COPPE
Instituto Alberto Luiz Coimbra de Pós-Graduação e Pesquisa de Engenharia
UFRJ

OTIMIZAÇÃO DO ALGORITMO DE BACKPROPAGATION PELO USO DA FUNÇÃO DE ATIVAÇÃO BI-HIPERBÓLICA

Geraldo Azar Miguez

Tese de Doutorado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Doutor em Engenharia de Sistemas e Computação.

Orientador: Nelson Maculan Filho

Rio de Janeiro

Setembro de 2012

OTIMIZAÇÃO DO ALGORITMO DE BACKPROPAGATION PELO USO DA FUNÇÃO DE ATIVAÇÃO BI-HIPERBÓLICA

Geraldo Azar Miguez

TESE SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Examinada por:

_____
Prof. Nelson Maculan Filho, D.Sc.

_____
Prof. Adilson Elias Xavier, D.Sc.

_____
Prof. Yuri Abitbol de Menezes Frota, D.Sc.

_____
Prof. Michael Ferreira de Souza, D.Sc.

_____
Prof. Felipe Maia Galvão França, D.Sc.

_____
Prof. Luiz Satoru Ochi, D.Sc.

RIO DE JANEIRO, RJ - BRASIL
SETEMBRO DE 2012

Miguez, Geraldo Azar

Otimização do Algoritmo de Backpropagation pelo Uso da Função de Ativação Bi-Hiperbólica / Geraldo Azar Miguez. – Rio de Janeiro: UFRJ/COPPE, 2012.

XI, 97 p.: il.; 29,7 cm.

Orientador: Nelson Maculan Filho

Tese (doutorado) – UFRJ/ COPPE/ Programa de Engenharia de Sistemas e Computação, 2012.

Referências Bibliográficas: p. 94-97.

1. Otimização. 2. Redes Neurais. 3. Algoritmo de Backpropagation. 4. Função Bi-hiperbólica. I. Maculan Filho, Nelson. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Titulo.

DEDICATÓRIA

Aos motivos da minha vida:
Lídia, Ricardo e Eduardo

AGRADECIMENTOS

A Deus.

Aos meus pais que me ensinaram a perseverar sempre.

Aos meus irmãos, Luiz, Luzia e Inês, pelo apoio e incentivo.

Ao Professor Nelson Maculan pelos ensinamentos e exemplo desde a minha graduação.

Ao Professor Adilson Elias Xavier pela ajuda e paciência na elaboração deste trabalho.

Ao Professor Felipe França pelo apoio nestes tempos turbulentos.

Aos Professores Michael Ferreira de Souza, Yuri Abitbol de Menezes Frota e Luiz Satoru Ochi pela participação na minha banca de doutorado.

À Doutora Alyne Escobar, minha nora, pelo incentivo.

Aos professores e funcionários do Programa de Engenharia de Sistemas e Computação pela ajuda e carinho com que me trataram durante esta jornada.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Doutor em Ciências (D.Sc.)

OTIMIZAÇÃO DO ALGORITMO DE BACKPROPAGATION PELO USO DA FUNÇÃO DE ATIVAÇÃO BI-HIPERBÓLICA

Geraldo Azar Miguez

Setembro/2012

Orientador: Nelson Maculan Filho

Programa: Engenharia de Sistemas e Computação

O Algoritmo de Backpropagation é uma das ferramentas mais utilizadas para o treinamento de Redes Neurais Artificiais. Entretanto, em algumas aplicações práticas, ele pode ser muito lento. Para permitir uma utilização mais ampla, muitas técnicas têm sido discutidas para acelerar o seu desempenho. Este trabalho apresenta uma nova estratégia baseada no uso da Função Bi-Hiperbólica, que oferece maior flexibilidade e uma avaliação computacional mais rápida. A eficiência e a capacidade de discriminação da metodologia proposta são demonstradas através de um conjunto de experimentos computacionais com problemas tradicionais da literatura.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.)

OPTIMIZING THE BACKPROPAGATION ALGORITHM USING OF BI-HYPERBOLIC ACTIVATION FUNCTION

Geraldo Azar Miguez

September/2012

Advisor: Nelson Maculan Filho

Department: Systems and Computing Engineering

The backpropagation algorithm is one of the most used tools for training artificial neural networks. However, in some practical applications, it may be very slow. Many techniques have been discussed so as to speed up the performance of this algorithm and allow its use in an even broader range of applications. This paper presents a new strategy based on the use of the Bi-hyperbolic function which offers more flexibility and a faster evaluation time. This use also makes the process of designing and configuring the network easier and faster. The efficiency and the discrimination capacity of the proposed methodology are shown through a set of computational experiments done with traditional problems of the literature.

SUMÁRIO

LISTA DE FIGURAS

Capítulo 1 Introdução

1.1. Contexto

Diversas aplicações de sistemas de inteligência artificial vêm, cada vez mais, sendo usados no nosso dia a dia. Segundo Taylor (2006), uma Rede Neural Artificial (RNA) é um desses sistemas que procura reproduzir o que se acredita que seja a modo pelo qual os seres humanos processam e armazenam as informações. Procura-se, desta forma, obter vantagens com relação aos sistemas tradicionais. O processamento biológico de informações é considerado, em primeiro lugar, como sendo robusto e tolerante a falhas, pois, mesmo com a perda diária de milhares de neurônios, continua funcionando sem uma deteriorização significativa associada a este fato. Outra característica importante buscada é a flexibilidade, pois quando expostos a um novo ambiente nós aprendemos e nos adaptamos. Podemos, também, lidar com informações incompletas, imprecisas, probabilísticas e inconsistentes. Para que um sistema computacional tradicional apresente estas características é necessária uma programação complexa e sofisticada, somente possível para ambientes em que seja possível uma análise detalhada de todos os dados.

As RNAs extraem o seu poder computacional de sua estrutura distribuída paralelamente e de sua habilidade de aprender e generalizar. Esta generalização permite que se obtenham saídas adequadas para entradas que não estavam presentes no treinamento. São essas características que permitem que se considere como sendo possível, com a utilização destas redes, a resolução de problemas complexos que atualmente são intratáveis.

Para possibilitar este comportamento, além de um bom projeto de sua estrutura, é necessário um algoritmo eficiente e eficaz de aprendizagem.

Este problema de treinamento de RNAs pode ser visto como um problema de otimização, onde é desejado minimizar o erro quadrático médio entre a saída desejada e aquela produzida pela RNAs. Existem vários tipos de algoritmos para treinamento das RNAs, alguns com forte supervisão e outros com necessidade reduzida de informações providas pelo meio ambiente. Em geral, sabe-se que nenhum dos algoritmos de treinamento de Redes Neurais Artificiais é completo. Alguns algoritmos apresentam boas características tais como alta velocidade de convergência, mas o erro quadrático médio na saída pode ainda ser relativamente alto. Outros algoritmos atingem pequeno erro quadrático médio na saída, porém apresentam, em geral, baixa velocidade de convergência. Existe ainda o problema de generalização da solução e de inicialização dos pesos da rede.

Um dos algoritmos mais utilizados no treinamento de RNAs é o conhecido por Backpropagation, que é um método baseado no uso de gradientes. Ele apresenta algumas limitações na sua utilização, dificultado a sua aplicação de uma forma mais ampla. Além da possibilidade de convergência para um mínimo pouco profundo, apresenta uma lentidão muito grande, mesmo nos casos em que consegue atingir o seu objetivo de apresentar um erro dentro dos limites desejados. Esta lentidão no processamento dificulta a sua utilização em uma gama maior de aplicações práticas, em especial em aplicações de médio e grande porte. Este é um dos problemas constantemente apresentados na literatura, como pode ser visto em Otair (2005) ou em Schiffmann (1994).

Um dos fatores possivelmente responsável pela lentidão deste processo de convergência é a função de ativação usada em seus neurônios, pois, sendo o processo de aprendizagem da rede essencialmente iterativo, uma função mais lenta para ser calculada torna todo o procedimento lento.

Esta lentidão no processo de convergência ocorre, especialmente, para redes com mais de uma camada oculta. A razão para isto pode estar na saturação da função de ativação usada para as camadas ocultas e de saída, pois, uma vez que a saturação de uma unidade ocorre, o gradiente descendente assume valores muito pequenos, mesmo quando o erro de saída ainda é grande.

Este problema de otimizar a eficiência e a taxa de convergência do algoritmo de backpropagation tem sido objeto de interesse de muitos pesquisadores.

O trabalho desenvolvido nas próximas seções apresenta resultados bastante satisfatórios em ambos os aspectos desta otimização.

1.2. Objetivo

O objetivo deste trabalho é avaliar a possibilidade de otimização do algoritmo de Backpropagation para o treinamento de redes neurais artificiais através do uso de uma nova função de ativação. Será usada a Função Bi-Hiperbólica Simétrica que apresenta características que atendem às necessidades deste algoritmo e oferece as vantagens de possibilitar uma maior flexibilidade na representação dos fenômenos modelados. Ela faz uso de dois parâmetros, um a mais do que as funções tradicionalmente utilizadas para esta finalidade, possibilitando um melhor enfrentamento do problema da saturação, além de permitir um tratamento mais adequado para evitar os mínimos locais. Outra vantagem, observada empiricamente, é a de ser computacionalmente 24,8% mais rápida de ser avaliada do que a Função Logística. Este resultado foi obtido por Xavier (2005) através de simulação

programada na linguagem FORTRAN, usando o compilador COMPAQ, em um computador tipo IBM PC, com 800 Mhz de clock.

Outra vantagem do uso desta Função Bi-Hiperbólica Simétrica, citada por Xavier (2005), reside em possibilitar, por sua maior flexibilidade, a capacidade de poder aproximar qualquer função de uma forma mais sintética e, com isso, permitir a utilização de um menor número de neurônios, o que melhora ainda mais o desempenho do Algoritmo Backpropagation, agindo diretamente na topologia da rede.

Foi desenvolvido um protótipo em MATLAB que, através de uma interface gráfica, permitiu a obtenção de resultados altamente favoráveis, apresentados posteriormente neste trabalho.

Os objetivos específicos incluem:

- Elaborar e implementar um modelo computacional para o treinamento de uma rede neural artificial do tipo Multilayer Perceptron que utilize diferentes tipos de função de ativação e arquiteturas básicas;
- Utilizar metodologias de avaliação do treinamento de redes neurais para permitir comparar as taxas de convergência e de generalização dos modelos treinados e validados com funções de ativação específicas;
- Obter medidas de tempo de processamento para comparar as vantagens oferecidas por cada modelo usado.
- Avaliar, através da alteração da arquitetura básica dos modelos usados, qual a redução no número de neurônios possível para cada tipo de função de ativação usada;
- Efetivar o treinamento e a validação dos modelos desenvolvidos pela utilização de bases de dados reconhecidas e disponíveis na literatura.

1.3. Justificativas

A utilização de modelos de inteligência artificial vem crescendo nas mais diversas áreas do conhecimento. As redes neurais artificiais desempenham um papel importante neste ambiente e uma melhoria no seu desempenho traria a possibilidade de utilização, ainda mais ampla, no tratamento de problemas reais.

Sendo o algoritmo conhecido por Backpropagation um dos mais utilizados no treinamento destas RNAs, a diminuição das limitações na sua utilização traria como consequência a sua aplicação de uma forma mais ampla em uma gama maior de aplicações reais práticas, em especial em aplicações de médio e grande porte, como já destacaram Otair (2005) e Schiffmann (1994).

Um dos fatores possivelmente responsável pela lentidão deste processo de convergência é a função de ativação usada em seus neurônios, pois, sendo o processo de aprendizagem da rede essencialmente iterativo, uma função mais lenta para ser calculada torna todo o procedimento lento. Outro problema associado a esta taxa de convergência relativamente lenta, como aparece em Lee (1991), pode ser atribuído à saturação da função de ativação usada para as camadas ocultas e de saída. Uma vez que a saturação de uma unidade ocorra, o gradiente descendente assume valores muito pequenos, mesmo quando o erro de saída ainda é grande, aumentando o número de iterações, ou épocas, necessárias ao treinamento.

A utilização da Função Bi-Hiperbólica Simétrica pode contribuir para a solução destes problemas. A utilização de dois parâmetros, um a mais do que nas demais funções de ativação, fornece a essa função uma maior flexibilidade, possibilitando a representação mais adequada dos fenômenos modelados com redes neurais. Numa rede neural multicamadas, por exemplo, essa maior flexibilidade possibilita à função de ativação Bi-Hiperbólica Simétrica o poder de aproximar qualquer função de uma forma mais sintética, com menor número de neurônios. Isto contribui, também, para a otimização do processo de aprendizagem e utilização da rede neural, pela redução da sua arquitetura necessária, conforme ressalta Xavier (2005).

1.4. Estrutura da Tese

Os demais capítulos da tese estão estruturados da forma descrita a seguir.

A partir do segundo capítulo, é apresentada a fundamentação teórica deste trabalho, delineando uma visão histórica do surgimento das redes neurais artificiais, as suas principais características e como é feito o desenvolvimento de um projeto de utilização de um sistema com uma rede neural artificial. Também são descritas as principais aplicações destes sistemas encontradas na literatura.

O terceiro capitulo apresenta os processos principais de aprendizagem da rede neural artificial e o funcionamento e as características do Algoritmo de Backpropagation.

O quarto capítulo introduz os métodos principais usados na verificação e validação das redes neurais artificiais.

No quinto capítulo será apresentado o protótipo desenvolvido para o treinamento, a validação e a avaliação comparativa dos modelos com as funções de ativação usadas.

No sexto capítulo, serão apresentados os processamentos realizados e a análise dos resultados obtidos, corroborando as hipóteses estabelecidas no modelo teórico.

No sétimo capítulo estão as considerações finais do presente trabalho, as contribuições que podem advir dele e propostas para trabalhos futuros.

Capítulo 2 Redes Neurais Artificiais

2.1. Introdução

Diversas aplicações de sistemas de inteligência artificial estão cada vez mais sendo usados no nosso dia a dia. Uma Rede Neural Artificial (RNA) é um desses sistemas que procura reproduzir o que se acredita que seja a modo pelo qual os seres humanos processam e armazenam as informações.

Ao procurar reproduzir o modo humano de tratar informações objetivamos obter vantagens, uma vez que o processamento biológico de informações é considerado primordialmente como sendo robusto e tolerante a falhas. Mesmo com a perda diária de milhares de neurônios, continuamos a funcionar sem uma deteriorização significativa associada a este fato. Além disso, como pode ser visto em Haykin (2001), outro ponto importante é a flexibilidade, pois, quando expostos a um novo ambiente, nós nos adaptamos pelo aprendizado. Podemos, também, lidar com informações imprecisas, probabilísticas e inconsistentes. Temos a capacidade inata de lidar com a incerteza. Para um sistema computacional tradicional, este desempenho exigiria uma programação extensa e sofisticada e somente seria factível para um contexto em que todos os dados pudessem ser detalhadamente analisados. Outro ponto favorável é que tudo isso é feito em uma estrutura altamente paralela, pequena, compacta e com o consumo e a dissipação de pouca energia.

As Redes Neurais Artificiais extraem o seu poder computacional de sua estrutura distribuída paralelamente e de sua habilidade de aprender e generalizar. Esta generalização permite que se obtenham saídas adequadas para entradas que não estavam presentes no treinamento. São essas características que permitem que se considere como sendo possível a resolução de problemas complexos que atualmente são intratáveis pelos métodos tradicionais.

Uma Rede Neural Artificial funciona pela criação de ligações entre unidades de processamento matemático, chamadas de Neurônios Artificiais. Como destaca Taylor (2006), o conhecimento é codificado na rede pela força destas conexões entre diferentes neurônios, chamadas de Peso, e pela criação de camadas de neurônios que trabalham em paralelo. O sistema aprende através de um processo de determinação do número de neurônios, ou nós na rede, e pelo ajuste dos pesos destas conexões, o que é feito com base em dados usados para o treinamento da rede. Existem diversas formas de treinamento como, por exemplo, o treinamento supervisionado, onde são apresentados à rede pares de dados de entrada e de saída

desejada e a RNA procura encontrar a função que melhor possa representar esta associação.

Podemos considerar dois tipos principais de redes neurais. As Redes Neurais Fixas, ou Não-adaptativas que, após serem submetidas ao treinamento, têm seus parâmetros fixados e a sua estrutura interna permanece sem alterações durante a sua fase operacional. Um exemplo de aplicação destas redes pode ser encontrado em sistemas de classificação de produtos em uma linha de produção. As Redes Neurais Dinâmicas, ou Adaptativas, ficam continuamente mudando sua estrutura interna, se adaptando aos novos dados do ambiente. Elas são usadas em situações onde o sistema deve aprender novas informações durante o uso. São úteis em aplicações onde ocorrem cenários não previstos como, por exemplo, falhas em equipamentos.

Na próxima seção será apresentado um breve histórico do surgimento e da evolução das redes neurais artificiais para permitir um melhor entendimento desta tecnologia.

2.2. Histórico

O estudo das redes neurais artificiais pode ser considerado como tendo início em 1943, quando Warren McCulloch e Walter Pitts (1943) publicaram um artigo no *Bulletin of Mathematical Biophysics* intitulado “A Logical Calculus of the Ideas Immanent in Nervous Activity”. O neurônio de McCulloch era bastante simples, apresentando como saída pulso ou não pulso, e as suas entradas tinham um ganho arbitrário, podendo ser excitatórias ou inibitórias. Para determinar a saída do neurônio, calculava-se a soma ponderada das entradas, usando-se os respectivos ganhos como fatores de ponderação. Os excitatórios eram tratados como positivos e os inibitórios como negativos. Se este resultado fosse maior ou igual a um determinado limiar a saída do neurônio era pulso e, no caso contrário, era não pulso. Com isso, conforme exemplifica Aleksander (1995), eles implementaram diversas funções booleanas usando ganhos iguais a ½ e limiares iguais a 1.

A proposta de McCulloch e Pitts era, como resume Kóvacs (1996), que a inteligência seria equivalente ao cálculo de predicados que, por sua vez, poderia ser implementado por funções booleanas. Por outro lado, o sistema nervoso seria composto por redes de neurônios que, com as devidas simplificações, teriam a capacidade básica de implementar estas funções booleanas. Procuravam, assim, estabelecer de forma científica a ligação entre a inteligência e a atividade neural.

No final da década de 1950, Rosenblatt (1958) criou o Perceptron, que era uma rede de múltiplos neurônios do tipo discriminadores lineares. Ele se propunha a

resolver o problema de como escolher os ganhos sinápticos para a implementação de uma função discriminatória arbitrária. Para ele, baseado no fato de que sistemas nervosos biológicos possuem a propriedade de serem capazes de aprender uma função, deveria existir uma forma de ensinar a uma rede artificial para que esta pudesse aprender a função desejada. O modo mais intuitivo de se fazer isto é o treinamento através de exemplos, no qual são apresentados exemplos de comportamento à rede onde para cada estímulo de entrada é associada uma saída desejada. Conforme é apresentado por Aleksander (1995), Rosenblatt construiu um Perceptron deste tipo, o Mark I, que possuía uma matriz de 20 X 20 fotocélulas que funcionavam como sendo os componentes do vetor de entrada. Com isso, o Mark I reconhecia caracteres grafados nesta matriz. Este treinamento por exemplos é feito fornecendo-se um conjunto de treinamento formado pelas entradas e respectivas saídas. Após a inicialização do discriminador linear com parâmetros arbitrários, estes vão sendo ajustados por algum algoritmo de forma a se obter a convergência dos valores de saída do discriminador com os valores do conjunto de treinamento. Para que se assegure esta convergência de maneira eficiente é preciso encontrar este algoritmo de ajuste dos parâmetros.

Hagan (1996) mostra que, em 1949, Hebb propôs um princípio pelo qual o aprendizado em sistemas nervosos complexos poderia ser reduzido a um processo puramente local, em que a intensidade das conexões sinápticas é alterada apenas em função dos erros detectáveis localmente. Este princípio local foi adaptado ao discriminador linear gerando um algoritmo no qual os parâmetros são ajustados a cada exemplo apresentado, sendo que os novos parâmetros são obtidos com base na entrada apresentada e no erro de saída encontrado ao usar os parâmetros anteriores do discriminador. Este é o Princípio Hebbiano de Treinamento que é considerado uma das primeiras leis de aprendizagem de redes neurais.

O Perceptron começava a ser considerado como a base de uma possível inteligência artificial. Ele apresentava características qualitativas superiores ao modelo de McCulloch e Pitts como, por exemplo, poder assumir valores contínuos e apresentar uma lei de treinamento. Entretanto, em 1969, Marvin Minsky e Seymour Pappert (1969) apresentaram um trabalho em que argumentavam que redes de Perceptrons não contavam com possibilidades reais, principalmente devido as suas limitações básicas. Para eles, Rosenblatt não conseguiu estender para as redes a lei de treinamento do Perceptron e a sua respectiva prova. Segundo Kasabov (1998), esta publicação foi o marco que paralisou, a partir da década de 1970, as pesquisas nesta área. Elas somente foram retomadas de forma mais expressiva no início da década de 1980 a partir do trabalho de Hopfield em memórias associativas.

Praticamente na mesma época em que o projeto do Perceptron era desenvolvido, Widrow desenvolveu, na Universidade de Stanford, um modelo neural linear muito simples, denominado ADALINE (“Adaptive Linear Element”) e posteriormente o MADALINE (“Multiple Adaline”). Por serem muito simples, estes modelos só tiveram relevância acadêmica. Entretanto, Widrow criou um princípio de treinamento extremamente poderoso, conhecido como a Regra Delta ou Método do Gradiente que, posteriormente, foi generalizado para redes mais elaboradas. Este princípio foi a base para o que é considerado como o mais poderoso método de treinamento de redes neurais: o Método de Retropropagação do Erro. O Algoritmo de Retropropagação do Erro, conhecido por Algoritmo Backpropagation para redes de neurônios de múltiplas camadas, foi desenvolvido e popularizado por Rummelhart e pode ser considerado como o fato que consolidou o paradigma conexionista. Inicialmente, foram utilizadas redes de neurônios com funções de ativação semi-lineares no desenvolvimento deste algoritmo. Ele parte do erro da saída e caminha com este erro retrocedendo em direção à entrada, atualizando os pesos de cada conexão entre os neurônios.

A descrição deste algoritmo, como apresenta Braga (2000), mostrou que a visão de Minsky e Papert sobre o Perceptron era muito pessimista. A partir de meados da década de 1980, houve uma nova explosão de interesse pelas Redes Neurais Artificiais, impulsionada pelo avanço da tecnologia, em especial da microeletrônica, e pelo fato de que a escola simbolista não conseguiu avanços significativos na resolução de alguns problemas simples para um ser humano. A seguir são apresentadas as principais características de uma rede neural artificial de forma a facilitar o entendimento do seu funcionamento.

2.3. Características das RNAs

O poder computacional de uma RNA é devido basicamente à sua estrutura paralela, pesadamente distribuída, e à sua habilidade de aprender e, consequentemente, generalizar.

Entre as características relevantes das Redes Neurais Artificiais, entre as apresentadas por Haykin (2001), podemos destacar as seguintes:

- Possibilidade de modelar o comportamento não-linear dos fenômenos físicos responsáveis pela geração dos dados de entrada;
- Necessidade de pouco conhecimento estatístico sobre o ambiente no qual a rede está inserida;

- Capacidade de aprendizagem, a qual é atingida através de uma sessão de treinamento com exemplos entrada/saída que sejam representativos do ambiente;
- Habilidade de aproximar qualquer mapeamento entrada/saída de natureza contínua;
- Capacidade de adaptação. As RNAs são ferramentas extremamente flexíveis em um ambiente dinâmico. Elas têm a capacidade de aprender rapidamente padrões complexos e tendências presentes nos dados e de se adaptar rapidamente às mudanças;
- Capacidade de generalizar, o que permite às RNAs um desempenho satisfatório (produzir saídas adequadas) em resposta a dados desconhecidos (não pertencentes ao conjunto de treino, mas que estejam em sua vizinhança);
- Tolerância a falhas, característica que permite à rede continuar a apresentar resultados aceitáveis, mesmo no caso de falha de alguns neurônios;
- Representação do conhecimento pela própria estrutura da RNA e pelo seu estado de ativação. Cada neurônio da rede é potencialmente afetado pela atividade global de todos os outros neurônios na rede. Consequentemente, a informação contextual é tratada com naturalidade pelas RNAs;
- Possibilidade da implementação em VLSI (Very Large Scale Integration). Esta característica permite considerar elevado grau de paralelismo no projeto da rede. A natureza fortemente paralela das RNAs as tornam potencialmente rápidas para computar determinadas tarefas e, esta mesma característica, possibilita que sejam implementadas usando tecnologia VLSI.

Para que seja possível a obtenção destes benefícios apresentados por suas características, o projeto adequado da rede deve levar em conta, principalmente, os fatores apresentados a seguir.

2.4. Projeto da RNA

2.4.1. Neurônio Artificial

Os neurônios são considerados as estruturas que constituem o cérebro. O cérebro tem a capacidade de organizar seus componentes estruturais de forma a desempenhar certas operações, tais como reconhecimento de padrões, controle de movimento entre outras, muitas vezes mais rápido do que o mais rápido computador digital existente.

O neurônio biológico é basicamente o dispositivo computacional elementar do sistema nervoso, que possui entradas e uma saída, conforme apresentado simplificadamente na Figura 1, reproduzida de Hagan (1996). As entradas ocorrem através das conexões sinápticas, que conectam a árvore dendrital aos axônios de outras células nervosas. Os sinais que chegam dos axônios de outras células nervosas são pulsos elétricos conhecidos como impulsos nervosos ou potenciais de ação e constituem a informação que o neurônio processará de alguma forma para produzir como saída um impulso nervoso no seu axônio.

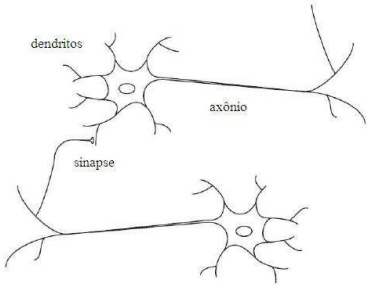


Figura 1: Neurônio do sistema nervoso central dos vertebrados

Os Neurônios Artificiais são as unidades de processamento das RNAs e, portanto, são fundamentais para o seu funcionamento. Estas unidades de processamento, ou nós, como ressalta Haykin (2001), são simplificações do conhecimento que se tinha do neurônio biológico, feitas por McCulloch e Pitts. O modelo desenvolvido, apresentado na Figura 2, apresenta vários terminais de entrada (X), representando os dendritos, e um terminal de saída (Y), representando o axônio. As sinapses têm seu comportamento simulado pelo acoplamento de pesos (W) a cada terminal de entrada do neurônio artificial, que podem assumir valores positivos ou negativos, emulando sinapses inibitórias ou excitatórias. A ativação do neurônio artificial é obtida através da aplicação de uma Função de Ativação que pode ativar ou não a saída, dependendo da soma ponderada dos valores de cada entrada atingir um limiar pré-determinado. A função de ativação limita a faixa de amplitude permitida do sinal de saída a algum valor finito. Tipicamente, a amplitude normalizada da saída de um neurônio é restrita ao intervalo unitário fechado [0, 1] ou, alternativamente [-1, 1]. O modelo neural apresentado inclui uma polarização externa (*bias*), que tem o efeito de aumentar ou diminuir o argumento da Função de Ativação (φ), que define a saída do neurônio em termos do potencial de ativação.

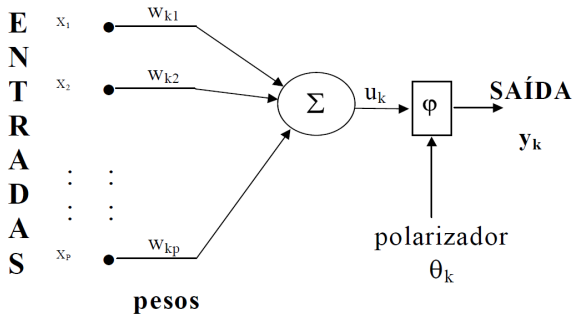


Figura 2: Neurônio Artificial

O neurônio é descrito, em termos matemáticos, por Haykin (2001) da seguinte forma:

$$u_k = \sum_{j=1}^{p} w_{kj} x_j \qquad (2.1)$$

$$v_k = u_k - \theta_k \qquad (2.2)$$

$$y_k = \varphi(v_k) \qquad (2.3)$$

Onde:

- $x_1, x_2, \ldots, x_p$ são os sinais de entrada;
- $w_{k1}, w_{k2}, \ldots, w_{kp}$ são os pesos sinápticos do neurônio k;
- u_k é a saída proveniente da combinação linear dos sinais de entrada e dos pesos obtida para os sinais de entrada;
- θ_k é o bias;
- $\varphi(\cdot)$ é a função de ativação;
- y_k é o sinal de saída do neurônio.

2.4.2. Funções de Ativação

Um dos componentes mais importantes do neurônio artificial é a sua Função de Ativação ou Transferência. Ela tem por objetivo limitar a amplitude válida do sinal de saída do neurônio a um valor finito.

Normalmente, esta amplitude normalizada se encontra em um intervalo fechado unitário [0, 1] ou, em alguns casos, [-1, 1].

As funções de ativação mais comumente utilizadas e disponibilizadas na literatura, entre outros autores por Xavier (2005), Haykin (2001), Duch (1999), e Ferreira (1997), são apresentadas a seguir. Também são descritas as suas derivadas, que têm grande importância no método de treinamento de Redes Neurais Artificiais conhecido como Algoritmo Backpropagation.

a) Função Degrau

$$\varphi_1(v) = \begin{cases} 1 & se \quad v \geq 0 \\ 0 & se \quad v < 0 \end{cases} \qquad (2.4)$$

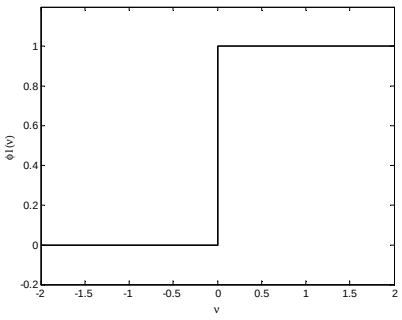


Figura 3: Função Degrau

A derivada desta função é $\varphi_1'(v) = 0$ para $\forall \quad v \neq 0$ e não é definida para $v = 0$. A descontinuidade na origem associada ao valor nulo da derivada nos demais pontos restringe muito a utilidade prática desta função.

Exemplo: Função AND

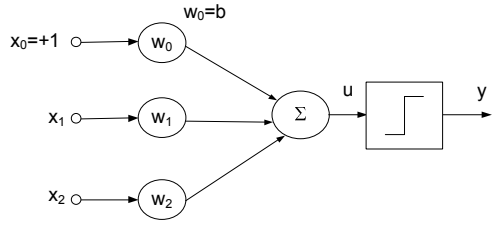


Figura 4: Rede Neural com bias

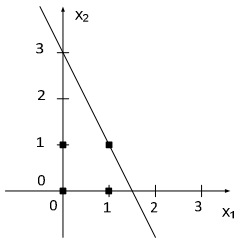


Figura 5: Representação Geométrica

Iniciando os pesos e o limiar em zero **w** = [0 0], ***b*** = *0* e ***a*** = *1*, tem-se ***w₁***=*2,* ***w₂*** = *1,* ***b*** = *-3* e a equação da reta $2x_1 + 1x_2 - 3 = 0$.

Temos como vetores de entrada e saída desejada:

$$\mathbf{X} = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix} \qquad \mathbf{d} = [0 \;\; 0 \;\; 0 \;\; 1]$$

b) **Função Patamar**

$$\varphi_2(v,b) = \begin{cases} 0, & se \;\; v \le -b; \\ (v+b)/2b, & se \;\; -b < v < b; \\ 1, & se \;\; v > b; \end{cases} \qquad (2.5)$$

Sendo $b = 1/(2\tan\alpha)$, onde α é o ângulo de inclinação.

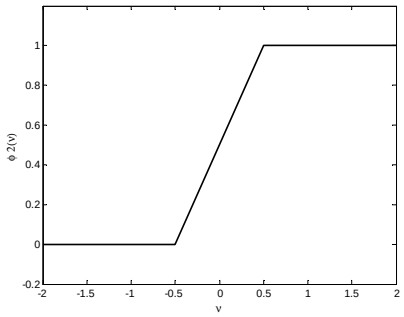


Figura 6: Função Patamar

A sua derivada não é definida nos pontos $v = -1/2$ e $v = 1/2$, nos demais valores assume:

$$\varphi_2'(v,b) = \begin{cases} 0, & para \;\; v < -b; \\ 1/2b, & para \;\; -b < v < b; \\ 0, & para \;\; v > b; \end{cases} \qquad (2.6)$$

A insensibilidade da derivada fora do intervalo (*-b, b*) limita consideravelmente o uso prático dessa função de ativação $\varphi_2(\bullet)$.

c) Funções Sigmoidais

As Funções Sigmoidais, conhecidas também como em forma de *S,* são encontradas na construção de diversos modelos nas mais variadas áreas, e muito aplicadas como função de ativação em neurônios artificiais. Elas são crescentes em todo intervalo de variação e não possuem pontos extremos com máximos e mínimos relativos. Possuem um ponto de inflexão onde ocorre a taxa máxima de variação da função e, até o ponto de inflexão, o gráfico representativo das funções sigmoidais apresenta concavidade para cima, com a derivada segunda positiva. No ponto de inflexão sua derivada segunda é nula, seguindo daí em diante com a concavidade para baixo e com a derivada segunda negativa. Cresce assintoticamente segundo a assíntota horizontal, o que leva a uma de suas características que é a saturação, ou seja, para valores grandes de argumento, a função opera numa região de saturação.

A utilização de funções de ativação Sigmoidais apresenta como vantagens terem uma não-linearidade fraca, já que o seu trecho central é quase linear, além do fato de suas derivadas serem fáceis de calcular. Permitem, também, uma interpretação da saída como taxa média de disparo, em vez de simplesmente indicar se o neurônio está ativado ou não.

Além do fato de poderem ser levadas a operar em uma região de saturação, as funções sigmoidais tradicionais apresentam a desvantagem de terem um elevado custo computacional para sua implementação e uso em sistemas de maior volume de dados. Isto se deve ao fato do principal algoritmo usado em sistemas computacionais para o seu cálculo, segundo Hahn (1993), ser o de expansão da Série de Taylor, apresentado a seguir:

$$e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots \qquad (2.7)$$

d) Função Logística

A função de ativação sigmoidal mais utilizada na construção de redes neurais artificiais é a Função Logística que é definida por:

$$\varphi_3(v,\alpha) = \frac{1}{1+e^{-\alpha v}} \qquad (2.8)$$

Onde, α é o parâmetro de declividade da função logística.

Segundo Xavier (2005), a Função Logística oferece a importante flexibilidade dada por sua inclinação na origem, $\varphi'_3(0,\alpha) = \alpha/4$, ser variável com o parâmetro α,

conforme ilustra a Figura 7. Além disso, a Função Logística apresenta propriedades de simetria e de completa diferenciabilidade, ou seja, pertence à classe de funções C^{∞}.

Com o uso da Equação 2.8 e pela variação do parâmetro α foram obtidos os gráficos da função logística de diferentes declividades. A curva com menor declividade foi obtida com o uso do menor valor de α utilizado no exemplo, correspondendo a $\alpha = 0{,}4$ e a com maior declive foi obtida com $\alpha = 1{,}2$. As curvas obtidas podem ser vistas na Figura 7.

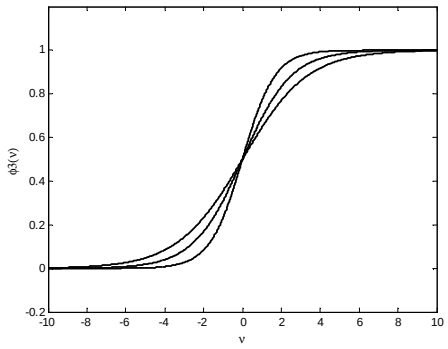


Figura 7: Função Logística – Efeito da variação do parâmetro α

A derivada da Função Logística é definida por:

$$\varphi_3'(v,\alpha) = \alpha\varphi_3(v,\alpha)(1-\varphi_3(v,\alpha)) \qquad (2.9)$$

Utilizando a Equação 2.9 e variando o parâmetro α, foram obtidas as curvas das derivadas da função logística de diferentes declividades. A curva com a maior altura corresponde à curva da função logística de maior declividade apresentada na Figura 7. Os valores de α utilizados foram $\alpha = 0{,}4$, $\alpha = 0{,}8$ e $\alpha = 1{,}2$. As curvas obtidas são apresentadas na Figura 8, correspondendo às curvas da função logística apresentadas anteriormente.

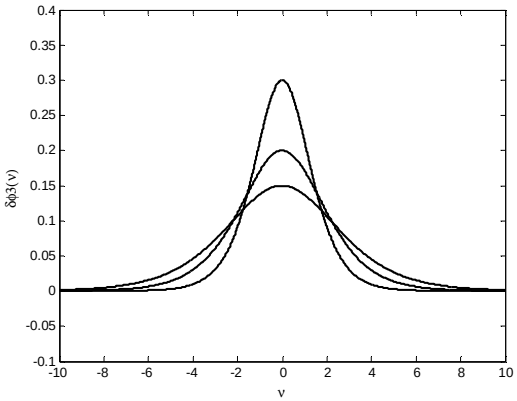


Figura 8: Derivadas da Função Logística variando o parâmetro α

e) Função Tangente Hiperbólica

Em alguns casos é interessante que a saída da função sigmoidal varie entre –1 e 1. Nestes casos utiliza-se a função tangente hiperbólica dada por:

$$\varphi_4(v) = \tanh(v) \qquad (2.10)$$

A função tangente hiperbólica apresenta as propriedades úteis de $\varphi_4(1) = 1$ e $\varphi_4(-1) = -1$. Também é utilizada uma variação desta equação, encontrada na literatura, como por exemplo em Kalman (1992), dada por:

$$\varphi_4(v) = a\tanh(bv) \qquad (2.11)$$

Onde, *a* e *b* são constantes e maiores que zero. Os valores indicados por Haykin (2001), para estas constantes são a = 1,7159 e b = 2/3.

Na realidade, segundo Haykin (2001), a Função Tangente Hiperbólica pode ser considerada como uma Função Logística re-escalada e modificada por um bias. A utilização dos parâmetros ***a*** e ***b*** serve para este tipo de adaptação, alterando a faixa de valores utilizados, como pode ser visto a seguir.

Com o uso da Equação 2.11 com o valor do parâmetro ***b*** = 2/3 e variando o valor do parâmetro ***a*** para ***a*** = 1; ***a*** = 1,7159 e ***a*** = 2,5 foram obtidos os gráficos da Função Tangente Hiperbólica, que podem ser vistas na Figura 9, onde pode ser observada a variação gerada no valor final da função pela alteração do parâmetro ***a*** mudando o intervalo de variação dos valores de saída.

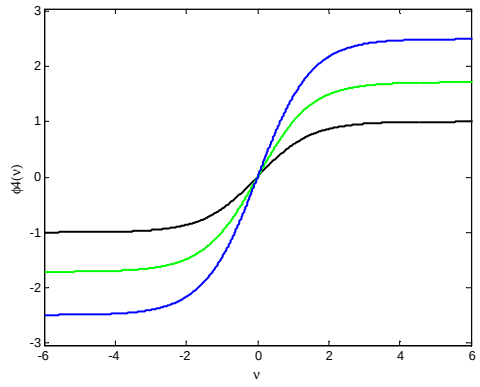


Figura 9: Função Tangente Hiperbólica variando o parâmetro ***a***

A derivada da Função Tangente Hiperbólica é definida por:

$$\varphi'_4(v) = ab(1 - \tanh^2(bv)) \qquad (2.12)$$

Com a utilização da Equação 2.12 e com o valor do parâmetro ***b*** = 2/3 e variando o valor do parâmetro ***a*** para ***a*** = 1; ***a*** = 1,7159 e ***a*** = 2,5 foram obtidos os gráficos das derivadas da Função Tangente Hiperbólica, que podem ser vistas na

Figura 10, onde pode ser observada a variação gerada no valor final da derivada da função pela alteração do parâmetro ***a***, onde a maior variação do parâmetro corresponde à curva de maior altura da derivada.

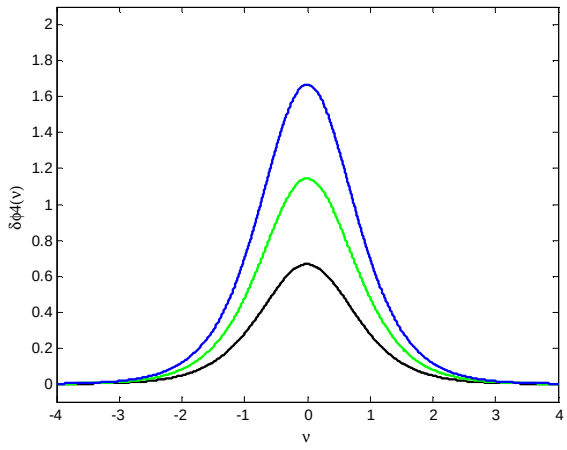


Figura 10: Derivadas da Função Tangente Hiperbólica variando ***a***

Com o uso da Equação 2.11 com o valor do parâmetro ***a*** igual a 1,7159 e variando o valor do parâmetro ***b*** para ***b*** = 0,5; ***b*** = 1,0 e ***b*** = 1,5 foram obtidos os gráficos da Função Tangente Hiperbólica, que podem ser vistos na Figura 11, onde pode ser observada a variação gerada na inclinação da curva da função pela alteração do parâmetro ***b***, onde a curva de menor inclinação corresponde ao menor valor atribuído ao parâmetro ***b***.

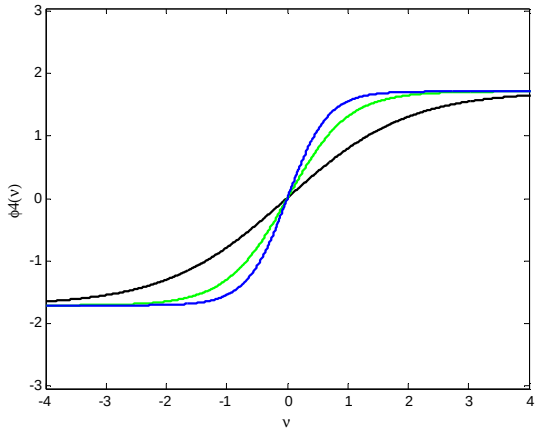


Figura 11: Função Tangente Hiperbólica variando o parâmetro ***b***

Com o uso da Equação 2.12 com o valor do parâmetro ***a*** igual a 1,7159 e variando o valor do parâmetro ***b*** para ***b*** = 0,5; ***b*** = 1,0 e ***b*** = 1,5 foram obtidos os gráficos das derivadas da Função Tangente Hiperbólica, que podem ser vistos na Figura 12, onde pode ser observada a variação gerada na altura da curva da derivada

da função pela alteração do parâmetro ***b***, onde a curva de menor altura corresponde ao menor valor atribuído ao parâmetro ***b***.

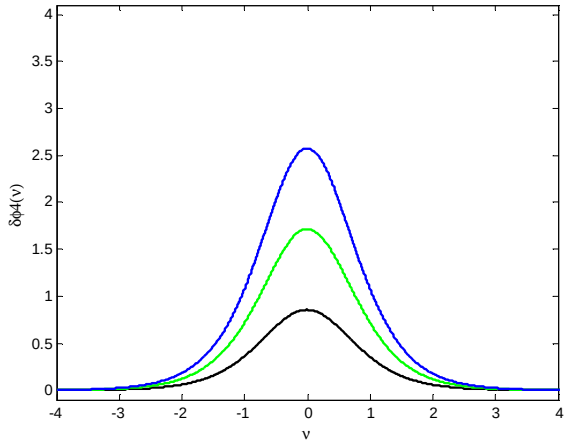


Figura 12: Derivadas da Função Tangente Hiperbólica variando ***b***

f) **Função de Elliott**

Esta função apresentada por Elliott (1993) e Xavier (2005), é definida por:

$$\varphi_5(v) = \left(\frac{v}{1+|v|} + 1 \right) / 2 \qquad (2.13)$$

A sua derivada é definida por:

$$\varphi_5'(v) = \frac{1}{2(1+|v|)^2} \qquad (2.14)$$

Ela apresenta a inclinação de sua derivada na origem invariante, $\varphi'_5(0)=1/2$, independente de qualquer transformação de escala, fato que limita fortemente a flexibilidade dessa função e seu decorrente uso prático.

Com o uso da Equação 2.13 foi obtido o gráfico desta função que pode ser observado na Figura 13.

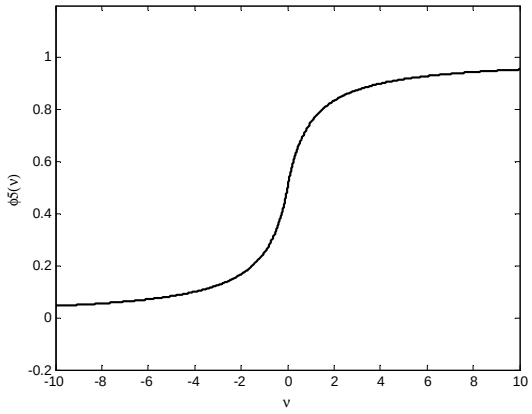


Figura 13: Função de Elliott

Com o uso da equação 2.15 foi obtido o gráfico da derivada da Função de Elliott, que é apresentado na Figura 14.

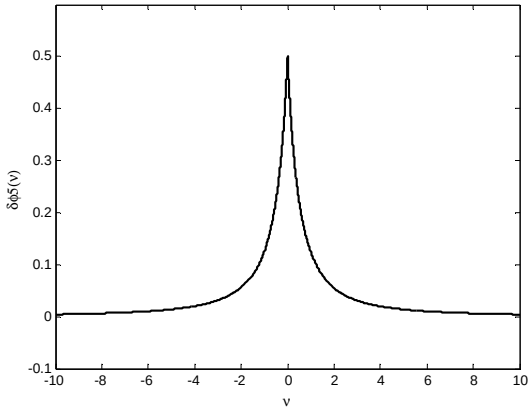


Figura 14: Derivada da Função de Elliott

g) **Função Bi-Hiperbólica Assimétrica**

A função Bi-Hiperbólica Assimétrica, apresentada por Xavier (2005), em sua forma mais geral é definida por:

$$\varphi_6(v,\lambda,\tau_1,\tau_2) = \sqrt{\lambda^2(v+\frac{1}{4\lambda})^2+{\tau_1}^2} - \sqrt{\lambda^2(v-\frac{1}{4\lambda})^2+{\tau_2}^2} + \frac{1}{2} \qquad (2.15)$$

Sendo sua derivada definida por:

$$\varphi_6{}'x(v,\lambda,\tau_1,\tau_2) = \frac{\lambda^2\left(v+\frac{1}{4\lambda}\right)}{\sqrt{\lambda^2\left(v+\frac{1}{4\lambda}\right)^2+{\tau_1}^2}} - \frac{\lambda^2\left(v-\frac{1}{4\lambda}\right)}{\sqrt{\lambda^2\left(v-\frac{1}{4\lambda}\right)^2+{\tau_2}^2}} \qquad (2.16)$$

A função $\varphi_6(\bullet,\lambda,\tau_1,\tau_2)$ apresenta a desejada propriedade de possuir diferenciabilidade infinita, ou seja, pertence à classe de funções c^{∞}, o que permitirá a sua utilização de algoritmos de otimização mais robustos, além de apresentar as seguintes propriedades triviais, consentâneas às demais funções de ativação:

$$\lim_{v\to-\infty}\varphi_6(v,\lambda,\tau_1,\tau_2)=0 \qquad (2.18)$$

$$\lim_{v\to\infty}\varphi_6(v,\lambda,\tau_1,\tau_2)=1 \qquad (2.19)$$

$$\lim_{v\to-\infty}\varphi_6'(v,\lambda,\tau_1,\tau_2)=0 \qquad (2.20)$$

$$\lim_{v\to\infty}\varphi_6'(v,\lambda,\tau_1,\tau_2)=0 \qquad (2.21)$$

Diferentemente das funções de ativação ortodoxas, quando $\tau_1 \neq \tau_2$, a função $\varphi_6(\bullet,\lambda,\tau_1,\tau_2)$ apresenta as diferenciadas propriedades de não oferecer simetria, bem como, de possuir uma imagem que ultrapassa o tradicional espaço definido pelo intervalo [0, 1].

Utilizando a Equação 2.15 e fazendo o parâmetro τ_1 maior que o parâmetro τ_2 o maior valor de saída da função $\varphi_6(\bullet,\lambda,\tau_1,\tau_2)$ ultrapassa o valor um e essa situação é retratada pelo gráfico da figura 15.

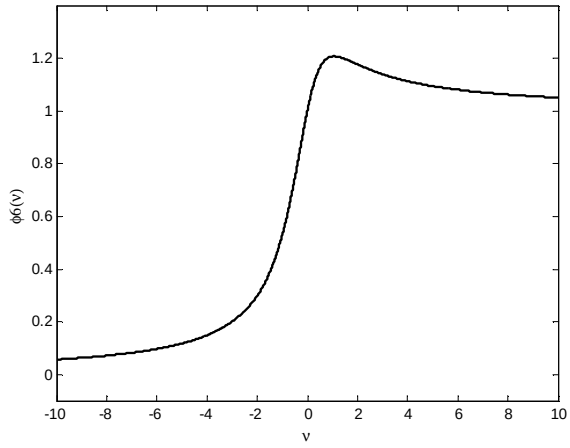


Figura 15: Função Bi-Hiperbólica Assimétrica

Usando a Equação 2.16 e a mesma parametrização aplicada para a geração do gráfico apresentado na Figura 15, foi obtido o gráfico para a sua derivada, que pode ser visto na Figura 16.

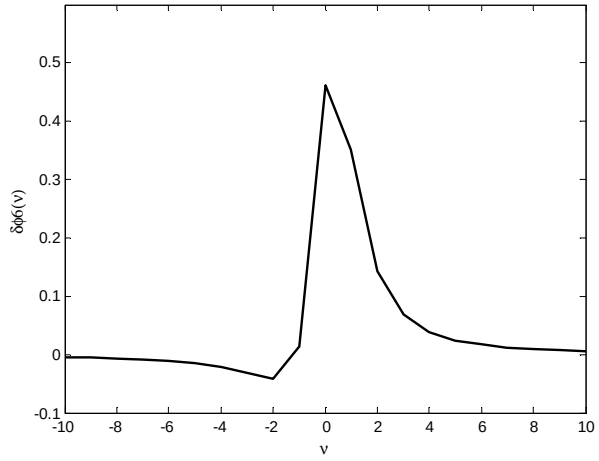


Figura 16: Derivada da Função Bi-Hiperbólica Assimétrica

Utilizando a Equação 2.15 com o parâmetro λ fixo e fazendo o parâmetro τ_1 menor que o parâmetro τ_2 o maior valor da função $\varphi_6(\bullet, \lambda, \tau_1, \tau_2)$ tende positivamente para um e o seu menor valor passa a ser negativo, situação retratada pelo gráfico da figura 17.

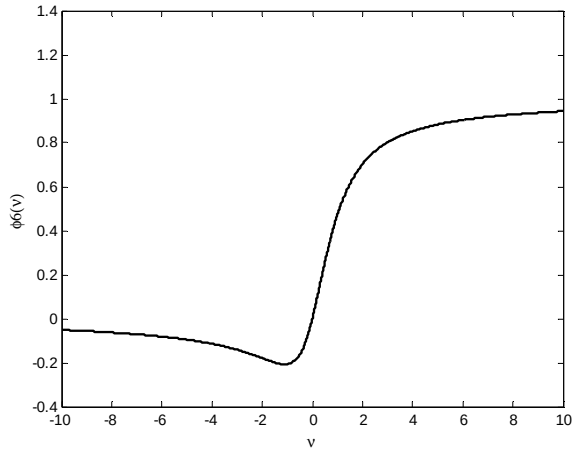


Figura 17: Função Bi-Hiperbólica Assimétrica quando $\tau_1 < \tau_2$

h) **Função Bi-Hiperbólica Simétrica**

Ao analisarmos a Função Bi-Hiperbólica em sua forma mais particular, como mostrado por Xavier (2005), igualando os valores dos parâmetros $\tau_1 = \tau_2 = \tau$, verificamos que a função $\varphi_6(\bullet, \lambda, \tau_1, \tau_2) \underline{\Delta} \varphi_7(\bullet, \lambda, \tau)$, assume uma forma mais próxima a das outras funções de ativação, tendo imagem no intervalo [0, 1] e oferecendo a propriedade de simetria, conforme pode ser visto na Equação 2.22.

$$\varphi_7(v,\lambda,\tau) = \sqrt{\lambda^2(v+\frac{1}{4\lambda})^2+\tau^2} - \sqrt{\lambda^2(v-\frac{1}{4\lambda})^2+\tau^2} + \frac{1}{2} \qquad (2.22)$$

A sua derivada é dada pela Equação 2.23, apresentada abaixo.

$$\varphi_7'(v,\lambda,\tau) = \frac{\lambda^2\left(v+\frac{1}{4\lambda}\right)}{\sqrt{\lambda^2\left(v+\frac{1}{4\lambda}\right)^2+\tau^2}} - \frac{\lambda^2\left(v-\frac{1}{4\lambda}\right)}{\sqrt{\lambda^2\left(v-\frac{1}{4\lambda}\right)^2+\tau^2}} \qquad (2.23)$$

A função $[\varphi_7(v,\lambda,\tau)-1/2]$ é anti-simétrica, ou seja:

$$\varphi_7(v,\lambda,\tau)-1/2 = -[\varphi_7(-v,\lambda,\tau)-1/2] \qquad (2.24)$$

No ponto $v=0$, são observados os seguintes valores para a função $\varphi_7 x$ e sua derivada:

$$\varphi_7(0,\lambda,\tau) = 1/2 \qquad (2.25)$$

$$\varphi_7'(0,\lambda,\tau) = \frac{\lambda}{2\sqrt{1/16+\tau^2}} \qquad (2.26)$$

$$\lim_{\tau\to 0}\varphi_7'(0,\lambda,\tau) = 2\lambda \qquad (2.27)$$

Com o uso da Equação 2.22 com o valor do parâmetro τ constante e variando o valor do parâmetro λ para λ = 3,0; λ = 5,0 e λ = 7,0 foram obtidos os gráficos da Função Bi-Hiperbólica Simétrica, que podem ser vistos na Figura 18, onde pode ser observada a variação gerada na inclinação da curva da função pela alteração do parâmetro λ. A curva de menor inclinação corresponde ao menor valor atribuído ao parâmetro λ. Pode-se ver um efeito similar àquele produzido pela variação do parâmetro α na Função Logística, conforme apresentado na figura 7. Dessa forma pode-se associar o parâmetro λ à inclinação da função na origem.

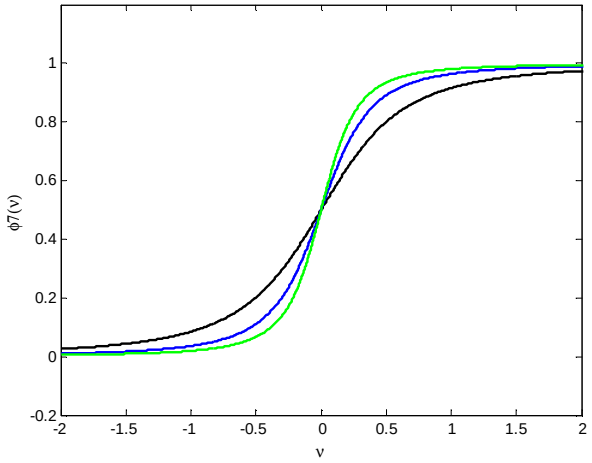


Figura 18: Função Bi-Hiperbólica Simétrica variando λ

Com o uso da Equação 2.23 com o valor do parâmetro τ constante e variando o valor do parâmetro λ para λ = 3,0; λ = 5,0 e λ = 7,0 foram obtidos os gráficos das derivadas da Função Bi-Hiperbólica Simétrica apresentados na Figura 19, que são correspondentes às funções apresentadas na Figura 18. A curva de menor altura corresponde ao menor valor atribuído ao parâmetro λ.

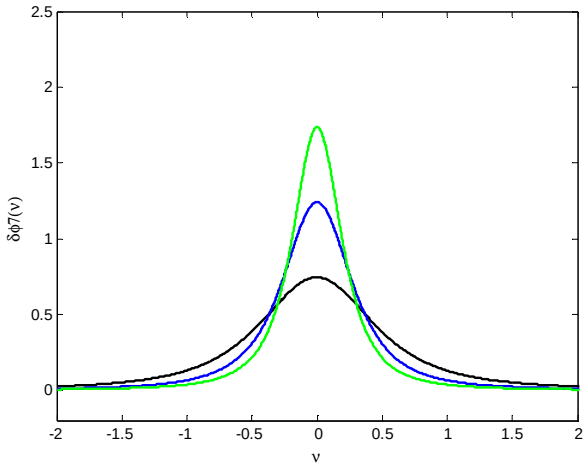


Figura 19: Derivada da Função Bi-Hiperbólica Simétrica variando λ

A função $\varphi_7(\bullet,\lambda,\tau)$ apresenta, ademais, os seguintes comportamentos assintóticos:

$$\lim_{\lambda\to\infty}\varphi_7(v,\lambda,\tau)=\varphi_1(v) \qquad (2.28)$$

$$\lim_{\tau\to 0}\varphi_7(v,\lambda,\tau)=\varphi_2\left(v,\frac{1}{4\lambda}\right) \qquad (2.29)$$

Podemos assim, por exemplo, aproximar a Função de Ativação Patamar $\varphi_2(\bullet,\bullet)$ por uma forma continuamente diferenciável usando a função $\varphi_7(\bullet,\bullet,\bullet)$, podendo essa aproximação ser tão próxima quanto se queira. Adicionalmente, pode-se efetuar essa aproximação assintótica mantendo constante uma inclinação com especificado ângulo α na origem, v = 0, bastando para isso se fazer a reparametrização

$$\lambda = 4\lambda'\sqrt{1/16+\tau^2} \qquad (2.30)$$

Sendo

$$\lambda' = \frac{1}{2}\tan\alpha \qquad (2.31)$$

O parâmetro τ está associado com o afastamento da curva às duas assíntotas horizontais, podendo variar a faixa de valores de saída da função, mantendo-a, entretanto, entre os limites de zero e um, conforme pode ser visto na Figura 20.

Com a utilização da Equação 2.22 com o valor do parâmetro λ constante e variando o valor do parâmetro τ para τ = 2,0; τ = 8,0 e τ = 14,0 foram obtidos os gráficos da Função Bi-Hiperbólica Simétrica apresentados na Figura 20, onde pode ser visto que o parâmetro τ está associado com o afastamento da curva às duas assíntotas horizontais. A curva com menor inclinação foi obtida para o valor mais alto do parâmetro τ.

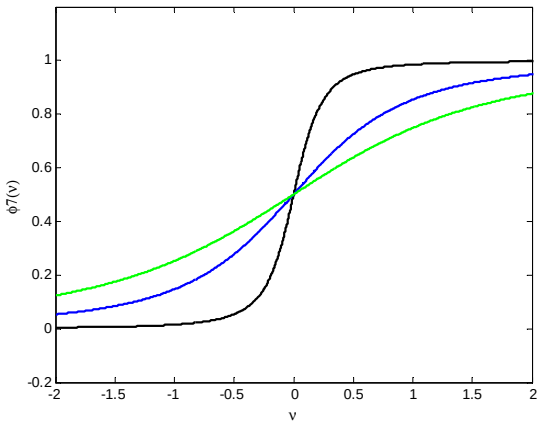


Figura 20: Função Bi-Hiperbólica Assimétrica variando τ

Com a utilização da Equação 2.23 com o valor do parâmetro λ constante e variando o valor do parâmetro τ para τ = 2,0; τ = 8,0 e τ = 14,0 foram obtidos os gráficos das derivadas da Função Bi-Hiperbólica Simétrica apresentados na Figura 20, onde a curva com maior altura foi obtida para o valor mais baixo do parâmetro τ.

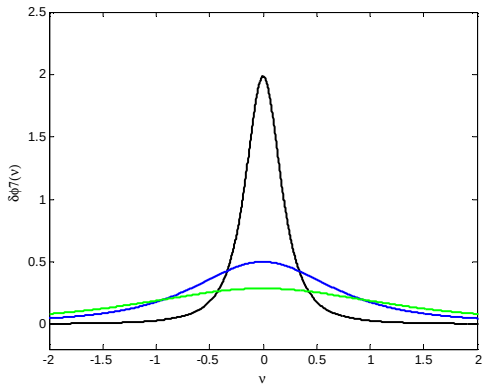


Figura 21: Derivada da Função Bi-Hiperbólica Simétrica variando τ

Como bem ressalta Xavier (2005), a existência de dois parâmetros, um a mais do que nas demais funções de ativação, fornece a essa função uma maior flexibilidade possibilitando a representação mais adequada dos fenômenos modelados com redes neurais.

Numa rede neural multicamadas, por exemplo, essa maior flexibilidade confere à função de ativação Bi-Hiperbólica Simétrica o poder de aproximar qualquer função de uma forma mais sintética, usando um menor número de neurônios na formação da arquitetura da rede.

Além disso, através da manipulação conveniente de seus parâmetros, a função $\varphi_7(\bullet,\lambda,\tau)$ oferece a possibilidade de poder enfrentar mais convenientemente o desastroso fenômeno de saturação, evitando, também, um indesejado mínimo pouco profundo. Um forte indicador destas possibilidades pode ser observado no gráfico de sua derivada, nas Figuras15 e 17, que apresenta uma taxa de variação bem maior do que a das derivadas das demais funções, muito importante para utilização em um método baseado no uso de gradientes.

2.4.3. Arquiteturas de Redes

A base para o estudo das Redes Neurais Artificiais é a estrutura do cérebro humano, onde cada neurônio processa sinais e se comunica com milhares de outros, continuamente, e em paralelo. Ao tentar imitar esta estrutura, estas redes se constituem em uma alternativa à computação algorítmica convencional. São sistemas paralelos distribuídos compostos por unidades de processamento simples que calculam funções matemáticas determinadas, sendo dispostas em uma ou mais camadas.

Esta arquitetura é um parâmetro muito importante no projeto de uma rede. Ela restringe o tipo de problema que pode ser tratado. Uma rede com uma única camada

de nós somente pode, por exemplo, tratar problemas que sejam linearmente separáveis.

A maneira pela qual os neurônios estão organizados em uma rede é intimamente ligada com o algoritmo de aprendizado usado para treinar a rede. Como é explicado por Haykin (2001), temos as redes com camada única onde temos uma camada de entrada que se projeta sobre uma camada de neurônios de saída de uma forma acíclica. Outra classe de rede do tipo acíclico se distingue pela presença de uma ou mais camadas ocultas, cujos nós computacionais são chamados de Neurônios Ocultos e têm por função intervir entre a camada de entrada e a de saída, conforme esquema apresentado na Figura 22. Com isso, a rede neural artificial adquire uma perspectiva global apesar de sua conectividade local, devido ao conjunto extra de conexões sinápticas e da dimensão extra de interações neurais.

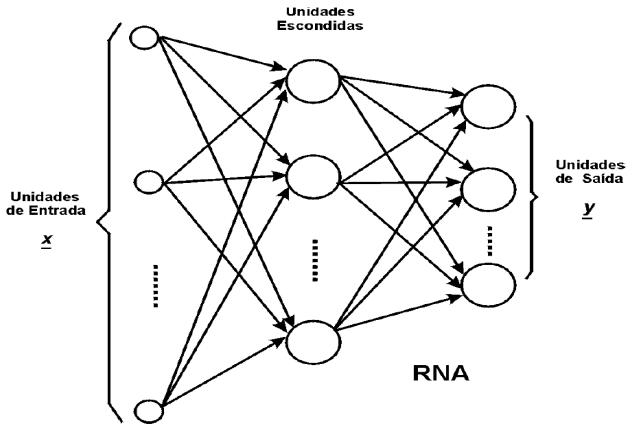


Figura 22: Rede Neural Artificial Multicamadas

As Redes Progressivas de Camada Única (*Single Layer Feedforward)* constituem a forma mais simples de redes em que os neurônios são organizados em camadas. Na Rede Progressiva de Camada Única, temos uma arquitetura com uma camada de entrada de nós fonte, conectada a uma camada de saída constituída de neurônios (nós computacionais), conforme mostrado na Figura 23, reproduzida de Haykin (2001). Esta rede é estritamente progressiva, em que não há conexões no sentido da camada de saída para a camada de nós fontes. A camada de nós de entrada não é contada por não ser formada por unidades processadoras ou neurônios.

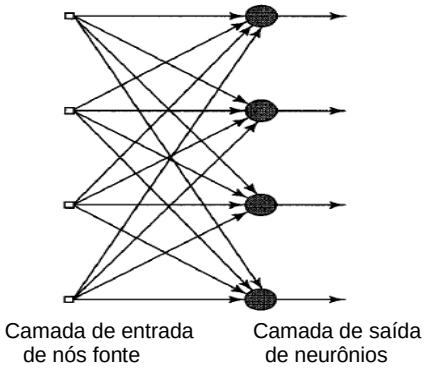


Figura 23: Redes Progressivas de Única Camada

As Redes Neurais Progressivas Multicamadas (Multilayer Perceptrons - MLP) têm por característica possuir uma ou mais camadas escondidas, cujos nós computacionais ou neurônios são correspondentemente chamados de neurônios escondidos ou unidades escondidas, conforme apresentado na Figura 24, reproduzida de Haykin (2001). A função dos neurônios escondidos é intervir entre a camada externa de entrada e a saída da rede de alguma forma útil. Adicionando uma ou mais camadas escondidas, a rede pode extrair estatísticas de ordem superior. Pode-se dizer que a rede adquire uma perspectiva global, apesar de sua conectividade local, devido ao conjunto extra de conexões sinápticas e à dimensão extra de interações neurais. Os nós fonte na camada de entrada da rede provêm dos vetores de entrada, que constituem os sinais de entrada aplicados aos neurônios da segunda camada (primeira camada escondida). Os sinais de saída da segunda camada são usados como entradas para a terceira camada, e, assim, sucessivamente, para o resto da rede. O conjunto de sinais, gerados pelos neurônios da camada de saída da rede, constituem a resposta global da rede ao padrão de ativação provido pelos nós fonte da camada de entrada. Uma rede multicamadas é dita totalmente conectada quando cada um dos nós de uma camada da rede está conectado a todos os nós da camada seguinte. Se estiverem faltando algumas das comunicações sinápticas, ela é dita parcialmente conectada.

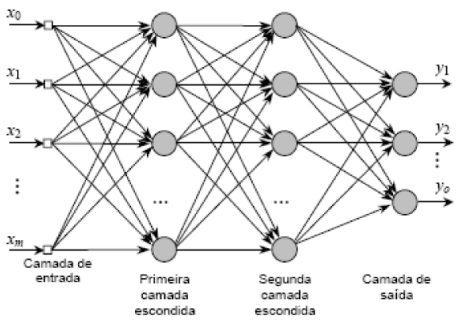


Figura 24: Redes Progressivas Multicamadas (MLP)

Além destas, temos também as Redes Recorrentes ou com Retropropagaçao de sinais, apresentadas na Figura 25, reproduzida de Haykin (2001), que se assemelham às anteriormente apresentadas, exceto pelo fato de terem pelo menos um laço de realimentação. Ou seja, podem se constituir de uma única camada de neurônios, com cada nó realimentando com seu sinal de saída o sinal de entrada de todos os outros neurônios ou de múltiplas camadas onde a realimentação se origina tanto dos neurônios ocultos quanto dos de saída.

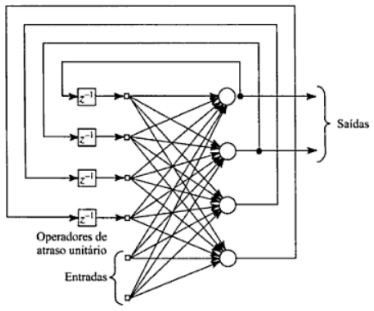


Figura 25: Redes Recorrentes com neurônios ocultos

Uma tarefa importante de uma rede neural é apreender o modelo do ambiente no qual ela está inserida e manter o modelo suficientemente consistente com o mundo real de maneira a atingir os objetivos específicos da sua aplicação. Este conhecimento do mundo abrange, além da chamada informação prévia, que consiste dos fatos sobre o que já é conhecido, as observações do mundo obtidas por sensores que sondam o ambiente no qual a rede deve operar. Essas informações, apesar de normalmente apresentarem erros devidos às imperfeições ou ruídos na sua obtenção, vão formar o conjunto de onde são retirados os exemplos usados para o treinamento da RNA.

O projeto de uma rede neural artificial começa com a seleção de uma arquitetura apropriada e com o treinamento através dos exemplos e de um algoritmo específico. Esta fase é a chamada de Aprendizagem. Em seguida é feita a avaliação com os dados não usados no treinamento para determinar o seu desempenho na tarefa específica. Esta fase é a chamada de Generalização. A diferença fundamental entre o projeto de uma RNA e o de um sistema de processamento de informações clássico está em que neste tipo de sistemas, normalmente, primeiro formula-se um modelo matemático das observações do ambiente, que é validado com os dados reais, e posteriormente se projeta o sistema com base neste modelo. Já o projeto de uma rede neural artificial é baseado diretamente nos dados do mundo real, fazendo com que a rede forneça um modelo implícito do ambiente no qual está inserida, além de realizar a função de processamento de informações.

No projeto de uma rede neural do tipo MLP o dimensionamento das camadas de entrada e de saída será sempre determinado pela natureza do próprio problema. Entretanto, como explicam Cybenko (1989), Hecht-Nielsen (1989) e Haykin (2001), a determinação de quantas camadas ocultas e de quantos neurônios estas devem possuir, não é uma tarefa que permita uma resposta exata. Existem, para este problema, soluções aproximadas, heurísticas, que procuram estimar estas variáveis. Estas heurísticas expõem sempre o compromisso entre a convergência e a generalização da rede. Considera-se Convergência a capacidade da rede de aprender todos os padrões de entrada usados no seu treinamento. Uma rede muito pequena em relação ao problema em análise não será capaz de aprender os dados de treinamento do problema, ou seja, a rede não possuirá parâmetros ou pesos sinápticos suficientes.

2.4.4. Principais aplicações

A utilização das redes neurais artificiais esta ingressando nas mais diversas áreas de aplicação, desde o reconhecimento de padrões, até a distribuição de energia elétrica, do mercado de capitais, até as aplicações navais, e também em sistemas especialistas, além de muitas outras.

Este uso crescente vem atingindo, como ressalta Taylor (2006), áreas em que a confiabilidade em sua capacidade de resolver os problemas é extremamente importante como, por exemplo, no apoio à direção de missões tripuladas para viagens fora do nosso planeta, para os casos de mudanças no ambiente previsto. Por isto, uma das principais dificuldades na distribuição e emprego destes sistemas inteligentes adaptativos está na capacidade de verificá-los e validá-los.

2.4.4.1. Reconhecimento de Padrões

O reconhecimento de padrões talvez tenha sido uma das primeiras aplicações das redes neurais. Na realidade, conforme exemplifica Hagan (1996), o Perceptron de Rosenblatt foi concebido com o principal objetivo de ser um instrumento capaz de reconhecer letras. O reconhecimento de padrões é uma tarefa considerada como sendo, geralmente, melhor desempenhada com o uso das capacidades cognitivas do homem do que executando um algoritmo.

Como exemplifica Barreto (2002), os seres humanos têm extrema facilidade no reconhecimento de rostos, músicas, da caligrafia de alguém conhecido, por sua vez,

cães são excelentes em reconhecer odores e gatos são capazes de sentir o humor de pessoas fugindo daquelas que exprimem características agressivas.

Isso pode ser atribuído a um sistema bastante desenvolvido de reconhecimento de padrões. A utilização de computação algoritmica para realizar estas tarefas tem encontrado sérias dificuldades. O reconhecimento de padrões é muito útil como classificador. Normalmente, o processamento inicia com uma etapa em que os atributos são selecionados para serem processados e este processamento atua como uma função fazendo a associação desses valores, pertencentes a um conjunto de atributos relevantes, com elementos de um conjunto de padrões viáveis, que forma a resposta do classificador.

O processamento de sinais visuais, como mostra Braga (2000), tem grande importância na implementação de robôs autônomos e requer um processamento maciço. Devido a isto, esta tem sido, já há algum tempo, a principal motivação da implementação de redes neurais artificiais em pastilhas de silício usando tecnologia VLSI (Very Large Scale Integration).

O paradigma mais comum de aprendizado, no caso do reconhecimento de padrões, é o supervisionado, associado a uma rede direta multicamadas. Devido a sua disponibilidade, a regra da retropropagação é frequentemente usada, bem como suas variantes. Entretanto, como ressalta Barreto (2002), a utilização de redes Kohonen também apresenta bons resultados com o aprendizado competitivo. Esta utilização é mais recomendada em casos onde não se conhece previamente o número de classes possíveis de serem identificadas. Este não é o caso do reconhecimento de padrões, onde estas classes já são de conhecimento prévio.

O reconhecimento de caracteres é uma aplicação tão bem sucedida de redes neurais, desde o Perceptron de Rosenblatt, que muitos sistemas prontos incluem alguma forma de reconhecimento de caracteres como programa de demonstração.

Normalmente, a letra é apresentada a uma grade, tal como a retina, e cada célula da grade serve de entrada a um neurônio da rede. Neurônios de saída da rede são associados a letras. A camada de saída normalmente contém tantos neurônios quantas são as letras que se deseja identificar. Por exemplo, para identificar todos os caracteres ASCII a rede terá 256 neurônios. Neste caso, as letras são representadas na entrada da rede usando uma representação distribuída (a uma letra correspondem vários neurônios ativados) e na de saída uma representação localizada (a uma letra corresponde apenas um neurônio).

Durante a fase de treinamento apresentam-se sucessivamente as letras à rede em sua forma ideal e com algumas variações mais comuns. Depois de treinada, a rede identificará as letras apresentadas aceitando uma variação.

Bastante sofisticado e dando resultados bastante bons, como apresentado em Freeman (1991), pode-se citar o neocognitron de Fukushima, evolução do cognitron. Esta rede é do tipo multicamadas. Permite reconhecer letras independentemente de sua posição e pode mesmo servir como modelo para o mecanismo biológico de reconhecimento de padrões visuais. Um exemplo de aplicação do neocognitron no reconhecimento de faces pode ser encontrado em Poli (2008).

Uma variante do reconhecimento de caracteres é a filtragem de caracteres. Neste caso, deseja-se poder apresentar na entrada da rede caracteres afetados por um ruído (mal representados) e ter na saída a mesma letra bem representada. Para este caso procede-se de modo inteiramente análogo com a diferença de que a representação da saída é também distribuída. O reconhecimento de caracteres foi também abordado com bastante sucesso por técnicas baseadas em manipulação simbólica, mas o reconhecimento de letras em posições diversas, o uso de caracteres distintos, o reconhecimento de letras manuscritas, por exemplo, continuam a ser problemas que a abordagem simbólica encontra dificuldades em resolver.

Bem mais complexo que o reconhecimento de caracteres é o reconhecimento de faces. Esta aplicação pode ser usada para identificar a quem pertence um rosto visto em posições variadas e também reconstituir um rosto visto apenas parcialmente. Finalmente os dois problemas podem ser associados no reconhecimento de um rosto apresentado parcialmente. O reconhecimento de um rosto segue esquema semelhante ao do reconhecimento de uma letra. Note-se, no entanto, que geralmente o problema exige uma grade muito mais fina, o que aumenta consideravelmente a quantidade de neurônios da rede. O reconhecimento de faces está em fase de desenvolvimento e constante aprimoramento, havendo um campo enorme de interesse ligado a problemas de segurança com novas propostas de soluções, como pode ser visto em Poli (2008). Problema análogo é o da identificação de impressões digitais.

2.4.4.2. Sistemas Especialistas Conexionistas

A inteligência artificial simbolista encontra dificuldades na solução de muitos problemas da vida real devido, principalmente, as suas inconsistências, exceções e especificações incompletas.

Conforme apresenta Barreto (2002), é possível identificar dois tipos de domínios em que um Sistema Especialista pode atuar: artificiais e naturais. Geralmente é mais fácil extrair regras para implementar um sistema especialista

usando a inteligência artificial simbolista em sistemas pertencentes a um domínio artificial, tal como panes de motores. Entretanto, é bem mais difícil obter regras quando o domínio é natural, como no caso de diagnóstico médico, previsões financeiras, entre outros.

Um dos primeiros a usar sistemas especialistas conexionistas foi Gallant em 1988, que considerou uma rede direta para, a partir de 6 sintomas, diagnosticar duas doenças. Gallant (1988) mostrou como é possível explicar o raciocínio para o caso estudado. Posteriormente, outros estudos mostraram como memórias associativas podem ser usadas em sistemas especialistas. Segundo Barreto (2002), estes dois paradigmas de implementação de sistemas conexionistas se encontram em fase de estudos e desenvolvimento.

Também estão sendo estudados os sistemas especialistas usando redes diretas, onde são considerados os exemplos disponíveis para o treinamento da rede por um algoritmo de aprendizado supervisionado. As redes diretas são capazes de aproximar uma função não linear qualquer. Imaginando-se que a solução do problema pode ser encontrada achando o valor da imagem de uma função, esta metodologia pode fornecer uma solução satisfatória.

Para os sistemas especialistas implementados com BAM (*Bi-directional Associative Memory*), conforme explica Barreto (2002), supõem-se conhecidas explicitamente as relações entre sintomas e diagnósticos, tal como é usual em um sistema especialista usando a inteligência artificial simbolista. Com isso, é também possível incorporar exemplos, como no caso de um diagnóstico médico no qual, inicialmente, os conceitos são organizados em conjuntos de conceitos semelhantes: doenças, diagnósticos e pacientes. A cada conjunto de conceitos corresponde um cluster de neurônios. As conexões entre objetos são colocadas representando o que se conhece da relação entre dois objetos. Assim, sabendo-se que um paciente apresenta um sintoma com certa gravidade, coloca-se a conexão entre este paciente e o sintoma com um valor que meça a gravidade do sintoma, ao menos subjetivamente.

Nesta abordagem, os exemplos são usados para colocar pesos nas conexões e não como elementos de um conjunto de treinamento. Além disso, o conhecimento é localizado e o aprendizado se reduz à colocação dos pesos, sendo que matriz sináptica é simétrica. A intensidade das conexões varia normalmente no intervalo [-1, 1] representando o valor nebuloso da importância da relação entre os dois objetos, sendo de [-1, 0] para inibição e de (0, 1] para excitação.

Uma consulta é feita excitando neurônios representativos de conceitos conhecidos e examinando os que aparecem excitados quando a rede atingir o equilíbrio. Este paradigma tem sido usado para diagnósticos em reumatologia.

2.4.4.3. Controle de Processos

Um sistema de controle, segundo Barreto (2002), é chamado de neural quando se usa, de alguma forma, uma rede neural artificial como parte do controlador. O principal interesse nestes sistemas se dá quando estamos lidando com processos em que não esteja disponível o seu modelo ou em casos em que se deseja usar as capacidades de aprendizado das redes neurais para obter a solução do problema.

Esses sistemas podem ser classificados, com base nas topologias de controle mais comuns, em malha aberta, realimentação, e modelo interno.

A topologia chamada de Malha Aberta, como explica Maya (2011), é a topologia mais simples possível. Ela é usada como entrada do sistema controlado e, se o controlador é escolhido como uma aproximação do modelo inverso do processo a controlar, a saída será uma aproximação da finalidade. Controle de malha aberta tem, geralmente, um funcionamento pobre, pois caso apareça uma perturbação no processo, o controlador não tem acesso a esta informação e não corrigirá o processo. Além disso, toda imperfeição na implementação do modelo inverso se refletirá na saída do processo.

Uma das várias maneiras de treinar uma rede neural artificial para representar o modelo inverso é fazer vários ensaios com o processo a controlar obtendo pares entrada/saída. Depois usar estes pares como exemplos.

Talvez a topologia de rede neural artificial mais utilizada, segundo Barreto (2002), seja a rede direta multicamadas, que pode ser treinada para aproximar uma função qualquer com uma precisão que dependerá do número de neurônios e da topologia da rede. Como o processo a controlar, frequentemente, não pode ser representado por uma função por se tratar de um sistema dinâmico, seria mais adequado o uso de uma rede neural artificial dinâmica, que se obtém usando uma rede com realimentação ou usando linha de retardo. Além disso, o modelo inverso de um sistema dinâmico real é normalmente um sistema dinâmico irrealizável e, consequentemente, pode ser implementado apenas aproximadamente.

Ainda, como explica Barreto (2002), outra topologia usada é a chamada de controle com retroação, que é a mais simples topologia que permite o controlador sentir os efeitos de uma perturbação. Os dois modos principais de empregar uma rede neural em um controle usando esta topologia são como controlador e como supervisor do controlador.

Uma rede pode ser usada diretamente na implementação do controlador e, para isso, usa-se um controlador para o processo durante algum tempo e registram-se

suas reações formando um conjunto de treinamento. Depois de treinada uma rede neural com esse conjunto de treinamento, a rede substitui o controlador.

Já no caso de se usar a rede neural como supervisor, usa-se um controlador convencional e a rede é usada para ajustar os parâmetros do controlador. Esta opção tem a vantagem de deixar o sistema funcionar do modo que vinha funcionando e a rede vem apenas melhorar o funcionamento global.

A topologia de controle conhecida como de Modelos Internos, segundo Barreto (2002), tem grande motivação biológica. Com efeito, supõe-se que muitos seres vivos constroem um modelo mental de seus corpos e do ambiente que os cerca. Baseado nestes modelos e por simulações mentais, eles decidem que ações devem realizar. Nesta topologia, um modelo do processo recebe o mesmo sinal atuante que o processo e suas respostas são comparadas. Se existir diferença entre as respostas do modelo e do processo, pode-se atribuir que houve uma perturbação. Então, esta diferença é realimentada para produzir um sinal de controle que corrija os efeitos da perturbação.

Esta topologia de controle pode ser implementada por uma rede neural treinada diretamente com sinais de excitação e resposta do processo. Deve-se, no entanto, lembrar que esta estrutura é aplicável somente no caso de processos estáveis ou que tenham sido previamente estabilizados.

2.4.4.4. Séries Temporais

Aplicações de redes neurais a séries temporais em diferentes áreas do conhecimento apresentam sucesso reconhecido, desde o mercado de capitais até a previsão meteorológica. Nestes casos, como cita Fyfe (2000), usa-se uma rede direta que é treinada com valores de uma série temporal que ocorreram em um determinado intervalo de tempo e com a saída como sendo um valor futuro da série.

2.4.4.5. Monitoramento

Redes neurais podem ser muito úteis em monitoramento, se explorada a capacidade de uma rede direta ter resposta praticamente imediata. Esta solução deve ser cogitada em casos onde a rapidez de resposta seja fator primordial.

Um exemplo, dado por Barreto (2002), é o da detecção de vibrações em um reator nuclear. Vibrações em uma instalação nuclear em frequências próximas das frequências de ressonância de partes importantes da instalação podem ter resultados

catastróficos. Elas devem ser detectadas urgentemente, antes que sua amplitude chegue a valores críticos.

Em muitas instalações, costuma-se, periodicamente, fazer inspeções que consistem muitas vezes em registrar o ruído neutrônico. Supondo que este ruído seja branco, qualquer alteração no espectro de freqüência deste ruído registrado é indicação que existe uma vibração nesta freqüência.

Havendo uma vibração do núcleo pode-se esperar que esta vibração se traduza por uma variação na espessura da proteção entre o núcleo e o captor. Isto faz variar a absorção de neutrons modulando o ruído neutrônico que deixa de se apresentar como ruído branco. A idéia foi colocar uma rede neural artificial, treinada para detectar o espectro de freqüência de um sinal, solução que deu um bom resultado.

Para que uma rede neural artificial seja capaz de executar esta diversidade de aplicações é fundamental a sua capacidade de apreender o meio ambiente no qual está inserida e poder generalizar gerando os resultados desejados. Para isto, existem algumas metodologias de aprendizagem disponíveis e na próxima seção iremos apresentar as principais, destacando-se o aprendizado supervisionado e com o uso do Algoritmo Backpropagation, objeto principal do objetivo deste trabalho.

Capítulo 3 Aprendizagem

3.1. Introdução

A propriedade mais importante de uma RNA é a sua capacidade de aprender, a partir do seu ambiente, e melhorar seu desempenho. O aprendizado, segundo Mackay (1998), se resume no problema de obter um conjunto de parâmetros livres que permita à rede atingir o desempenho desejado. O tipo de aprendizagem é determinado pela forma através da qual é efetuada a mudança nestes parâmetros.

Neste processo de aprendizagem, a rede é estimulada pelo ambiente e sofre mudanças em seus parâmetros livres como resultado deste estímulo. Devido às mudanças ocorridas em sua estrutura interna, ela passa a responder de uma nova forma ao ambiente.

Os dois paradigmas básicos de aprendizagem, como apresenta Haykin (2001), são o aprendizado através de um tutor (Aprendizado Supervisionado) e o aprendizado sem um tutor (Aprendizado Não-Supervisionado). Além destes, como veremos mais adiante, existe uma terceira forma chamada de Aprendizagem por Reforço, que utiliza um crítico.

No Aprendizado Supervisionado, uma série de padrões, representados pelos vetores de entrada, é associada aos resultados desejados como resposta e é apresentado à rede, conforme esquema apresentado na Figura 26. Os parâmetros internos da rede, chamados de pesos sinápticos, são alterados sistematicamente de forma a aproximar os resultados obtidos aos das respostas desejadas. Este procedimento consiste em minimizar os erros obtidos na comparação entre os resultados desejados e os calculados para os padrões usados no treinamento**.**

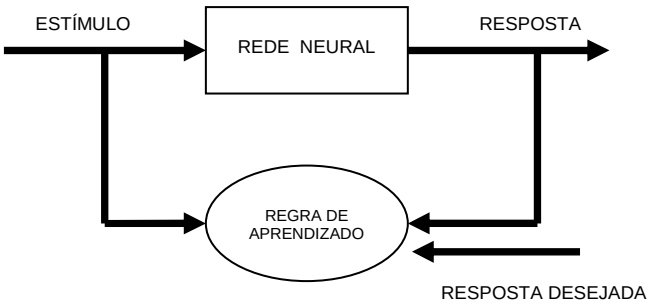


Figura 26: Aprendizado Supervisionado

Outra maneira de treinamento é representada pelo Aprendizado Não Supervisionado ou Auto-Organizado, onde não existe um tutor externo ou um crítico para supervisionar o processo de aprendizagem, conforme apresentado esquematicamente na Figura 27. Estes algoritmos não necessitam do conhecimento

das saídas desejadas. Durante o treinamento, somente padrões de entrada são apresentados à RNA até que ela forme novas representações internas e possa agrupar os padrões de entrada em grupos com características similares.

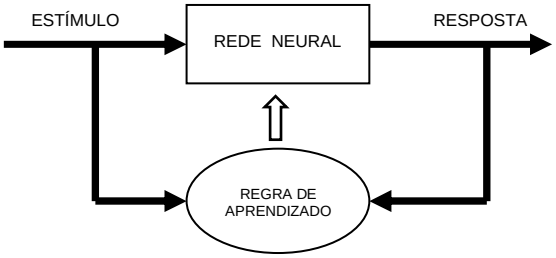


Figura 27: Aprendizado Não Supervisionado

No Aprendizado por Reforço não existe um agente externo indicando a resposta desejada para os padrões de entrada. Neste caso, a aprendizagem de um mapeamento de entrada-saída é obtida através de uma interação contínua com o ambiente, de forma a melhorar o seu desempenho. A Figura 28 representa um destes modelos com o uso de um crítico.

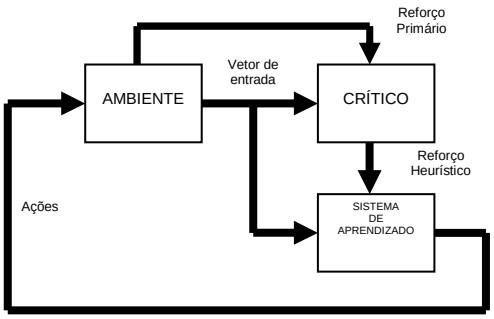


Figura 28: Aprendizado por Reforço

3.2. Algoritmo de Backpropagation

O treinamento de um Perceptron de Múltiplas Camadas (Multi-Layer Perceptron – MLP), segundo apresenta Haykin (2001), consiste em ajustar os pesos e os limiares de disparo de suas unidades para que a classificação desejada seja obtida. Quando um padrão é inicialmente apresentado à rede, ela produz uma saída e, após medir a distância entre a resposta atual e a desejada, são realizados os ajustes apropriados nos pesos de modo a reduzir esta distância. Este procedimento é conhecido como Regra Delta

Esse tipo de rede apresenta soluções para funções linearmente não-separáveis e necessita de um algoritmo de treinamento capaz de definir de forma automática os pesos. O algoritmo mais utilizado para o treinamento destas redes MLP é uma generalização da Regra Delta denominado de Algoritmo Backpropagation.

Durante o treinamento com o Algoritmo Backpropagation, como visto em Haykin (2001), a rede opera em uma sequência de dois passos. No primeiro, um padrão é apresentado à camada de entrada da rede. O sinal resultante flui através da rede, camada por camada, até que a resposta seja produzida pela camada de saída. No segundo passo, a saída obtida é comparada à saída desejada para esse padrão particular. Se esta não estiver correta, o erro é calculado. Este erro é propagado a partir da camada de saída até a camada de entrada, e os pesos das conexões das unidades das camadas internas vão sendo modificados na medida em que o erro é retropropagado.

Na Regra Delta padrão, conforme esquematização apresentada na Figura 29[1], é implementado um gradiente descendente no quadrado da soma dos erros para funções de ativação lineares. Entretanto, como a superfície do erro pode não ser tão simples, as redes ficam sujeitas aos problemas de mínimos locais.

A Regra Delta Generalizada ou Backpropagation funciona quando são utilizadas na rede unidades com uma função de ativação semi-linear, que é uma função diferenciável e não decrescente. Uma função de ativação amplamente utilizada, nestes casos, é a Função Logística.

A Taxa de Aprendizado é uma constante de proporcionalidade no intervalo [0 1], pois este procedimento de aprendizado requer apenas que a mudança no peso seja proporcional à meta. Entretanto, como o verdadeiro gradiente descendente requer que sejam tomados passos infinitesimais, quanto maior for essa constante, maior será a mudança nos pesos, aumentando a velocidade do aprendizado. Tal situação pode levar a uma oscilação do modelo na superfície de erro. Procura-se, então, utilizar a maior taxa de aprendizado possível que não leve a esta oscilação, resultando em um aprendizado mais rápido. O treinamento das redes MLP com backpropagation pode demandar muitos passos no conjunto de treinamento, resultando em um tempo de treinamento consideravelmente longo. Se for encontrado um mínimo pouco profundo, o erro no treinamento para de diminuir e estaciona em um valor maior que o aceitável. Uma maneira de aumentar a taxa de aprendizado sem levar à oscilação é modificar a Regra Delta generalizada para incluir o termo de Momentum, uma constante que determina o efeito das mudanças passadas dos pesos

[1] Disponível em http://www.icmc.usp.br/~andre/research/neural/index.htm#intro

na direção atual do movimento no espaço de pesos. O termo Momentum torna-se útil em espaços de erro que contenham longas gargantas, com curvas acentuadas ou vales com descidas suaves.

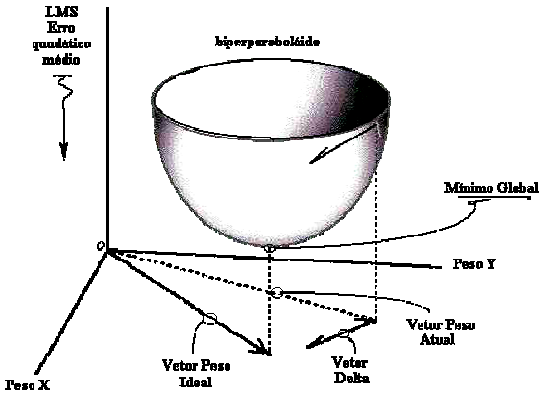


Figura 29: Regra Delta

3.3. Calibração de Parâmetros

No projeto de uma rede neural do tipo MLP, o dimensionamento das camadas de entrada e de saída será sempre determinado pela natureza do próprio problema.

Entretanto, a determinação de quantas camadas ocultas e de quantos neurônios estas devem possuir, não é uma tarefa que permita uma resposta exata, como pode ser visto em Stathakis (2009). Existem, para este problema, heurísticas, ou soluções aproximadas, que procuram estimar estas variáveis, como explicado em Hecht-Nielsen (1989). Estas heurísticas expõem sempre o compromisso entre a convergência e a generalização da rede. Considera-se Convergência a capacidade da rede de aprender todos os padrões de entrada usados no seu treinamento. Uma rede muito pequena em relação ao problema em análise não será capaz de aprender os dados de treinamento do problema, ou seja, a rede não possuirá parâmetros ou pesos sinápticos suficientes.

Generalização é a capacidade da RNA responder adequadamente a padrões fora dos usados no treinamento. Uma rede muito grande, com número de neurônios muito superior ao necessário, não responderá corretamente a estes novos padrões e perderá a capacidade de generalizar, como mostrado em Hornik (1989). Neste caso, os pesos sinápticos da rede aprenderão os vetores de entrada e, também, o ruído presente nestes dados perdendo esta desejada capacidade.

A capacidade de generalização de uma rede neural é afetada pelo tamanho e eficiência dos dados de treinamento; pela arquitetura da rede; pelo número de nós

processadores nas camadas ocultas e pelas características intrínsecas do próprio problema a ser resolvido. Na prática, as heurísticas são utilizadas em conjunto com séries de tentativas e ajustes na arquitetura e nas definições da rede. O principal objetivo é obter uma rede que generalize, ao invés de memorizar os padrões usados no treinamento.

Capítulo 4 Avaliação

4.1. Verificação e Validação

Para que seja possível a utilização de novos modelos, especialmente em aplicações críticas em segurança, é necessário assegurar a sua confiabilidade operacional. Em sistemas não-críticos, a falha pode representar desde uma mera inconveniência, até perdas de produção e lucros. Entretanto, em sistemas críticos, onde a alta confiabilidade é um requisito primordial, as falhas podem resultar em perdas de vidas humanas.

A verificação avalia a correção do produto final do desenvolvimento do modelo, além dos resultados dos testes realizados de forma a possibilitar medir a confiabilidade esperada do sistema.

A validação examina o sistema através de outra perspectiva. Procura garantir que todos os requisitos tenham sido satisfeitos, inclusive a confiabilidade, a taxa de falhas e outras características importantes para o uso a que se destina o sistema.

As redes neurais, diferentemente dos sistemas tradicionais, são treinadas com o uso de um algoritmo de aprendizagem e de um conjunto de dados de treinamento. Devido à sua adaptabilidade, como explica Taylor (2006), as redes neurais artificiais são consideradas como “caixas pretas”, uma vez que a sua resposta pode não ser previsível ou bem definida em todas as regiões do espaço de entrada.

O projeto de uma rede neural artificial é um processo interativo que sofre alterações durante o treinamento. É necessário que se tenha um gerenciamento das configurações testadas, para que seja possível acompanhar as alterações necessárias na estrutura, funções e arquiteturas utilizadas, além dos conjuntos de treinamento usados. Com isso é possível o retorno a um estágio anterior se as alterações trouxerem perda de desempenho. Também as técnicas de validação usadas em RNAs, tais como Validação Cruzada e *Bootstraping*, devem ser controladas.

Um ponto importante no projeto e treinamento de uma rede neural artificial é, como pode ser visto em Geman (1992), o conhecido dilema bias/variância, onde estes dois estimadores estatísticos de desempenho são considerados.

O *bias* estatístico representa a restrição de complexidade que a arquitetura da rede neural impõe no grau em que a função proposta pelo modelo se ajusta aos dados disponíveis ou, com é mais conhecida na área de redes neurais, a Convergência do modelo. A *variância* estatística é o desvio da eficácia do aprendizado que a rede neural apresenta, de uma amostra para a outra, ao ser descrita pela mesma função proposta pelo modelo. Este fator é conhecido como a Generalização do modelo, que

é a sua capacidade de representar dados que não estavam presentes no conjunto de treinamento. Embora estas medidas estatísticas estejam interrelacionadas, a melhoria de uma delas, não necessariamente, garante a melhoria da outra. Deve-se procurar um ponto de equilíbrio, para evitar um ajustamento excessivo ou insuficiente aos dados disponíveis, ou seja, entre a complexidade da rede e a complexidade dos dados.

A Figura 30 apresenta as curvas do erro esperado de treinamento e do erro esperado do teste. São representadas duas regiões separadas entre subajustamento, que é devido ao bias excessivo (convergência), e de sobreajustamento, devido ao aumento da variância (generalização). A seleção do melhor modelo corresponde a um compromisso ótimo entre o menor erro global de treinamento combinado com o menor erro global do teste da rede.

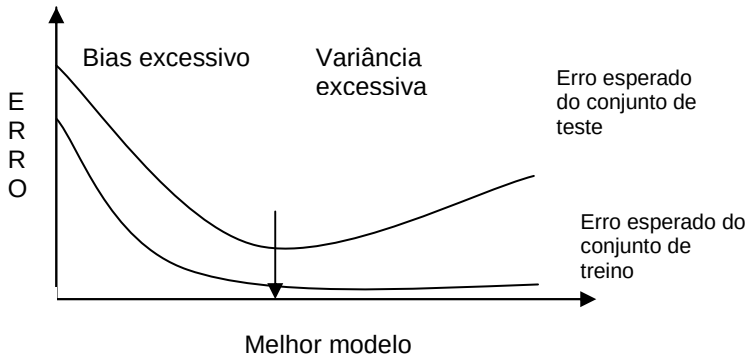


Figura 30: Dimensionamento do melhor modelo

A situação de ajuste insuficiente decorre da falta de treinamento adequado, enquanto que o treinamento em demasia pode resultar em ajustamento excessivo ao conjunto de treinamento, perdendo o modelo a sua capacidade de generalização.

O aumento do número de parâmetros de uma RNA pode contribuir para uma maior variância, possibilitando o aumento da generalização. Já um menor número de parâmetros aumenta o bias, o que pode acarretar em uma rede mais especializada em um dado domínio do problema, podendo-se esperar um melhor desempenho neste caso, mas uma baixa generalização para dados fora deste domínio.

Observando a Figura 30, pode-se notar que o processo de treinamento pode ser dividido em duas partes. A primeira vai do início do treinamento até o ponto de mínimo da curva de erro de validação. Nesta fase, a rede se adapta somente às principais características dos dados, ou seja, aprende a função geradora. Na fase seguinte, o ruído presente nos dados começa a ser modelado. Na maioria das vezes, temos redes mais complexas do que o necessário. Neste caso devemos interromper o

treinamento quando o erro de validação atingir o seu mínimo de forma a evitar que o ruído presente nos dados seja modelado.

A interrupção do treinamento de uma rede antes de seu término é uma estratégia, denominada Parada Antecipada, que é utilizada para a obtenção de redes com boa capacidade de generalização, sendo sugerida através do comportamento do erro de generalização em relação a um dado conjunto de validação.

4.2. Critério de Parada

Um dos problemas que surge na utilização do algoritmo de retropropagação está ligado ao critério a ser adotado para terminar o ajuste dos pesos, uma vez que nem sempre é possível demonstrar a convergência deste algoritmo. Como estamos procurando um mínimo na superfície de erro, segundo Haykin (2001), uma condição necessária é dada para o caso em que o vetor gradiente da superfície de erro em relação ao vetor de peso seja zero para o peso ótimo. Com isso, podemos adotar o critério de considerar que houve a convergência do algoritmo quando a norma euclidiana do vetor gradiente for suficientemente pequena. Este procedimento pode ser demorado, levando a um alto tempo de aprendizagem. Pode ser utilizado outro critério relativo ao mínimo já que a função de custo, ou de medida do erro, é estacionária no ponto onde o vetor de pesos é ótimo. Neste caso, verifica-se o valor da taxa absoluta de variação de Erro Médio Quadrático por época e usa-se como critério de convergência o fato dela apresentar um valor suficientemente pequeno. Tipicamente, valores no intervalo entre 0,1 e 1,0 por cento são considerados suficientemente pequenos. Este critério pode resultar em falha no treinamento, dificultado a generalização.

Para permitir uma melhor generalização, outro critério estabelece que a rede seja testada após cada interação de aprendizagem. Conforme pode ser visto na Figura 30, o processo será encerrado quando se conseguir uma generalização adequada ou ficar aparente que o desempenho de generalização atingiu o máximo. Em muitos casos, ao se verificar o comportamento do erro de validação, pode-se perceber que o seu mínimo não corresponde ao mínimo obtido na curva correspondente no treinamento. Assim, se considerarmos o ponto de parada com base somente na minimização do erro de treinamento, podemos obter uma rede cujo desempenho, ao ser submetida a padrões desconhecidos, não seja o melhor possível.

4.3. Validação Cruzada

O processo de treinamento de uma rede neural artificial visa à obtenção do melhor conjunto de parâmetros, segundo algum critério, de forma a poder generalizar o futuro de um determinado conjunto de dados, com base no seu passado.

Uma ferramenta estatística utilizada para auxiliar na solução deste problema é a Validação Cruzada. É feita a divisão aleatória do conjunto de dados disponíveis em um conjunto de treinamento e em um conjunto de testes. O conjunto de treinamento é subdividido em dois subconjuntos disjuntos, um de estimação, usado para selecionar o modelo ou ajustar os parâmetros livres, e outro de validação, usado para testar ou validar o modelo.

O que se procura com isto é a obtenção de um modelo com os valores de parâmetros com melhor desempenho, mas que não esteja excessivamente ajustado ao conjunto de treinamento. O uso da validação cruzada é interessante por ser útil para projetarmos redes neurais com uma boa capacidade de generalização.

A divisão do conjunto de treinamento em subconjunto de estimação e subconjunto de validação é um fator de muita importância para este objetivo. Não existe uma fórmula exata para a determinação do tamanho de cada uma destas partições. Entretanto, foram identificadas algumas propriedades qualitativas para o particionamento ótimo do conjunto de dados. Como pode ser visto em Kearns (1996), um dimensionamento descrito na literatura é a separação de 80% do conjunto de treinamento para o subconjunto de estimação e os 20% restantes são atribuídos ao subconjunto de validação.

Com o decorrer do processo de treinamento, as funções de mapeamento vão aumentado de complexidade, com o erro médio quadrático decrescendo com o aumento do número de épocas do treinamento. Quando o objetivo é a obtenção de uma boa generalização, se impõe o problema de decidir quando parar o treinamento avaliando apenas a curva de aprendizagem.

Para identificar o início do treinamento em excesso, uma ferramenta muito útil, como explica Haykin (2001), é a validação cruzada. Ela pode ser muito útil para identificar o início do treinamento em demasia. Após um determinado número de épocas de treinamento, o processo é interrompido e, com os pesos sinápticos calculados e com os níveis de bias determinados, a rede é avaliada com o subconjunto de dados de validação. Normalmente, o modelo funciona melhor para o conjunto de treinamento e a sua curva de aprendizagem decresce monotonamente para um número crescente de épocas. Já a curva de aprendizagem de validação, usualmente, decresce monotonamente para um mínimo e então começa a crescer

com o aumento do treinamento. Como pode ser observado na Figura 30, após este ponto, a rede começa a assimilar o ruído presente nos dados. Este ponto pode, então, ser considerado como o Ponto de Parada.

4.4. Validação Cruzada Múltipla (*k-folds*)

A Validação Cruzada Múltipla, ou Validação Cruzada de Fator K, como mostram Grenn (2007) e Haykin (2001), é uma variação da validação cruzada, onde o conjunto de exemplos disponíveis é divido em K partições mutuamente exclusivas (*folds*). O modelo é treinado com todas as partições, exceto uma, e o erro de validação é obtido pela avaliação do modelo para esta partição não usada no treinamento. Este procedimento é repetido para um total de K vezes, usando uma partição diferente para a validação, conforme apresentado na Figura 31. Para medir o desempenho do modelo é utilizada a média dos erros médios quadráticos obtidos na validação de todas as repetições do processo.

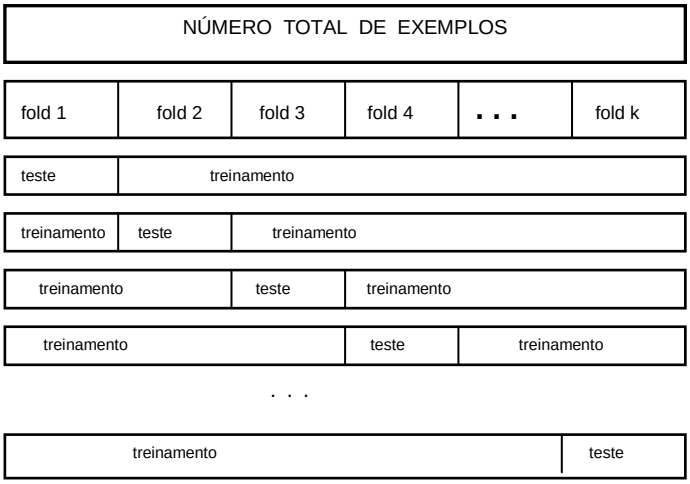


Figura 31: Método de Validação Cruzada Múltipla

4.5. Método Holdout

Este método consiste em dividir o conjunto total de dados em dois subconjuntos disjuntos, um para treinamento (estimação dos parâmetros) e outro para teste (validação), conforme apresentado na Figura 32.

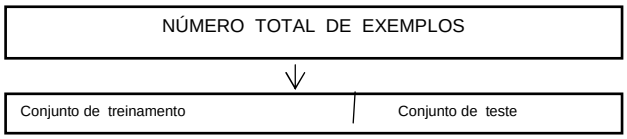


Figura 32: Método Holdout

A divisão do conjunto de dados pode ser feita colocando-se quantidades iguais ou diferentes em cada um dos subconjuntos. Como pode ser visto em Green (2007), uma proporção muito comum encontrada na literatura é utilizar 2/3 dos dados para treinamento e o 1/3 restante para teste. Após o particionamento, é realizado o treinamento da rede neural utilizando o subconjunto de dados próprios, obtendo, assim, os parâmetros de configuração. Após este treinamento, é feita a validação do modelo usando o subconjunto de dados de teste. Com isso, é possível calcular o erro de predição esperado. Esta abordagem é indicada quando está disponível uma grande quantidade de dados. Caso o conjunto total de dados seja pequeno, o erro calculado na predição pode sofrer muita variação. A acurácia deste modelo é calculada dividindo-se o número de amostras bem classificadas pelo número total de amostras no subconjunto de teste.

Uma vantagem desse método é que ele é simples e computacionalmente rápido. Porém, sua avaliação pode ter uma alta variância. O resultado da avaliação pode depender, por exemplo, do ponto em que terminaram os dados de treinamento e começaram os dados de teste, ou seja, da quantidade de padrões existente em cada conjunto. A avaliação depende, então, da maneira com que é feita esta divisão. Outro fator que influencia na avaliação é a quantidade de padrões existentes de cada classe no conjunto de treinamento. Por exemplo, se uma das classes se apresenta com uma grande quantidade de padrões, ela deverá influenciar mais o resultado final, ao contrário de uma outra classe com poucos padrões do seu tipo. A rede treinada terá, então, uma melhor generalização para os dados da classe predominante do que para os da outra classe. Este problema, como mostra Green (2007), pode ser contornado por outras formas de amostragem e de um maior número de execuções do modelo. Sua utilidade se estende ao problema, apresentado anteriormente, de definir o ponto de parada do treinamento, conforme pode ser visto na Figura 30.

4.6. Método Deixe Um de Fora (*Leave-one-out Cross-Validation)*

Outra forma de se implementar a validação cruzada é o método conhecido como Deixe Um de Fora. Este método é especialmente útil quando dispomos de uma pequena quantidade de exemplos para o treinamento da rede. Ele aproveita ao máximo os dados disponíveis, o que é importante quando esta quantidade é pequena, e não envolve sub-amostragens aleatórias. Entretanto, é considerado computacionalmente custoso. Ele apresenta boas estimativas para o erro de generalização para funções contínuas. No caso de funções descontínuas, este método pode apresentar baixo desempenho.

Considerando ***n*** a quantidade de exemplos disponíveis, este método utiliza *n-1* exemplos para treinar o modelo, que é validado pelo teste realizado com o exemplo deixado de fora no treinamento. Este procedimento é repetido ***n*** vezes, cada vez deixando de fora um exemplo diferente para ser usado na validação. O Erro Médio Quadrático considerado é a média dos erros obtidos nas ***n*** execuções deste procedimento.

4.7. Método Bootstrap

Como comenta Franke (2006), em muitas aplicações, para usar bem um estimador estatístico é muito importante o conhecimento sobre a sua distribuição de probabilidade. Entre estas aplicações podemos citar a construção de um intervalo de confiança para parâmetros de modelos ou a determinação de valores críticos para testes. Um método bastante usado nestes casos é o conhecido como bootstrap. Ele é baseado na estimativa da estrutura probabilística do processo de geração dos dados que é obtida pelas informações obtidas em um conjunto de observações aleatórias.

Bootstrap é um método de estimação que usa amostragem com reposição para formar o conjunto de treinamento. Retira-se uma amostra aleatória de tamanho *n* de um conjunto de *m* exemplos com reposição e esta amostra é usada para o treinamento. Os exemplos dos dados originais, que não estão no conjunto de treino, são usados como teste. É a melhor maneira quando o conjunto de dados é pequeno

Este é um procedimento computacional que utiliza amostragens com reposição de uma forma intensiva, objetivando reduzir as incertezas e imitar o componente randômico de um processo, além de reduzir a variância pela utilização da média de muitas partições diferentes dos dados. Entretanto, segundo Abrahart (2001), a decisão de quais itens serão reamostrados não é uma tarefa simples, e deve ser determinada a partir de considerações dos componentes estocásticos do processo de modelagem. Normalmente, se processam centenas ou milhares de subconjuntos para produzir uma estimativa empírica da distribuição de saída, a partir da qual certas características fundamentais da população podem ser calculadas, como por exemplo, médias e variâncias. Essa estimativa la também é usada para produzir informações sobre probabilidades, para gerar inferências sobre parâmetros reais e para determinar intervalos de confiança. O uso de aplicação da metodologia de bootstrap para a construção de redes neurais é objeto de pesquisa atualmente. Existem dois caminhos para incorporar a randomização na operação de construção de um modelo neural: através de diferentes escolhas sobre a divisão dos dados, ou pelas diferentes escolhas na inicialização, arquitetura e treinamento da rede. Qualquer um destes

caminhos, ou mesmo os dois, pode se beneficiar da metodologia de bootstrap. O bootstrap neural já foi usado para produzir agregação de montagem multi-modelos objetivando produzir saídas médias e uma solução mais estável. Mais exemplos de utilização podem ser encontrados em Abrahart (2001), em White (2011) e em Lebaron (1994).

Na próxima seção será apresentado o planejamento dos experimentos computacionais efetuados para permitir a avaliação das características impactadas pela utilização da Função de Ativação Bi-Hiperbólica, proposta neste trabalho, em comparação com as principais funções atualmente utilizadas.

Capítulo 5 Modelagem Computacional

5.1 Plano dos Experimentos

Um experimento pode ser definido como um conjunto de procedimentos que objetiva coletar evidências destinadas a avaliar uma hipótese formulada teoricamente.

A Função de Ativação Bi-Hiperbólica apresenta diversas características computacionais importantes previstas teoricamente. Entre elas, quando comparada com as funções de ativação atualmente utilizadas, possibilitar um desempenho superior nas fases de treinamento e de projeto da arquitetura da rede neural.

Para possibilitar a avaliação dessas características computacionais será apresentado, nas seções seguintes, o resultado computacional dos experimentos realizados com a utilização de modelos de uma rede neural artificial, do tipo Multilayer Perceptron, onde foram utilizados a mesma arquitetura, os mesmos parâmetros básicos e diferentes funções de ativação.

Cada modelo foi submetido aos mesmos experimentos de treinamento e validação, com os mesmos conjuntos de dados, objetivando a obtenção de parâmetros comparativos relacionados com a eficiência e eficácia das funções de ativação Bi-Hiperbólica, Logística e Tangente Hiperbólica. Serão registrados indicadores dos processos de convergência e generalização das redes, bem como o tempo de processamento da fase de treinamento dos modelos.

O foco principal deste trabalho foi comparar o desempenho destes modelos e não o ajuste diferenciado de cada um deles para a obtenção de resultados mais favoráveis individualmente. A comparação se dá, portanto, em modelos de características semelhantes.

As bases de dados utilizadas nos testes referidos acima foram obtidas no site do Centro para Aprendizado por Máquinas e Sistemas Inteligentes, da Universidade da Califórnia, Irvine[2].

O primeiro experimento utilizou a base de dados conhecida como “Wisconsin Breast Cancer Data”, formada por dados obtidos pelo Dr. William H. Wolberg do Hospital da Universidade de Wisconsin em Madison (Wolfberg, 1990).

O segundo experimento usou a base de dados biomedicos organizada pelo Dr. Henrique da Mota durante o período de residência médica junto ao *Group of Applied Research in Orthopaedics (GARO)* do Centre *Médico-Chirurgical de Réadaptation des Massues*, Lyon, França, como pode ser visto em Netto (2011).

[2] disponíveis no endereço http://archive.ics.uci.edu/ml/

Ambas as bases possuem características básicas semelhantes, facilitando a comparação dos resultados dos experimentos.

5.2 Comparação de Modelos de Redes Neurais

As redes do tipo Perceptron de Múltiplas Camadas (Multilayer Perceptrons – MLP) têm sido um dos modelos de Redes Neurais Artificiais mais amplamente utilizado na construção de sistemas para a solução de diferentes problemas, tais como classificação de padrões (reconhecimento), controle e processamento de sinais.

Para atender a esta diversidade de aplicações é necessário um eficiente algoritmo de treinamento e o mais utilizado tem sido o algoritmo de Retro-Propagação do Erro, mais conhecido por Backpropagation. Ele se baseia no aprendizado por correção de erro, em que o erro é retro-propagado da camada de saída para as camadas intermediárias da RNA. É um método computacionalmente eficiente para o treinamento de redes MLPs e resolveu o problema de realizar a propagação reversa do erro em RNAs com múltiplas camadas. Este problema atrasou, por muitos anos, o desenvolvimento da área de redes neurais artificiais. Entretanto, esse algoritmo apresenta algumas dificuldades na sua utilização, impedindo sua aplicação de uma forma mais ampla em problemas reais. Sendo um método baseado no uso de gradientes, existe sempre a possibilidade de convergência para um mínimo pouco profundo. Apresenta, também, como ressalta Schiffmann (1994), uma lentidão muito grande no seu processamento, mesmo nos casos em que consegue atingir o seu objetivo de apresentar um erro dentro dos limites desejados. Esta demora no processamento dificulta a sua utilização ainda mais ampla em aplicações reais práticas.

Um dos fatores possivelmente responsável pela lentidão deste processo de convergência, como ressalta Kamruzzaman (2002), é a função de ativação usada em seus neurônios, pois, sendo o processo de aprendizagem da rede essencialmente iterativo, uma função mais lenta para ser calculada torna todo o procedimento lento. Outro importante fator a ser considerado é a saturação da função de ativação usada para as camadas ocultas e de saída, pois, uma vez que a saturação de uma unidade ocorre, o gradiente descendente assume valores muito pequenos, mesmo quando o erro de saída ainda é grande, aumentando o número de iterações em busca da minimização deste erro. A taxa de convergência deste método é relativamente lenta, especialmente para redes com mais de uma camada oculta.

O problema de otimizar a eficiência e a taxa de convergência do algoritmo de backpropagation tem sido objeto de estudo de muitos pesquisadores, sendo ainda uma área aberta a novos estudos e de grande importância para a evolução do uso das redes neurais.

A proposta apresentada neste trabalho prevê a utilização de uma nova função de ativação, a Função Bi-Hiperbólica Simétrica, com características que atendem às necessidades do Algoritmo de Backpropagation e oferece a vantagem de possibilitar maior flexibilidade na representação dos fenômenos modelados. O uso de dois parâmetros, como ressalta Xavier (2005), um a mais do que nas funções tradicionalmente utilizadas para esta finalidade, implica na possibilidade de melhor enfrentar o problema da saturação, além de permitir melhor tratamento para evitar os mínimos locais. Outra vantagem, observada empiricamente, é a de ser computacionalmente mais rápida de ser avaliada do que a Função Logística, que é atualmente a função de ativação mais usada. Além disso, o uso da Função Bi-Hiperbólica Simétrica possibilita, por sua maior flexibilidade, a capacidade de poder aproximar qualquer função de uma forma mais sintética, permitindo a utilização de um menor número de neurônios nas camadas ocultas da rede, melhorando ainda mais o desempenho do algoritmo Backpropagation agindo diretamente na topologia da rede (XAVIER, 2005).

5.3 Protótipos Desenvolvidos

Para permitir uma avaliação do desempenho da função de ativação Bi-hiperbólica Simétrica, comparando-a com as funções de ativação tradicionalmente utilizadas, foram desenvolvidos protótipos em MATLAB que, através de uma interface gráfica, permitiram a avaliação da possibilidade de uso da função Bi-hiperbólica Simétrica como função de ativação em neurônios artificiais. Foram obtidos resultados altamente favoráveis, apresentados posteriormente neste trabalho. Esta interface possibilitou a análise gráfica do comportamento das funções Bi-Hiperbólica, Logística e Tangente Hiperbólica, o que foi importante para a definição inicial dos valores a serem utilizados na parametrização dos modelos de Backpropagation desenvolvidos.

Para o desenvolvimento e processamento dos testes apresentados dos modelos, foi utilizada a ferramenta de programação MATLAB 7.7, release R2008b. Este software é fornecido pela empresa The MathWorks e pode ser considerado como uma linguagem técnica de computação de alto nível, bem como um ambiente interativo para o desenvolvimento de algoritmos, visualização de dados, análise de dados e computação numérica. Com o seu uso, como ressalta Hanselman (2003), é

possível o desenvolvimento de modelos técnicos de computação mais rapidamente do que com as linguagens tradicionais. Para que se possa ter uma noção mais aproximada do tempo esperado de processamento, os valores indicados nos testes se referem à utilização de um sistema computacional cuja configuração básica é um micro-computador Dell XPS 154370, equipado com um processador Intel Core i7-2670QM, com 8 Gb de memória DDR3. O sistema operacional usado foi o Microsoft Windows 7 Home Premium.

O protótipo desenvolvido apresenta, apesar de sua simplicidade, as funções necessárias aos treinamentos e testes realizados, permitindo a execução do ciclo de treinamento e a verificação dos resultados obtidos pela comparação de desempenho entre os modelos usados.

Foram adotadas redes neurais artificiais do tipo Perceptron Multicamadas (Multilayer Perceptron – MLP) com o objetivo de aprender as formas apresentadas. Cada rede é constituída por um conjunto de nós fonte, que formam a camada de entrada da rede, uma camada escondida e uma camada de saída. Todas as camadas são constituídas por neurônios e, portanto, apresentam capacidade computacional.

É uma rede progressiva (feedforward), completamente conectada, com cada nó em uma camada conectado a todos os outros nós da camada adjacente.

Na definição da arquitetura da rede, o número de nós na camada de entrada da rede é determinado pela dimensionalidade do espaço de observação responsável pela geração dos sinais de entrada, enquanto que o número de neurônios na camada de saída é determinado pela dimensionalidade requerida da resposta desejada.

A existência de camadas ocultas se deve para permitir a extração de estatísticas de ordem superior de algum processo aleatório subjacente, responsável pelo "comportamento" dos dados de entrada, que é basicamente o processo sobre o qual a rede está tentando adquirir conhecimento.

Para a determinação do número de camadas escondidas e do número de neurônios em cada uma destas camadas, como não existem regras específicas para tal especificação, foi adotado o uso de uma camada oculta. Este é um valor arbitrário, podendo variar em função da análise de desempenho do modelo.

A decisão de utilizar uma única camada oculta é baseada na demonstração feita por Robert Hecht-Nielsen (1989) de que, teoricamente, o uso de três camadas é sempre suficiente para a aproximação de qualquer função. Entretanto, ele resalta que, em se tratando de problemas do mundo real, esta aproximação por apenas três camadas poderia resultar na necessidade de uma quantidade de neurônios na camada oculta extremamente grande, fazendo com que seja mais prático o uso de um

maior número de camadas para a obtenção de uma solução tratável (Hecht-Nielsen, 1989).

Robert Hecht-Nielsen (1989) indica uma heurística para a determinação do número de neurônios nesta camada oculta. Com base no Teorema de Kolmogorov de que qualquer função de ***n*** variáveis pode ser representada por ***2n + 1*** funções de uma variável, ele define a seguinte expressão para a determinação desta quantidade de neurônios:

$$N_o = 2\,N_e + 1 \qquad (5.1)$$

Onde,

N_o = número de neurônios ocultos

N_e = número de entradas.

Para permitir que a rede neural artificial adquira uma perspectiva global do processo aleatório, apesar de sua conectividade local, é fundamental a utilização de um algoritmo de treino supervisionado adequado. O algoritmo de treino consagrado pela literatura para isto é o Algoritmo de Retro-propagação do Erro (Backpropagation), que foi o utilizado neste trabalho.

Para possibilitar a avaliação comparativa da função proposta, o protótipo desenvolvido faz o treinamento em três redes distintas, com os mesmos parâmetros básicos e com o uso de funções de ativação diferenciadas. Em todos os casos, o modelo de neurônio de cada rede inclui uma função de ativação não-linear, diferenciável em qualquer ponto, com a forma de não-linearidade sigmoidal.

A primeira rede utiliza como função de ativação a Função Logística que é dada pela equação:

$$y_j = 1/(1+e^{(-\alpha v_j)}) \qquad (5.2)$$

Onde,

α é o parâmetro associado à inclinação da Função Logística,

$\boldsymbol{v_j}$ é o potencial de ativação (isto é, a soma ponderada de todas as entradas sinápticas mais a polarização) do neurônio **j**,

$\mathbf{y_j}$ é a saída do neurônio **j**.

Esta função assume valores contínuos no intervalo entre [0, 1].

A segunda rede utiliza como função de ativação a Função Tangente Hiperbólica. Esta função pode ser considerada, como indica Haykin (2001), como uma Função Logística re-escalada e modificada por um bias. Para sua utilização neste trabalho foi adotada uma versão mais geral com os parâmetros apontados na literatura como os mais adequados, que é dada pela equação:

$y_j = a \tanh(b v_j)$ (5.3)

Onde,

a assume o valor constante de 1,7159,

b assume o valor constante de 2/3,

v_j é o potencial de ativação, isto é, a soma ponderada de todas as entradas sinápticas mais a polarização do neurônio **j,**

y_j é a saída do neurônio **j**.

Esta função assume valores contínuos no intervalo entre [-1, 1].

A terceira rede utiliza como função de ativação a Função Bi-Hiperbólica Simétrica, apresentada por Xavier (2005), dada pela equação:

$$y_j = \sqrt{\lambda^2 (v_j + \frac{1}{4\lambda})^2 + \tau^2)} - \sqrt{\lambda^2 (v_j - \frac{1}{4\lambda})^2 + \tau^2)} + \frac{1}{2} \quad (5.4)$$

onde

v_j é o potencial de ativação (isto é, a soma ponderada de todas as entradas sinápticas mais a polarização) do neurônio **j**, e

y_j é a saída do neurônio **j**.

Os parâmetros λ e τ são utilizados para controlar o funcionamento da função. Esta função assume valores contínuos no intervalo entre [0, 1].

As características das redes MLP são também responsáveis pelas dificuldades encontradas na análise de tais redes. Em uma rede MLP, o conhecimento aprendido sobre o ambiente é representado pelos valores assumidos pelos pesos sinápticos da rede. A natureza distribuída deste conhecimento ao longo da rede a torna de difícil interpretação. Além disso, o uso de neurônios escondidos torna o processo de aprendizado mais difícil de ser "visualizado" na estrutura da rede.

5.4 Bases de Dados para Teste do Modelo

5.4.1 *Wisconsin Breast Cancer Data*

Para possibilitar a obtenção de dados comparativos, foi utilizada a base de dados conhecida como "*Wisconsin Breast Cancer Data*", disponível no site UC Irvine Machine Learning Repository[3], do *Center for Machine Learning and Intelligent Systems* da Universidade da Califórnia em Irvine. Ela é formada por dados disponibilizados originalmente pelo Dr. William H. Wolberg (1990), do *University of Wisconsin Hospitals* e mais detalhes sobre sua utilização podem ser encontrados em Mangasarian (1990) e em Frank (2010).

Ela tem sido bastante utilizada em artigos publicados na área médica e de reconhecimento de padrões, facilitando as comparações, como recomenda Prechelt (1994), dos resultados a serem obtidos

É uma base com um razoável número de amostras e com atributos e padrões bem definidos. É formada pelos valores obtidos das características de cada instância das amostras de células obtidas nos exames de Aspiração por Agulha Fina de nódulos com suspeitas de malignidade, encontradas em exames de mamas humanas, como apresentado por Mangasarian, Street e Wolberg (1990).

Cada amostra apresenta um identificador e nove atributos descritivos das características observadas, que utilizam uma escala numérica padronizada. A cada amostra está associado o resultado da avaliação feita por especialistas, classificando-as como benignas (resultado negativo) ou malignas (resultado positivo).

Foram utilizadas 683 amostras, sendo 444 classificadas como benignas (65 %) e 239 classificadas como malignas (35 %).

[3] disponível em *http://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Original)*,

As amostras são formadas pelos seguintes atributos:

#	Atributos	Domínio
1	*Sample code number*	identificação
2	*Clump Thickness*	1 - 10
3	*Uniformity of Cell Size*	1 - 10
4	*Uniformity of Cell Shape*	1 - 10
5	*Marginal Adhesion*	1 - 10
6	*Single Epithelial Cell Size*	1 - 10
7	*Bare Nuclei*	1 - 10
8	*Bland Chromatin*	1 - 10
9	*Normal Nucleoli*	1 - 10
10	*Mitoses*	1 - 10
11	*Class*	(2 - benigno, 4 - maligno)

5.4.2 *Vertebral Column Data Set*

Para permitir mais avaliações dos modelos, foi utilizada uma nova base de dados para a classificação de condições encontradas em exames de colunas vertebrais, como mostrado em Neto (2011). Ela contém valores para seis características biomecânicas usadas para a classificação de pacientes ortopédicos em duas classes (normal ou anormal). Disponibilizada por Guilherme de Alencar Barreto, Ajalmar Rêgo da Rocha Neto do Departamento de Engenharia de Teleinformática, da Universidade Federal do Ceará e Henrique Antonio Fonseca da Mota Filho do Hospital Monte Klinikum de Fortaleza, pode ser encontada no site *UCI Machine Learning Repository*[4]*.*

Estes dados foram obtidos pelo Dr. Henrique da Mota durante seu período de residência médica no *Group of Applied Research in Orthopaedics (GARO)* do *Centre Médico-Chirurgical de Réadaptation des Massues* em Lyon, França. Os dados utilizados foram agrupados em duas classes: a primeira, com pacientes classificados como situação normal, com 100 instâncias, e, a segunda, com pacientes portadores de Hérnia de Disco ou Espondilolistese, contendo 210 instâncias, classificados como situação anormal.

[4] Disponível em *http://archive.ics.uci.edu/ml*

Foram utilizadas 310 amostras, sendo 100 classificadas como normais (32,3 %) e 210 classificadas como anormais (67,7 %)

As amostras são formadas pelos seguintes atributos:

#	Atributos	Domínio
1	*pelvic_incidence*	numérico
2	*pelvic_tilt*	numérico
3	*lumbar_lordosis_angle*	numérico
4	*sacral_slope*	numérico
5	*pelvic_radius*	numérico
6	*degree_spondylolisthesis*	numérico
7	*class*	(Anormal, Normal)

Capítulo 6 Resultados Computacionais

6.1. Introdução

Para análise do desempenho computacional dos modelos foram utilizados na comparação os métodos Holdout e de Validação Cruzada k-folds com k igual a dez. Os resultados obtidos permitem avaliar melhor o desempenho de cada função de ativação usada, com relação ao tempo de processamento do treinamento, convergência e acurácia. A topologia da rede utilizada nos três modelos é de uma Rede Neural Progressiva Multicamadas (Multilayer Perceptrons - MLP) com uma camada escondida conforme apresentado na Figura 33 e descrita abaixo.

a) Topologia inicial da Rede Neural:
   - Uma camada externa com 10 nós, um para cada um dos nove atributos descritivos das características observadas e mais um para o controle do bias;
   - Uma camada escondida com 21 nós, definida com base na heurística proposta por Hecht-Nielsen (1989).
   - Uma camada de saída com 1 nó;
b) Nível de Erro Médio Quadrático considerado: menor que 0,001;
c) Taxa de aprendizado: 0,05

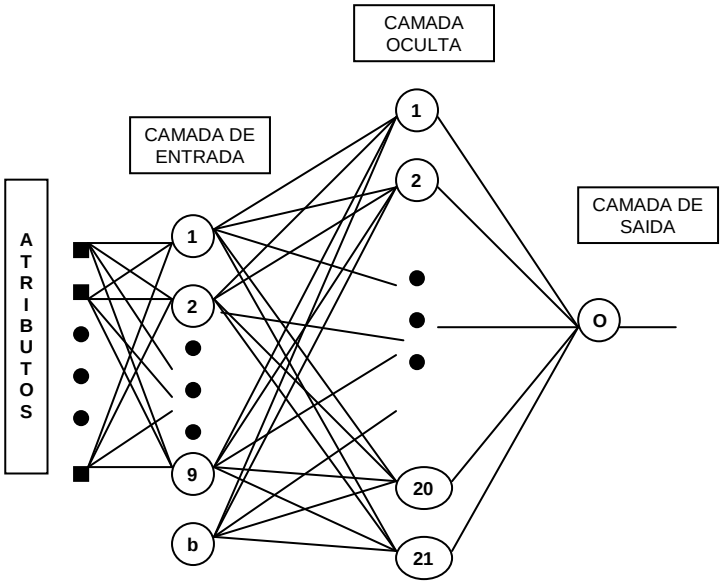


Figura 33: Topologia Inicial da Rede Neural

Foi feita uma aleatorização das instâncias na preparação inicial dos dados para evitar alguma tendência não conhecida devido a, por exemplo, a temporalidade da obtenção das amostras. Foi feita também uma normalização dos atributos originais

para uma escala de valores entre zero e um. Nenhum destes procedimentos altera as características das amostras, visando apenas facilitar a visualização e o manuseio dos dados.

Foram feitos testes de sensibilidade para os parâmetros dos modelos usando métodos de validação tradicionalmente usados para redes neurais. A seguir, são apresentadas as configurações e parametrizações utilizadas e os resultados obtidos, considerando os melhores resultados em termos de acurácia na obtenção de diagnósticos corretos, do número de épocas, ou iterações, usadas no treinamento com o conjunto de amostras destacado para tal fim e, também, qual a arquitetura utilizada na rede de melhor desempenho.

6.2 Resultados obtidos com a Base de Dados de Cancer de Mama

6.2.1 Resultados com o Método Holdout

Este método é simples e computacionalmente rápido. Para evitar o problema de alta variância, relativamente comum neste método, foram tomadas algumas precauções. Foram aleatorizadas as amostras, e foi feita uma estratificação inicial para garantir que a distribuição de probabilidades, dos padrões de resultados positivos e negativos dos exames, existente no conjunto total, fosse mantida tanto na partição destinada ao treinamento quanto na destinada à validação dos modelos obtidos.

Como o objetivo deste trabalho é avaliar a utilização da função Bi-hiperbólica Simétrica como função de ativação em neurônios artificiais, verificando se ofereceria as vantagens, previstas teoricamente, em relação às funções de ativação já usadas tradicionalmente, o conjunto de amostras disponíveis para o teste foi dividido em duas partições a serem usadas para treinamento e validação dos modelos. Inicialmente, optou-se por uma quantidade de 200 instâncias para o treinamento e 483 instâncias para a validação. Esta não é uma divisão tradicional pois, para agilizarmos os testes iniciais e verificarmos a capacidade de desempenho dos modelos, utilizamos uma quantidade de instâncias para treinamento bem menor do que o usualmente apresentado na literatura deste método. Estas amostras foram aleatorizadas sem estratificação.

Posteriormente, foi feito um experimento complementar usando uma divisão mais compatível com o sugerido na literatura especializada, ou seja, uma partição com aproximadamente 2/3 das amostras sendo utilizadas para o treinamento e o restante sendo usado para a validação dos resultados obtidos.

Em ambos os experimentos, após o particionamento do conjunto de amostras, foi realizado o treinamento da rede neural utilizando o subconjunto de dados próprios, obtendo-se, assim, os parâmetros de configuração. Após este treinamento, foi feita a validação do modelo usando o subconjunto de dados de teste. O Critério de Parada utilizado foi o de obter um Erro Médio Quadrático menor do que 0,001 ou, quando isto não foi possível, foi estabelecido um limite arbitrário para o número de épocas de treinamento igual a 1500. Com isso, foi possível calcular o erro de predição esperado. A acurácia deste modelo é calculada dividindo-se o número de amostras bem classificadas pelo número total de amostras no subconjunto de teste.

Para evitar a influência causada pelo uso de valores aleatórios favoráveis na inicialização dos pesos, todos os testes para cada um dos modelos foram feitos com o mesmo conjunto de valores iniciais para os pesos das ligações entre os neurônios das diversas camadas.

6.2.1.1 Experimento Inicial

6.2.1.1.1 Modelo com a Função de Ativação Logística

Para possibilitar uma avaliação comparativa com o desempenho dos modelos que utilizam outras funções de ativação foram feitos testes variando o parâmetro alfa e o número de neurônios na camada oculta. Utilizando a indicação encontrada em Haykin (2001) de considerar alfa, que é o parâmetro que define a inclinação da curva logística, como sendo positivo, foram feitos testes variando este parâmetro no intervalo entre 0,1 e 4,9, considerando um acréscimo de 0,2 em cada iteração. Isto gerou 25 opções de valores para os testes. Combinando estas opções com a variação da arquitetura obtida pela utilização de um número de neurônios na camada oculta desde 21, valor obtido pelo uso da heurística proposta por Hecht-Nielsen, até o limite arbitrário de cinco neurônios ocultos, foi gerada uma quantidade de 425 combinações a serem testadas.

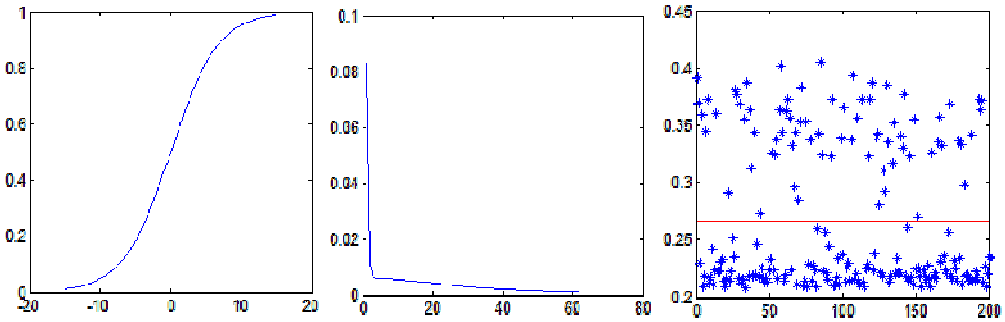
O parâmetro variável da curva Logística, o **α**, que apresentou o melhor resultado, em termos de acertos, foi **α** igual a 0,3 que convergiu em 62 épocas, em uma arquitetura composta por 21 neurônios na camada oculta, e apresentando sete diagnósticos errados, o que corresponde a um percentual menor que 1,45% de classificações erradas.

Considerando-se apenas as dezoito redes que apresentaram o melhor resultado obtido, com sete diagnósticos errados em 483 instâncias avaliadas, correspondendo a menos de 1,45% de erros, foram obtidas as configurações e valores apresentados na Tabela 1, ordenados pelo número de épocas necessárias à convergência do modelo.

AVALIAÇÃO DAS ARQUITETURAS FUNÇÃO LOGÍSTICA		
Épocas de convergência	Parâmetro alfa	Neurônios na camada oculta
62	0,3	21
64	0,3	20
75	0,1	20
76	0,1	19
77	0,1	18
78	0,1	17
79	0,1	16
81	0,1	15
83	0,1	14
85	0,3	9
85	0,1	13
87	0,1	12
89	0,1	11
92	0,1	10
95	0,1	9
175	0,9	13
176	0,9	12
177	0,9	11

Tabela 1: Configuração das redes com melhor generalização

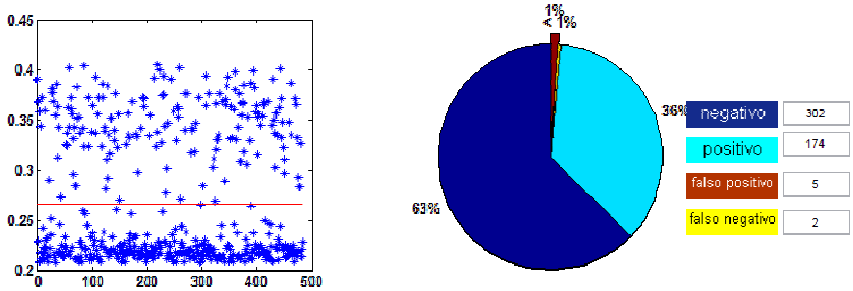
Para a arquitetura que apresentou o menor erro e convergência mais rápida, foram obtidos os gráficos de treinamento apresentados nas Figuras 34 (a), (b) e (c).



(a) curva usada (b) EQM (c) amostras - treinamento

Figura 34: Saídas obtidas na etapa de treinamento do modelo

Para esta mesma configuração foram obtidos os gráficos de validação apresentados nas Figuras 35 (a) e (b).



(a) amostras – validação (b) classificação das amostras

Figura 35: Resultados obtidos na avaliação do modelo

A Figura 36 apresenta o número de épocas de convergência para configurações com diferentes quantidades de neurônios ocultos e usando três diferentes valores para o parâmetro alfa. Nela podemos verificar que, em alguns casos, o mesmo resultado foi obtido por uma mesma arquitetura, mas com a utilização de parâmetros alfa diferentes e com a consequente variação do número de épocas em que a rede convergiu para apresentar o resultado igual.

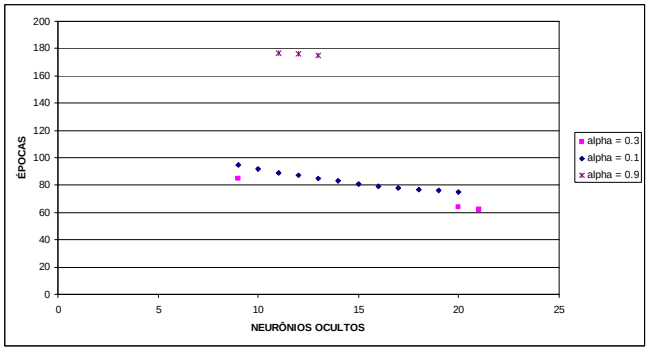


Figura 36: Avaliação das arquiteturas para a Função Logística

6.2.1.1.2 Modelo com a Função de Ativação Tangente Hiperbólica

Para possibilitar uma avaliação comparativa com o desempenho dos modelos que utilizam outras funções de ativação, foram feitos testes utilizando a função Tangente Hiperbólica em sua versão parametrizada, usando os valores recomendados para os parâmetros **a** e **b** encontrados em Haykin (2001), ou seja, considerar o valor de **a** igual a 1,7159 e o de **b** igual a 2/3.

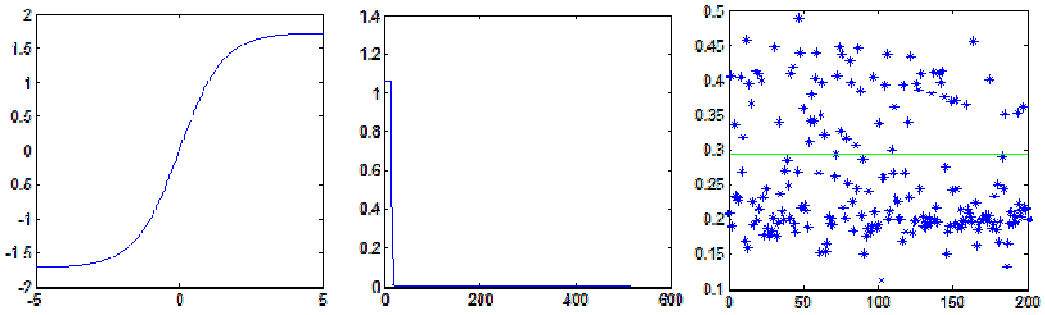
Estas opções foram combinadas com a variação da arquitetura obtida pela utilização de um número de neurônios na camada oculta desde 21, quantidade obtida pelo uso da heurística proposta por Hecht-Nielsen (1989), até o limite arbitrário de cinco neurônios ocultos, o que gerou, no total, uma quantidade de 17 combinações testadas.

O treinamento de cada rede foi feito até que o Erro Médio Quadrático atingisse um valor menor do que 0,001 ou fosse atingido o limite estipulado de 1.500 épocas. A configuração que obteve o melhor resultado apresentou uma quantidade de erros de diagnósticos igual a 13, ou seja, um percentual de erros menor do que 2,7%. Esta configuração está representada na Tabela 2.

AVALIAÇÃO DAS ARQUITETURAS FUNÇÃO TANGENTE HIPERBÓLICA			
Neurônios na camada oculta	Parâmetro a	Parâmetro b	Épocas de convergência
16	1,7159	0,666667	516

Tabela 2: Configuração da rede com melhor generalização

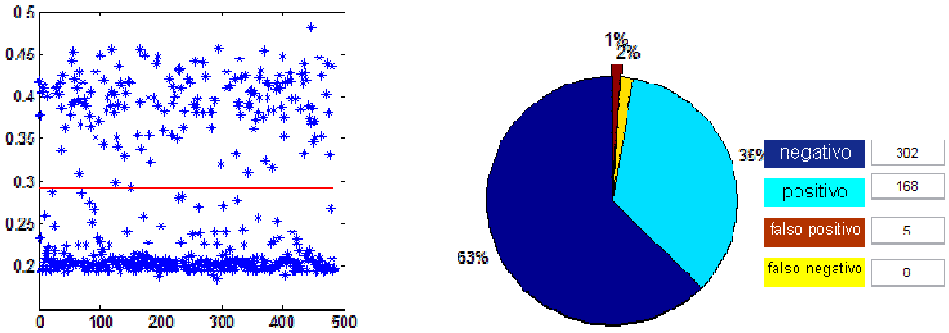
A arquitetura com melhor generalização apresentou os gráficos para a fase de treinamento mostrados nas Figuras 37 (a), (b) e (c).



(a) curva usada (b) EQM (c) amostras - treinamento

Figura 37: Saídas obtidas na etapa de treinamento do modelo

Os gráficos para a fase de validação são mostrados nas Figuras 38 (a) e (b).



(a) amostras – validação (b) classificação das amostras

Figura 38: Saídas obtidas na avaliação do modelo

6.2.1.1.3 Modelo com a Função de Ativação Bi-Hiperbólica Simétrica

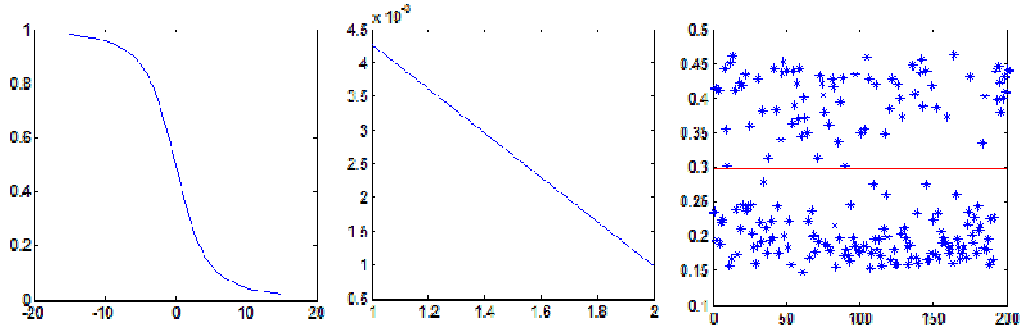
Para possibilitar uma avaliação comparativa do desempenho do modelo que utiliza a Função Bi-Hiperbólica Simétrica, foram feitos testes variando o parâmetro λ, que pode ser associado com a inclinação da curva na origem, no intervalo [-6, -0.5] e [0.5, 6], considerando um acréscimo de 0.5 a cada iteração. O parâmetro τ, que pode ser associado com o afastamento da curva às duas assíntotas horizontais, variou no intervalo [0.5, 5], considerando um acréscimo de 0.5 a cada iteração. Isto gerou 240 opções de valores para os testes. Combinando estas opções com a variação da arquitetura obtida pela utilização de um número de neurônios na camada oculta desde 21, quantidade obtida pelo uso da heurística proposta por Hecht-Nielsen (1989), até o limite arbitrário de cinco neurônios ocultos, foi gerada uma quantidade de 4080 combinações a serem testadas.

O treinamento de cada rede foi feito até que o Erro Médio Quadrático atingisse um valor menor do que 0,001 ou fosse atingido o limite estipulado de 1.500 épocas. Das configurações testadas 287 delas obtiveram como melhor resultado uma quantidade de sete diagnósticos errados, ou seja, uma taxa de erros menor do que 1,45 %. Deste total, 21 configurações convergiram em apenas duas épocas e estão representadas na Tabela 3.

AVALIAÇÃO DAS ARQUITETURAS FUNÇÃO BI-HIPERBÓLICA			
Neurônios na camada oculta	Parâmetro lambda	Parâmetro tau	Épocas de convergência
7	-6	4,5	2
7	-6	5	2
7	-5,5	4	2
7	-5,5	4,5	2
7	-5	4	2
7	-4,5	3,5	2
7	-4,5	4	2
7	-4	3	2
7	-4	3,5	2
7	-3,5	3	2
7	-3	2,5	2
7	-2,5	2	2
7	-2	1,5	2
9	-6	5	2
9	-5,5	4,5	2
9	-5,5	5	2
9	-5	4,5	2
9	-4,5	4	2
9	-4	3,5	2
9	-3,5	3	2
9	-3	2,5	2

Tabela 3: Redes com melhor convergência e generalização

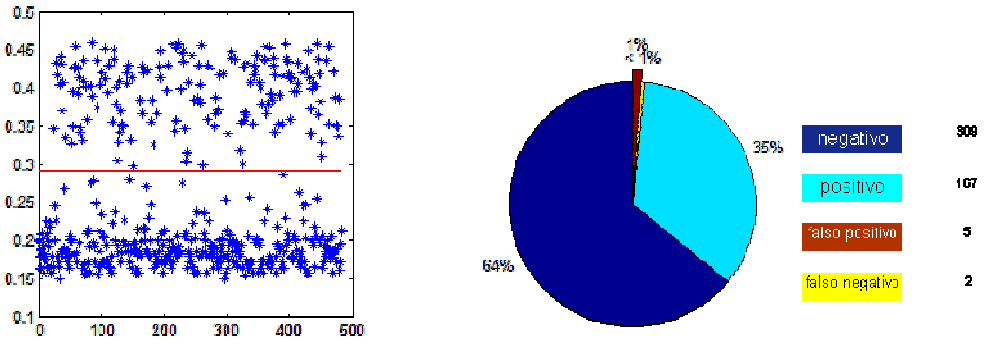
Como exemplo, dentre estas arquiteturas que apresentaram o menor número de erros e obtiveram a convergência mais rápida, utilizando um menor número de neurônios na camada oculta, a configuração correspondente à da primeira linha da Tabela 3 apresentou os gráficos para a fase de treinamento mostrados nas Figuras 39 (a), (b) e (c).



(a) curva usada b) EQM (c) amostras - treinamento

Figura 39: Saídas obtidas na etapa de treinamento do modelo

Os gráficos para a fase de validação desta rede são mostrados nas Figuras 40 (a), (b).



(a) amostras – validação (b) classificação das amostras

Figura 40: Saídas obtidas na avaliação do modelo

O processamento deste modelo com a variação dos parâmetros citada anteriormente permitiu, também, a obtenção de outros resultados muito interessantes. Por exemplo, considerando como melhor resultado o número de épocas, importante no caso do uso em sistemas computacionais mais lentos ou da necessidade de treinamento mais rápido, foram obtidas 68 combinações que convergiram em apenas uma época e que apresentaram entre oito e nove diagnósticos errados, uma acurácia de mais de 98,1%.

6.2.1.1.4 Avaliação do experimento preliminar

Esta avaliação pelo Método Holdout apresentou um resultado bastante animador em relação à hipótese inicial de possibilidade de uso da Função Bi-hiperbólica como função de ativação do neurônio artificial. Para atingir o mesmo grau de acertos, com a mesma arquitetura, a convergência do modelo usando a Função Bi-Hiperbólica Simétrica necessitou de apenas duas épocas, enquanto que o modelo equivalente, utilizando a Função Logística, convergiu em 62 épocas. Ou seja, com número de iterações 31 vezes maior do que o observado no modelo que utilizou a ativação pela Função Bi-hiperbólica. Além do número de épocas, deve ser ressaltado que a arquitetura do modelo que utilizou a Função Bi-hiperbólica contava com apenas 7 neurônios na camada oculta, enquanto que o baseado na Função Logística apresentava uma arquitetura formada por 21 neurônios na camada oculta.

O modelo utilizando a função Tangente Hiperbólica obteve, como melhor resultado, um nível de erros maior, com 13 diagnósticos errados, convergindo neste caso em 516 épocas.

A robustez deste modelo com a função de ativação bi-hiperbólica fica fácil de ser verificada quando se analisa o amplo espectro da parametrização que possibilitou atingir os melhores resultados obtidos, ou seja, sete diagnósticos errados. A variação dos parâmetros lambda e tau utilizados pode ser vista na Figura 41, que mostra as combinações destes parâmetros nas configurações que obtiveram a melhor convergência.

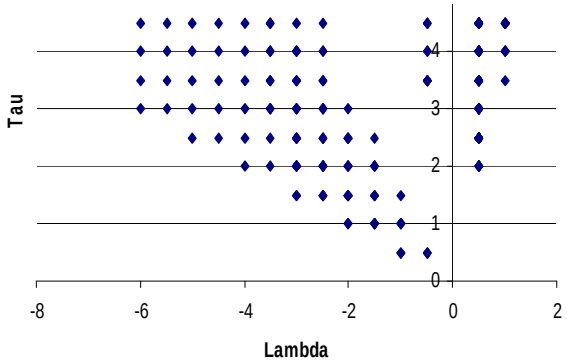


Figura 41: Combinações dos parâmetros com maior acerto

Complementando esta abrangência de parâmetros, outra importante característica deste modelo com a função de ativação bi-hiperbólica pode ser verificada ao se examinar o efeito da redução do número de neurônios na camada oculta. Foram feitos processamentos variando-se este número de neurônios desde 21 até 5. O modelo conseguiu convergir em todas estas topologias reduzidas, conforme pode ser visto na Figura 42, a distribuição da quantidade de épocas de processamento para cada topologia estudada. É importante lembrar que estamos falando somente dos casos que apresentaram o menor percentual de erros.

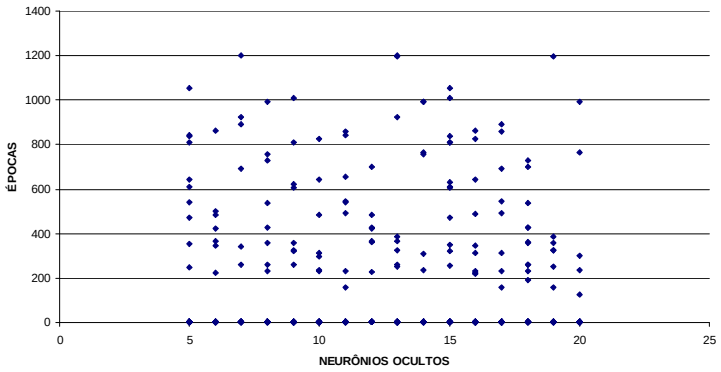


Figura 42: Número de neurônios ocultos e quantidade de épocas

A combinação destas características tem um forte impacto na melhoria do processo de modelagem de uma rede neural, uma vez que uma das maiores dificuldades para isto está na definição da melhor topologia e da parametrização que

permita a convergência e generalização do modelo. Este é um processo, essencialmente, de tentativas e erros. A robustez do modelo caracterizada pela ampla faixa de parametrização e de topologia que pode ser empregada para a obtenção dos melhores resultados, sem dúvida, facilita este processo de modelagem, uma vez que as opções de escolha para a obtenção dos melhores resultados é muito ampla.

6.2.1.2 Experimento Complementar

Para complementar a avaliação preliminar, foram executados novamente os testes apresentados anteriormente, mantendo-se a mesma gama de parâmetros, alterando apenas a composição dos subconjuntos utilizados para testes e validação dos modelos. As amostras foram aleatorizadas e estratificadas de forma a serem mantidas nos subconjuntos disjuntos de teste e avaliação as mesmas proporções de resultados existentes no conjunto total de amostras. Foram utilizadas 450 instâncias para formar o conjunto destinado aos testes e 233 instâncias para a execução da validação.

Para permitir a comparação do tempo de processamento de cada modelo foi registrado o tempo decorrido no treinamento através do uso das funções *tic* / *toc*, disponibilizadas pelo MATLAB. Estes valores não são uma medida de complexidade computacional e não devem ser considerados em sua forma absoluta. Servem apenas para efeitos de comparação entre os modelos analisados ao serem processados no mesmo ambiente computacional.

6.2.1.2.1 Modelo com ativação pela Função Logística

Seguindo o utilizado no teste anterior, o parâmetro alfa variou no intervalo entre 0,1 e 4,9, considerando um acréscimo de 0,2 em cada iteração. Estas instâncias foram combinadas com a variação da arquitetura obtida pela utilização de um número de neurônios na camada oculta desde 21, quantidade obtida pelo uso da heurística proposta por Hecht-Nielsen (1989), até o limite experimental de cinco neurônios ocultos.

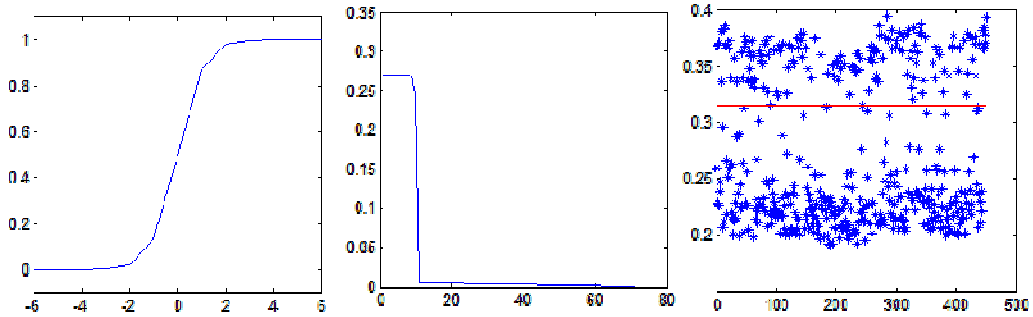
O treinamento de cada rede foi feito até que o Erro Médio Quadrático atingisse um valor menor do que 0,001 ou fosse atingido o limite arbitrário de 1.500 épocas. Das configurações testadas, oito obtiveram como melhor resultado a quantidade de três erros, configurando uma taxa de erros de 1,29%. As configurações que apresentaram estes resultados estão apresentadas na Tabela 4.

AVALIAÇÃO DAS ARQUITETURAS FUNÇÃO LOGÍSTICA			
Épocas de convergência	Parâmetro alfa	Neurônios na camada oculta	Tempo de processamento (u.t.)
71	1,9	12	2,85
82	1,9	13	3,52
116	2,1	13	4,98
199	2,3	13	8,36
66	1,7	14	2,85
82	1,9	14	3,54
440	2,1	15	19,40
1287	2,3	15	56,63

Tabela 4: Configuração das redes com melhor generalização

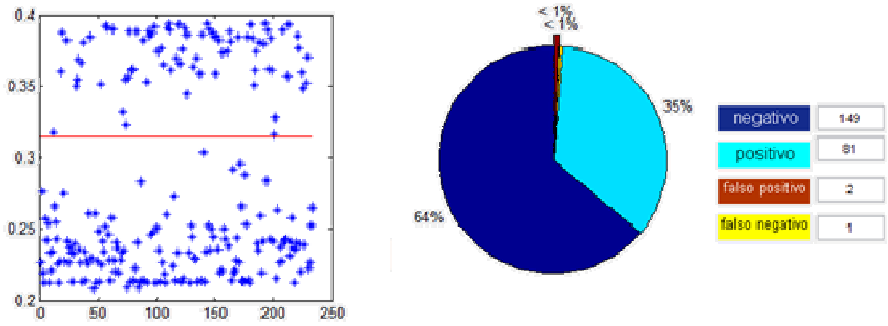
Este aumento na quantidade de instâncias disponibilizadas para o treinamento permitiu uma melhoria na capacidade de generalização do modelo, que apresentou um percentual maior de acertos na classificação dos padrões. Também a sua capacidade de convergência foi favorecida.

Como exemplo, dentre estas arquiteturas que apresentaram o menor número de erros, a que obteve a convergência mais rápida e utilizou o menor número de neurônios na camada oculta, apresentou os gráficos para a fase de treinamento mostrados nas Figuras 43 (a), (b) e (c). Os gráficos para a fase de teste são mostrados nas Figuras 44 (a), (b).



(a) curva usada (b) EQM (c) amostras - treinamento

Figura 43: Saídas obtidas na etapa de treinamento do modelo



(a) amostras – validação (c) classificação das amostras

Figura 44: Saídas obtidas na avaliação do modelo

Podemos verificar na Tabela 4 que o mesmo resultado foi obtido por diferentes configurações com a utilização de parâmetros alfa diferentes e com a consequente variação do número de épocas em que a rede convergiu para apresentar o resultado igual, bem como com números diferentes de neurônios na camada oculta.

6.2.1.2.2 Modelo com Ativação pela Função Tangente Hiperbólica

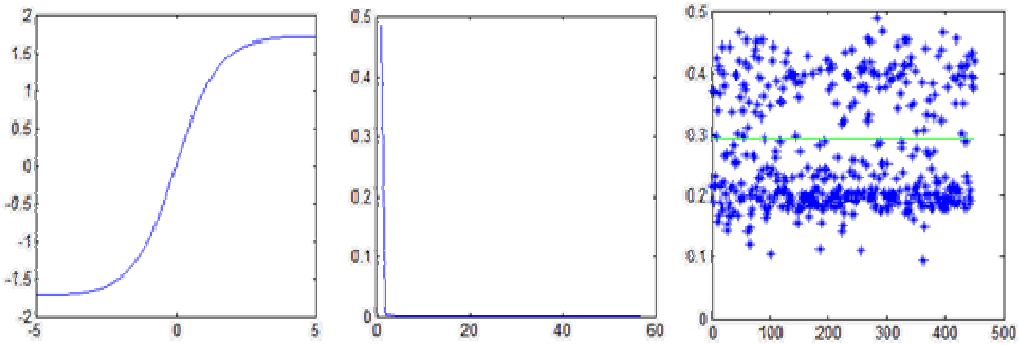
Seguindo o utilizado no teste anterior, foram feitos testes utilizando a função Tangente Hiperbólica em sua versão parametrizada, usando os valores recomendados para os parâmetros a e b encontrados em Haykin (2001), ou seja, considerar o valor de a igual a 1,7159 e o de b igual a 2/3. Estas instâncias foram combinadas com a variação da arquitetura obtida pela utilização de um número de neurônios na camada oculta desde 21, quantidade obtida pelo uso da heurística proposta por Hecht-Nielsen (1989), até o limite arbitrário de cinco neurônios ocultos, gerando uma quantidade de 17 combinações testadas.

O treinamento de cada rede foi feito até que o Erro Médio Quadrático atingisse um valor menor do que 0,001 ou fosse atingido o limite arbitrário de 1.500 épocas. A configuração que obteve o melhor resultado apresentou uma quantidade de erros de diagnósticos igual a cinco, ou seja, um percentual de erros igual a 2,15%. A configuração que apresentou este resultado está representada na Tabela 5.

	AVALIAÇÃO DAS ARQUITETURAS FUNÇÃO TANGENTE HIPERBÓLICA			
Épocas de convergência	Parâmetro a	Parâmetro b	Neurônios na camada oculta	Tempo de processamento (u.t.)
57	1,7159	0,666667	12	3,6

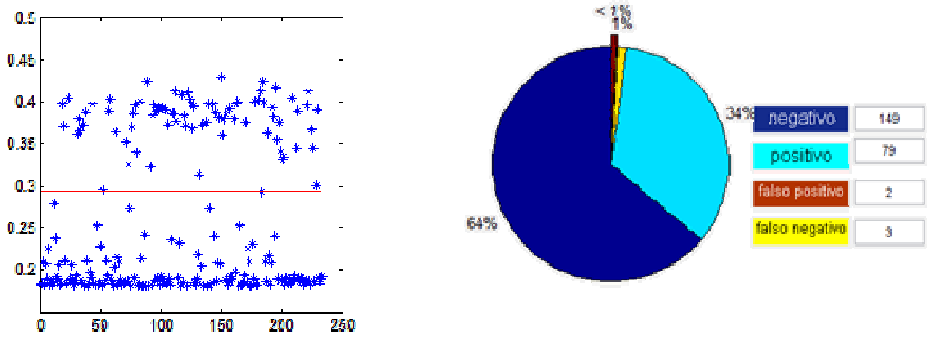
Tabela 5: Configuração da rede com melhor generalização

Esta arquitetura, que apresentou o menor número de erros, forneceu os gráficos para a fase de treinamento, mostrados nas Figuras 40 (a), (b) e (c). Os gráficos para a fase de validação são mostrados nas Figuras 41 (a), (b).



(a) curva usada (b) EQM (c) amostras - treinamento

Figura 45: Saídas obtidas na etapa de treinamento do modelo



(a) amostras – validação (c) classificação das amostras

Figura 46: Saídas obtidas na avaliação do modelo

6.2.1.2.3 Modelo com Ativação pela Função Bi-Hiperbólica Simétrica

Seguindo o utilizado no teste anterior, foram feitos testes variando o parâmetro λ no intervalo [-6, -0,5] e [0,5, 6], considerando um acréscimo de 0,5 a cada iteração. O parâmetro τ variou no intervalo [0.5, 5], considerando um acréscimo de 0,5 a cada iteração. Isto gerou 240 opções de valores para os testes. Combinando estas opções com a variação da arquitetura obtida pela utilização de um número de neurônios na camada oculta desde 21, quantidade obtida pelo uso da heurística proposta por Hecht-

Nielsen (1989), até o limite arbitrário de cinco neurônios ocultos, foi gerada uma quantidade de 4080 combinações a serem testadas.

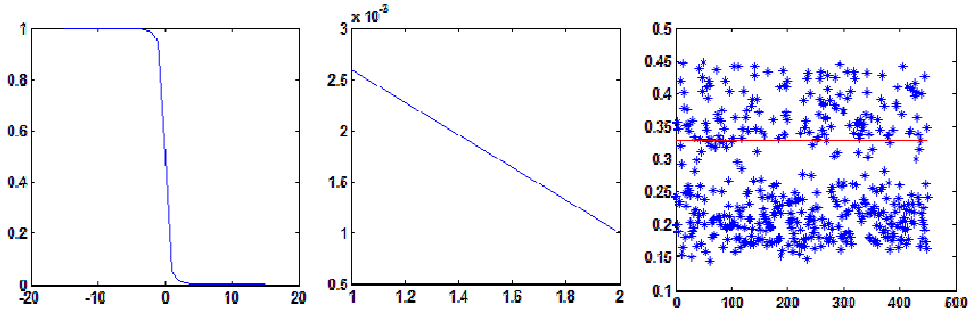
O treinamento de cada rede foi feito até que o Erro Médio Quadrático atingisse um valor menor do que 0,001 ou fosse atingido o limite arbitrário de 1.500 épocas. Vinte e quatro configurações obtiveram como melhor resultado uma quantidade de erros igual a três, ou seja, apresentou uma taxa de erros de 1,29 %. As configurações que apresentaram estes resultados estão apresentadas na Tabela 6.

AVALIAÇÃO DAS ARQUITETURAS FUNÇÃO BI-HIPERBÓLICA				
Neurônios na camada oculta	Parâmetro lambda	Parâmetro tau	Número de épocas	Tempo de processamento (u.t.)
9	-1,5	0,5	3	0,15
11	0,5	0,5	34	1,64
11	0,5	0,5	34	1,64
11	1,0	1,0	35	1,69
11	1,5	1,5	35	1,69
11	2,0	2,0	35	1,69
11	2,5	2,5	35	1,68
11	2,5	3,0	35	1,69
11	3,0	3,0	35	1,68
11	3,0	3,5	35	1,69
11	3,5	3,5	35	1,69
11	3,5	4,0	35	1,69
11	4,0	4,0	35	1,69
11	4,0	4,5	35	1,68
11	4,5	4,5	35	1,70
11	4,5	5,0	35	1,67
11	5,0	5,0	35	1,69
11	5,0	4,5	37	1,78
11	5,5	5,0	37	1,78
11	4,0	3,5	38	1,83
11	3,5	3,0	39	1,88
11	3,0	2,5	40	1,94
11	5,5	4,5	41	1,97
11	2,5	2,0	42	2,02

Tabela 6: Configuração das redes com melhor generalização

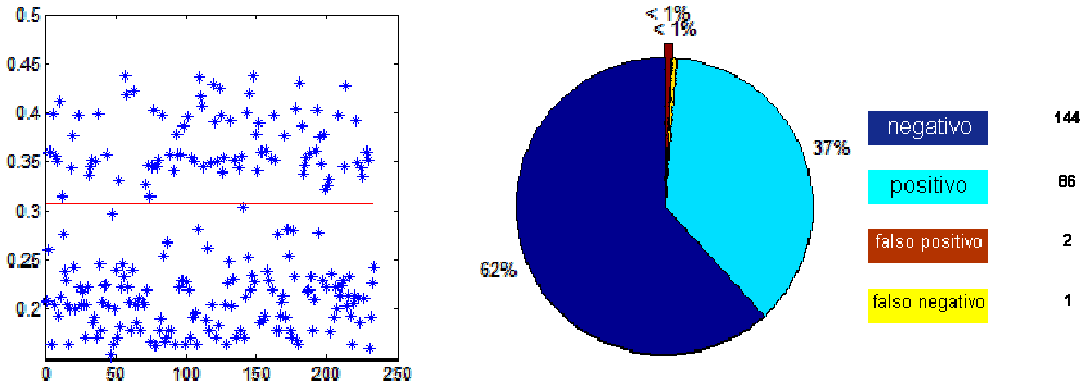
Como exemplo, dentre estas arquiteturas que apresentaram o menor número de erros, a que obteve a convergência mais rápida e utilizou o menor número de neurônios na camada oculta, apresentou os gráficos para a fase de treinamento

mostrados nas Figuras 47 (a), (b) e (c). Os gráficos para a fase de validação são mostrados nas Figuras 48 (a), (b).



(a) curva usada (b) EQM (c) amostras - treinamento

Figura 47: Saídas obtidas na etapa de treinamento do modelo



(a) amostras – validação (c) classificação das amostras

Figura 48: Saídas obtidas na avaliação do modelo

Podemos verificar na Figura 49 que, em alguns casos, o mesmo resultado foi obtido por uma mesma arquitetura, mas com a utilização de parâmetros lambda e tau diferentes e com a consequente variação do número de épocas em que a rede convergiu para apresentar o resultado igual.

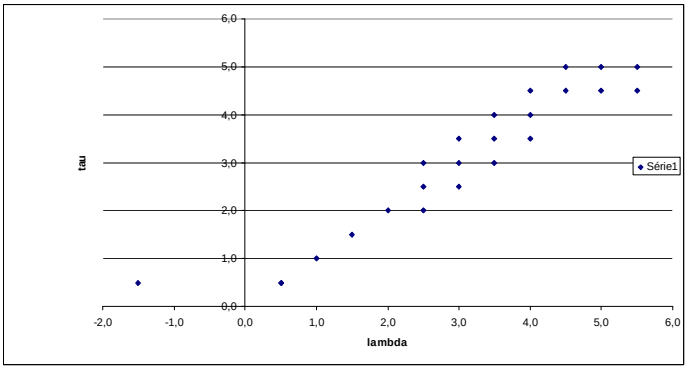


Figura 49: Combinações de parâmetros lamba e tau

6.2.1.2.4 Avaliação do experimento complementar

Este novo processamento usando 450 amostras no subconjunto de treinamento levou a uma melhor generalização para todos os modelos. Entretanto, houve uma mudança nas convergências com um aumento no número de épocas para atingir os melhores resultados.

O modelo que usou como função de ativação a função logística e o que usou a função tangente hiperbólica apresentaram melhores resultados e usaram arquiteturas com um menor número de neurônios na camada oculta, certamente se beneficiando do treinamento com um maior número de amostras.

Já o modelo que usou a função bi-hiperbólica apresentou um ligeiro aumento tanto no número de neurônios utilizados na camada oculta, quanto no número de épocas necessárias para convergir. Isto pode demonstrar que o aumento do número de amostras no treinamento foi desnecessário e excessivo.

Ficou clara a robustez deste modelo que, mesmo em condições desfavoráveis, apresentou resultados melhores em relação à arquitetura da rede e no tempo de processamento necessário para o seu treinamento. Ficou comprovada a sua necessidade de menos treinamento, de uma arquitetura mais enxuta, e da redução do tempo necessário para o treinamento, além da sua maior versatilidade com a obtenção dos melhores resultados para uma maior variedade de combinações de parâmetros e arquiteturas.

Podemos considerar que este teste demonstrou a superioridade do modelo que usou como função de ativação a função bi-hiperbólica, tanto em relação ao tempo de processamento, ao menor número de neurônios na camada oculta, quanto à necessidade de treinamento, confirmando as hipóteses previstas.

6.2.2 Resultados com a Validação Cruzada

Para permitir uma validação mais completa foram utilizados métodos de validação cruzada múltipla que possibilitam avaliar o comportamento do modelo para diversas partições dos dados disponíveis para o treinamento. Estes testes buscam avaliar se existe algum desvio significativo dos resultados para partições diferentes dos dados.

O teste utilizado foi o da validação cruzada com 10 partições. Neste método, o conjunto é dividido em 10 partições iguais em números de elementos que são utilizadas combinadamente de forma a que o conjunto usado para treinamento tenha

80% das amostras usadas para a definição dos parâmetros livres da rede e os 20% restantes sejam usados para a validação do resultado obtido. Para melhor avaliação dos modelos, foi feita a aleatorização das amostras, mas sem a estratificação que deixaria todas as partições com a mesma probabilidade de resultados. Para evitar a influência da inicialização dos pesos por valores aleatórios, todos os testes foram feitos com os mesmos valores iniciais para os pesos das ligações entre os neurônios das diversas camadas.

6.2.2.1 Modelo com ativação pela Função Logística

O modelo usando como função de ativação a Função Logística foi submetido a este processo variando o parâmetro alfa no intervalo entre 0,1 e 4,9, considerando um acréscimo de 0,2 em cada iteração, possibilitando 25 opções. Combinando estas opções com a variação da arquitetura obtida pela utilização de um número de neurônios na camada oculta desde 21, até o limite arbitrário de 5 neurônios ocultos, gerou uma quantidade de 425 combinações testadas.

Cada teste é dividido em 10 experimentos (10-fold), obtidos pela combinação das partições definidas. O resultado final para cada conjunto formado por uma arquitetura e um valor para o parâmetro alfa foi obtido pelas médias dos valores observados no processamento de cada conjunto.

Considerando-se o número de acertos, o melhor resultado obtido foi de 1,8 diagnósticos errados. Isso equivale a 2,65% de erros e foi obtido com a configuração apresentada na Tabela 7.

AVALIAÇÃO DAS ARQUITETURAS FUNÇÃO LOGÍSTICA			
Épocas de convergência (média)	Parâmetro alfa	Neurônios na camada oculta	Tempo de processamento (u.t.)
29,5	0,5	8	13,29

Tabela 7: Configuração da rede com melhor generalização

6.2.2.2 Modelo com ativação pela Função Tangente Hiperbólica

Para possibilitar uma avaliação comparativa com o desempenho dos demais modelos que utilizam outras funções de ativação, foram feitos testes com o parâmetro a constante e igual a 1,7159 e o parâmetro b com o valor de 2/3, que são os valores

encontrados em Haykin (2001) para estas constantes. Combinando estes valores com a variação da arquitetura obtida pela utilização de um número de neurônios na camada oculta desde 21, quantidade obtida pelo uso da heurística proposta por Hecht-Nielsen (1989), até o limite arbitrário de 5 neurônios ocultos, foi gerada uma quantidade de 17 combinações testadas.

A configuração que apresentou o melhor resultado convergiu em 12,3 épocas, com uma arquitetura composta por 7 neurônios na camada oculta, e apresentou 1,9 diagnósticos errados, na média, em 68 amostras avaliadas o que corresponde a um percentual menor que 2,80% de respostas erradas.

AVALIAÇÃO DAS ARQUITETURAS FUNÇÃO TANGENTE HIPERBÓLICA				
Épocas de convergência	Parâmetro a	Parâmetro b	Neurônios na camada oculta	Tempo de processamento (u.t.)
12,3	1,7159	0,666667	7	7,44

Tabela 8: Configuração da rede com melhor generalização

6.2.2.3 Modelo com ativação pela Função Bi-Hiperbólica Simétrica

Para possibilitar uma avaliação comparativa do desempenho do modelo que utiliza a Função Bi-Hiperbólica Simétrica, foram feitos testes variando o parâmetro λ no intervalo [-6, -0,5] e [0,5, 6], considerando um acréscimo de 0,5 a cada iteração. O parâmetro τ, variou no intervalo [0,5, 5], considerando um acréscimo de 0,5 a cada iteração. A sua arquitetura teve uma variação no número de neurônios da camada oculta desde 21 neurônios, quantidade obtida pelo uso da heurística proposta por Hecht-Nielsen (1989), até o limite experimental de cinco neurônios ocultos. Com isso, foram testadas 4080 combinações.

Considerando-se o número de diagnósticos corretos obtidos na fase de testes, o melhor resultado encontrado apresentou na média 1,6 diagnósticos errados em 68 amostras avaliadas, ou seja, menos de 2,35%. Este resultado foi obtido em 12 combinações, sendo que em, 8 delas, foi necessário o processamento de 19,1 épocas, conforme pode ser visto na Tabela 9. Isto demonstra o enorme poder de convergência do modelo, bem como a sua capacidade de operar com uma rede de arquitetura com menos neurônios, o que facilita o seu uso em ambientes computacionais com menos recursos disponíveis.

AVALIAÇÃO DAS ARQUITETURAS FUNÇÃO BI-HIPERBÓLICA				
Neurônios na camada oculta	Parâmetro lambda	Parâmetro tau	Número de épocas	Tempo de processamento (u.t.)
7	1,5	1,5	19,1	8,89
7	2	2	19,1	8,67
7	2,5	2,5	19,1	8,70
7	3	3	19,1	8,64
7	3,5	3,5	19,1	8,59
7	4	4	19,1	8,55
7	4,5	4,5	19,1	8,60
7	5	5	19,1	8,41
8	2	2,5	20,8	9,61
8	2,5	3	20,8	9,47
8	4	5	20,8	9,56
18	5	1,5	427,9	256,11

Tabela 9: Configuração da rede com melhor generalização

6.2.2.4 Avaliação dos Testes com a Validação Cruzada

Este novo processamento usando 680 amostras no subconjunto de treinamento levou a uma melhor generalização média para todos os modelos. Entretanto, houve uma mudança nas convergências com um aumento no número de épocas para atingir os melhores resultados.

O modelo que usou a função bi-hiperbólica apresentou o melhor resultado em termos de generalização com 1,6 erros em média. Ele teve uma convergência em um número de épocas equivalente a 64,7% do que o apresentado pelo modelo que usou a função logística. Em comparação com o modelo que utilizou a função tangente hiperbólica, ele teve um percentual maior do número de épocas utilizado, em um valor equivalente a 55,3% a mais.

O aumento do número de amostras no treinamento certamente favoreceu ao modelo dotado da função tangente hiperbólica como função de ativação. Entretanto, o modelo com a função bi-hiperbólica continuou demonstrando a sua robustez, conseguindo obter a melhor generalização e um bom nível de convergência, apesar de ter seu desempenho prejudicado pelo treinamento excessivo.

6.3 Resultados obtidos com a Base de Dados de Coluna Vertebral

6.3.1 Resultados com o Método Holdout

Este é um método simples e computacionalmente rápido. Para evitar o problema de alta variância relativamente comum neste método, foram tomadas algumas precauções, aleatorizando as amostras e feita uma estratificação para garantir que a distribuição de probabilidades dos padrões de resultados positivos e negativos dos exames fosse mantida tanto na partição destinada ao treinamento quanto na destinada à validação dos modelos obtidos.

O experimento foi feito usando uma divisão compatível com o sugerido na literatura especializada, ou seja, uma partição com aproximadamente 2/3 das amostras utilizadas para o treinamento e o restante sendo usada para a validação dos resultados obtidos.

Neste experimento, após o particionamento do conjunto de amostras, foi realizado o treinamento da rede neural utilizando o subconjunto de dados próprios, obtendo-se assim os parâmetros de configuração. Após este treinamento, foi feita a validação do modelo usando o subconjunto de dados de teste. O Critério de Parada utilizado foi o de obter um Erro Médio Quadrático menor do que 0,003 ou, quando isto não foi possível, foi estabelecido um limite arbitrário para o número de épocas de treinamento igual a 1500. Com isso, foi possível calcular o erro de predição esperado. A acurácia deste modelo é calculada dividindo-se o número de amostras bem classificadas pelo número total de amostras no subconjunto de teste.

Para evitar a influência da inicialização dos pesos por valores aleatórios, todos os testes foram feitos com os mesmos valores iniciais para os pesos das ligações entre os neurônios das diversas camadas.

6.3.1.1 Modelo com a Função de Ativação Logística

Para possibilitar uma avaliação comparativa com o desempenho dos modelos que utilizam outras funções de ativação, foram feitos testes variando o parâmetro alfa e o número de neurônios na camada oculta. Utilizando a indicação encontrada em Haykin (2001) de considerar alfa, que é o parâmetro que define a inclinação da curva logística, como sendo positivo, foram feitos testes variando este parâmetro no intervalo

entre 0,1 e 4,9, considerando um acréscimo de 0,2 em cada iteração. Isto gerou 25 opções de valores para os testes. Combinando estas opções com a variação da arquitetura obtida pela utilização de um número de neurônios na camada oculta desde 15, valor obtido pelo uso da heurística proposta por Hecht-Nielsen (1989), até o limite arbitrário de cinco neurônios ocultos, foi gerada uma quantidade de 275 combinações a serem testadas. Foram utilizadas 206 instâncias para formar o conjunto destinado aos testes e 104 instâncias para a execução da validação.

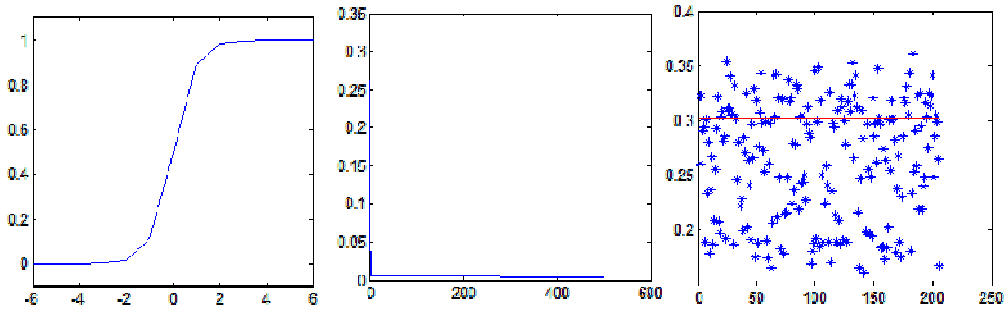
O parâmetro variável da curva Logística, o **α**, que apresentou o melhor resultado, em termos de acertos, foi **α** igual a 2,1 que convergiu em 504 épocas, em uma arquitetura composta por seis neurônios na camada oculta, e apresentando doze diagnósticos errados, o que corresponde a um percentual menor que 11,6% de respostas erradas.

A rede que apresentou o melhor resultado, com doze diagnósticos errados em 104 instâncias avaliadas, foi gerada com a configuração apresentada na Tabela 10.

AVALIAÇÃO DAS ARQUITETURAS FUNÇÃO LOGÍSTICA			
Épocas de convergência	Parâmetro alfa	Neurônios na camada oculta	Tempo de processamento (u.t.)
504	2,1	6	6,47

Tabela 10: Configuração da rede com melhor generalização

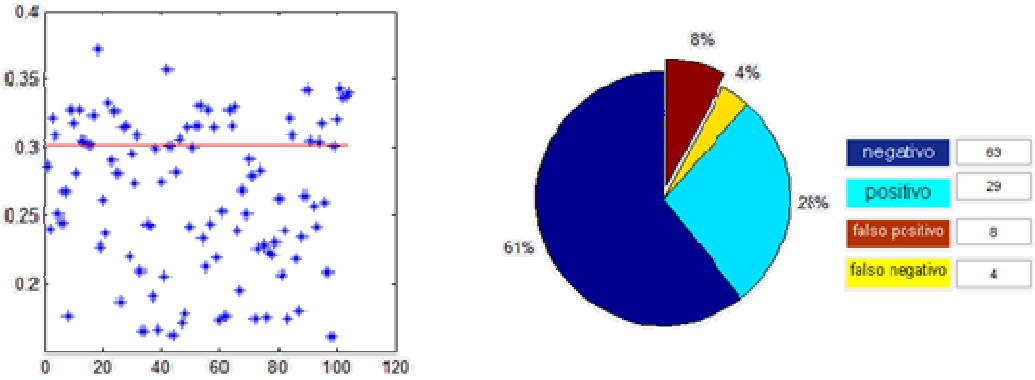
Para a arquitetura que apresentou o menor erro e convergência mais rápida, foram obtidos os gráficos de treinamento apresentados na Figuras 50 (a), (b) e (c).



(a) curva usada (b) EQM (c) amostras - treinamento

Figura 50: Saídas obtidas na etapa de treinamento do modelo

Para a arquitetura que apresentou o menor erro e convergência mais rápida, foram obtidos os gráficos de validação apresentados na Figuras 35 (a) e (b).



(a) amostras – validação (b) classificação das amostras

Figura 51: Resultados obtidos na avaliação do modelo

6.3.1.2 Modelo com a Função de Ativação Tangente Hiperbólica

Para possibilitar uma avaliação comparativa com o desempenho dos modelos que utilizam outras funções de ativação, foram feitos testes utilizando a função Tangente Hiperbólica em sua versão parametrizada, usando os valores recomendados para os parâmetros a e b encontrados em Haykin (2001), ou seja, considerar o valor de a igual a 1,7159 e o de b igual a 2/3.

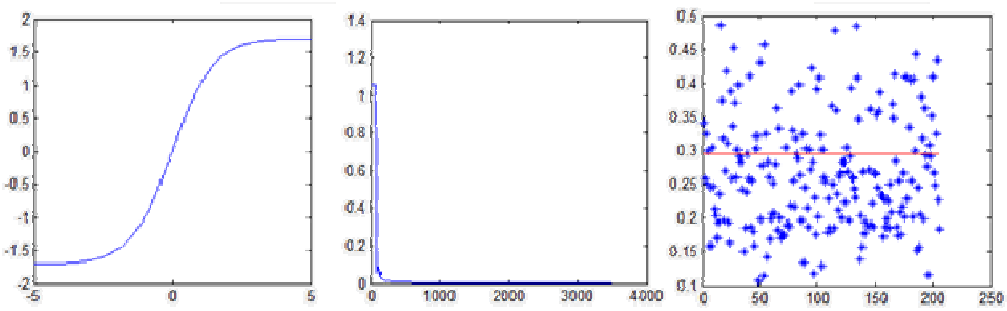
Estas opções foram combinadas com a variação da arquitetura obtida pela utilização de um número de neurônios na camada oculta desde 15, quantidade obtida pelo uso da heurística proposta por Hecht-Nielsen (1989), até o limite arbitrário de cinco neurônios ocultos, o que gerou, no total, uma quantidade de 11 combinações testadas.

O treinamento de cada rede foi feito até que o Erro Médio Quadrático atingisse um valor menor do que 0,003 ou fosse atingido o limite estipulado de 1.500 épocas. A configuração que obteve o melhor resultado apresentou uma quantidade de erros de diagnósticos igual a treze, ou seja, um percentual de erros menor do que 12,5%. Esta configuração está representada na Tabela 11.

AVALIAÇÃO DAS ARQUITETURAS FUNÇÃO TANGENTE HIPERBÓLICA				
Épocas de convergência	Parâmetro a	Parâmetro b	Neurônios na camada oculta	Tempo de processamento (u.t.)
1500	1,7159	0,666667	15	26,07

Tabela 11: Configuração da rede com melhor generalização

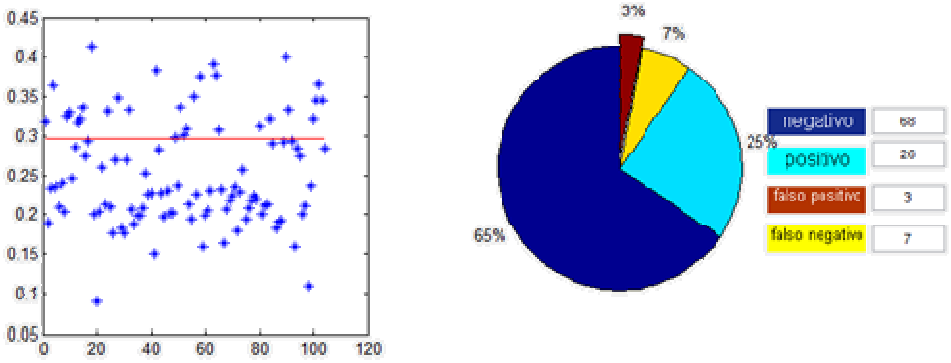
A arquitetura com melhor generalização apresentou os gráficos para a fase de treinamento mostrados nas Figuras 37 (a), (b) e (c).



(a) curva usada (b) EQM (c) amostras - treinamento

Figura 52: Saídas obtidas na etapa de treinamento do modelo

Os gráficos para a fase de validação são mostrados nas Figuras 53 (a) e (b).



(a) amostras – validação (b) classificação das amostras

Figura 53: Saídas obtidas na avaliação do modelo

6.3.1.3 Modelo com a Função de Ativação Bi-Hiperbólica Simétrica

Para possibilitar uma avaliação comparativa do desempenho do modelo que utiliza a Função Bi-Hiperbólica Simétrica, foram feitos testes variando o parâmetro λ, no intervalo [-6, -0,5] e [0,5, 6], considerando um acréscimo de 0,5 a cada iteração. O parâmetro τ, variou no intervalo [0,5, 5], considerando um acréscimo de 0,5 a cada iteração. Isto gerou 240 opções de valores para os testes. Combinando estas opções com a variação da arquitetura obtida pela utilização de um número de neurônios na camada oculta desde 15, quantidade obtida pelo uso da heurística proposta por Hecht-Nielsen (1989), até o limite arbitrário de cinco neurônios ocultos, foi gerada uma quantidade de 2.640 combinações a serem testadas.

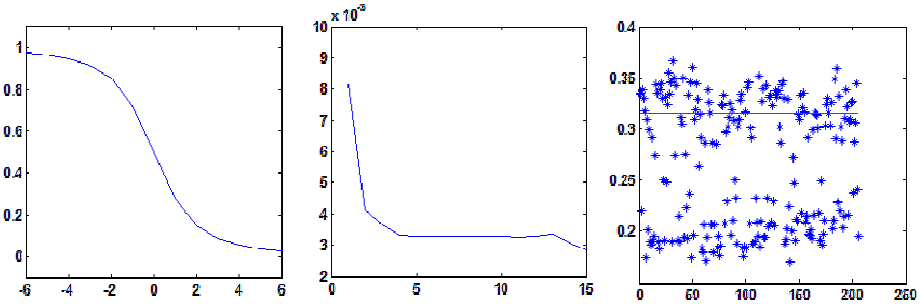
O treinamento de cada rede foi feito até que o Erro Médio Quadrático atingisse um valor menor do que 0,003 ou fosse atingido o limite estipulado de 1.500 épocas.

Das configurações testadas uma obteve como melhor resultado uma quantidade de oito diagnósticos errados, ou seja, uma taxa de erros menor do que 7,69%. Esta configuração está representada na Tabela 12.

AVALIAÇÃO DAS ARQUITETURAS FUNÇÃO BI-HIPERBÓLICA				
Épocas de convergência	Parâmetro lambda	Parâmetro tau	Neurônios na camada oculta	Tempo de processamento (u.t.)
15	-4	2	11	0,31

Tabela 12: Redes com melhor convergência e generalização

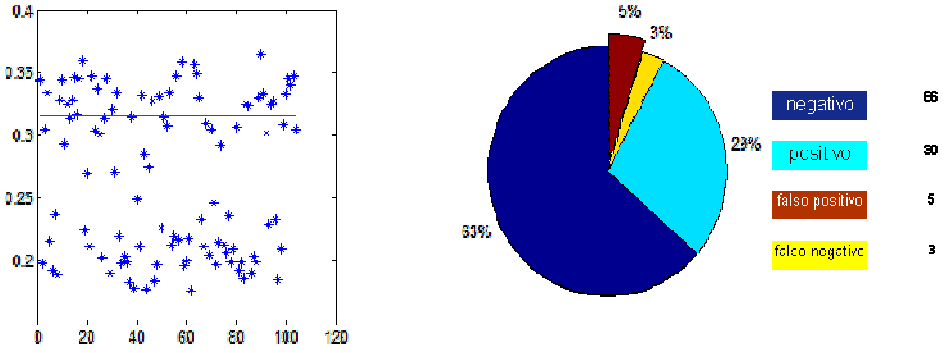
Como exemplo, dentre estas arquiteturas que apresentaram o menor número de erros, a que obteve a convergência mais rápida e utilizou o menor número de neurônios na camada oculta apresentou os gráficos para a fase de treinamento mostrados nas Figuras 39 (a), (b) e (c).



(a) curva usada b) EQM (c) amostras - treinamento

Figura 54: Saídas obtidas na etapa de treinamento do modelo

Os gráficos para a fase de validação desta rede são mostrados nas Figuras 55 (a), (b).



(a) amostras – validação (b) classificação das amostras

Figura 55: Saídas obtidas na avaliação do modelo

6.3.1.4 Avaliação do experimento com o Método Holdout

Esta avaliação pelo Método Holdout apresentou um resultado bastante animador em relação à hipótese inicial de possibilidade de uso da Função Bi-hiperbólica como função de ativação do neurônio artificial. O modelo usando a Função Bi-Hiperbólica Simétrica conseguiu obter vantagens na generalização, obtendo oito erros na classificação, com uma acurácia maior do que 92,31% enquanto que o modelo equivalente, utilizando a Função Logística, obteve uma acurácia de 88,46%. O modelo que utilizou a ativação pela Função Tangente Hiperbólica obteve uma acurácia de 87,50%.

Ao se considerar a capacidade de convergência dos modelos, também foi ressaltada a capacidade do modelo usando a Função Bi-Hiperbólica Simétrica que conseguiu convergir em 15 épocas. O modelo equivalente utilizando a Função Logística, que obteve uma acurácia menor, convergiu em 504 épocas, ou seja, 33,6 vezes o número de épocas utilizado pelo modelo que usou a função de ativação Bi-hiperbólica Simétrica. Neste atributo, o modelo que utilizou a ativação pela Função Tangente Hiperbólica não conseguiu convergir para o limite estabelecido de Erro Quadrado Médio igual a 0,003 dentro do número de épocas de processamento limitado a 1.500. Também no tempo de processamento necessário para atingir os melhores graus de convergência de cada modelo, o modelo que utilizou a Função Bi-Hiperbólica Simétrica necessitou de 0,31 unidades de tempo, enquanto que o modelo equivalente, utilizando a Função Logística, utilizou 6,47 unidades de tempo. Ou seja, o primeiro modelo foi mais de 20 vezes mais rápido. O modelo que utilizou a ativação pela Função Tangente Hiperbólica não conseguiu convergir para o limite estabelecido e utilizou 26,07 unidades de tempo para processar as 1500 épocas, demorando aproximadamente 0,02 unidades de tempo por época, contra o índice de aproximadamente 0,002 unidades de tempo por época do modelo que utilizou a Função Bi-Hiperbólica Simétrica.

Além do número de épocas, deve ser ressaltado que a arquitetura do modelo que utilizou a Função Bi-hiperbólica contava com 11 neurônios na camada oculta. O modelo baseado na Função Logística apresentava uma arquitetura formada por 6 neurônios na camada oculta e o modelo utilizando a função Tangente Hiperbólica utilizou 15 neurônios na camada oculta. Deve ser lembrado que o modelo que utilizou a Função Bi-hiperbólica obteve generalização e convergência superiores.

Se considerarmos o mesmo número de acertos do modelo com a Função Logística, o modelo que utilizou a Função Bi-Hiperbólica Simétrica consegiu este

mesmo número com 15 parametrizações diferentes, o que mostra a robustez deste modelo e a maior facilidade no processo de modelagem da rede neural.

6.3.2 Resultados com a Validação Cruzada

Para permitir uma validação mais completa foram utilizados métodos de validação cruzada múltipla que possibilitam avaliar o comportamento do modelo para diversas partições dos dados disponíveis para o treinamento. Estes testes buscam avaliar se existe algum desvio significativo dos resultados para partições diferentes dos dados.

O teste utilizado foi o da validação cruzada com 10 partições. Neste método o conjunto é dividido em 10 partições iguais em números de elementos que são utilizadas combinadamente de forma a que o conjunto usado para treinamento tenha 80% das amostras usadas para a definição dos parâmetros livres da rede e os 20% restantes sejam usados para a validação do resultado obtido. Para melhor avaliação dos modelos, foi feita a aleatorização das amostras, mas sem a estratificação que deixaria todas as partições com a mesma probabilidade de resultados. Para evitar a influência da inicialização dos pesos por valores aleatórios, todos os testes foram feitos com os mesmos conjuntos de valores iniciais para os pesos das ligações entre os neurônios das diversas camadas.

6.3.2.1 Modelo com ativação pela Função Logística

O modelo usando como função de ativação a Função Logística foi submetido a este processo variando o parâmetro alfa no intervalo entre 0,1 e 4,9, considerando um acréscimo de 0,2 em cada iteração, possibilitando 25 opções. Combinando estas opções com a variação da arquitetura obtida pela utilização de um número de neurônios na camada oculta desde 15, até o limite arbitrário de 5 neurônios ocultos, gerou uma quantidade de 275 combinações testadas.

Cada teste é dividido em 10 experimentos (10-fold), obtidos pela combinação das partições definidas. O resultado final para cada conjunto formado por uma arquitetura e um valor para o parâmetro alfa foi obtido pelas médias dos valores observados no processamento de cada conjunto. Tomando-se por base o número de acertos, o melhor resultado obtido foi de 6,4 diagnósticos errados, equivalente a 20,65% de erros, e foi obtido com as configurações apresentadas na Tabela 13.

AVALIAÇÃO DAS ARQUITETURAS FUNÇÃO LOGÍSTICA			
Épocas de convergência (média)	Parâmetro alfa	Neurônios na camada oculta	Tempo de processamento (u.t.)
79,6	1,1	6	14,11
117	1,5	6	20,85

Tabela 13: Configuração das redes com melhor generalização

6.3.2.2 Modelo com ativação pela Função Tangente Hiperbólica

Para possibilitar uma avaliação comparativa com o desempenho dos demais modelos que utilizam outras funções de ativação, foram feitos testes com o parâmetro a constante e igual a 1,7159 e o parâmetro b com o valor de 2/3, que são os valores encontrados em Haykin (2001) para estas constantes. Combinando estes valores com a variação da arquitetura obtida pela utilização de um número de neurônios na camada oculta desde 15, quantidade obtida pelo uso da heurística proposta por Hecht-Nielsen (1989), até o limite arbitrário de 5 neurônios ocultos, foi gerada uma quantidade de 17 combinações testadas.

A configuração que apresentou o melhor resultado convergiu em 1035,7 épocas, com uma arquitetura composta por 6 neurônios na camada oculta, e apresentou 6,2 diagnósticos errados, o que corresponde a um percentual de 20,0% de respostas erradas.

AVALIAÇÃO DAS ARQUITETURAS FUNÇÃO TANGENTE HIPERBÓLICA				
Épocas de convergência	Parâmetro a	Parâmetro b	Neurônios na camada oculta	Tempo de processamento (u.t.)
1035,7	1,7159	0,666667	6	262,99

Tabela 14: Configuração da rede com melhor generalização

6.3.2.3 Modelo com ativação pela Função Bi-Hiperbólica Simétrica

Para possibilitar uma avaliação comparativa do desempenho do modelo que utiliza a Função Bi-Hiperbólica Simétrica, foram feitos testes variando o parâmetro λ, no intervalo [-6, -0,5] e [0,5, 6], considerando um acréscimo de 0,5 a cada iteração.

O parâmetro τ variou no intervalo [0,5, 5], considerando um acréscimo de 0,5 a cada iteração. A sua arquitetura teve uma variação no número de neurônios da

camada oculta desde 15 neurônios, quantidade obtida pelo uso da heurística proposta por Hecht-Nielsen (1989), até o limite experimental de cinco neurônios ocultos. Com isso, foram testadas 2640 combinações.

Considerando-se o número de diagnósticos corretos obtidos na fase de testes, o melhor resultado encontrado apresentou, na média 5,8 diagnósticos errados em 31 amostras avaliadas, ou seja, menos de 18,71%. Esse resultado foi obtido em um processamento de 258,3 épocas, utilizando a configuração apresentada na Tabela 15.

AVALIAÇÃO DAS ARQUITETURAS FUNÇÃO BI-HIPERBÓLICA				
Neurônios na camada oculta	Parâmetro lambda	Parâmetro tau	Número de épocas	Tempo de processamento (u.t.)
13	3	1	258,3	57,76

Tabela 15: Configuração da rede com melhor generalização

6.3.2.4 Avaliação dos Testes com a Validação Cruzada

Este novo processamento usando 206 amostras no subconjunto de treinamento levou a uma melhor generalização percentual média para todos os modelos com a função de ativação Logística e a Tangente Hiperbólica. Já o modelo com a Função Bi-Hiperbólica registrou um aumento no número de épocas para convergir em comparação com os valores obtidos pelo método Holdout. Pode-se atribuir este fato ao treinamento excessivo gerado pelo aumento das amostras usadas no processo de aprendizagem.

O modelo que usou a função bi-hiperbólica apresentou o melhor resultado em termos de generalização com 5,8 erros, em média, equivalente a uma acurácia de 81,29%.

O aumento do número de amostras no treinamento certamente favoreceu aos modelos dotado das funções logísticas e tangente hiperbólica como função de ativação. Entretanto, o modelo com a função bi-hiperbólica continuou demonstrando a sua robustez, conseguindo obter a melhor generalização e um bom nível de convergência, apesar de ter seu desempenho prejudicado pelo treinamento excessivo.

6.4 Comparação das derivadas das funções de ativação

Como a Função Tangente Hiperbólica pode ser considerada, como apresenta Haykin (2001), como uma Função Logística re-escalada e modificada por um bias, a comparação das derivadas será feita considerando as Funções Logística e a Bi-Hiperbólica.

Para a obtenção das curvas Logística e Bi-hiperbólica com a mesma inclinação na origem, foram adotados os critérios de cálculo dos parâmetros λ e τ propostos por XAVIER (2005) e apresentados a seguir. Foi escolhido um valor para α igual a 0,3 para a Função Logística e, para a determinação do valor de λ na parametrização da Função Bi-Hiperbólica foi considerada a expressão $\lambda = 4\lambda'\sqrt{1/16 + \tau^2}$, onde λ' é igual ½ do valor da derivada da Função Logística para o ponto de origem considerado e o valor adotado para τ foi de 0,5. As curvas obtidas estão representadas na Figura 56, onde o gráfico representativo da função Bi-hiperbólica é traçado com uma linha pontilhada, enquanto que o da Função Logística é traçado com uma linha contínua. As duas funções possuem a mesma inclinação na origem, conforme descrito anteriormente.

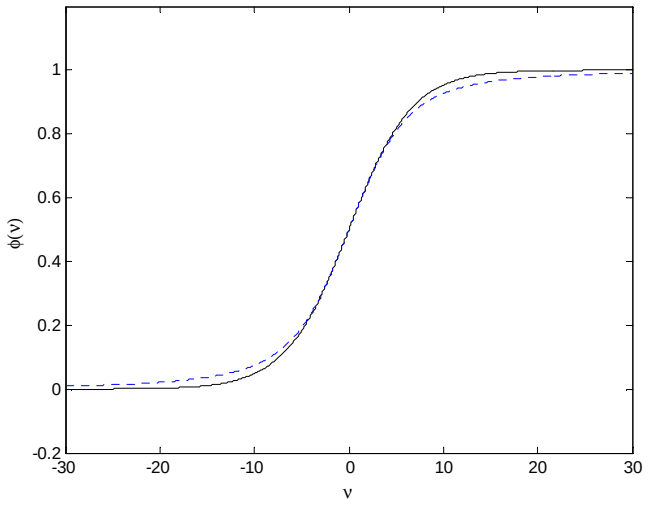


Figura 56: Bi-Hiperbólica (....) e Logistica - mesma inclinação na origem

As respectivas derivadas destas funções são apresentadas na Figura 57, onde a derivada da função Bi-hiperbólica é traçada com uma linha pontilhada e a derivada da função Logística é traçada de forma contínua.

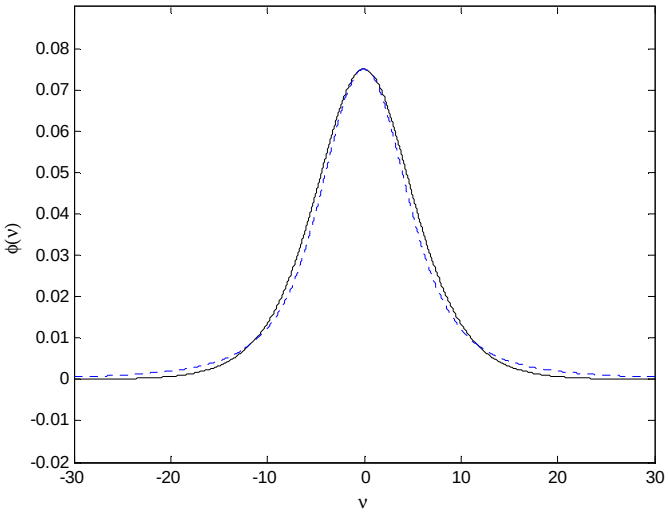


Figura 57: Derivada da função Bi-Hiperbólica (....) e da Logística

Como pode ser visto na Figura 57, ambas as derivadas tendem para zero a medida em que os valores fornecidos para a entrada das funções se afastam positiva ou negativamente do valor zero. O que faz a diferença é o maior vigor no crescimento da taxa de variação da derivada da função bi-hiperbólica em comparação com o da função Logística.

Para podermos avaliar este fator, temos que considerar o comportamento da razão entre o valor da derivada da função Logística e o da derivada da função Bi-hiperbólica. Para representar o impacto desta taxa de variação foram utilizadas curvas com a mesma inclinação na origem, conforme pode ser visto na Figura 56, e as suas derivadas para este caso que são apresentadas na Figura 57. Mantidos os parâmetros básicos, foram desenhadas as relações entre a derivada da Função Bi-hiperbólica e a da Função Logística, variando-se a amplitude dos valores de entrada fornecidos. Podemos verificar que esta razão entre as derivadas apresenta um comportamento que justifica a afirmação feita: Após um período de pequenas variações, esta razão cresce exponencialmente, corroborando a afirmação de ser este o fator preponderante que evita a saturação e agiliza o processo de convergência da rede.

Na Figura 58, traçada para um intervalo com amplitude de valor 10 em relação à origem, em um pequeno intervalo em torno da origem a derivada da função Logística se apresenta ligeiramente maior do que a da função Bi-hiperbólica. Logo depois, essa situação se inverte e o crescimento do valor da derivada da função Bi-hiperbólica é bem superior ao da outra derivada. Esse é um fato muito importante em se tratando de um algoritmo baseado em gradientes, como o Backpropagation.

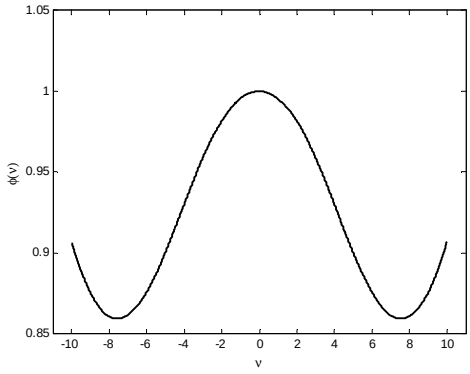


Figura 58: Razão entre derivadas - amplitude de -10 a 10

Uma vantagem adicional da função Bi-hiperbólica reside na maior facilidade de modelagem possibilitada pela utilização de mais um parâmetro que pode, inclusive, permitir que ela assuma uma forma bastante similar à apresentada pela função Logística, caso isto seja necessário. Com isso, esta amplitude, onde o comportamento da derivada da função Logística é maior do que o da derivada da função Bi-hiperbólica, pode ser controlado, fazendo com que ele aumente ou diminua pela simples alteração do valor do parâmetro τ.

Utilizando os parâmetros apresentados acima para a obtenção das curvas com mesma inclinação na origem, foram traçados outros gráficos representativos da razão entre as derivadas, variando o intervalo dos valores fornecidos na entrada, comprovando o maior vigor na inclinação da derivada da função Bi-hiperbólica. Estes gráficos podem ser vistos nas Figuras 59 a 62, onde a amplitude do intervalo varia desde -15 a 15, até o intevalo de entrada variando de -100 a 100.

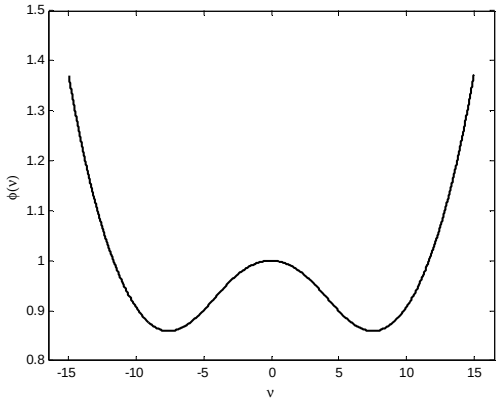


Figura 59: Razão entre derivadas - amplitude de -15 a 15

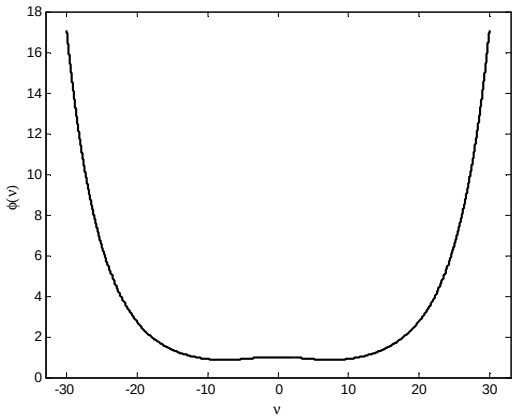


Figura 60: Razão entre derivadas - amplitude de -30 a 30

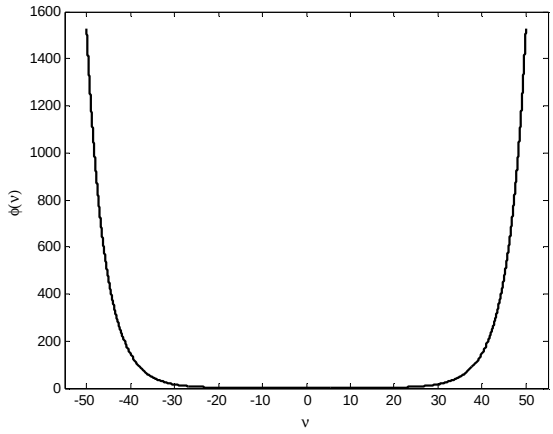


Figura 61: Razão entre derivadas - amplitude de -50 a 50

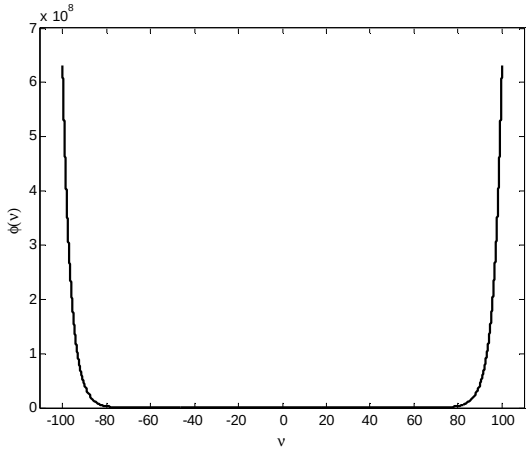


Figura 62: Razão entre derivadas - amplitude de -100 a 100

Capítulo 7 Conclusões

Este capítulo apresenta as conclusões obtidas nesta tese, destacando a confirmação das hipóteses apresentadas inicialmente e a apresentação de contribuições relevantes relacionadas com o tema da pesquisa, além de apontar perspectivas futuras para a expansão de pesquisas sobre este tema.

Os experimentos computacionais desenvolvidos demonstraram a viabilidade da da utilização da Função Bi-Hiperbólica como função de ativação dos neurônios componentes de uma Rede Neural Artificial do tipo Perceptron de Múltiplas Camadas. As hipóteses levantadas teoricamente com relação ao fato desta função possibilitar a otmização do algoritmo de treinamento conhecido como Algoritmo Backpropagation também se confirmaram, pois o modelo que se utilizou desta função de ativação conseguiu melhores taxas de convergência e de generalização, além da utilização de uma arquitetura com um menor número de neurônios na camada oculta. A velocidade do processamento também mostrou uma sensível melhora para o modelo que se utilizou desta função de ativação.

Esta maior rapidez no treinamento, menor consumo de recursos e maior precisão na obtenção de resultados é uma melhoria muito importante uma vez que pode permitir a utilização destes modelos em aplicações reais, que atualmente se ressentem desta agilidade.

Outro fator importante, que se pode inferir dos resultados obtidos neste trabalho, é que a atividade de configuração da estrutura da rede, que normalmente é feita através de processos heurísticos e de tentativas e erros, fica facilitada pelo uso desta função, uma vez que uma ampla combinação de parâmetros e topologias diferentes possibilita a obtenção dos resultados desejados.

Ficou clara a robustez deste modelo que, mesmo em condições desfavoráveis, apresentou resultados melhores em relação à arquitetura da rede e ao tempo de processamento necessário para o seu treinamento. Ficou comprovada a sua necessidade de menos treinamento, de uma arquitetura mais enxuta, e da redução do tempo necessário para o treinamento, além da sua maior versatilidade com a obtenção dos melhores resultados para uma maior variedade de combinações de parâmetros e arquiteturas.

Os experimentos computacionais realizados partiram da definição dos modelos a serem adotados na comparação e da seleção das bases de dados a serem utilizadas nos experimentos computacionais. Foram executados os testes descritos anteriormente e os resultados obtidos demonstraram a superioridade esperada da nova função de ativação proposta.

A utilização do método Holdout permitiu avaliar que o modelo com a função de ativação Bi-hiperbólica tem o comportamento mais robusto em relação às demais funções atualmente em uso. Ela necessita de menos recursos computacionais e informacionais, uma vez que pode ser avaliada que a sua necessidade de treinamento requer menos instâncias e pode ser conduzida em um tempo bem mais reduzido.

A utilização do método de Validação Cruzada permitiu avaliar o uso do modelo para diferentes partições das bases de dados utilizadas, confirmando a superioridade do modelo que usou como função de ativação a função Bi-hiperbólica, tanto em relação ao tempo de processamento, ao menor número de neurônios na camada oculta, quanto à necessidade de treinamento, confirmando as hipóteses previstas. Estas características podem ser atribuídas à maior taxa de variação de sua derivada em relação às das outras funções de ativação atualmente em uso.

A Função Bi-Hiperbólica Simétrica tem imagem no intervalo [0, 1] além de oferecer a propriedade de simetria. Ela é descrita pela Equação 2.22, enquanto que a sua derivada é dada pela Equação 2.23, apresentadas no capítulo 2 deste trabalho.

Por analogia com outras funções sigmoidais, o parâmetro λ pode ser associado à inclinação da função na origem, enquanto que o seu parâmetro τ está associado com o afastamento da curva às duas assíntotas horizontais, podendo variar a faixa de valores de saída da função, mantendo-a, entretanto, entre os limites de zero e um.

A existência de dois parâmetros, como ressalta Xavier (2005), um a mais que as demais funções de ativação, fornece à essa função uma maior flexibilidade possibilitando a representação mais adequada dos fenômenos modelados com redes neurais.

Como foi mostrado nos experimentos computacionais, essa maior flexibilidade conferiu à função de ativação Bi-Hiperbólica Simétrica o poder de melhor convergência e generalização, além de ter possibilitado a criação de redes usando um menor número de neurônios e com um período de treinamento reduzido e mais rápido. A sua rapidez é função, principalmente, dos algoritmos mais simples usados na sua resolução pelos sistemas computacionais.

A determinação destes parâmetros nos experimentos computacionais foi feita empiricamente através de experimentos, uma vez que se objetivava principalmente a avaliação da possibilidade do seu uso como função de ativação que fornecesse uma rede mais enxuta e uma fase de treinamento da rede mais rápida, como eram as hipóteses teóricas.

Após as comprovações destas características, surge uma nova perspectiva para futuras pesquisas, uma vez que, estando estas características melhor definidas,

esta parametrização poderá permitir uma melhor aproximação através de processos de automatização a serem elaborados, onde a distribuição dos valores iniciais de entrada para a rede neural sejam tratados de forma a que, por exemplo, obtenham-se automaticamente os valores para estes parâmetros de configuração que melhor se ajustem aos objetivos do modelo.

Isto poderá diminuir a probabilidade da geração do fenômeno de saturação e, também, uma melhor discriminação dos valores obtidos pelo treinamento, melhorando o processo de generalização da rede. Estas sugestões indicam um novo caminho para futuras pesquisas e trabalhos que venham a complementar e aperfeiçoar o modelo aqui gerado.

Referências

ABRAHART, Robert J. “Single-Model-Bootstrap Applied to Neural Network Rainfall-Runoff Forecasting” In: *Proceedings of the 6th International Conference on GeoComputation*, Universidade de Queensland, Austrália, Set. 2001. Disponível em: www.geocomputation.org/2001/papers/abrahart2.pdf Acessado em: 05/10/2010.

ALEKSANDER, I., MORTON, H., *Introduction to Neural Computing*, Londres, International Thomsom Computer Press, 1995.

BARRETO, J.M., *Introdução às Redes Neurais Artificiais*, Florianópolis, 2002. Disponível em: www.inf.ufsc.br/~barreto/tutoriais/Survey.pdf. Acesso em 01/04/2006.

BATTITI, Roberto. “Accelerated Backpropagation Learning: Two Optimization Methods”, In: *Complex Systems* 3, pp. 331- 342, Champaign (Ilinois), EUA, 1989.

BENVENUTO, N., PIAZZA, F. “On the Complex Backpropagation Algorithm”, In: *IEEE Transactions on Signal Processing*, Vol. 40, No. 4, Abril, 1992. Disponível em: ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=127967&url=http%3A%2F%2Fieeexplore.ieee.org%2Fstamp%2Fstamp.jsp%3Ftp%3D%26arnumber%3D127967 Acesssado em: 05/10/2010.

BRAGA, A. P., CARVALHO, A. P. L. F., LUDERMIR, T. B., *Redes Neurais Artificiais: Teoria e Aplicações*, Rio de Janeiro, Livros Técnicos e Científicos Editora S.A., 2000.

CHANDRA, P., SINGH, Y., “An activation function adapting training algorithm for sigmoidal feedforward networks”, In: *Neurocomputing* 61, pp. 429 – 437. Disponível:dspace.ipu.edu:8080/xmlui/bitstream/handle/1/106/PC_YS%20NC%2061.pdf?sequence=1 Acessado em: 01/08/2010.

CYBENKO, G., “Approximation by Superpositions of a Sigmoidal Function”, In: *Math. Control Signals Systems* 2: pp. 303-314. Springer-Verlag New York Inc. 1989. Disponível em: www.ise.ncsu.edu/fangroup/ie789.dir/Cybenko.pdf Acessado em 10/09/2010.

DUCH, W., JANKOWSKI, N., “Survey of Neural Transfer Functions”, *Neural Computing Surveys* 2, pp. 163-212, 1999. Disponível em: ftp.icsi.berkeley.edu/ftp/pub/ai/jagota/vol2_6.pdf Acessado em: 20/08/2010.

DUDA, R., HART,P.E.,STORK, D.G., *Pattern Classification*. Hoboken, NJ, John Wiley & Sons, 2001.

ELLIOTT, D. L., *A Better Activation Function for Artificial Neural Networks*, Institute for Systems Research, University of Maryland, ISR Technical Report TR 93-8, 1993. Disponível: drum.lib.umd.edu/bitstream/1903/5355/1/TR_93-8.pdf Acessado: 10/09/2010.

FERREIRA, A. A., *Comparação de Arquiteturas de Redes Neurais para Sistemas de Reconhecimento de Padrões em Narizes Artificiais*, Dissertação de Mestrado, Universidade Federal de Pernambuco, Recife, 2004.

FERREIRA, P. , "Neural networks and approximation by superposition of Gaussians," *Icassp*, vol. 4, pp.3197, 1997 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'97) - Volume 4, 1997 Disponível em: ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=595472&url=http%3A%2F%2Fieeexplore.ieee.org%2Fstamp%2Fstamp.jsp%3Ftp%3D%26arnumber%3D595472 Acessado em: 07/09/2010.

FRANK, A. & ASUNCION, A. *UCI Machine Learning Repository Irvine*, CA: University of California, School of Information and Computer Science. Diponível em archive.ics.uci.edu/ml, 2010. Acessado em: 01/08/2010

FRANKE ,J., NEUMANN, MICHAEL H., **"**Bootstrapping Neural Networks", *Neural Computation,* Agosto 2000, Vol. 12, No. 8, pp. 1929-1949, Massachusetts Institute of Technology. DIsponível em: dl.acm.org/citation.cfm?id=1121340 Acessado em: 12/11/2010.

FREEMAN, James A.; SKAPURA, David M. *Neural networks : algorithms, applications, and programming techniques*. Addison-Wesley Publishing Company, Nova Iorque, 1991.

FYFE, C. *Artificial Neural Networks*, Departament of Computing and Information Systems. The University of Paisley, 2000. Disponível em: http://ebookee.org/Artificial-Neural-Networks_400952.html Acessado em: 17/11/2010.

GALLANT, STEPHAN I., "Connectionist expert systems". *Communications of the ACM*, Volume 31 Issue 2, Fevereiro 1988. Disponível em: http://dl.acm.org/citation.cfm?id=42377 Acessado em: 09/10/2010.

GEMAN, S., BIENENSTOCK, E., DOURSAT, R., "Neural networks and the bias/variance dilemma", *Neural Computation* 4:1-58. 1992. Disponível em: http://dl.acm.org/citation.cfm?id=148062 Acessado em: 05/11/2010.

GOH, A., ABIDI, S.S.R., "Re-Visiting Backpropagation Network Optimization: Towards Maximally Pruned Networks" In *International Conference on Artificial Intelligence (IC-AI'99),* Junho 28-Julho 1 1999, Las Vegas. Disponível em: http://web.cs.dal.ca/~sraza/papers/ICAI99-Goh.pdf Acessado em: 5/12/2010.

GREEN, M., OHLSSON, M. "Comparison Of Standard Resampling Methods For Performance Estimation Of Artificial Neural Network Ensembles". In *Proceedings of The third international conference on Computational Intelligence in Medicine and Healthcare*, Plymouth, England, eds. E. Ifeachor (2007) Disponível em: http://cbbp.thep.lu.se/pub/Preprints/07/lu_tp_07_16.pdf Acessado em: 04/09/2010.

HAGAN, Martin T.; DEMUTH, Howard B.; BEALE, Mark. *Neural Network Design*. PWS Publishing Co. Boston, 1996.

HAHN, Brian D., *Fortran 90 for Scientists and Engineers*. Londres, CUP, 1993

HANSELMAN, D.; LITTLEFIELD, B., *MATLAB 6 Curso Completo*, São Paulo, Prentice Hall, 2003.

HAYKIN, S., *Redes Neurais: Princípios e Prática*. 2a. ed. Porto Alegre, Bookman, 2001.

HECHT-NIELSEN, R., “Theory of the Backpropagation Neural Network”; In: *IJCNN International Joint Conference Neural Networks*, pp 593 – 605. Washington, USA, 1989. Disponível em: s112088960.onlinehome.us/annProjects/ Research%20Paper%20Library/backPropTheory.pdf Acessado em: 2/10/2010.

HORNIK, K. “Multilayer Feedforward Networks are Universal Approximators”, *Neural Networks*, Vol. 2, pp. 359-366, 1989. Disponível em: http://weber.ucsd.edu/~hwhite/pub_files/hwcv-028.pdf Acessado em: 07/10/2010.

KALMAN, B. L., KWASNY, B. L., “Why Tanh: Choosing a Sigmoidal Function”, In International Joint Conference on Neural Networks, 1992. IJCNN, 0-*7803-0559-0 192 Q 1992 IEEE* Disponível em: ieeexplore.ieee.org/xpl/login.jsp?tp =&arnumber=227257&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs _all.jsp%3Farnumber%3D227257. Acessado em: 05/10/2010.

KAMRUZZAMAN, J., AZIZ,S.M. “A Note on Activation Function in Multilayer Feedforward Learning”, In 0-7803-7278-6/02 2002 IEEE. Disponível em: http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=1005526&url=http%3A%2F %2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D1005526 Acessado em: 08/10/2010.

KASABOV, Nikola K., *Foundations of Neural Networks, Fuzzy Systems, and Knowledge Engineering*, The MIT Press, Cambridge, Massachusetts, 1998.

KEARNS, Michael. “A Bound on the Error of Cross Validation Using the Approximation and Estimation Rates, with Consequences for the Training-Test Split”, *Advances in Neural Information Processing Systems*. Vol. 8, pp.183-189, Cambridge, MA:MIT Press. Disponível em: dl.acm.org/citation.cfm?id=255120 Acessado em: 04/08/2010.

KÓVACS, Z. L., *Redes Neurais Artificiais: Fundamentos e Aplicações*. São Paulo, Edição Acadêmica, 1996.

LAWRENCE, S., GILES, C.L., TSOI, A. C., “What Size Neural Network Gives Optimal Generalization? Convergence Properties of Backpropagation”, Technical Report, UMIACS-TR-96-22 and CS-TR-3617, Institute for Advanced Computer Studies, University of Maryland, 1996. Disponível em: clgiles.ist.psu.edu/papers/UMD-CS-TR-3617.what.size.neural.net.to.use.pdf Acessado em: 15/09/2010.

LEBARON, Blake, WEIGEND, A.S., Evaluating Neural Network Predictors by Bootstrapping, Computer Science Department, University of Colorado at Boulder, Disponível em: ftp.cs.colorado.edu/pub/Time-series/ MyPapers/ bootstrap.ps, 1994 Acessado em: 28/12/2011.

LEE, Y., OH, Sang-Hoon, KIM, M. W., “The Effect of Initial Weights on Premature Saturation in Back-Propagation Learning” In *Seattle International Joint Conference on Neural Networks*, 1991., IJCNN-91, @7803-0164-1/91/0000-M65 1991 IEEE Disponível em: ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber= 155275&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Far number%3D155275 Acessado em: 20/12/2011.

MACKAY, D. J. C., *Information Theory, Inference, and Learning Algorithms*, Cambridge: Cambridge University Press, 1998.

MANGASARIAN, O. L., WOLBERG, W. H. "Cancer Diagnosis Via Linear Programming", *SIAM News*, Volume 23, Number 5, September 1990, pp 1 - 18. Disponível em: ftp://ftp.cs.wisc.edu/math-prog/tech-reports/94-10.pdf Acessado em: 10/11/2010.

MAYA, Paulo A., LEONARDI, Fabrizio, Controle Essencial, São Paulo, Pearson Education do Brasil, 2011.

McCULLOCH, W.S., PITTS, H., "A Logical Calculus of the Ideas Immanent in Nervous Activity", *Bulletin of Mathematical Biophysics*, Vol. 5, 1943, pp.115-133. Disponível em: dl.acm.org/citation.cfm?id=104377. Disponível em: 15/10/2010.

MINSKY, M., PAPERT, S. *Perceptrons, an Introduction to Computational Geometry*. MIT Press, Cambridge, Mass., 1969. Disponivel em: dl.acm.org/ citation.cfm?id=1096885 Acessado em: 22/09/2010.

NETO, A.R.R., SOUSA, R., BARRETO, G.A., CARDOSO, "J.S. Diagnostic of Pathology on the Vertebral Column with Embedded Reject Option", In *Proceedings of Iberian Conference on Pattern Recognition and Image Analysis (IbPRIA),* 2011. Disponível em: http://www.inescporto.pt/~jsc/publications /conferences/2011AjalmarIbPRIA.pdf Acessado em: 7/12/2012.

OTAIR, Mohammed A., SALAMEH, Walid A., "Speeding Up Back-Propagation Neural Networks", In *Proceedings of the 2005 Informing Science and IT Education Joint Conference*, Flagstaff, Arizona, USA. Disponível em: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.107.8544. Acessado em: 08/09/2010.

POLI, Gustavo; SAITO, J. Hiroki; MARI, J. F.; ZORZAN, Marcelo R. "Processing Neocognitron of Face Recognition on High Performance Environment Based on GPU with CUDA Architecture", In *20th International Symposium on Computer Architecture and High Performance Computing*, IEEE Computer Society, 2008. Disponível em: www.uweb.ucsb.edu/~yichuwang/ecv/paper/face_recognition _with_cuda.pdf Acessado em: 16/12/2011.

PRECHELT, Lutz. *Proben1 - A Set of Neural Network Benchmark Problems and Benchmarking Rules*. Fakultat fur Informatik, Universitat Karlsruhe, Germany, Technical Report 21/94, Setembro 30, 1994. Disponível em: http://page.mi.fu-berlin.de/prechelt/Biblio/1994-21.pdf. Acessado em: 5/12/2010.

ROSENBLATT, F., "The PERCEPTRON : a Probabilistic Model for Information Storage and Organization in the Brain". *Psychological Review*, Vol. 65, No. 6, 1958. Disponível em: psycnet.apa.org/journals/rev/65/6/386/ Acessado em: 14/9/2010.

SCHIFFMANN, W., JOOST, M., WERNER, R., *Optimization of the Backpropagation Algorithm for Training Multilayer Perceptrons*, University of Kblenz, Institute of Physics, Koblenz, 1994. Disponível em: www.cs.bham.ac.uk/~pxt/ NC/schiffmann.bp.pdf Acessado em: 19/12/2010.

STATHAKIS, D. "How many hidden layers and nodes?", In *International Journal of Remote Sensing,* Vol. 30, No. 8, 20 Abril 2009, pp 2133–2147. Disponível em: dstath.users.uth.gr/papers/IJRS2009_Stathakis.pdf Acessado em: 05/11/2010.

TAYLOR, Brian J., *Methods and Procedures for the Verification and Validation of Artificial Neural Networks*, Institute for Scientific Research, Inc. ,Springer

Science+Business Media, Inc., 2006 Disponível em: pt.scribd.com/doc/47548054/Methods-and-Procedures-for-the-Verification-and-Validation-of-Artificial-Neural-Networks Acessado em: 14/10/2010.

WHITE, H., RACINE, J., *Statistical Inference, the Bootstrap, and Neural Network Modeling with application to foreign exchange rates*, Disponível em: *weber.ucsd.edu/~hwhite/pub_files/hwcv-080.pdf*. Acessado em: 28/12/2011.

WOLBERG, William H. ; MANGASARIAN, O.L. "Multisurface method of pattern separation for medical diagnosis applied to breast cytology", In *Proceedings of the National Academy of Sciences*, U.S.A., Volume 87, Dezembro 1990, pp 9193-9196. Disponível em: www.pnas.org/content/87/23/9193 Acessado: 19/4/2010.

XAVIER, Adilson Elias, "Uma Função de Ativação para Redes Neurais Artificiais Mais Flexível e Poderosa e Mais Rápida". *Learning and Nonlinear Models – Revista da Sociedade Brasileira de Redes Neurais* (SBRN), Vol. 1, No. 5. pp. 276-282, 2005.

ZHANG, Jing-Ru , ZHANG, J., LOK, Tat-Ming, LYU, Michael R. ,"A hybrid particle swarm optimization–back-propagation algorithm for feedforward neural network training", In *Applied Mathematics and Computation* 185 (2007) 1026–1037. Disponível em: 200.17.137.109:8081/xiscanoe/Members/taef/taef/material-pso/A%20hybrid%20particle%20swarm%20optimization-back-propagation%20algorithm%20for%20feedforward%20neural%20network%20training.pdf Acessado em: 13/11/2010.

ZWEIRI, Yahya , "Optimization of a Three-Term Backpropagation Algorithm Used for Neural Network Learning", In *International Journal of Engineering and Mathematical Sciences* 3:4 2007 Disponível em: www.waset.org/journals/ijims/v3/v3-4-38.pdf Acessado em: 29/11/2010.