em seu livro *Principles of neurodynamics* (Princípios da neurodinâmica), em 1958, forneceu várias idéias a respeito dos *Perceptrons*, que são modelos de neurônios baseados nos conceitos de McCulloch e Pitts (LOESCH & SARI, 1996). Uma das principais idéias de Rosenblatt foi a elaboração da arquitetura *backcoupled Perceptron* e do algoritmo *back-coupled error correction algorithm* (algoritmo de correção do erro *back-coupled*), que é capaz da adaptar os pesos de um conjunto de entradas de uma unidade de associação de acordo com uma saída desejada.Widrow e Hoff desenvolveram o ADALINE (*ADAptive LINear Element*) e o MADALINE (*Many* ADALINE) *Perceptron* como dispositivos práticos para resolver tarefas de reconhecimento de padrões. O ADALINE/MADALINE usou saídas analógicas ao invés das binárias originalmente propostas por McCulloch e Pitts.

Minsky e Papert fizeram um estudo cuidadoso desses algoritmos e publicaram, em 1969, seu livro *Perceptrons*. Provaram formalmente que uma rede formada de uma única camada de neurônios, independente do algoritmo de aprendizagem, é capaz de resolver o problema de associação de padrões apenas quando os conjuntos são linearmente separáveis. Estes resultados e observações feitas por Minsky e Papert foram devastadores, e a abordagem conexionista ficou renegada a um plano secundário durante toda a década de 70 e início da década de 80.

A impotência das redes *perceptron* na resolução do problema de associação de padrões para um conjunto de padrões não-linear foi eliminada por Rumelhart, Hinton e Willians. A solução encontrada foi a *Regra Delta Generalizada*, mais conhecida como *Algoritmo de Correção de Erros de Retropropagação*, em 1986, para redes *Perceptron* de várias camadas de neurônios, com entradas e saídas analógicas.

Essas Redes Neurais com várias camadas (*Multilayer Perceptron)* são majoritariamente indicadas como ferramenta de análise devido à sua elevada capacidade de reconhecimento de padrões. Sendo que até aqui se falou sobre o histórico das Redes Neurais, faz-se agora necessário um esclarecimento dos componentes envolvidos nestas, a partir do seu elemento fundamental: o neurônio artificial, que é de certa forma baseado no funcionamento do neurônio biológico.

Neurônio Artificial

O Neurônio Artificial é a unidade fundamental de processamento de uma RNA, o qual recebe uma ou mais entradas, transformando-as em saídas. HAYKIN (2001) escreve que cada entrada tem um peso associado, que determina sua intensidade. O esquema de neurônio artificial pode ser visualizado na Figura 1.

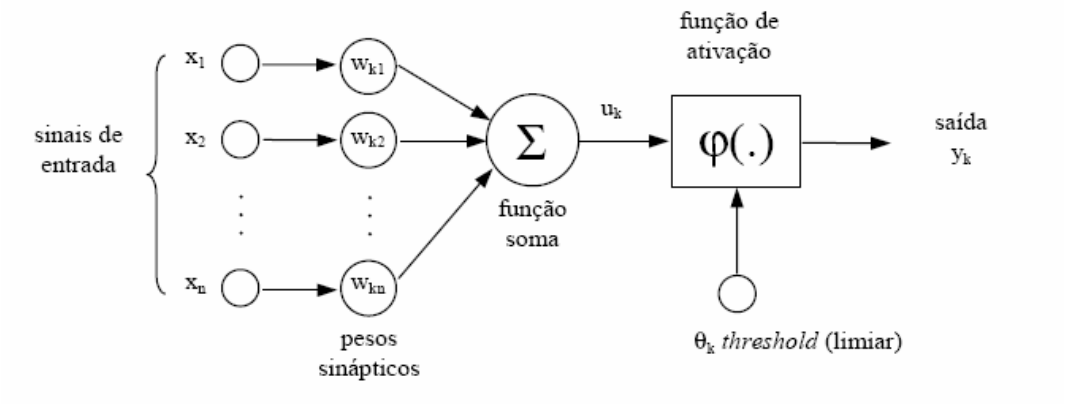

FIGURA 1 – MODELO NÃO LINEAR DE UM NEURÔNIO.

Com base na Figura 1 é possível distinguir alguns elementos considerados importantes na estrutura de um neurônio:

- Sinapses, caracterizadas por um peso (*w*), que pode representar a sua intensidade. O papel do peso w_{ij} é multiplicar o sinal x_j na entrada da sinapse *j*, conectada a um neurônio *k*. O peso w_{ij} é positivo se a sinapse associada é excitatória e negativo se a sinapse associada é inibitória;
- Somatório, adiciona as entradas ponderadas multiplicadas pelos seus pesos respectivos de acordo com a Equação 2.1.

$$u_k = \sum_{j=1}^{n} w_{kj} \cdot x_j \tag{2.1}$$

- Limiar de Excitação (*threshold*), θ_i, tem um papel determinante na saída de neurônio. Sua função é controlar a intensidade da função de ativação para se obter o desempenho desejado na rede. Se o valor de u_i for menor que este limiar, então, a saída do neurônio fica inibida. Caso contrário, o neurônio fica ativo;
- Função de ativação, que funciona como um limitante à amplitude da saída do neurônio, ou seja, a entrada é normalizada dentro de um intervalo fechado, geralmente [0,1] ou [-1,1];
- saída do neurônio, y_i, representado pela Equação 2.2:

$$y_i = \varphi(u_i - \theta_i) \tag{2.2}$$

- onde φ é a função de ativação.

A seguir, serão apresentados alguns tipos de função de ativação.

Funções de Ativação

Segundo HAYKIN (2001), a função de ativação denotada por φ (.) define a saída de um neurônio de acordo com o nível de atividade da sua entrada. Ele mesmo identifica três tipos básicos de função de ativação:

1. *Função Limiar* - para este tipo de função de ativação, mostrada na Equação 2.3, tem-se que:

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

Correspondentemente, a saída do neurônio k que utiliza a função limiar é expressa de acordo com a equação 2.4:

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

Onde v_k, representado pela Equação 2.5, é o nível de atividade interna do neurônio, ou seja, a descrição de sua função de ativação.

$$v_k = \sum_{j=1}^{n} W_{kj} \cdot X_j - \theta_k \tag{2.5}$$

Tal neurônio é citado na literatura como o modelo McCulloch-Pitts, em reconhecimento ao trabalho pioneiro feito por McCulloch e Pitts no ano de 1943. Neste

modelo, a saída do neurônio recebe o valor 1 se o nível total de atividade interna desse neurônio é não negativo, e 0 se é o contrário.

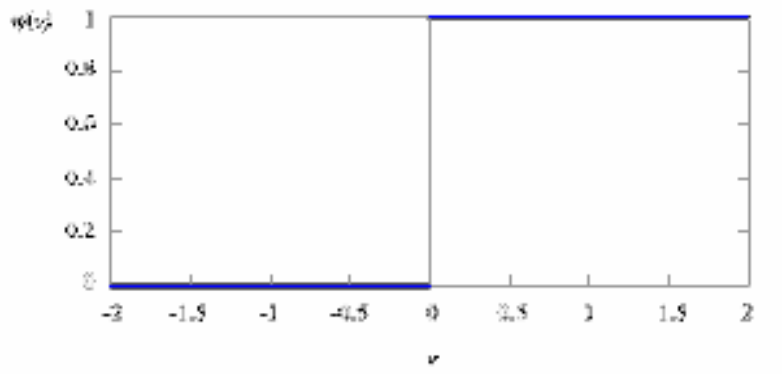

FIGURA 2 - Representação da função limiar.

2. *Função Parcialmente Linear* - no caso desta função, representada graficamente na Figura 3, utiliza-se um valor unitário para apresentar o fator de graduação do sinal de saída em uma região de comportamento linear. Excluindo esta região, a função assume características similares a Função Limiar.

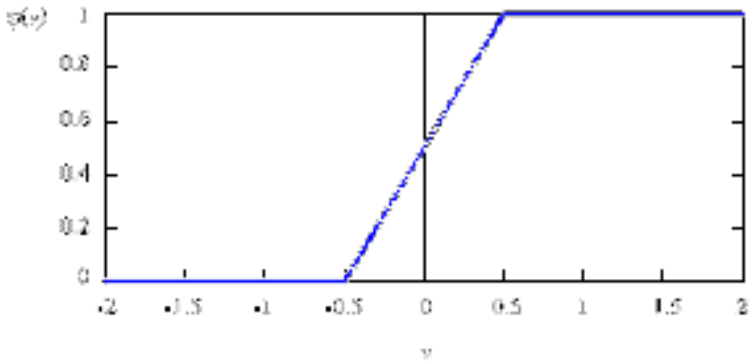

FIGURA 3 - Representação da função parcialmente linear.

3. *Função Sigmóide* - esta é para muitos a forma mais comum de função de ativação usada na construção de Redes Neurais Artificiais. É definida como uma função de caráter

estritamente crescente, que mostra propriedades homogêneas e assintóticas. Um exemplo é a função logística, definida de acordo com a Equação 2.6:

$$\varphi(v) = \frac{1}{1 + exp(-av)} \tag{2.6}$$

Onde *a* é representado pelo parâmetro de inclinação da Função Sigmóide. Variando o parâmetro *a*, são obtidas funções sigmóides de diferentes inclinações, como ilustrado na Figura 4.

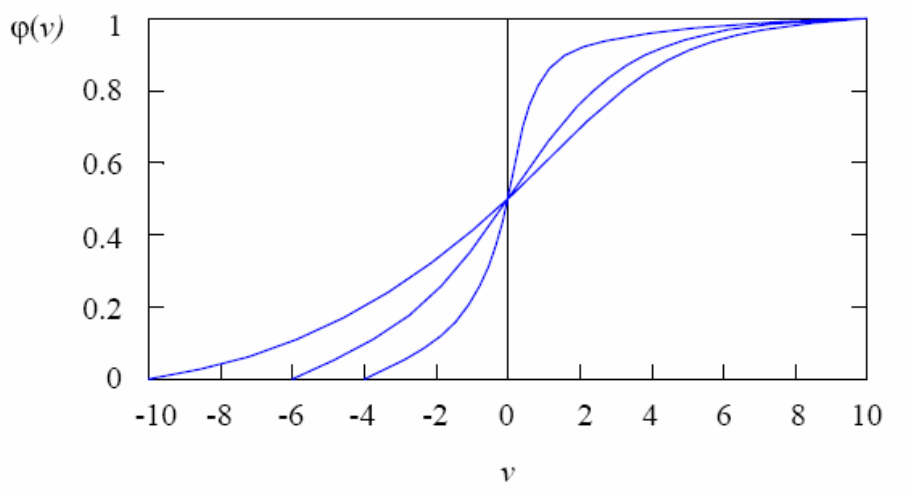

FIGURA 4 - Representação da função sigmóide.

No limite, o parâmetro de inclinação se aproxima tanto do infinito, que a Função Sigmóide torna-se uma Função Limiar. Enquanto que uma função limiar assume o valor de 0 ou 1, uma função sigmóide assume valores em uma faixa contínua entre 0 e 1. Além disso, a função sigmóide é diferenciável, enquanto que a função limiar não.

As funções de ativação definidas nas Equações (2.3), (2.5) e (2.6) estão na faixa de 0 a 1. Algumas vezes é desejável ter a faixa da função de intervalo de -1 a 1, caso em que a função de ativação assume uma forma anti-simétrica com respeito à origem. Especificamente,

a Função Limiar da equação (2.3) pode ser redefinida como na Equação (2.7), geralmente citada como Função Signo.

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

No caso da Função Sigmóide, a equação (2.6) pode ser substituída pela Função Tangente Hiperbólica, definida como em (2.8).

$$\varphi(v) = tg\left(\frac{v}{2}\right) = \frac{1 - exp(-v)}{1 + exp(-v)} \tag{2.8}$$

Assim como a Função Logística, a Função Tangente Hiperbólica também pode assumir valores unipolares (0 e 1) ou bipolares (-1, 1). A diferença entre estes dois tipos de função está no fato de que a Função Hiperbólica se trata de uma Função Logística Modificada.

As várias topologias de redes neurais podem ser agrupadas, basicamente, em duas classes:

- Não-Recorrentes
- Recorrentes

As RNAs não-recorrentes são aquelas que não possuem realimentação de suas saídas para suas entradas e por isso são também ditas "sem memória". A estrutura das RNAs ditas não-recorrentes é em camadas, podendo estas RNAs serem formadas por uma (RNA de camada única) ou mais camadas (RNA multicamadas). Redes Neurais Multicamadas contêm um conjunto de neurônios de entrada, uma camada de saída e uma ou mais camadas escondidas. Segundo WASSERMAN (1989) a entrada não é considerada uma camada da rede, pelo fato de apenas distribuir os padrões. A camada com os neurônios que fornecem a saída da rede é chamada camada de saída.

Na Figura 5 é apresentado um exemplo de uma RNA não recorrente. As RNAs de uma só camada, também chamadas de "perceptrons" não serão tratadas neste trabalho, por possuírem um espectro limitado de representações. As Redes Neurais Multicamadas, por suprirem as deficiências das redes de uma única camada, são utilizadas neste trabalho e serão estudadas com mais detalhes na seção 2.7.

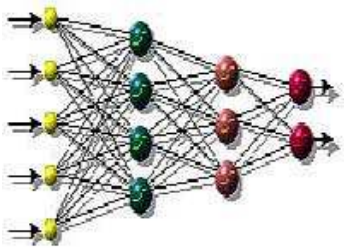

FIGURA 5 - Exemplo de uma RNA não recorrente.

As RNAs recorrentes são redes mais gerais, que contêm realimentação das saídas para as entradas, sendo suas saídas determinadas pelas entradas atuais e pelas saídas anteriores. Além disso, suas estruturas não são obrigatoriamente organizadas em camadas. Quando o são,

estas redes podem possuir interligações entre neurônios da mesma camada e entre camadas não consecutivas, gerando interconexões bem mais complexas que as RNAs não-recorrentes, como mostrado na Figura 6.

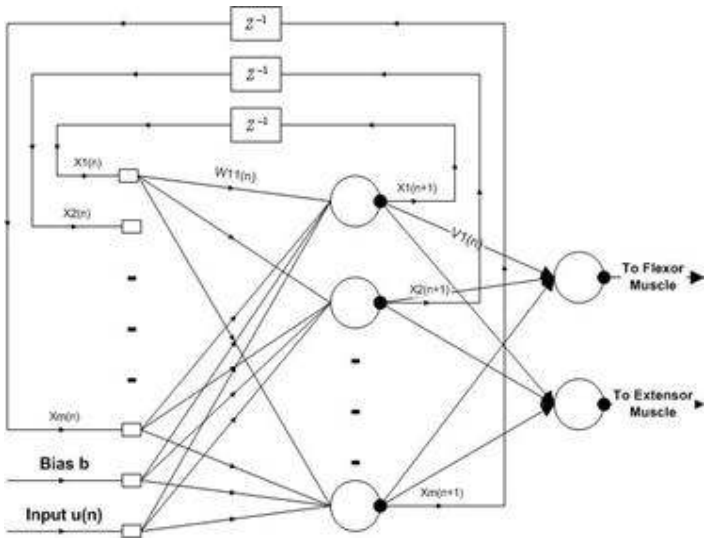

FIGURA 6 - Exemplo de uma RNA recorrente.

Nas RNAs recorrentes os neurônios têm conexões com eles mesmos e com outros neurônios, excitatorios ou inibidores. Depois de um certo intercâmbio de sinais, os neurônios que têm uma saída ativa ganham e permitem aos neurônios do seu mesmo grupo atualizar seus pesos. Para as redes estáveis, este processo é repetido várias vezes, produzindo pequenas mudanças nas saídas, até estas ficarem constantes. Ainda, as RNAs recorrentes nem sempre são estáveis, mesmo com entradas constantes. O fato de não se conseguir prever quais redes seriam estáveis foi um problema que preocupou os pesquisadores até o início da década de 80, quando Cohen e Grossberg provaram um teorema para definir quando as RNAs eram estáveis. Este teorema determina que para as RNAs recorrentes alcançarem um estado estável é necessário que possuam conexões simétricas. Contribuições importantes também foram dadas por HOPFIELD (1984), tanto para a teoria como para a prática, sendo que algumas

configurações passaram a ser chamadas de redes de Hopfield, em sua homenagem. HINTON & SEJNOWSKI (1986) também tiveram méritos neste campo, introduzindo regras gerais de treinamento para redes, denominadas por eles de máquina de Boltzmann, devido ao cálculo das saídas dos neurônios artificiais ser efetuado através de probabilidades segundo a distribuição de Boltzmann.

Tipos de Treinamento

De todas as propriedades interessantes das Redes Neurais Artificiais, nenhuma "captura" tão bem uma característica humana como a habilidade de aprender. O objetivo do treinamento de uma RNA é fazer com que a aplicação de um conjunto de entradas produza um conjunto de saídas desejado ou no mínimo um conjunto de saídas consistentes. Cada conjunto de entrada ou saída é chamado de vetor.

O treinamento é realizado pela aplicação seqüencial dos vetores de entradas (e em alguns casos também os de saída), enquanto os pesos da rede são ajustados de acordo com um procedimento de treinamento pré-determinado. Durante o treinamento, os pesos da rede gradualmente convergem para determinados valores, de maneira tal que a aplicação dos vetores de entrada produza as saídas necessárias. Os procedimentos de treinamento que levam as RNAs a aprender determinadas tarefas podem ser classificados em duas classes de treinamento:

- Supervisionado
- Não Supervisionado

O treinamento *supervisionado* necessita de um par de vetores composto do vetor de entrada e do vetor alvo que se deseja como saída. Juntos, estes vetores são chamados de pares

de treinamentos ou vetor de treinamento, sendo interessante ressaltar que geralmente a rede é treinada com vários vetores de treinamento.

O procedimento de treinamento funciona da seguinte forma: o vetor de entrada é aplicado. A saída da rede é calculada e comparada com o correspondente vetor alvo. O erro encontrado é então realimentado através da rede e os pesos são atualizados de acordo com um algoritmo determinado a fim de minimizar este erro. Este processo de treinamento é repetido até que o erro para os vetores de treinamento tenha alcançado níveis bem baixos.

O treinamento *não supervisiona*do, por sua vez, não requer vetor alvo para as saídas e, obviamente, não faz comparações para determinar a resposta ideal. O conjunto de treinamento modifica os pesos da rede de forma a produzir saídas que sejam consistentes, isto é, tanto a apresentação de um dos vetores de treinamento, como a apresentação de um vetor que é suficientemente similar, irão produzir o mesmo padrão nas saídas. O processo de treinamento extrai as propriedades estatísticas do conjunto de treinamento e agrupa os vetores similares em classes. A aplicação de um vetor de uma determinada classe à entrada da rede irá produzir um vetor de saída específico, mas não existe maneira de se determinar, antes do treinamento, qual o padrão que será produzido na saída para um vetor de entrada de uma determinada classe. Desta forma, a saída de algumas RNAs deve ser transformada em uma forma compreensível após o processo de treinamento, o que é um simples problema de identificação das relações entrada-saída estabelecidas pela rede.

No que diz respeito aos algoritmos de treinamento usados, existe uma grande variedade, tanto para o treinamento supervisionado, como para o não supervisionado. Entre esses, um dos mais difundidos com certeza é o algoritmo utilizado neste trabalho, conhecido por algoritmo *backpropagation* (retro-propagação).

Redes Multilayer Perceptron

As arquiteturas do tipo *perceptron* de múltiplas camadas (MLP) constituem os modelos neurais artificiais mais utilizados e conhecidos atualmente. Tipicamente, esta arquitetura consiste de um conjunto de unidades sensoriais que formam uma camada de entrada, uma ou mais camadas intermediárias (ou escondidas) de unidades computacionais e uma camada de saída. Os sinais de entrada são propagados camada a camada pela rede em uma direção positiva, ou seja, da entrada para a saída. Esta arquitetura representa uma generalização do *perceptron* apresentado anteriormente.

As redes do tipo MLP tem sido utilizadas com sucesso para a solução de vários problemas envolvendo altos graus de não-linearidade. Seu treinamento é do tipo supervisionado e utiliza um algoritmo muito popular chamado retro-propagação do erro *(error backpropagation*). Este algoritmo é baseado numa regra de aprendizagem que "corrige" o erro durante o treinamento (HAYKIN, 2001).

Basicamente, o processo de retro-propagação do erro é constituído de duas fases: uma fase de propagação do sinal funcional *(feedforward*) e uma de retropropagação do erro *(backpropagation*). Na fase positiva, os vetores de dados são aplicados às unidades de entrada, e seu efeito se propaga pela rede, camada a camada. Finalmente, um conjunto de saídas é produzido como resposta da rede. Durante a fase positiva, os pesos das conexões são mantidos fixos. Na retro-propagação do erro, por outro lado, os pesos são ajustados de acordo com uma regra de correção do erro.

Especificamente, a resposta da rede em um instante de tempo é subtraída da saída desejada *(target*) para produzir um *sinal de erro*. Este sinal de erro é propagado da saída para a entrada, camada a camada, originando o nome "retro-propagação do erro". Os pesos são

ajustados de forma que a “distância” entre a resposta da rede e a resposta desejada seja reduzida.

NOÇÕES GERAIS

A Figura 7 apresenta uma arquitetura do tipo MLP com duas camadas intermediárias. A rede apresentada aqui possui todas as conexões, o que significa que um neurônio em qualquer camada da rede está conectado a todas as outras unidades (neurônios) na camada anterior. O fluxo de sinais através da rede é feito positivamente, da esquerda para a direita, camada a camada.

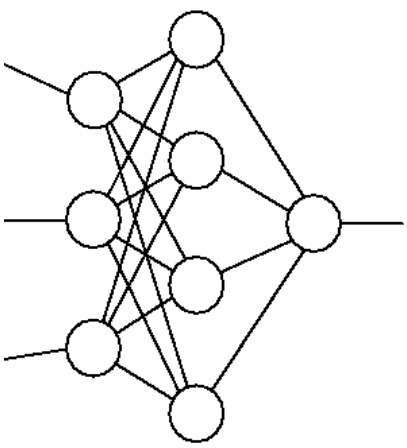

FIGURA 7 - Arquitetura MLP com uma camada intermediária.

A Figura 8 mostra apenas uma parte da rede. Nesta rede, dois tipos de sinais podem ser identificados:

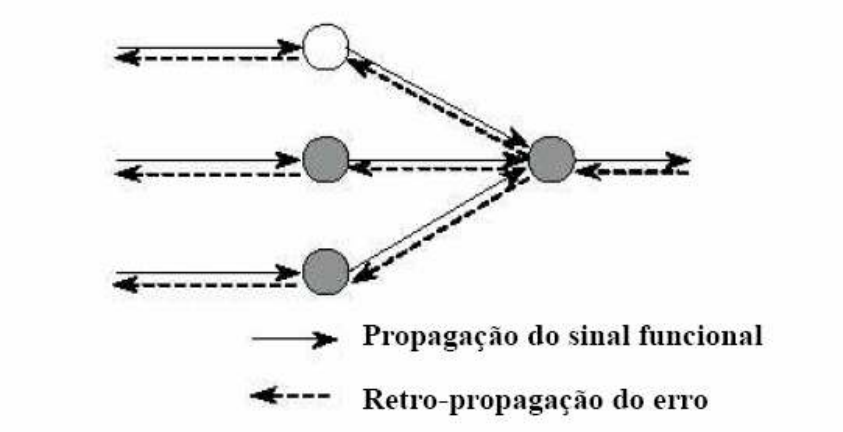

FIGURA 8 - Ilustração das direções de propagação.

- *Sinal funcional*: um sinal funcional é um sinal de entrada (estímulo) que chega na entrada e é propagado positivamente (neurônio a neurônio) através da rede, e aparece na saída como um sinal de saída.
- *Sinal de err*o: os sinais de erro originam-se nas saídas e são retro-propagados (neurônio a neurônio) através da rede.

A camada de entrada geralmente é composta por neurônios sensoriais, ou seja, unidades que não modificam os sinais externos, apenas os distribuem para a primeira camada intermediária. As unidades de saída constituem a camada de saída da rede e as demais unidades constituem as camadas intermediárias. As camadas intermediárias são todas aquelas que não fazem parte da entrada e nem da saída. Cada unidade intermediária ou de saída é responsável por duas tarefas:

- Calcular o sinal na saída da unidade, que geralmente é expresso como uma função não-linear do sinal de entrada e pesos sinápticos associados; e
- Calcular uma estimativa instantânea do vetor gradiente, que é necessário para a retro-propagação do erro através da rede.

O gradiente é um vetor que nos informa a direção em que devemos movimentar o espaço de pesos para reduzir w_{ij}. A fim de encontrar um vetor de peso de solução, simplesmente mudamos os pesos na direção do gradiente, recalculamos w_{ij}, recalculando assim o novo gradiente e iteramos até a regra conseguir convergir à deixar todos os pesos ponderados.

ALGORITMO DE APRENDIZADO PARA UMA REDE MLP

A regra de aprendizado utilizada para treinar uma rede *Multilayer Perceptron* é denominada Regra Delta Generalizada ou, mais comumente, *backpropagation*, já discutida anteriormente. O algoritmo *backpropagation* é primordial para muitos trabalhos atuais sobre aprendizado em redes neurais.

O treinamento usando esta regra consiste em fornecer à rede um conjunto de pares de entradas e saídas, onde a cada entrada do treinamento tem-se uma saída desejada. Este algoritmo é um método de gradiente descendente, que não garante chegar ao mínimo erro global, e que pode ser dividido em 5 passos:

- Passo 1: Apresente um padrão de entrada e a saída desejada:

Utilizando uma determinada estratégia de apresentação coloque um dos padrões de entrada na camada de entrada e saída desejada nas suas respectivas camadas, então ative o passo *calcule saída*.

- Passo 2: Calcule saída

A partir da primeira camada, permita que cada camada produza os valores de saída até atingir a camada de saída e, então ative o passo *ajuste pesos da camada de saída.*

- Passo 3: Ajuste dos pesos da camada de saída

Para cada neurônio j da camada de saída, atualize todos os pesos w_{ij} conforme equação 2.9, então ative o passo *ajuste pesos das camadas escondidas*.

$$W(n+1) = W(n) + \Delta W \tag{2.9}$$

Nesta equação temos como sendo W(n+1) e novo peso atualizado da rede, e ΔW expressando a variação das entradas do neurônio, representada pela Equação 2.10

$$\Delta W_{ij} = \eta.\delta_j.y_i \tag{2.10}$$

onde:

η: taxa de aprendizado

y_i: saída do neurônio

δ_j: diferença (erro) entre a saída computada e a saída desejada do neurônio *j*, que pode ser calculada segundo equação 2.11:

$$\delta_j = y_j(d_j - y_j)(1 - y_j) \tag{2.11}$$

onde:

d_j: saída desejada de dos padrões de entrada nj.

a) Passo 4: Ajuste de pesos das camadas escondidas

b) Para ajustar os pesos de todas as camadas escondidas, atualize o peso w_{ij} de um neurônio n_i de uma camada escondida que está ligado a outro neurônio n_i na camada anterior conforme equação 2.12

c) $$\Delta W_{ij} = \eta \quad \delta / 1 \quad y_k \tag{2.12}$$

d) onde:

e) η : taxa de aprendizagem

f) δ/i: erro relativo do neurônio n_i, dado pela equação 2.13

g)

h) $$\delta / 1 = y_i (1 - y_i) \sum w_{ij} \delta_j \tag{2.13}$$

i) onde:

j) y_i: saída do neurônio n_i, que estimula n_i via w_{ij}

Depois que todas as conexões tenham sido ajustadas ative o passo *cheque magnitude do erro*.

k) Passo 5: Cheque magnitude do erro

l) Para se checar a magnitude o erro, pode-se adotar várias estratégias diferenciadas quanto à forma de aceitar o erro como desprezível. Uma regra muito comum é verificar se o erro global de saída da rede em relação à saída desejada é menor que um dado erro predefinido. Se a condição adotada é satisfeita, então a rede aprendeu o conjunto de treinamento, caso contrário volte a ativar o passo *apresente um padrão de entrada e a saída desejada.*

- FIM DO ALGORITMO

ESTE ALGORITMO PODE SER UTILIZADO EM UMA ARQUITETURA COM QUALQUER NÚMERO DE CAMADAS. O PASSO 4 DEVE SER ATIVADO RECURSIVAMENTE ATÉ ATINGIR A CAMADA DE ENTRADA. PARA SE AUMENTAR A VELOCIDADE DE CONVERGÊNCIA É POSSÍVEL INTRODUZIR NA EQUAÇÃO (2.11) UM MULTIPLICADOR CHAMADO DE *MOMENTUM*, ONDE ESTE VALOR REPRESENTA O EFEITO DOS AJUSTES ANTERIORES NO AJUSTE ATUAL.

Arquiteturas de Redes

Uma das redes mais simples que podemos destacar é a rede linear. Nesta rede os elementos de processamento, ou neurônios artificiais, foram apresentado por McCulloch e Pitts por volta de 1943. Toda rede com camada linear é equivalente a outra rede linear com apenas uma camada e, portanto, não é capaz de resolver problemas que não sejam linearmente separáveis.

Um dos modelos mais conhecidos de redes neurais como já foi citado é o "*Perceptron*", proposto por Frank Rosenblatt em 1957. Foi utilizado inicialmente para reconhecimento de caracteres. Este modelo já pode possuir varias camadas escondidas, associadas. Tipicamente, diversos elementos (neurônio) excitatórios e inibitórios da camada anterior são conectados à cada elemento da camada posterior. No entanto, apenas uma camada é adaptativa, ou seja, pode ter seus pesos modificados através de um algoritmo de aprendizado o qual, hoje, é conhecido como o algoritmo de aprendizado do perceptron. Neste modelo Rosenblatt provou que o algoritmo utilizado para o treinamento permite encontrar, um conjunto de pesos que classifique corretamente os vetores de entrada, caso este conjunto exista. Esta prova ficou conhecida como o *Teorema de Convergência do Perceptron*. O grande problema apresentado é que, conforme Minsky e Papert, eles não são capazes de aprender a classificar funções não linearmente separáveis.

O modelo *ADALINE*, proposto por Widrow, constitui-se um estudo independente e paralelo ao do perceptron. Os elementos de processamento do ADALINE é muito parecido com o do perceptron, pois este também possui um peso de conexão, w_i, conhecido como "*bias*", ou polarização. Este termo é um peso de conexão que apresenta seu valor de entrada u_i

sempre igual a 1. Os ADALINE, da mesma forma do perceptron, não são capazes de aprender a classificar funções não linearmente separáveis.

Para formar uma rede neural artificial, Widrow criou o *MADALINE*, que é um sistema que apresenta uma camada de unidades ADALINE que são conectadas a uma única unidade ADALINE de saída. Um ponto importante a ser observado é que não existem pesos associados as interconexões entre as unidades da camada ADALINE para a unidade de saída.

Em um trabalho publicado em 1982, Hopfield introduziu uma arquitetura de rede que veio a ser conhecida como *Redes de Hopfield*. Ele descreveu em termos claros e simples como capacidades computacionais podem ser obtidas utilizando-se redes com elementos tipo neurônio ilustrando um problema de memórias associativas e, posteriormente, demonstrou como problemas de otimização poderiam ser resolvidos por esta rede. As Redes de Hopfield são caracterizadas por operações assíncronas, ou seja, em cada instante de tempo só um neurônio pode mudar de estado. Além disso, os neurônios têm a mesma probabilidade de ser o escolhido para ser o neurônio a ser analisado para uma possível transição de estados. A característica mais importante deste modelo é que se pode associar a cada estado um índice de quantidade de energia, que diminui cada vez que um neurônio muda de estado.

Teuvo Kohonen, em 1984, desenvolveu uma família de redes que possui como uma de sua maior característica a de auto-organização, conhecidas como redes "*Self-Organizing Map*" ou simplesmente como *Mapas de Kohonen*. Estas redes são úteis principalmente em reconhecimento de padrões, quando as classes a que devem pertencer os elementos a serem reconhecidos não são conhecidas a principio. Os neurônios de uma camada competem entre si pelo privilégio de permanecerem ativos, tais que o neurônio com maior atividade seja o único a participar do processo de aprendizado.

A Lógica Fuzzy

Na mesma linha das Redes Neurais Artificiais surgiu em 1969 a Lógica Fuzzy diante da dificuldade, ou a impossibilidade, de se obter todas as informações e de equacionar a realidade imprecisa do mundo, levou alguns cientistas a propor lógicas alternativas que seriam mais propícias à representação daquele mundo particular. Uma destas proposições é a lógica "*fuzzy*" de Zadeh. Esta teoria permite dar forma matemática às expressões próprias. A força da Lógica Fuzzy deriva da sua habilidade em inferir conclusões e gerar respostas baseadas em informações vagas, ambíguas e qualitativamente incompletas e imprecisas. Neste aspecto, os sistemas de base Fuzzy têm habilidade de raciocinar de forma semelhante à dos humanos. Seu comportamento é representado de maneira muito simples e natural, levando à construção de sistemas compreensíveis e de fácil manutenção. A Lógica Fuzzy é baseada na teoria dos Conjuntos Fuzzy. Esta é uma generalização da teoria dos Conjuntos Tradicionais para resolver os paradoxos gerados à partir da classificação "verdadeiro ou falso" da Lógica Clássica. Tradicionalmente, uma proposição lógica tem dois extremos: ou "completamente verdadeiro" ou "completamente falso". Entretanto, na Lógica Fuzzy, uma premissa varia em grau de verdade de 0 a 1, o que leva a ser parcialmente verdadeira ou parcialmente falsa.

Conclusão Sobe as Redes Neurais Artificias

Conectividades diferentes conduzem a redes com comportamentos diferentes. Redes Diretas não possuem memória no sentido que a sua resposta a uma entrada é independente do estado anterior da rede. Redes Recorrentes, ou de realimentação, por outro lado, são sistemas dinâmicos, pois quando se é apresentado um novo padrão de entrada os neurônios de saída

são novamente computados. Devido aos caminhos de realimentação, as entradas de cada neurônio são então modificados, o que faz com que a rede entre em novo estado.

Arquiteturas diferentes de redes requerem algoritmos de aprendizagem diferentes. Há três paradigmas de aprendizagem: supervisionado, não-supervisionado e híbrido. Estes algoritmos de aprendizagem encaixam-se na maioria dos modelos de redes neurais citados. Na aprendizagem supervisionada, ou aprendizagem com "professor", fornece-se à rede uma resposta correta (saída) para cada padrão de entrada. Os pesos são determinados a fim de permitir que a rede produza respostas tão próximas quanto possíveis das respostas corretas. A aprendizagem com reforço é uma variante da aprendizagem supervisionada na qual se fornece à rede somente uma crítica da correção da saída da rede e não a resposta correta em si. Em contraste, a aprendizagem não-supervisionada, ou aprendizagem sem "professor", não necessita de uma resposta correta associada com cada padrão de entrada no conjunto de dados de treinamento. Ela explora a estrutura subjacente dos dados, ou correlações entre padrões dos dados e organiza os padrões em categorias a partir destas correlações. A aprendizagem híbrida combina aprendizagem supervisionada e não-supervisionada, onde parte dos pesos é normalmente determinada através de aprendizagem supervisionada enquanto que os demais pesos são obtidos através da aprendizagem não-supervisionada.

m) A complexidade computacional relaciona-se com o tempo requerido para um algoritmo de aprendizagem estimar a solução a partir de parâmetros de treinamento.

Utilização do Software MatLab

ORIGINALMENTE O MATLAB FOI IMPLEMENTADO PARA TRABALHAR COM MATRIZ E VETORES DE GRANDE PROPORÇÕES, COMO O NOME MESMO NOS SUGERE: MATRIX LABORATORY. FOI DESENVOLVIDO NA UNIVERSIDADE DE STANFORD E NOVO MÉXICO NA DÉCADA DE 70. ESTE PODEROSO SOFTWARE PROPORCIONA ALÉM DE UMA ALTA PERFORMANCE NA COMPUTAÇÃO NUMÉRICA, TAMBÉM A INTEGRAÇÃO DE CÁLCULOS MATEMÁTICOS, VISUALIZAÇÕES GRÁFICAS E PROGRAMAÇÃO COM AMBIENTE ATIVO OFERECENDO UM ALTO DESEMPENHO PARA A COMPUTAÇÃO CIENTIFICA.

MATLAB É UM PODEROSO SOFTWARE INTERATIVO DE ALTA PERFORMANCE VOLTADO PARA EFETUAÇÃO DE CÁLCULOS NUMÉRICOS. INTEGRA ANÁLISE NUMÉRICA, CÁLCULO COM MATRIZES, PROCESSAMENTO DE SINAIS E CONSTRUÇÃO DE GRÁFICOS EM AMBIENTE FÁCIL DE USAR, ONDE PROBLEMAS E SOLUÇÕES SÃO EXPRESSOS SOMENTE COMO ESCRITAS MATEMÁTICAS, AO CONTRÁRIO DA PROGRAMAÇÃO TRADICIONAL. UM SISTEMA CUJO ELEMENTO BÁSICO DE INFORMAÇÃO É UMA MATRIZ QUE NÃO REQUER DIMENSIONAMENTO. ESSE SISTEMA PERMITE A RESOLUÇÃO DE MUITOS PROBLEMAS NUMÉRICOS EM APENAS UMA FRAÇÃO DO TEMPO QUE SE GASTARIA PARA ESCREVER UM PROGRAMA SEMELHANTE EM LINGUAGEM FORTRAN, BASIC OU C, POR EXEMPLO. ALÉM DISSO, AS SOLUÇÕES DOS PROBLEMAS SÃO EXPRESSAS NO MATLAB QUASE EXATAMENTE COMO ELAS SÃO ESCRITAS NAS FORMULAS MATEMÁTICA.

O MÉTODO MAIS FÁCIL DE ENTRAR COM PEQUENAS MATRIZES NO MATLAB É USANDO UMA LISTA EXPLÍCITA. OS ELEMENTOS DE CADA LINHA DA MATRIZ SÃO SEPARADOS POR ESPAÇOS EM BRANCO OU VÍRGULAS E AS COLUNAS SEPARADAS POR PONTO E VÍRGULA, COLOCANDO-SE COLCHETES EM VOLTA DO GRUPO DE ELEMENTOS QUE FORMAM A MATRIZ. POR EXEMPLO, AO ENTRARMOS COM A EXPRESSÃO:

```
>> A=[ 1 2 3;4 5 6;7 8 9 ]
```

OU

>> A=[1,2,3;4,5,6;7,8,9]

PRESSIONANDO <ENTER> O MATLAB MOSTRA O RESULTADO

A =

1	*2*	*3*
4	*5*	*6*
7	*8*	*9*

A MATRIZ *A* É SALVA NA MEMÓRIA RAM DO COMPUTADOR, FICANDO ARMAZENADA PARA USO POSTERIOR.

ASSIM COMO QUALQUER OUTRA LINGUAGEM DE PROGRAMAÇÃO, O MATLAB TEM REGRAS A RESPEITO DOS NOMES DE VARIÁVEIS. AS VARIÁVEIS SÃO CASE-SENSITIVE (SENSÍVEIS A MAIÚSCULAS E MINÚSCULAS), PODEM CONTER ATÉ 19 CARACTERES, O NOME DEVE COMEÇAR COM UMA LETRA, SEGUIDA DE UM NÚMERO, ALGARISMOS OU SUB-LINHAS, CARACTERES DE PONTUAÇÃO NÃO SÃO PERMITIDOS POR ALGUNS TEREM SIGNIFICADOS PRÓPRIOS DENTRO DA LINGUAGEM. POSSUI CERTAS VARIÁVEIS DE NOMES PADRÕES, POR EXEMPLO, *ANS* USADA PARA RESULTADOS, *PI* QUE É RAZÃO ENTRE O PERÍMETRO DA CIRCUNFERÊNCIA E SEU DIÂMETRO COM VALOR APROXIMADO DE 3,14, DENTRE OUTRAS.

POR CAUSA DE SUA FILOSOFIA VOLTADA PARA MATRIZES, O MATLAB EXECUTA PRONTAMENTE ANÁLISES ESTATÍSTICAS EM CONJUNTOS DE DADOS, QUE SÃO ARMAZENADOS EM MATRIZES ORIENTADAS POR COLUNAS. ISTO É, CADA COLUNA DA MATRIZ REPRESENTA AMOSTRAS INDIVIDUAIS DAS VARIÁVEIS.

HÁ NO MATLAB UM NÚMERO CONSIDERAVELMENTE GRANDE DE COMANDOS QUE PARA SEREM MEMORIZADOS LEVARIAM UM GRANDE TEMPO DE ESTUDO. MAS ISTO É RESOLVIDO COM UM DOS RECURSOS PODEROSOS EXISTENTE, A AJUDA ON-LINE, QUE ESTÁ DISPONÍVEL EM TRÊS

formas: o comando "*MATLAB help*", o comando "*MATLAB lookfor*" e o uso interativo do help a partir da barra de menu.

A construção de gráficos no MatLab é mais uma das facilidades do sistema. Através de comandos simples podemos obter gráficos bidimensionais ou tridimensionais com qualquer tipo de escala e coordenada. Existe no MatLab uma vasta biblioteca de comandos gráficos. A função *plot* do MatLab é extremamente poderosa. Ela automaticamente escolhe os limites dos eixos, marca os pontos individuais e desenha linhas retas entre eles. As opções do comando *plot* nos permitem representar graficamente inúmeros conjuntos com múltiplos de dados no mesmo eixo em um mesmo plano bidimensional ou tridimensional, usar tipos diferentes de linhas, tais como pontilhada ou tracejada, marcar somente os pontos de dados sem interligá-los, usar cores distintas para as diferentes curvas. Além disso, é possível colocar nomes nos eixos, um título na parte superior, desenhar uma grade nas marcas mais grosas, e daí por diante.

Em problemas simples é mais rápido e eficiente introduzir os comandos no prompt do MatLab. Entretanto, à medida que se aumenta o número de comandos e nos casos em que você desejar mudar o valor de uma ou mais variáveis ou re-executar separadamente alguns comandos, pode-se tornar extenso demais introduzir todas as vezes necessárias os comandos no prompt. No entanto, o MatLab apresenta uma solução lógica para este problema. Ele permite colocar os comandos em um arquivo de texto simples e depois somente informar o MatLab para abrir este arquivo e executar os comandos exatamente como se você os tivesse introduzido manualmente no prompt. Esses arquivos são chamados arquivos de instrução ou arquivos M. O termo instrução representa o fato do MatLab simplesmente seguir as instruções contidas no arquivo. O termo M refere-se ao

FATO DE QUE OS NOMES DOS ARQUIVOS DE INSTRUÇÕES SÃO SALVOS COM A EXTENSÃO ‘M’, POR EXEMPLO, *TESTE.M*.

OUTRAS FUNÇÕES ESTÃO DISPONÍVEIS EM BIBLIOTECAS EXTERNAS DISTRIBUÍDAS COM O PROGRAMA ORIGINAL (TOOLBOX), QUE SÃO NA REALIDADE ARQUIVOS COM A EXTENSÃO ".M" CRIADOS A PARTIR DAS FUNÇÕES ESSENCIAIS. ESTA TOOLBOX PODE SER CONSTANTEMENTE ATUALIZADA À MEDIDA QUE NOVAS APLICAÇÕES SÃO DESENVOLVIDAS. ESTA CAIXA DE FERRAMENTAS VEM A SER UMA COLEÇÃO DE FUNÇÕES PARA O MATLAB USADAS PARA MANIPULAR E RESOLVER EXPRESSÕES SIMBÓLICAS. HÁ DIVERSAS FERRAMENTAS PARA COMBINAR, SIMPLIFICAR, DERIVAR, INTEGRAR, RESOLVER EQUAÇÕES DIFERENCIAIS E ALGÉBRICAS, ENTRE OUTRAS VÁRIAS EXISTENTES.

O MATLAB JÁ POSSUI NO PACOTE DE INSTALAÇÃO VÁRIOS DESTES TOOLBOXES, O QUE NÃO IMPEDE QUE CADA UM DESENVOLVA SUA PRÓPRIA CAIXA DE FERRAMENTA. IREMOS UTILIZAR NESTE TRABALHO O TOOLBOX DE REDES NEURAIS QUE POSSUI VÁRIAS FUNÇÕES QUE NOS AUXILIA NAS MODELAGENS E APRENDIZADO DAS REDES NEURAIS ARTIFICIAIS.

TOOLBOX de Redes Neurais

O TOOLBOX DE REDE NEURAIS ESTENDE O AMBIENTE COMPUTACIONAL DO MATLAB PARA FORNECER FERRAMENTAS AO PROJETO, A EXECUÇÃO, A VISUALIZAÇÃO, E A SIMULAÇÃO DE REDES NEURAIS. O TOOLBOX DE REDES NEURAIS ARTIFICIAIS FORNECE UMA SUSTENTAÇÃO DETALHADA PARA MUITOS PARADIGMAS JÁ ANTERIORMENTE PROVADOS NAS TEORIAS, TAMBÉM UMA RELAÇÃO COM OS USUÁRIO QUE CONSEGUEM PROJETAR E CONTROLAR SUAS REDES A NÍVEIS GRÁFICOS. O PROJETO MODULAR, ABERTO, E EXTENSÍVEL DO TOOLBOX SIMPLIFICA A CRIAÇÃO DE FUNÇÕES E DE REDES CUSTOMIZADAS.

O GUIA DE USUÁRIO DO TOOLBOX FOI ESCRITA PELO PROFESSOR EMERITUS HOWARD DEMUTH E MARK BEALE. ESTE HELP TEM A QUALIDADE DE LIVRO-TEXTO, NO QUAL FORNECE UM TRATAMENTO COMPLETO DE ARQUITETURAS DAS REDES NEURAIS ARTIFICIAIS, DE SEUS PARADIGMAS, E DE SUA APLICAÇÕES. INCLUI TAMBÉM UM TUTORIAL E EXEMPLOS DAS APLICAÇÕES. AS DEMONSTRAÇÕES E OS EXEMPLOS ADICIONAIS TAMBÉM ESTÃO INCLUÍDOS COM O PRODUTO.

PELO MOTIVO DAS REDES NEURAIS ARTIFICIAIS REQUEREM CÁLCULOS INTENSIVOS UTILIZANDO MATRIZES, O MATLAB FORNECE UMA ESTRUTURA IDEAL E COM RÁPIDA EXECUÇÃO NO ESTUDO DOS COMPORTAMENTOS DA REDE E NA SUA APLICAÇÃO.

Descrições das funções utilizadas

A PRIMEIRA ETAPA PARA SE TREINAR UMA REDE DE RETROPROPAGAÇÃO NO MATLAB É A DECLARAÇÃO DO OBJETO DA REDE. A FUNÇÃO *NEWFF* É UTILIZADA PARA SE CRIAR UMA REDE DO TIPO FEEDFORWARD (RE-ALIMENTAÇÃO). ESTA FUNÇÃO REQUER COMO PARÂMETRO QUATRO ENTRADAS E RETORNA O OBJETO DA REDE INSTANCIADO. A PRIMEIRA ENTRADA É DECLARADA POR UMA MATRIZ 2X2, DE VALORES MÍNIMOS E MÁXIMOS PARA CADA UM DOS ELEMENTOS DOS PARÂMETROS DE ENTRADA DO VETOR DA ENTRADA. A SEGUNDA ENTRADA CONTÉM A QUANTIDADE DE NODOS ESCONDIDOS E DA SAÍDA DA REDE. A TERCEIRA ENTRADA CONTEM OS NOMES DAS FUNÇÕES DE TRANSFERÊNCIA, ATIVAÇÃO, A SER USADA EM CADA CAMADA. A ENTRADA FINAL CONTÉM O NOME DA FUNÇÃO DO TREINAMENTO A SER USADA. PARA O EXEMPLO, O COMANDO ABAIXO CRIA UMA REDE DE DUAS CAMADAS.

```
NET=NEWFF([-1 2; 0 5], [3, 1], {'LOGSIG', 'LOGSIG'}, 'TRAINGD ');
```

Há um vetor de entrada com dois elementos. Os valores para o primeiro

elemento do vetor de entrada variam entre -1 e 2, e os do segundo elemento vetor ficam entre 0 e 5. Há três neurônios na primeira camada e em um neurônio na segunda camada (da saída). A função de ativação utilizada será do tipo sigmóide tangencial. O ultimo parâmetro descrito é o traingd, que indica que o treinamento será feito com gradiente descendente, que nada mais é o método backpropagation de aprendizagem.

ESTE COMANDO CRIA O OBJETO, INICIALIZA OS PESOS E DEFINE AS POLARIZAÇÕES DA REDE, CONSEQÜENTEMENTE ESTÁ PRONTA PARA O TREINAMENTO, INSTANCIADO NA VARIÁVEL *NET*.

FEITO ISSO, OS PESOS DEVEM SER INICIALIZADOS UTILIZANDO A FUNÇÃO *INIT*. É IMPORTANTE RESSALTAR QUE SEMPRE ANTES DE TREINAR UMA REDE BACKPROPAGATION, OS PESOS E AS POLARIZAÇÕES DEVEM SER INICIALIZADOS. O COMANDO DO *NEWFF* INICIALIZARÁ AUTOMATICAMENTE OS PESOS. ISTO PODE SER FEITO COM O INIT DO COMANDO. ESTA FUNÇÃO FAZ O EXAME DE UM OBJETO DA REDE COMO ENTRADA E RETORNA UM OUTRO COM TODOS OS PESOS E POLARIZAÇÕES INICIALIZADOS. E APÓS ISSO SE PODE DIZER QUE TEMOS A REDE INICIALIZADA; COMO EXEMPLIFICADO ABAIXO:

```
NET = INIT(NET);
```

A FUNÇÃO *SIM* É NECESSÁRIA PARA SIMULAR A REDE. ESTA FUNÇÃO UTILIZA AS ENTRADAS *P* (PADRÕES DE TREINAMENTO) DA REDE, E O OBJETO DA REDE *NET* (REDE JÁ INICIALIZADA), E RETORNA A SAÍDA DA REDE *A*. A UTILIZAÇÃO DA FUNÇÃO DE SIMULAÇÃO SIM PODE SER UTILIZADA NA SEGUINTE SÍNTESE:

```
P = [ 1;2 ];
A = SIM(NET, P)
A = -0.1011
```

ESTES COMANDOS REPRESENTADO POR *P* QUE É O VETOR COM OS PADRÕES DE ENTRADA DA REDE E POR *A* COMO SENDO O VALOR RESULTANTE DO PESO DA REDE TREINADA, CUJA SAÍDA *A* PODE SER DIFERENTE, DEPENDENDO DO ESTADO DE SEU GERADOR DO NÚMERO ALEATÓRIO QUANDO A REDE FOI INICIALIZADA.

NOVAMENTE, O *SIM* É CHAMADO PARA CALCULAR AS SAÍDAS PARA UM EXEMPLO DE TRÊS VETORES DE ENTRADA. NO EXEMPLO ABAIXO, TEMOS A SIMULAÇÃO EM QUE TODOS OS VETORES DA ENTRADA OCUPAM UM LUGAR EM UMA MATRIZ, COMO NO ANTERIOR TAMBÉM. ISTO É MUITO MAIS EFICIENTE DO QUE APRESENTANDO OS VETORES UM DE CADA VEZ.

```
P = [ 1 3 2;2 4 1 ];

                                        a=sim(net, p)
A = -0.1011 -0.2308 0.4955
```

UMA VEZ QUE OS PESOS E AS POLARIZAÇÕES DA REDE FORAM INICIALIZADOS, A REDE ESTÁ PRONTA PARA O TREINAMENTO. O PROCESSO DO TREINAMENTO REQUER ALGUNS EXEMPLOS JÁ CONHECIDOS PARA A REDE (ENTRADAS P DA REDE E SAÍDAS T DO PADRÃO DESEJADO). DURANTE O TREINAMENTO DOS PESOS E AS POLARIZAÇÕES DA REDE OS PESOS SERÃO AJUSTADOS PARA MINIMIZAR A FUNÇÃO DE DESEMPENHO DA REDE (NET.PERFORMFCN). TODOS ESTES PASSOS DO ALGORITMO USAM O GRADIENTE DA FUNÇÃO DO DESEMPENHO (ALGORITMO BACKPROPAGATION) PARA DETERMINAR COMO AJUSTAR OS PESOS PARA MINIMIZAR O DESEMPENHO.

A UTILIZAÇÃO PRÁTICA DO ALGORITMO, COM A ENTRADA DOS DADOS E RESULTADO COM SAÍDAS RECONHECIDAS PELOS PADRÕES DESEJADOS, DENTRO DO MATLAB SERÁ DESCRITA MAIS A FRENTE. NESTE PASSO O IMPORTANTE É SE TER UMA IDÉIA APENAS DE QUE JÁ EXISTEM FUNÇÕES PRÉ-IMPLEMENTADAS DENTRO DO TOOLBOX DE REDES NEURAIS ARTIFICIAIS DO SOFTWARE QUE NOS AJUDARÃO NA INICIALIZAÇÃO, TREINAMENTO E RESPOSTA (SAÍDA) DA REDE.

DESENVOLVIMENTO DO PROTÓTIPO

O presente capítulo trata da especificação e implementação de teorias de Redes Neurais Artificiais na identificação do diabetes em exames de sangue efetuados em índias da Aldeia de Pima.

Com uma base de dados referente a amostras de sangue das índias da aldeia de Pima, o sistema visa identificar, após o treinamento da rede, se os novos testes que forem sendo efetuados sejam reconhecidos pelos padrões da rede treinada e como resultado informe se o determinado exame da paciente contem ou não essa enfermidade metabólica, que é a presença do diabetes mellitus.

O propósito do projeto é identificar se determinado exame efetuado na paciente possui ou não a presença da deficiência de produção e/ou de ação da insulina, que leva a sintomas agudos e a complicações crônicas características do diabetes.

Será definido também parâmetros que identifiquem a presença do diabetes no sangue, para melhor compreensão do problema e organização do protótipo.

Identificação da Doença

n) O diabetes mellitus, popularmente conhecido apenas por DIABETES, é um distúrbio do metabolismo que afeta primeiramente os açúcares (glicose e outros), mas que também tem repercussões importantes sobre o metabolismo das gorduras (lipídios) e das proteínas. Muita gente pensa que o diabetes é uma doença simples e benigna, um problema banal de teor de açúcar alto no sangue. Na verdade, infelizmente não é bem assim. O diabetes é uma disfunção que, se não tratada e bem controlada, acaba produzindo, com o correr do tempo, lesões graves e potencialmente fatais, como o infarto do miocárdio, derrame cerebral,

cegueira, impotência, nefropatia, úlcera nas pernas e até amputações de membros. Por outro lado, quando bem tratado e bem controlado todas essas complicações crônicas podem ser evitadas e o paciente diabético pode ter uma vida perfeitamente normal. Recentemente, foi concluído um grande estudo, nos Estados Unidos, o qual demonstrou que o controle adequado do diabetes é, realmente, o único caminho para se evitar as complicações mencionadas.

o) O pâncreas é o órgão responsável pela produção do hormônio denominado insulina. Este hormônio é responsável pela regulação da glicemia (nível de glicose no sangue). Para que as células das diversas partes do corpo humano possam realizar o processo de respiração aeróbica (utilizar glicose como fonte de energia), é necessário que a glicose esteja presente na célula. Portanto, as células possuem receptores de insulina, que quando acionados "abrem" a membrana celular para a entrada da glicose presente na circulação sanguínea. Uma falha na produção de insulina resulta em altos níveis de glicose no sangue, já que a mesma não é devidamente dirigida ao interior das células.

p)

Diabéticos da Aldeia Indígena de Pima

A tribo indígena Pima vive nos Estados Unidos e, historicamente, eram nômades. Migravam de norte a sul do país acompanhando o movimento dos bisontes, grandes mamíferos ungulados, pré-históricos, parecido com os búfalos e que estavam sempre em busca de pasto verdejante. Em função desse estilo de vida eram magros e saudáveis.

Com a civilização, o homem branco confinou os índios em apenas um pedaço de terra, localizado próximo à cidade de Phoenix, no Arizona, evitando assim a sua maratona atrás de comida. Resultado disso é que eles se tornaram sedentários, pois trocaram os bisontes por hamburguês, deixando assim de serem nômades.

Com o tempo se verificou que os Pima, apesar de terem ficado com o estilo de vida igual ao do homem branco, tinham uma população de obesos e diabéticos muito mais significativa. Descobriu-se, então, que os Pima possuíam uma pré-disposição genética à obesidade, que não vinha a tona quando faziam sua marcha atrás de comida.

Base de Dados

A base de dados utilizada como padrão de treinamento contém 768 amostras, divididas em duas classes que representam a presença ou não de diabetes mellitus nas mulheres da aldeia de Pima:

Classe -1: Não-diabéticos, 478 amostras (~65,5%)

Classe 1: Diabéticos, 251 amostras (~34,5%)

As classes são definidas em termos de oito atributos numéricos e que trazem as seguintes informações:

Número de gestações: quantidade de vezes em que foi constatado que a paciente estava grávida.

Taxa de glicose no plasma sanguíneo após 2 horas do teste de tolerância oral de glicose : segundo a American Diabetes Association, um dos sintomas do diabetes mellitus é a confirmação da glicemia (taxa de glicose no sangue), independente do jejum, uma taxa diagnosticada em maior ou igual a 180 ml/dl.

Pressão sanguínea diastólica: chamada popularmente de pressão mínima, medida em mmHg (milímetro de mercúrios). Quando observando o manômetro, corresponde ao ultimo batimento regular audível.

ESPESSURA (MM) DA DOBRA DA PELE DO TRÍCEPS: A DIMINUIÇÃO DA FLEXIBILIDADE DESTE MÚSCULO É PRESENTE EM PACIENTES QUE POSSUEM DIABETES, CONSIDERADA DENTRO DOS PADRÕES ATÉ CERCA DE 3,5CM MEDIDOS PELO GONIÔMETRO.

TAXA DE INSULINA NO SORO SANGUÍNEO: A INSULINA É UM HORMÔNIO SECRETADO PELO PÂNCREAS RESPONSÁVEL PELA REDUÇÃO DE GLICEMIA AO PROMOVER O INGRESSO DE GLICOSE NAS CÉLULAS. A TAXA MÉDIA CONSIDERADA COMO NORMAL É A DE ATÉ 0,30% DE GLICOSE POR INSULINA.

ÍNDICE DE MASSA CORPORAL: CALCULADO ATRAVÉS DE UMA FORMULA QUE INDICA SE O INDIVIDUO ESTÁ ACIMA, OBESO, OU ABAIXO DO PESO IDEAL.

TENDÊNCIA A TER DIABETES (FUNÇÃO DE PEDIGREE): ABORDAGEM UTILIZADA COM RELAÇÃO A TENDÊNCIA GENÉTICA DO PACIENTE CONSULTADO.

IDADE: DATA DO EXAME SUBTRAÍDO DA DATA DE NASCIMENTO DA PACIENTE.

Essa base apresentada é o resultado da pesquisa do "The National Institute of Diabetes and Digestive and Kidney Diseases" (NIDDK), um laboratório americano fundado em meados de 1950 pelo então presidente dos Estados Unidos da América Harry S. Truman, que possui inúmeras pesquisas voltadas à área biológica.

Patrocinado pelo governo e liderada pelo cientista e pesquisador Dr. Vincent Sigillito, o laboratório NIDDK desenvolveu o projeto após ter percebido a grande presença de diabetes na população de Pima. O grande incentivo caracterizado pelo instituto é que depois de efetuada a pesquisa, retirada as conclusões pelos pesquisadores, todos os resultados são disponibilizados para serem reutilizados cientificamente.

As bases divulgadas pelo laboratório NIDDK, especificamente a *"Pima Indians Diabetes*", como é conhecida internacionalmente a base de dados utilizada nesse projeto, e outros centros de estudos, ficam disponíveis para downloads em uma central de dados da Universidade da Califórnia situada em Irvine, através do endereço "*ftp://ftp.ics.uci.edu/pub*".

Estas bases já são formatadas em arquivos *data* aptas a serem lidas por diversos aplicativos que oferecem manipulação de dados. A base de Pima pode ser interpretada em arquivos texto, de acordo com a Figura 9 abaixo:

```
6,148,72,35,0,33.6,0.627,50,1
1,85,66,29,0,26.6,0.351,31,-1
8,183,64,0,0,23.3,0.672,32,1
1,89,66,23,94,28.1,0.167,21,-1
```

FIGURA 9 - AMOSTRA DA BASE DE DADOS

ONDE:

6 – NÚMERO DE GESTAÇÕES;

148 – TAXA DE GLICOSE NO PLASMA SANGUÍNEO;

72 – PRESSÃO SANGÜÍNEA DIASTÓLICA;

35 – ESPESSURA DA DOBRA DA PELE DO TRÍCEPS;

0 – TAXA DE INSULINA NO SORO SANGÜÍNEO;

33.6 – ÍNDICE DE MASSA CORPORAL;

0.627 – TENDÊNCIA A TER DIABETES (FUNÇÃO PEDIGREE);

50 – IDADE;

1 – RESULTADO DO EXAME.

DE FORMA A PROCEDER A CLASSIFICAÇÃO DOS DADOS, DISTRIBUIU-SE AS AMOSTRAS POR CONJUNTOS DE DADOS ESPECÍFICOS (CONJUNTO DAS AMOSTRAS CUJA CLASSE É CONHECIDA, CONJUNTO DAS AMOSTRAS CUJAS CLASSES SERVIRÃO DE TREINO PARA O ALGORITMO E CONJUNTO DAS AMOSTRAS DE TESTE) CONFORME DESCRITO NA TABELA 1.

TABELA 1: DISTRIBUIÇÃO DAS AMOSTRAS POR CONJUNTOS DE DADOS

TIPO DE AMOSTRA	DISTRIBUIÇÃO
TREINAMENTO	60% (**460)**
VALIDAÇÃO	27% (**207)**

TESTE	13% (**101**)

De acordo com a distribuição da base de dados definido pela Tabela 1, a quantidade de padrões de entradas destinados ao treinamento é satisfatório para que uma rede neural artificial possa ser treinada pois passa dos 60% do total dos dados. Os 40% restantes dos dados contidos na base serão divididos em 27% e 13% para validação e testes, respectivamente, também é considerado um número consideravelmente adequado pois estes dados serão apenas utilizados após a rede já treinada, ou seja, para que tenhamos a certeza de que esta rede conseguiu realmente aprender os dados a ela apresentado.

Modelagem de uma RNA no MatLab

O sistema utilizará as informações contidas na base de dados descrita no Item 4.3 para que possa ser efetuado o treinamento da rede (460 padrões). Com isso ao termino do aprendizado será possível fazer um teste de validação com os 207 padrões. Após a validação será realizado testes com 101 padrões a fim de verificar a consistência da rede.

Baseando-se nas técnicas de reconhecimentos de padrões e utilizando os componentes do modelo de rede perceptron multicamadas com aprendizado de retropropagação se pretende caracterizar matematicamente a funcionalidade de uma Rede Neural Artificial

Para que seja feito o treinamento de uma rede neural, com um desempenho computacional de relevância, é preciso saber além de qual a melhor arquitetura a escolher, também saber qual o melhor algoritmo de aprendizado para o modelo que se quer treinar. A escolha do modelo backpropagation de treinamento de uma rede neural para a implementação do protótipo, deu-se principalmente pela grande capacidade de generalização e na sua rápida operacionalização.

O TREINAMENTO DE UMA REDE DE RETROPROPAGAÇÃO REQUER MUITAS ÉPOCAS E RECÁLCULO DOS PESOS DAS SINAPSES, SENDO ESTE UM DOS MOTIVOS DA UTILIZAÇÃO DO SOFTWARE MATLAB ALÉM DO TOOLBOX ESPECIFICO DE REDES NEURAIS ARTIFICIAIS.

Treinamento de uma RnA para Reconhecimento de Padrões

As várias arquiteturas existentes das Redes Neurais Artificiais nos permitem escolher para cada tipo de implementação qual a melhor se adapta quanto ao desempenho computacional apresentado para seu treinamento.

Neste capítulo discutiremos o treinamento de uma rede neural simples, mais conhecida como *Perceptron* e uma mais desenvolvida com vários neurônios distribuídos em camadas intermediários que utilizam o algoritmo backpropagation para que seja efetuado seu treinamento.

TREINAMENTO DO PERCEPTRON

Como discutido anteriormente, o treinamento da rede neural artificial utilizando perceptron é uma das mais simples implementações utilizada, por isso também é sujeita a restrições.

Na implementação de um perceptron devemos fornecer como parâmetros de entradas os valores dos padrões de treinamento, a taxa de aprendizagem, threshold e para que não seja atribuído zero aos pesos sinápticos das conexões entraremos também com seus valores como atributos para que seja iniciado o treinamento da rede.

O ANEXO 1 MOSTRA O CÓDIGO UTILIZADO PARA A IMPLEMENTAÇÃO DESTA REDE PERCEPTRON DESCRITA NO SOFTWARE MATLAB.

A função *newp* contida no TOOLBOX de Redes Neurais Artificiais do MatLab é utilizada para a criação de um objeto (variável) que representará um perceptron. Os padrões de entradas serão os contidos na base de dados já descrita. Entraremos somente como atributos aleatórios os valores dos pesos das conexões sinápticas. O valor da taxa de aprendizagem e o de threshold serão valores fixos para que possamos fazer a comparação entre os resultados obtidos com redes de arquiteturas diferentes.

Com a introdução dos perceptrons nos anos 50, como já visto anteriormente, se criou uma grande agitação junto aos entendidos da computação. Pois era um dispositivo muito semelhante a um neurônio para os quais estavam disponíveis algoritmos de aprendizagem

bem definidos. Embora o teorema que demonstra o perceptron garanta a classificação correta de dados linearmente separáveis, a maioria não fornece dados de maneira a conseguirmos fazer sua classificação

Diante do problema do perceptron só classificar problemas linearmente separáveis, na identificação dos pacientes com diabetes mellitus temos a necessidade de um algoritmo que reconheça muitas variações dos padrões de entradas que necessitam da mesma saída levando aprendizado do perceptron a encontrar-se com o problema do mínimo local.

A questão do mínimo local é tratado sobre a existência do risco que possui o algoritmo de treinamento do perceptron em ficar preso em valores dos sinais sinápticos. Isso significa que a cada atualização dos pesos, com o decorrer das épocas de interações, entram e um "*loop*" onde não conseguem mais serem atualizados para novos valores para que possam ser convergidos.

A Figura 10 nos mostra o gráfico que demonstra como está ocorrendo o treinamento do perceptron. O valor da taxa de aprendizagem está à 0.00001 e 0.3 está atribuído ao valor de threshold. O valor dos pesos das conexões sinápticas estão representadas pelo vetor w=[0.2 -0.1 0.4 -0.3 0.1 -0.2 0.5 -0.3] e o número máximo de iterações está atribuído à 200.

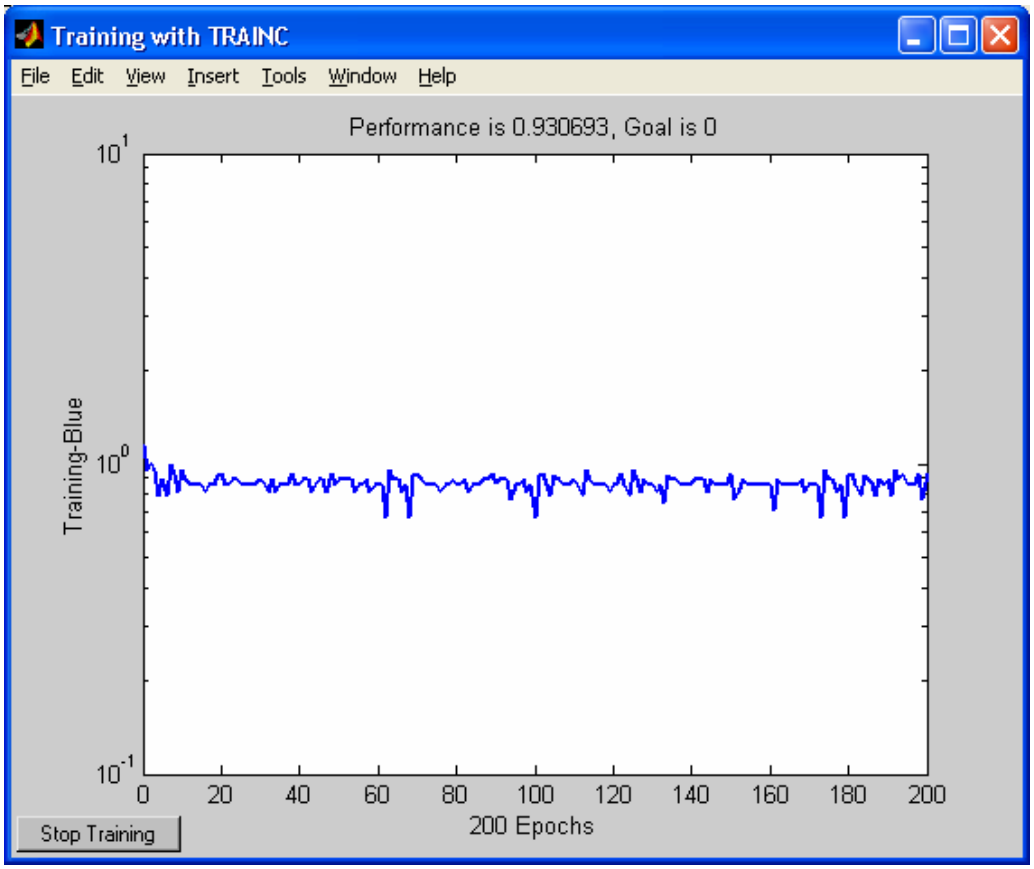

FIGURA 10 – GRÁFICO DE TREINAMENTO DO PERCEPTRON COM 200 ÉPOCAS

Finalizando as 200 iterações, ainda não temos como resposta um perceptron treinado. Já observamos que a linha do gráfico representado pelo treinamento ainda não conseguiu convergir à um número no qual seja satisfatório seu desempenho. O tempo de processamento gasto para estas primeiras iterações é de 33.0940 segundos. Entretanto aumentando-se o número de iterações para 1000, o tempo de processamento fica em média de 162.7350 segundos e obtemos o seguinte gráfico representado pela figura 11.

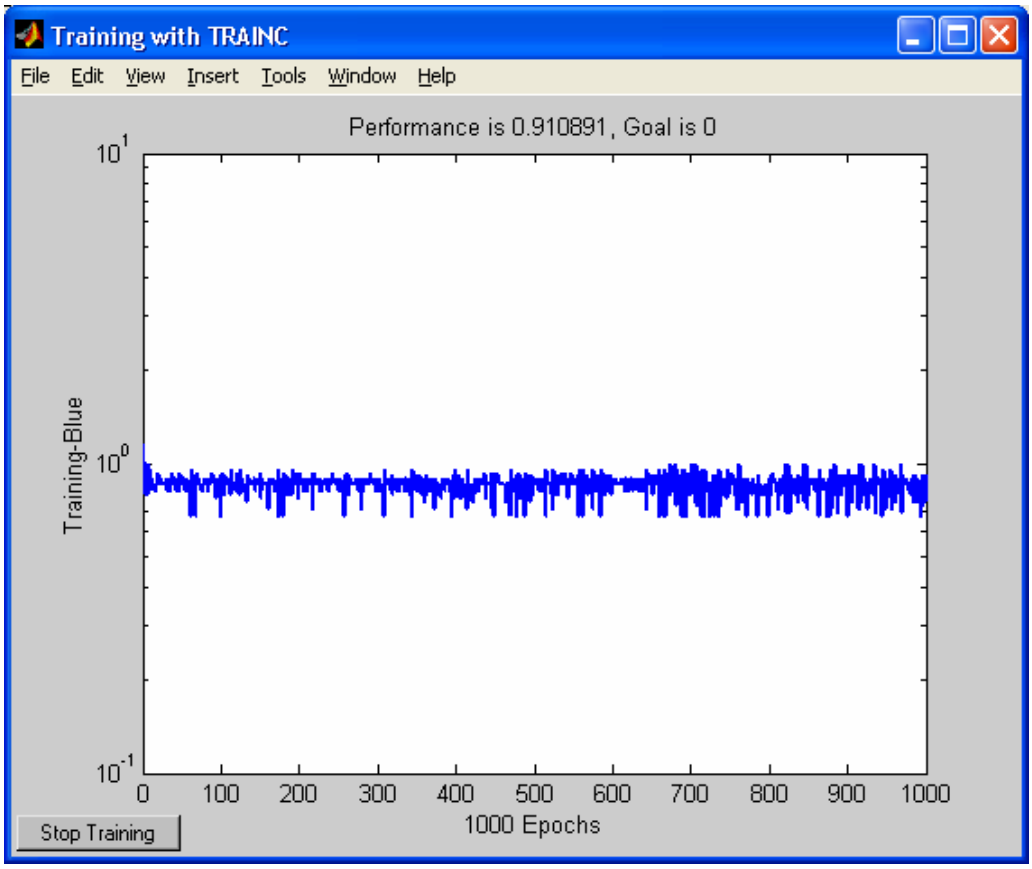

FIGURA 11 – Gráfico de Treinamento do Perceptron com 1000 épocas

NO TÉRMINO DAS 1000 ITERAÇÕES DAS ÉPOCAS PODEMOS COMPARAR A ÉPOCA AO SEU VALOR DA PERFORMANCE, POIS POR "DEFAULT" A REDE PERCEPTRON NÃO DEVE DEIXAR DE RECONHECER NENHUM PADRÃO DE TREINAMENTO A ELA APRESENTADO COMO ENTRADA. ANALISANDO O GRÁFICO REPRESENTADO ACIMA PELA FIGURA 11, OBSERVAMOS QUE A REDE APRESENTOU O PROBLEMA DO MÍNIMO LOCAL, NÃO CONSEGUINDO MAIS CONVERGIR OS PESOS DAS CONEXÕES SINÁPTICAS. REPRESENTADO POR "*PERFORMANCE*" NO GRÁFICO, TEMOS A QUANTIDADE DE PADRÕES DE TREINAMENTO QUE A REDE AINDA NÃO CONSEGUIU RECONHECER. DIANTE DISSO PODEMOS MONTAR A TABELA 2 RESULTANTE DO NÚMERO DE ÉPOCAS JÁ INTERADAS PELA PERFORMANCE APRESENTADA.

TABELA 2: DISTRIBUIÇÃO DAS AMOSTRAS POR CONJUNTOS DE DADOS

NÚMERO DE ÉPOCAS	PERFORMANCE
25	0.851485
50	0.891089
75	0.851485
100	0.673267
125	0.950495
150	0.930693
175	0.891089
200	0.930693
225	0.899189
250	0.811881
275	0.891089
300	0.851485
325	0.851485
350	0.851485
375	0.851485
400	0.851485
425	0.891089
450	0.792079
475	0.811881
500	0.851485
525	0.950495
550	0.712871
575	0.891089
600	0.891089
625	0.891089
650	0.811881
675	0.990099
700	0.792079
725	0.871287
750	0.851485
775	0.851485
800	0.871287
825	0.891089
850	0.891089
875	0.871287
900	0.673267
925	0.811881
950	0.871287
975	0.851485
1000	0.910891

Observando a Tabela 2 podemos perceber que em várias épocas o valor da performance, que nos indica uma rede treinada quando converge a 0, se repete, alternando-se com o passar das iterações. O que nos deixa bem claro que o perceptron não pode ser treinado para o reconhecimento deste determinado caso.

Notamos que o perceptron não pode aprender a esta superfície por não ser capaz de resolver o problema diante de decisões não lineares que deve separa as duas saídas diferentes. Isto reforça a primeira crítica real ao perceptron feita por Minsky e Papert (1969).

O perceptron mostrou-se merecedor de estudos apesar de suas severas limitações por características que atraem a atenção: sua linearidade, seu teorema de aprendizagem e sua simplicidade de treinamento para reconhecimento de padrões. A Figura 12 nos representa um perceptron devidamente treinado, com seus pesos convergindo e sua performance aceitável.

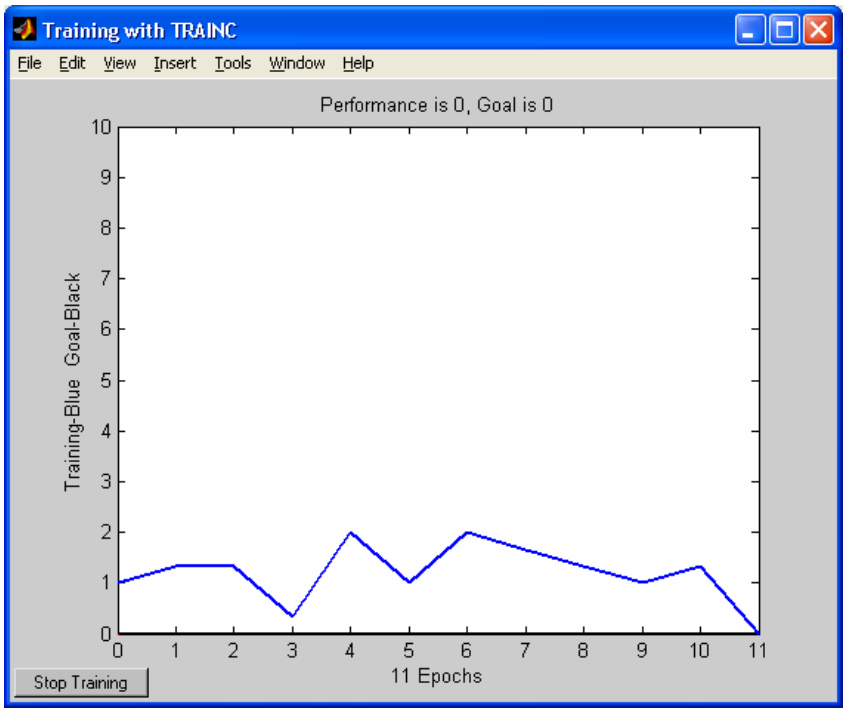

FIGURA 12 – Gráfico do Perceptron Treinado

TREINAMENTO DE UMA REDE MULTI-CAMADAS

Uma das decisões mais importantes para a implementação de uma rede é a escolha das variáveis nas quais a rede ira se basear para poder aprender. Quando as redes neurais artificiais são utilizadas para reconhecimento de padrões, dados sobre diversos índices, ou indicadores, devem ser lhe apresentado para que, como já visto, na sua saída obtenha-se alguma informação.

Considerando que o objetivo da rede neural neste trabalho é a realização do diagnóstico da paciente quanto a presença do diabetes mellitus descreveremos a seguir os passos até chegarmos na saída desejada.

PARA OBTERMOS OS PADRÕES DE ENTRADA DA REDE, DEVEMOS FAZER LEITURA DA BASE DE DADOS. SOMENTE CARREGANDO A VARIÁVEL CONTENDO A BASE INTEIRA PODEMOS FAZER A SEPARAÇÃO DOS ARQUIVOS DE TREINAMENTO, VALIDAÇÃO E TESTES. DISTRIBUIÇÃO ESTA FEITA DE ACORDO COM O APRESENTADO ANTERIORMENTE NA TABELA 1.

- Em uma rede multicamadas é extremante indispensável que saibamos quantas camadas (nodos) ocultas ela deverá possuir para que se tenha uma rede mais eficiente. Devido a isso o número de nodos é colocado como atributo livre para que, além do controle, possam ser efetuados testes de eficiência da rede quanto ao número de camadas ocultas.

ANTES DE INICIARMOS O TREINAMENTO DA REDE NECESSITAMOS AINDA EXTRAIR DA BASE OS VALORES MÁXIMO E MÍNIMO DOS DADOS UTILIZADOS COMO PADRÕES DE ENTRADA. O MATLAB NOS FORNECE ESSE RECURSO ATRAVÉS DA FUNÇÃO *MINMAX*. COM ISSO É FORMADO UM VETOR DE 2 POSIÇÕES QUE NOS INDICARÁ O INTERVALO DE CADA PADRÃO.

DISTRIBUÍDO OS ARQUIVOS PARA A ALIMENTAÇÃO DA REDE E TODOS OS PARÂMETROS E SEUS VALORES ATRIBUÍDOS CORRETAMENTE DEVE-SE INICIAR A MODELAÇÃO DA REDE. SENDO

ESTE O PONTO DO PROCESSO ONDE INFORMAMOS QUAL É A ARQUITETURA QUE DESEJAMOS QUE A REDE NEURAL ARTIFICIAL POSSUA.

NA ESTRUTURA DA REDE CRIADA NESSE PROTÓTIPO OS NODOS DAS CAMADAS ESCONDIDAS E DE SAÍDA POSSUIRÃO A FUNÇÃO DE ATIVAÇÃO DO TIPO SIGMÓIDE LOGÍSTICA, REPRESENTADA PELA FUNÇÃO *LOGSIG*. A ATUALIZAÇÃO DOS PESOS SERÁ FEITA DE ACORDO COM OS ALGORITMOS DE OTIMIZAÇÃO GRADIENTE DESCENDENTE E DE LEVENBERG-MARQUARDT, AMBOS VARIAÇÕES DO BACKPROPAGATION. COM FUNÇÕES DIFERENTES PARA CÁLCULO DOS PESOS DAS CONEXÕES SINÁPTICAS TEREMOS AS COMPARAÇÕES DE QUAL SE ADAPTA MELHOR PARA ESTA APLICAÇÃO. O CÁLCULO DO ERRO SERÁ ATRAVÉS DA SOMA DOS ERROS QUADRÁTICOS.

APÓS A CRIAÇÃO DO OBJETO COM A ESTRUTURA DESEJADA DA REDE DEVEMOS ATRIBUIR OS PARÂMETROS NECESSÁRIOS PARA O TREINAMENTO. MAXIMO NÚMERO DE ITERAÇÕES, TAXA DE APRENDIZADO, GRADIENTE MÍNIMO, TEMPO MÁXIMO (EM SEGUNDOS) PARA O TREINAMENTO E LIMIAR DE EXCITAÇÃO SÃO TODOS PARÂMETROS AJUSTÁVEIS PARA MELHOR DESEMPENHO DA REDE AO SER TREINADA.

ATRIBUÍDOS TODOS OS PARÂMETROS NECESSÁRIOS, COM A FUNÇÃO DO TOOLBOX DE REDES NEURAIS ARTIFICIAIS DO MATLAB DESCRITA POR *TRAIN* INICIA-SE O TREINAMENTO DA REDE. NESTA FUNÇÃO, UTILIZA OS DADOS DO TREINAMENTO COMO PARÂMETROS DE ENTRADA E OS DA VALIDAÇÃO PARA QUE AO TERMINO DO TREINAMENTO SEJA EFETUADO A CONFIRMAÇÃO DE QUE O PROCESSO FOI BEM SUCEDIDO.

COM UMA REDE TREINADA AINDA SE TEM A OPÇÃO DE FAZER SUA SIMULAÇÃO. QUANDO TODOS OS PARÂMETROS DA REDE ESTIVEREM DE ACORDO COM OS DESEJADOS, APÓS TODAS AS ITERAÇÕES NECESSÁRIAS, E SUA VALIDAÇÃO UTILIZAMOS OS DADOS DOS TESTES PARA EFETUARMOS A SIMULAÇÃO. COM A FUNÇÃO *SIM* USAMOS O RESTANTE DOS DADOS PARA QUE

SEJA FEITA A CONFIRMAÇÃO DE QUE A REDE ESTA REALMENTE COM OS PESOS PONDERADOS CORRETAMENTE.

COMO RESULTADO TEMOS A REDE TREINADA, COM SEUS PESOS AJUSTADOS E A SAÍDA DESEJADA. APÓS ISSO, TODOS OS PADRÕES QUE FOREM APRESENTADOS A REDE DEVERÃO TER A SAÍDA RESULTANTE CORRETA. PODEMOS AINDA FAZER A ANÁLISE DE VÁRIOS RESULTADOS QUANTO AO DESEMPENHO COMPUTACIONAL QUE A REDE APRESENTOU DURANTE O TREINAMENTO, QUAL FOI O ERRO APRESENTANDO, ENTRE OUTROS ASPECTOS QUE SERÃO DESCRITOS NOS PRÓXIMOS TÓPICOS DO CAPÍTULO.

A REDE FOI IMPLEMENTADA ATRAVÉS DO TREINAMENTO SUPERVISIONADO, POIS CONHECEMOS AS SAÍDAS DESEJADAS. COMO DESCRITO ANTERIORMENTE, A BASE DE DADOS POSSUI SUA SAÍDA VARIANDO EM 1 E –1, DIABÉTICO OU NÃO. A FUNÇÃO DE ATIVAÇÃO *LOGSIG*, NO QUAL UTILIZAMOS, TEM SUA VARIAÇÃO ENTRE AS PROXIMIDADES DE 0 E 1, CONFORME REPRESENTADA NA FIGURA 13

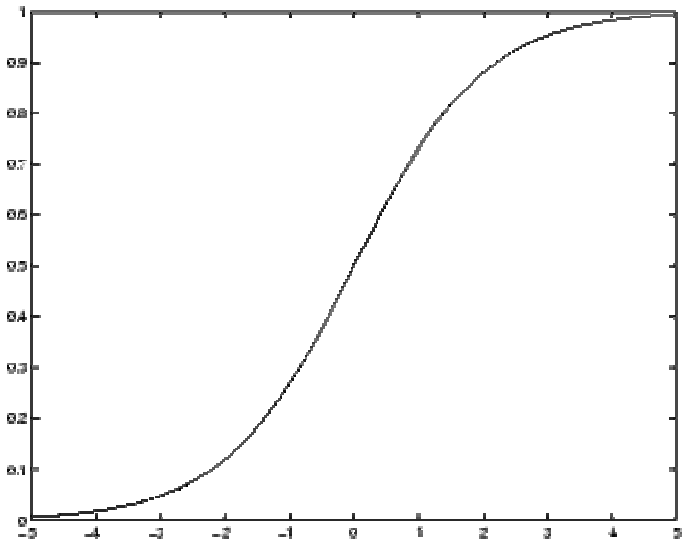

FIGURA 13 – GRÁFICO DA FUNÇÃO DE ATIVAÇÃO *LOGSIG*

COMO A ESCOLHA DA FUNÇÃO DE ATIVAÇÃO FOI A SIGMÓIDE LOGÍSTICA, A SAÍDA DA REDE DEVE OBEDECER A SUAS LIMITAÇÕES. PARA ISSO NA BASE FOI EFETUADO UM TRATAMENTO QUANTO AOS VALORES DA SAÍDA DESEJADA. UTILIZANDO A FUNÇÃO *MASKM* CONSEGUIMOS COM QUE TENHAMOS UM NOVA BASE COM A SAÍDA DESEJADA, DE ACORDO COM *LOGSIG*. O VALOR DA SAÍDA FOI ATUALIZADO PARA 0.4 COMO SENDO O PACIENTE PORTADOR DO DIABETES MELLITUS E 0.6 COMO NÃO PORTADOR DA DOENÇA. A NÃO ATUALIZAÇÃO DOS VALORES DA SAÍDA FAZ COM QUE NÃO CONSEGUIMOS OBTER UMA REDE CONVERGINDO A UM RESULTADO ACEITÁVEL. A FIGURA 14 NOS REPRESENTA O GRÁFICO DE SAÍDA DE UMA REDE MULTI-CAMADAS QUE NÃO CONSEGUIU PONDERAR SEUS PESOS PARA QUE TENHAMOS UMA SAÍDA COM O RESULTADO INICIALMENTE CONTIDO NA BASE NEM MESMO COM UM GRANDE NÚMERO DE ITERAÇÕES DAS ÉPOCAS, CERCA DE 952.100 .

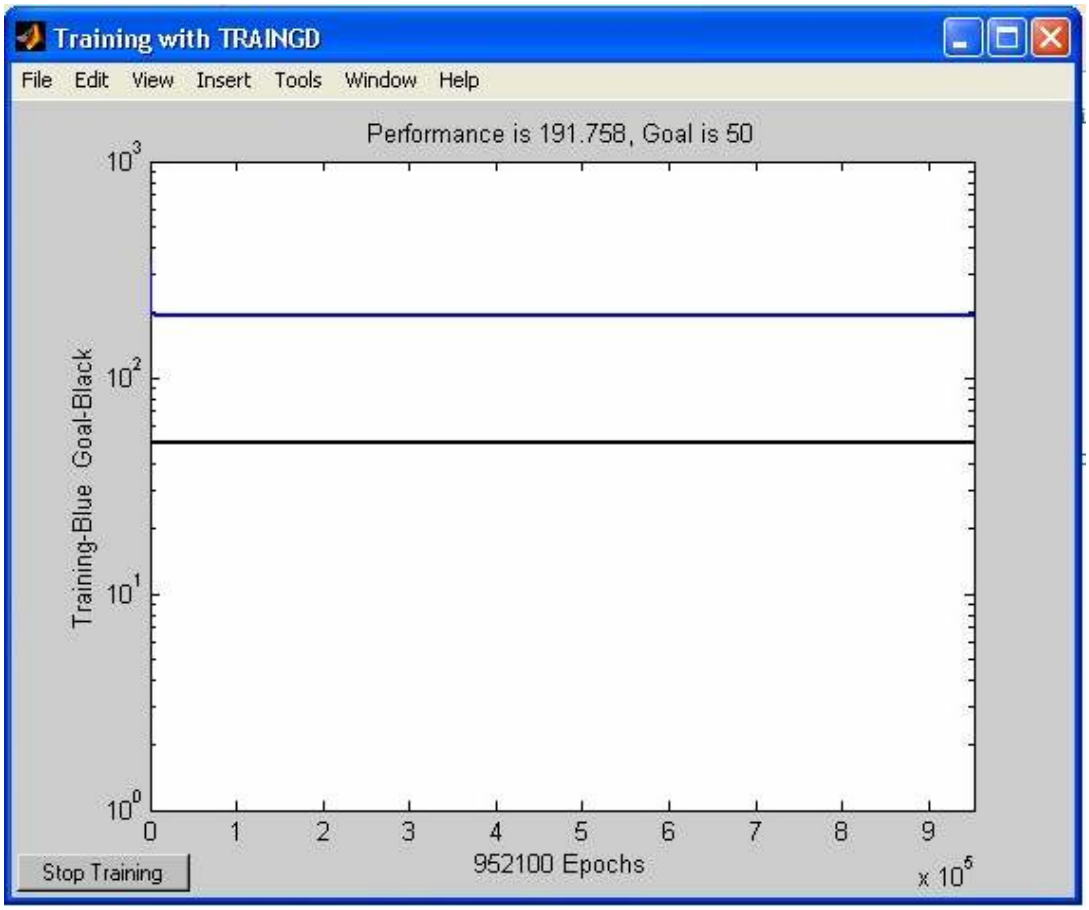

FIGURA 14 – GRÁFICO DA REDE SEM O TRATAMENTO DA SAÍDA DA BASE

APÓS O TRATAMENTO DAS SAÍDAS DA BASE, TEMOS UMA REDE NEURAL ARTIFICIAL CONVERGINDO SEU RESULTADO AO CRITÉRIO MÍNIMO DE ERRO DE TREINAMENTO COM A FUNÇÃO DE ATIVAÇÃO SIGMÓIDE LOGÍSTICA DEVIDAMENTE ESTIPULADA E OS PESOS DE DUAS CAMADAS INTERNAS OCULTAS SENDO CORRETAMENTE PONDERADOS PELOS ALGORITMOS DE TREINAMENTO GRADIENTE DESCENDENTE E COM A OTIMIZAÇÃO DE LEVENBERG-MARQUARDT.

O ANEXO 2 APRESENTA O CÓDIGO UTILIZADO PARA A IMPLEMENTAÇÃO DESTA REDE NEURAL ARTIFICIAL DE MULTI-CAMADAS COM ALGORITMO DE TREINAMENTO BACKPROPAGATION IMPLEMENTADA NO SOFTWARE MATLAB.

OS RESULTADOS OBTIDOS COM A VARIAÇÃO DA TAXA DE APRENDIZAGEM E AS FUNÇÕES QUE PONDERAM OS PESOS DA REDE SERÃO APRESENTADOS NA SEÇÃO 5.3, ANALISE DE RESULTADOS,.

Operacionalidade da Implementação

Ao iniciarmos o treinamento de uma rede neural artificial, não sabemos qual será seu comportamento quanto à saída de resultados. Diante disso, após a implementação do script dentro do software MatLab, possuímos uma ferramenta específica para reconhecimento de determinado padrão cuja rede foi treinada.

PARA UMA APLICAÇÃO SER CONSIDERADA REALMENTE TREINADA, DEVE SER ATRIBUÍDA A ELA VALORES QUE AJUDARÃO NO SEU MELHOR DESEMPENHO, POIS O GANHO COMPUTACIONAL QUANTO AO TEMPO DO PROCESSAMENTO É UM DOS FATORES PRIMORDIAIS PARA SER TRATADO.

DIANTE DESSA NOVA FAZE, DEVEMOS SABER QUANTOS NODOS ESCONDIDOS A REDE MULTICAMADA DEVE POSSUIR EM CADA CAMADA OCULTA, QUAL A MELHOR TAXA DE

aprendizagem que deve ser relacionada. Um bom exemplo para se descrever o ganho computacional, é o fato da necessidade ou não de um certo número estipulado de épocas para o treinamento. Se com uma quantidade menor de épocas já conseguimos ter uma rede treinada, não há a necessidade de se continuar a fazer as interações. A vantagem que o MatLab já trata isso por si, mas no caso dos critérios de erro e de falha, só saberemos qual o melhor desempenho efetuando-se vários testes sobre o treinamento da rede.

Entrando com os padrões de treinamento iniciais da rede, ela devera dar como resposta se um paciente é ou não portador do diabetes mellitus. Para isso, usamos o erro SSE (soma dos erros quadráticos) para representar os resultados da rede. O erro SSE é muito utilizado na representação de problemas de classificação pelo motivo de fornecer uma idéia de como a rede se comportou para classificar os padrões apresentados, pois ele representa uma medida da diferença entre as saídas da rede e as saídas desejadas. Outro resultado muito utilizado também é o índice do erro de classificação, que representa o percentual de padrões que foram classificados erradamente.

No treinamento da rede os aspectos fixos devem ser estipulados inicialmente, pois nos dirão quais as características gerais da rede. Devido a isso, atribuímos ao número de entrada da rede correspondente aos 8 atributos da base e a saída com 1 valor, a saída desejada. O algoritmo utilizado foi o backpropagation, com a função de ativação determinada pela sigmóide.

Para encontrarmos o melhor desempenho no treinamento de uma Rede Neural Artificial outros aspectos devemos deixar sendo variáveis. A quantidade de neurônios na camada intermediaria, a taxa de aprendizado e a quantidade máxima de iterações serão os atributos que nos levará a uma combinação onde o

DESEMPENHO DA REDE FOI MAIS SATISFATÓRIO QUANTO AO ERRO SSE E DE CLASSIFICAÇÃO DIANTE DO TEMPO DE PROCESSAMENTO.

TERMINADO O TREINAMENTO DA REDE NEURAL ARTIFICIAL JÁ POSSUÍMOS OS SSE DO CONJUNTO DE TREINAMENTO E O DE VALIDAÇÃO. SE O VALOR DO ERRO FOR SATISFATÓRIO, QUE REPRESENTA UMA REDE TREINADA, COMEÇAMOS ENTÃO A EFETUAR OS TESTES COM O CONJUNTO DE PADRÕES DE TREINAMENTO DESTINADOS À ISSO. SIMULANDO A REDE, TEMOS A COMPARAÇÃO REAL DA SAÍDA DO CONJUNTO COM A SAÍDA DESEJADA. SOMENTE ENTÃO, CONSEGUE-SE OBTER O VALOR SSE DA REDE DIZENDO O DESEMPENHO REAL OBTIDO ATRAVÉS DO ALGORITMO DE TREINAMENTO.

Análise de Resultados

Finalizamos a implementação com a analise dos dados de saídas dos diferentes métodos de implementação de Rede Neural Artificial à começar com o Perceptron, depois uma outra implementação com uma arquitetura de três camadas ocultas além da camada de entrada e saída.

Diante do problema apresentado anteriormente, não conseguimos o treinamento de uma rede baseada no modelo de Rossenblatt pelo motivo da não classificação de problemas linearmente separáveis. A fácil implementação que possui o Perceptron entra em contraste com suas limitações. Com cerca de 1000 interações de cálculos das épocas não se chega a uma rede considerada treinada por isso descarta-se os resultados apresentado pelo Perceptron.

A rede multi-camadas com algoritmo de aprendizagem backpropagation mostrou-se apta ao aprendizado dos padrões de entradas representados pelos dados dos resultados de exames para detecção de diabetes mellitus contidos na base. A analise dos resultados obtidos

no final do treinamento da rede nos permite encontrar qual a melhor combinação dos parâmetros livres da entrada para o desempenho mais satisfatório.

Uma analise da performance da rede multi-camadas pode se dar através da variação da taxa de aprendizado e a quantidade de neurônios em cada uma das 3 camadas ocultas. A Tabela 3 representa a saída de uma rede com 16, 8 e 4 neurônios respectivamente nas camadas ocultas variando a taxa de aprendizagem em 0.01, 0.0001 e 0.000001 para que possamos verificar as interações de cada passo utilizando a função *traingd* para a atualização dos pesos.

TABELA 3: RESULTADO OBTIDO UMA REDE COM 16, 8 E 4 NEURÔNIOS RESPECTIVAMENTE NAS CAMADAS OCULTAS

CONFIG.	TAXA DE APRENDIZADO	Nº DE ITERAÇÕES	SSE TREINAMENTO	SSE VALIDAÇÃO	SSE TESTE	PERFORMANCE	THRESHOULD	TEMPO CPU (SEGUNDOS)
01	0.01	12	4.28778	1.71738	1.10432	4.28778	7.807775	2.172000
02	0.001	29	4.06413	1.58539	0.88792	4.06413	-8.803837	2.907000
03	0.0001	726	2.97593	1.12127	0.61686	2.97593	-4.062370	27.203000

Observando a Tabela 3 podemos destacar que quanto menor for o valor da taxa de aprendizagem mais tempo se gasta com o processamento do treinamento da rede. Isso devido ao algoritmo backpropagation requer muitos cálculos para chegar ao treinamento correto de uma Rede Neural Artificial com múltiplas camadas entre a entrada e a saída.

Foi atribuído o valor 0 em todas as configurações ao erro de treinamento, mas como a função de aprendizado do TOOLBOX de Redes Neurais do MatLab quando os pesos não podem mais serem atualizados ela faz com que o treinamento seja encerrado. Para uma rede que contem 768 padrões de entrada a performance da rede chegando abaixo de 5 padrões com

erro, nos representa que apenas 0,651% dos testes serão falhos, valendo destacar o ótimo desempenho da rede.

O gráfico da Figura 15 corresponde à configuração número 03 da Tabela 3 com a taxa de aprendizagem atribuída em 0.0001 a rede converge o resultado com um erro de apenas 2.97593 em 726 épocas.

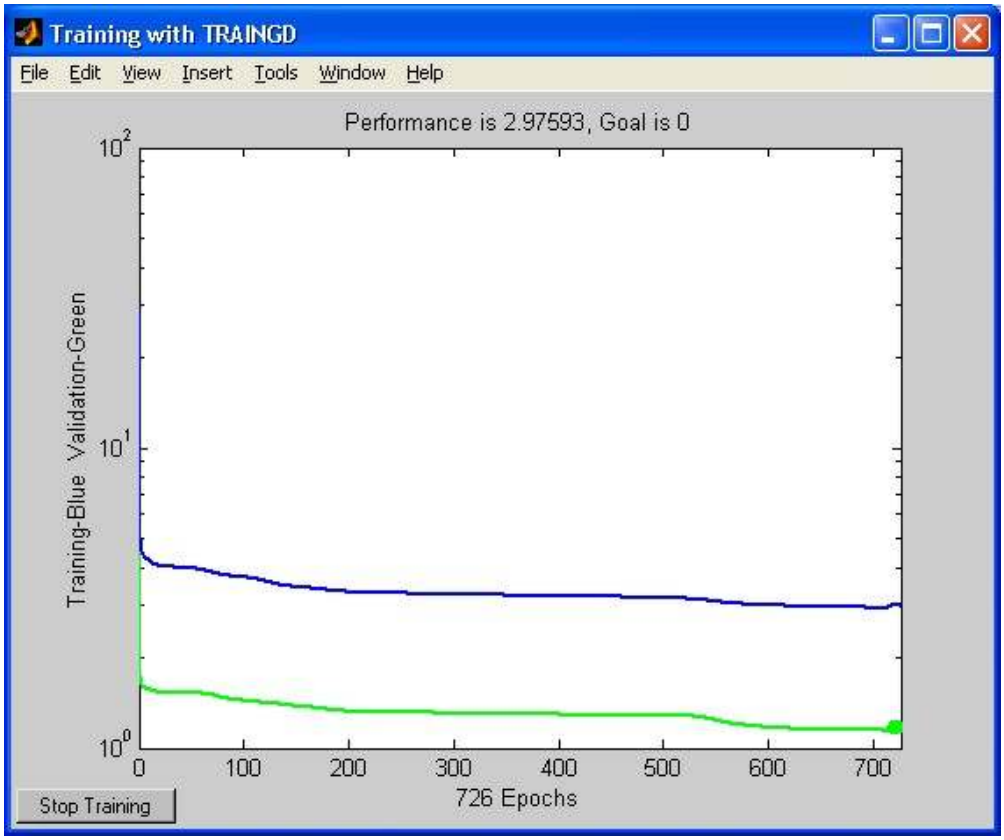

FIGURA 15 – Resultado da rede com taxa de aprendizado de 0.0001

Para o preenchimento dos dados da Tabela 3 utilizamos as atualizações dos pesos de acordo com o algoritmo gradiente descendente descrito pela função *traingd* do TOOLBOX de Redes Neurais do MatLab. A Tabela 4 nos mostra os resultados obtidos com o algoritmo de otimização dos pesos de Levenberg-Marquardt, função *trainlm*.

TABELA 4: RESULTADO OBTIDO DE UMA REDE COM ATUALIZAÇÃO DOS PESO PELO MÉTODO LEVENBERG-MARQUARDT

CONFIG.	TAXA DE APRENDIZADO	Nº DE ITERAÇÕES	SSE TREINAMENTO	SSE VALIDAÇÃO	SSE TESTE	PERFORMANCE	THRESHOULD	TEMPO CPU (SEGUNDOS)
01	0.01	8	2.43795	1.31923	0.65563	2.43795	0.192307	4.969000
02	0.001	17	3.92488	1.59747	0.96476	3.92488	7.904522	10.969000
03	0.0001	19	3.02856	1.37038	0.97154	3.02856	10.255640	10.500000

Analisando a Tabela 4 preenchida com os resultados da rede com atualização dos pesos pela otimização de Levemberg-Marquardt, percebemos que este modelo apresenta uma rede treinada com mais eficiência à uma taxa de aprendizagem maior. Ao diminuir esse valor o desempenho apresentado pela rede diminui, não chega a representa a perca de sua eficiência total, apenas sua performance fica com um índice menor.

A Figura 16 nos mostra o gráfico gerado pela rede treinada com a função *trainlm* com a sua taxa de aprendizagem em 0.01 e a arquitetura da rede sem modificações, continuando com as 3 camadas ocultas e os respectivos números de neurônios em cada uma representado na Tabela 4 pela configuração número 4.

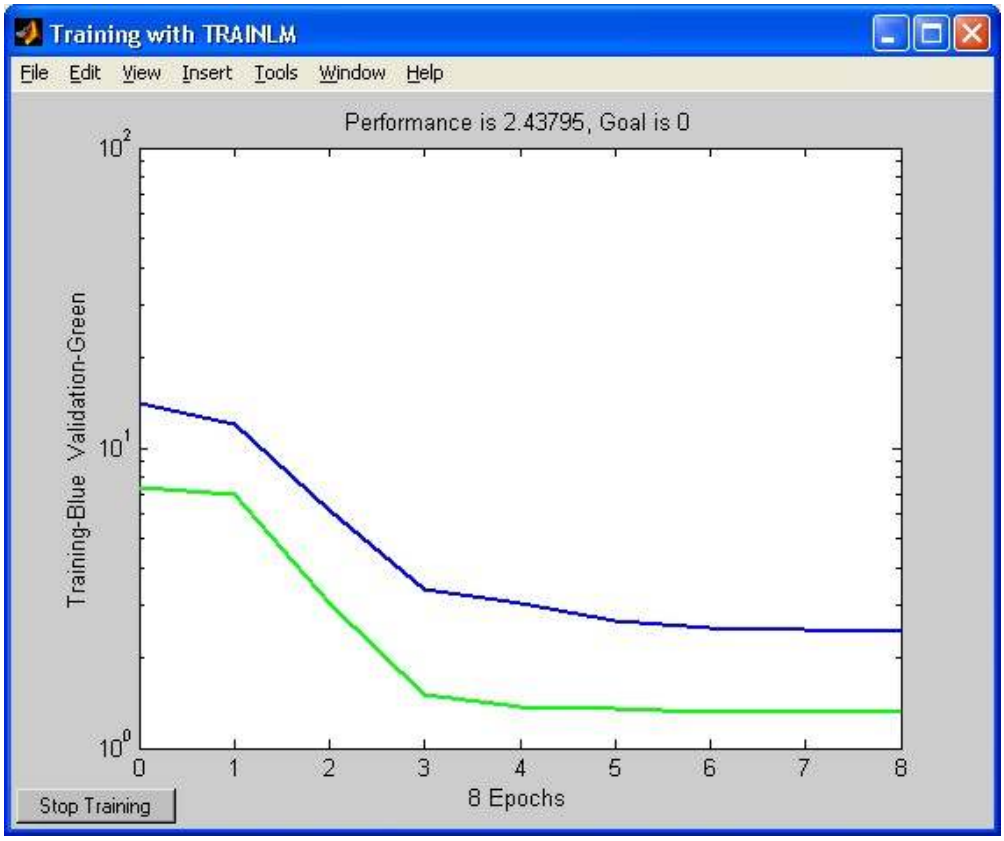

FIGURA 16 – RESULTADO DA REDE COM TAXA DE APRENDIZADO DE 0.01

O MODELO LEVEMBERG-MARQUARDT NA PRÁTICA TEM CARACTERÍSTICAS DE CONVERGÊNCIA GLOBAL (CONVERGE PARA O MÍNIMO LOCAL A PARTIR DE QUALQUER VALOR APROXIMADO), O LIMITE DO TAMANHO DO PASSO É FEITO ATRAVÉS DO USO DA REGIÃO DE CONFIANÇA.

INDIFERENTE DA FUNÇÃO DE ATUALIZAÇÃO DOS PESOS QUE UTILIZAMOS, TEMOS UMA REDE NEURAL ARTIFICIAL TREINADA PARA A IDENTIFICAÇÃO DE PACIENTES COM DIABETES MELLITUS. A COMPARAÇÃO ENTRE AS DUAS FORMAS DE SE OBTER OS PESOS PONDERADOS É REFERENTE A EFICIÊNCIA QUE ELA FORNECE. OS DOIS MÉTODOS APRESENTARAM UM ÓTIMO RESULTADO, NÃO TENDO MUITA DIFERENÇA ENTRE A EFICIÊNCIA DA FUNÇÃO *TRAINLM* E *TRAINGD*. PARA APRESENTAÇÃO DE UMA DAS DUAS COM O MELHOR DESEMPENHO PODEMOS

CITAR A CONFIGURAÇÃO 04 DA TABELA 04, POIS UTILIZANDO A FUNÇÃO *TRAINLM* CONSEGUIMOS UM EXCELENTE RESULTADO COM A PERFORMANCE CHEGANDO A 2.43795.

CONCLUSÕES

Este trabalho teve como objetivo geral o desenvolvimento de um procedimento capaz de tornar mais efetivo o uso de um modelo de Redes Neurais Artificiais gerado para auxiliar o reconhecimento de caso de pacientes com a presença de uma disfunção no organismo. Mais do que isso, os procedimentos aqui apresentados podem ser utilizados para quaisquer outros modelos que tenham sido desenvolvidos a partir de Redes Neurais Artificiais com a mesma característica dos padrões de treinamentos apresentados a esta rede, sejam eles ou não para saber se uma índia da Aldeia de Pima, ou qualquer outro grupo de pessoas que contenham tais informações possuem a presença de diabetes.

Todos os procedimentos aqui desenvolvidos (conceitos e aprendizado de Redes Neurais Artificiais junto ao programa desenvolvido no MatLab) apresentaram bom desempenho como ferramentas para um futuro uso prático, tendo sido demonstrado, através do exemplo, a viabilidade de uso dos mesmos. Além da capacidade de realizar estimativas, o uso das ferramentas desenvolvidas permite observar de forma clara, por exemplo, o comportamento dos valores de saída como conseqüência de variações nos dados de entrada, ou seja, análises de sensibilidade. A utilização dos vários modelos para demonstração da eficiência das Redes Neurais Artificiais nos mostra a complexidade e variedade que as arquiteturas dos modelos nos fornecem.

Finalmente este trabalho com uma estima elevada quanto à grande capacidade computacional que uma Rede Neural Artificial possui. A sua característica de adquirir uma forma de inteligência através apenas de exemplos à ela fornecida incentiva-nos a prosseguir nas pesquisas para que um dia, quem sabe, possamos realmente reproduzir todo o complexo do cérebro humano através de paradigmas da computação.

REFERÊNCIAS BIBLIOGRÁFICAS

AZEVEDO, Fernando Mendes de; BRASIL, Lourdes Mattos; OLIVEIRA, Roberto Célio Limão de. **Redes Neurais com Aplicações em controle e sistemas especialistas.** Florianópolis: Editora Bookstore, 2000.

CALIFORNIA, UNIVERSIDADE. *CENTRAL DE DADOS*. **FTP://FTP.ICS.UCI.EDU/PUB**. ACESSO EM: 10.ABR.2005.

ENGEL, Paulo Matirns. Desenvolvimento de um módulo neural para tarefas de reconhecimento de padrões. In: Simpósio Brasileiro de Inteligência Artificial, 10., 1993, Porto Alegre. **Anais**... Porto Alegre: UFRGS, 1993. p. 445-457.

HANSELMAN, D; LITTLEFIELD, B. **MATLAB VERSÃO DO ESTUDANTE.** TRADUÇÃO HÉRCULES P. NEVES. SÃO PAULO: MAKRON BOOKS, 1997.

HAYKIN, Simon. **Redes Neurais:** princípios e prática. Porto Alegre: Bookman, 2001.

HINTON, G.E.; SEJNOWSKI, T.J. **LEARNING AND RELEARNING IN BOLTZMANN MACHINES.** IN: RUMELHART, D.E. & MCCLELLAND, J.L. *PARALLEL DISTRIBUTED PROCESSING: EXPLORATIONS IN THE MICROSTRUCTURE OF COGNITION*. CAMBRIDGE: EDITORA MIT PRESS, (1986).

HOPFIELD, J.J. **NEURONS WITH GRADED RESPONSE HAVE COLLECTIVE COMPUTATIONAL PROPERTIES LIKE THOSE OF TWO-STATE NEURONS**. EUA, (1984).

KOHONEN, TEUVO. **SELF-ORGANIZING MAP:** *EXTENDED EDITION*. BERLIN: EDITORA SPRINGER (2001).

LOESCH, Cláudio; SARI, Solange Teresinha. **Redes Neurais Artificiais:** fundamentos e modelos. Blumenau: Editora da FURB, 1996.

MCCULLOCH, Warren S.; PITTS, Walter. **A Logical Calculus of the Ideas Immanent in Nervous Activity**. Mathematical Biophysics: 1943.

OLIVEIRA, M. E. (2000) *REDES NEURAIS ARTIFICIAIS*. HTTP://GEOCITIES.COM/CAPECANAVERAL/RUNWAY/ 4303/ ENGENHAR.HTML. ACESSO EM: 08.MAI.2005.

PARDINI, Hermes. **MANUAL DE EXAMES 2003/2004** – Instituto de patologia clínica H. Pardini.

ROSENBLATT, R.. **PRINCIPLES OF NEURODYNAMICS**: PERCEPTRONS AND THE THEORY OF BRAIN MECHANISMS. NOVA YORK: EDITORA SPARTAN BOOKS, 1962.

TAFNER, Malcon Anderson. **Redes Neurais Artificiais:** introdução e princípio de neurocomputação. Blumenau: Editora da FURB, 1995.

WASSERMAN, P.D. **ADVANCED METHODS IN NEURAL COMPUTING**. NOVA YORK: EDITORA VAN OSTRAND REINHOLD, 1989.

ANEXO II

```
CLC;
CLEAR;
ECHO ON
TIC;
numEscondidos1 = 16;    % Quantidade de nodos escondidos na 1ª camada oculta
numEscondidos2 = 8;     % Quantidade de nodos escondidos na 2ª camada oculta
numEscondidos3 = 4;     % Quantidade de nodos escondidos na 3ª camada oculta

numEntradas  = 8;    % Quantidade de nodos de entrada
numSaidas    = 1;    % Quantidade de nodos de saida

numTr       = 460;   % Quantidade de padroes de treinamento
numVal      = 207;   % Quantidade de padroes de validacao
numTeste    = 101;   % Quantidade de padroes de teste

numBase     = numTr + numVal + numTeste; %Numero total de padroes de treinamento

ECHO OFF

%   Abrindo arquivo
arquivoBase     = fopen('PIMA2.data','rt');


%   Montando e distribuindo os padroes de treinamento

% Montando a base completa
dadosBase       = fscanf(arquivoBase,'%f',[(numEntradas + numSaidas), numBase]);

%Tratamento dos valores de saida da base
entradas = dadosBase(1:8,:);
saida = dadosBase(9:9,:);
saida=maskm(saida,(saida<1),0.4);
saida=maskm(saida,(saida>0.4),0.6);

dadosBase = [entradas; saida];

%Separando valores de entradas de Treinamento, Validaçao e Teste
entradasBase     = dadosBase(1:numEntradas, 1:numBase);
saidasBase       = dadosBase((numEntradas + 1):(numEntradas + numSaidas), 1:numBase);

%TREINAMENTO
dadosTreinamento = [dadosBase(:,1:460)];
entradasTreinamento = dadosTreinamento(1:numEntradas, 1:numTr);
saidasTreinamento  = dadosTreinamento((numEntradas + 1):(numEntradas +
numSaidas), 1:numTr);

%VALIDACAO
aux = numTr+1;
aux2 = numTr+numVal;
dadosValidacao = [dadosBase(:,aux:aux2)];
entradasValidacao  = dadosValidacao(1:numEntradas, 1:numVal);
saidasValidacao    = dadosValidacao((numEntradas + 1):(numEntradas + numSaidas),
1:numVal);
```

```
%TESTES
aux = numTr+numVal+1;
aux2 = numTr+numVal+numTeste;
dadosTeste = [dadosBase(:,aux:aux2)];
entradasTeste     = dadosTeste(1:numEntradas, 1:numTeste);
saidasTeste       = dadosTeste((numEntradas + 1):(numEntradas + numSaidas),
1:numTeste);

%   Fechando arquivo da base
fclose(arquivoBase);

%   Adquirindo valores maximo e minimos da base
pr = minmax(entradasBase);


%  Criando a rede

%Estrutura Utilizando o recalculo dos pesos de acordo com o Gradiente Descendente
rede = newff(pr,[numEscondidos1 numEscondidos2 numEscondidos3
numSaidas],{'logsig','logsig','logsig','logsig'},'traingd','learngd','sse');

%Estrutura Utilizando o recalculo dos pesos de acordo com a otimizaçao de
Levemberg-Marquardt
%rede = newff(pr,[numEscondidos1 numEscondidos2 numEscondidos3
numSaidas],{'logsig','logsig','logsig','logsig'},'trainlm','learngd','sse');

%Para a variaçao dos recalculos dos pesos basta apenas retirar o comentario de uma
funçao e coloca-la na outra
%%%%%%%%%%


%Inicializando a Rede
rede = init(rede);

%   Parametros do treinamento
rede.trainParam.epochs   = 1000;   % Maximo numero de iteracoes
rede.trainParam.lr       = 0.0001;   % Taxa de aprendizado
rede.trainParam.goal     = 0;     % Criterio de minimo erro de treinamento
rede.trainParam.max_fail = 1;     % Criterio de quantidade maxima de falhas na
validacao
rede.trainParam.min_grad = 0;     % Criterio de gradiente minimo
rede.trainParam.show     = 50 ;    % Iteracoes entre exibicoes na tela (preenchendo com
'NaN', nao exibe na tela)
rede.trainParam.time     = inf;   % Tempo maximo (em segundos) para o treinamento

echo on %Funcao que nao deixa aparecer os comandos efetuados dentro do MatLab,
somente os resultados por ele apresentado

%   Treinando a rede
fprintf('\nTreinando ...\n')

conjuntoValidacao.P = entradasValidacao; % Entradas da validacao
conjuntoValidacao.T = saidasValidacao;   % Saidas desejadas da validacao
```

```
FPRINTF('%6.5F\T\T',ENTRADASTREINAMENTO);

 [REDENOVA,DESEMPENHO,SAIDASREDE,ERROS] =
TRAIN(REDE,ENTRADASTREINAMENTO,SAIDASTREINAMENTO,[],[],CONJUNTOVALIDACAO);


%   TESTANDO A REDE
FPRINTF('\NTESTANDO ...\N');

[SAIDASREDETESTE,PF,AF,ERROSTESTE,DESEMPENHOTESTE] =
SIM(REDENOVA,ENTRADASTESTE,[],[],SAIDASTESTE);

ECHO OFF
TEMPO = TOC;

%RESULTADOS
FPRINTF('SSE PARA O CONJUNTO DE TREINAMENTO : %6.5F
\N',DESEMPENHO.PERF(LENGTH(DESEMPENHO.PERF)));
FPRINTF('SSE PARA O CONJUNTO DE VALIDACAO: %6.5F
\N',DESEMPENHO.VPERF(LENGTH(DESEMPENHO.VPERF)));
FPRINTF('SSE PARA O CONJUNTO DE TESTE: %6.5F \N',DESEMPENHOTESTE);


PESOSAIDA = REDE.IW{1,1};
PESOS = PESOSAIDA(4,:);
FPRINTF('PESOS DA REDE: \N'); PESOS

THRESHOULD=REDE.B{1};   % MOSTRA O VALOR TRESHOULD
THRESHOULD=THRESHOULD(1,:);
FPRINTF('VALOR DE THRESHOULD : %F \N',THRESHOULD);

FPRINTF('\N\N\NTEMPO DE PROCESSAMENTO : %F \N',TEMPO);
FPRINTF('\N\N\N\NFINALIZANDO O TREINAMENTO...\N\N');
```

ANEXO I

```
CLEAR;
CLC;
TIC;
%-------------------------------------------------------------
%AS ENTRADAS ABAIXOS ESTAO COMENTADAS PARA SE TER VALORES PADROES
%W(1) = INPUT('PESO 1: ');
%W(2) = INPUT('PESO 2: ');
%W(3) = INPUT('PESO 3: ');
%W(4) = INPUT('PESO 4: ');
%W(5) = INPUT('PESO 4: ');
%W(6) = INPUT('PESO 4: ');
%W(7) = INPUT('PESO 4: ');
%W(8) = INPUT('PESO 4: ');

%N = INPUT('TAXA DE APRENDIZADO: ');

%TETA = INPUT('VALOR THRESHLD: ');
%-------------------------------------------------------------

ARQUIVOBASE = LOAD('PIMA4.DATA');
W = [0.2 -0.1 0.4 -0.3 0.1 -0.2 0.5 -0.3];
N=0.00001;
TETA=0.3;

P_AUX = ARQUIVOBASE(:,1:8);
P = P_AUX';

T_AUX = ARQUIVOBASE(:,9:9);
T = T_AUX';

PR = MINMAX(ARQUIVOBASE(1:8,:));

REDE = NEWP( [PR],1,'HARDLIMS');

REDE.IW{1,1} = W;

REDE.B{1} = TETA;

REDE.TRAINPARAM.LR = N;
REDE.TRAINPARAM.EPOCHS = 1000;
REDE = TRAIN(REDE,P,T);

TETA2=REDE.B{1}*(-1);

PESOS = REDE.IW{1,1}

A=SIM(REDE,P);

TOC;
```