Raquel Esperanza Patiño Escarcina

COMPUTAÇÃO INTERVALAR EM REDES NEURAIS PERCEPTRON

Natal, Rio Grande do Norte - Brasil
Novembro de 2004

Universidade Federal do Rio Grande do Norte
Centro de Ciências Exatas e da Terra
Departamento de Informática e Matemática Aplicada
Programa de Pós-Graduação em Sistemas e Computação

COMPUTAÇÃO INTERVALAR EM REDES NEURAIS PERCEPTRON

Raquel Esperanza Patiño Escarcina

Orientador
Prof. Dr. Benjamín René Callejas Bedregal
Prof. Dr. Aarão Lyra

Dissertação submetida e aprovada no Programa de Pós-Graduação em Sistemas e Computação do Departamento de Informática e Matemática Aplicada da Universidade Federal do Rio Grande do Norte como parte dos requisitos para a obtenção do grau de Mestre em Sistemas e Computação (MSc.).

Natal, Rio Grande do Norte - Brasil
Novembro de 2004

'Esta Dissertação foi julgada adequada para a obtenção do título de Mestre em Sistemas e Computação (MSc.), Área de Concentração em Inteligência Computacional, e aprovada em sua forma final pelo Programa de Pós-Graduação em Sistemas e Computação do Departamento de Informática e Matemática Aplicada da Universidade Federal do Rio Grande do Norte.'

Banca Examinadora:

Prof. Dr. Aarão Lyra

Prof. Dr.Adrião Duarte Dória Neto

Profa. Dra.Anne Magaly de Paula Canuto

Prof. Dr. Wilson Rosa de Oliveira Junior

PATIÑO ESCARCINA, Raquel Esperanza

Computação Intervalar em Redes Neurais Perceptron / Raquel Esperanza Patiño Escarcina. Natal, Rio Grande do Norte - Brasil: 2004.

100p. : xx.

Orientador:
Prof. Dr. Benjamín René Callejas Bedregal
Prof. Dr. Aarão Lyra

Dissertação (Mestrado) – Universidade Federal do Rio Grande do Norte. Centro de Ciências Exatas e da Terra. Departamento de Informática e Matemática Aplicada.

1. Inteligência Computacional – Dissertação. 2. Matemática Intervalar 3. Redes Neurais Artificiais.

Este trabalho é dedicado a meu pai Jesus, minha mãe Teresa, meus irmãos Maria Teresa, Amparo e Enrique pela confiança que depositaram em mim e por seu imenso carinho. A Dennis por seu apoio incondicional, por seu carinho e compreensão.

AGRADECIMENTOS

No processo para terminar este trabalho, vi como cada pessoa que passou pela minha vida foi importante e contribuiu de alguma forma na realização deste objetivo alcançado. Mas, existem pessoas que nos marcam a vida e entre as demais merecem um realce especial.

Assim, quero agradecer primeiramente a Deus, porque sempre esteve a meu lado;

Aos meus pais *Jesus Patiño Sanchez* e *Teresa Escarcina de Patiño* que são meu apoio e minha força para continuar lutando para alcançar meus objetivos;

Às minhas irmãs *María Teresa Patiño Escarcina*, *Amparo Patiño Escarcina* e ao meu irmão *Jesus Enrique Patiño Escarcina* por sua compreensão e seu carinho que na distância une-nos ainda muito mais;

Ao meu amor e futuro esposo *Dennis Barrios Aranibar* por seu carinho, seu apoio incondicional e por seu incentivo para chegar um passo na frente do esperado;

Ao professor e orientador *Benjamín René Callejas Bedregal* pela oportunidade que me deu de poder fazer o mestrado e por sua confiança. Por sua orientação ao longo deste trabalho e pela amizade;

Ao professor e orientador *Aarão Lyra* pela tão competente orientação, meu agradecimento pelo apoio constante e tempo dedicado e pelo incentivo durante todo o decorrer do trabalho;

De forma especial quero agradecer ao professor Benjamín, a sua esposa Ivanosca, a Lúcia e a Ruan, pela confiança e por seu apoio. Porque me abriram as portas de sua casa mesmo sem me conhecer;

Aos professores do DIMAp e do DCA pelos conhecimentos compartilhados. Em especial ao professor *Regivan Hugo Nunes Santiago* e à professora *Anne Magaly de Paula Canuto*, a minha admiração pela sua dedicação na sala de aula e firmeza mostrada;

Ao pessoal que trabalha no departamento de *Assuntos Estudantis* da UFRN, pelo seu apoio.

Aos amigos da turma, pela amizade e os momentos compartilhados ao longo destes dois anos, em especial a *Manuel, Macilón, Adriana* con quem compartilhei muitas coisas que ficarão na minha lembrança;

Ao pessoal do DIMAp, *a Dona Graça, ao Senhor Valberto, Dona Celma e ao Senhor Gaspar*, por seu apoio;

Finalmente quero agradecer a todas as pessoas que participaram deste trabalho. Neste momento me lembro de tantos amigos que me ajudaram em algum momento mas se faz impossível citar todos eles, então só posso dizer que fico grata a todos.

COMPUTAÇÃO INTERVALAR EM REDES NEURAIS PERCEPTRON

Raquel Esperanza Patiño Escarcina

Orientador:
Prof. Dr. Benjamín René Callejas Bedregal
Prof. Dr. Aarão Lyra

RESUMO

As redes neurais artificiais computacionais são utilizadas atualmente com o objetivo de resolver problemas em diferentes áreas das ciências. Elas representam uma forma simples de se encontrar soluções de problemas que possuem alta complexidade.

Para que as redes possam ter um funcionamento satisfatório, elas têm que passar por uma série de etapas, dentre elas, a escolha dos dados a serem fornecidos como exemplos, sua melhor representação e a escolha da melhor configuração da rede.

Usualmente, a obtenção dos dados conserva um erro numérico devido à limitação dos equipamentos, como por exemplo, erro na leitura ou erro na captura. Além disso podem existir erros devido à própria computação numérica da rede neural, ao efetuar arredondamento ou truncamento. Assim, é necessário ter o controle destes erros para que os resultados obtidos sejam satisfatórios.

Dentre os métodos de resolução e estimativa do erro, o método clássico é o que obtém um resultado de melhor qualidade, porém extenso, dispendioso e às vezes, inviável. Algoritmos convencionais conciliados com o método clássico de estimativa de erro fornecem resultados errados, sem garantia alguma de que o resultado obtido equivale ao esperado. Assim, Santos, Aguiar e Dimuro em [SB04], observaram através de

estudos, que as técnicas intervalares propostas em [Moo66] são alternativas boas para resolver esses tipos de problemas.

Dados intervalares estão presentes em diversos modelos de problemas a se resolver através de redes neurais sendo necessário a construção de algoritmos computacionais para utilizá-las.

Neste trabalho são apresentadas as *redes neurais artificiais com aprendizado supervisionado* sob a abordagem da *Matemática Intervalar* aplicada a um estudo de caso das redes neurais artificiais Perceptron. Estas redes neurais intervalares propostas são baseadas nas redes neurais pontuais, mas buscam solucionar o problema dos erros de precisão nos cálculos e o tratamento de dados intervalares presentes em diversos problemas a solucionar.

Assim são propostas as arquiteturas destas redes intervalares junto com os algoritmos para seu treinamento. É feita uma análise da estatística das redes comparando com as redes neurais tradicionais. A diferença de outros trabalhos feitos dentro desta área de redes neurais intervalares, nosso trabalho aqui apresentado tenta não só propor as redes e reconstruir os algoritmos tradicionais mas também fazer uma análise do comportamento destas redes neurais intervalares.

O software utilizado para os testes foi o Matlab 6.0 junto com o toolbox para intervalos b4m [Zem02] .

INTERVAL COMPUTING IN PERCEPTRON NEURAL NETWORK

Raquel Esperanza Patiño Escarcina

Advisor:
Prof. Dr. Benjamín René Callejas Bedregal
Prof. Dr. Aarão Lyra

ABSTRACT

Artificial Neural Networks are used to solve problems in deferent science fields. They represent a simple way to find solutions for highly complex problems.

In general, Neural Networks work with given data that include information of the problem (examples with inputs and desired outputs from the model) from the knowledge is extracted, and after it, for new data they give solutions with a low error rate.

In order to Neural Networks work satisfactorily, they have to pass for several steps. One of this steps is the selection of the examples and their best representation to be given to the Neural Network, and the selection of the best configuration for the Neural Network.

Obtaining data usually implies numerical errors because of the limitation of the equipment, for example, reading error or capture error. Beyond it, numeric computing of the Neural Network can give errors of rounding or truncation. Thus, it is necessary to have control of these errors in order to obtain satisfactory results.

Among resolution methods, the classic method of error estimate have a high quality result, however, it is extensive, expensive and, some times, impracticable. Conventional

algorithms conciliated with the classic method give erroneous results, without guarantee of the obtained result is equivalent to the expected result. [SAD94]

Through studies, we see that intervalar technics proposed in [Moo66] are a good alternative for solving this kind of problems. Intervalar data is present in almost all models to being solved with Neural Networks. It is necessary make algorithms for using intervalar data.

In this work are presented the *Intervalar Neural Networks* with supervised training. This Neural Networks are based on Punctual Neural Networks, but try to be a solution for the problems of calculus precision error and the treatment of intervalar data present in several problems to solve. Beyond it, it is believed that intervalar connections between neurons permit Neural Network convergence time to be lower than punctual networks without loss efficiency.

It was made a statistic analysis of this approach in comparison with the traditional punctual neural networks. Deferent of other works in this fill of intervalar neural networks this work makes an analysis of the behavior of the networks proposed.

It was used Matlab 6.0 software and b4m toolbox [Zem02] for programming a testing this approach.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Imaginação é mais importante que conhecimento!

Albert Einstein

1.1 Preâmbulo

Em muitas situações da vida real, é necessário mensurar o valor de uma quantidade física y que é difícil (ou impossível) de medir diretamente. Por exemplo, é impossível a medição direta da distância da terra a uma estrela. Ainda que não possamos medir tal quantidade diretamente, mediremos indiretamente utilizando algumas outras quantidades x_i.

Sendo os dados de entrada imprecisos para os algoritmos de processamento, o resultado y do processamento é também impreciso. Em outras palavras, são necessários métodos de processamento de dados que produzam resultados aproximados, talvez não exatos, mas resultados com verificação automática [KLRK98]. Pode-se dizer que a Computação Intervalar começou com Arquimedes, que estimou dois valores (um intervalo) com a garantia de que entre eles estava o número π. Em 1914 Norbert Wiener fez uma análise da exatidão nas medidas, mas o principal aporte foi em 1962 com Ramon E. Moore [Moo62], de Standford que trabalhava para Lockheed Missiles and Space Co., que publicou em 1959 um *technical report* no qual desenvolveu uma nova técnica chamada de computação intervalar e aplicou essa técnica no problema de computar a trajetória de um propulsor da Terra à Lua.

Em diferentes áreas da ciência e tecnologia, a precisão dos dados usados é muito importante e faz-se necessário o desenvolvimento de algoritmos que garantam uma computação precisa. O trabalho de Moore foi uma alternativa de solução computacional

para os problemas que não têm uma entrada de dados exatos ou uma representação finita. Estes dados, chamados de dados intervalares, carregam um erro máximo de um dado numérico. Um resultado intervalar possui o erro propagado pelas operações aritméticas intervalares. A aritmética intervalar de Moore, possibilitou um enorme desenvolvimento de pesquisas nesta área.

Por outro lado, as redes neurais artificiais é um conceito da computação inspiradas no cérebro humano, que visa trabalhar no processamento de dados O cérebro é tido como um processador altamente complexo e que realiza processamentos de maneira paralela. Para isso, ele organiza sua estrutura, ou seja, os neurônios, de forma que eles realizem o processamento necessário.

Nas redes neurais artificiais, a idéia é realizar o processamento de informações tendo como princípio a organização de neurônios do cérebro. Como o cérebro humano é capaz de aprender e tomar decisões baseadas na aprendizagem, as redes neurais artificiais devem fazer o mesmo. Assim, uma rede neural pode ser interpretada como um esquema de processamento capaz de armazenar conhecimento baseado em aprendizagem (experiência) e disponibilizar este conhecimento para a aplicação em novos casos apresentados.

Hoje, um dos principais desafios enfrentados pelas redes neurais, consiste na escolha dos algoritmos de treinamento e memorização, ou seja, como alterar os pesos sinápticos de forma a produzir os sinais desejados na saída.

Além disso, está longe a data onde será possível vislumbrar redes neurais com ligações tão complexas e perfeitas como as do cérebro, em especial no que diz respeito às ligações maciças existentes neste último.

Entretanto, à medida que os modelos de representação avançam, conseguir-se-á construir soluções para problemas atuais que passo a passo poderão conduzir à meta desejada.

1.2 Motivação

Em muitas aplicações, o conjunto de dados usado pode não ser exato o qual é um problema serio, pois os resultados obtidos também serão não exatos. A representação de conjuntos de dados através de intervalos tenta solucionar estes problemas, porém, as ferramentas de análises têm que ser reconstruídos de forma natural para efetuar o processamento com intervalos.

As abordagens existentes até agora das redes neurais com dados intervalares foram propostos de diferentes formas. Alguns deles são: métodos que utilizam os valores

extremos, onde são computados os valores máximo e mínimo dos intervalos; outro é o método probabilístico, onde o intervalo é considerado um conjunto de dados, no qual cada número no intervalo tem uma probabilidade igual de ser escolhido. Estas abordagens não são soluções ideais para o problema levando a uma computação com erros e perdas de informação. Todos os trabalhos usados como referências no tema de redes neurais intervalares têm uma contribuição importante, mas, é necessário definir como seria uma rede com o processamento realmente intervalar e suas características como o aprendizado, o algoritmo de treinamento, etc.

Com base nas idéias anteriores, vimos que é preciso estudar as redes neurais preparadas para receber dados de natureza intervalar conservando suas propriedades básicas e acolhendo as características do processamento intervalar.

1.3 Objetivos

Para os problemas que necessitam utilizar dados intervalares é necessário propor uma reconstrução das redes neurais para solucioná-los com facilidade. Então, o objetivo deste trabalho é utilizar a matemática intervalar que permite o controle nos erros da computação numérica, para fornecer às redes neurais de aprendizado supervisionado (no caso específico as redes neurais Perceptron) a capacidade de processar dados intervalares.

1.4 Organização do Trabalho

O trabalho apresentado, requer o estudo das redes neurais, em geral conceitos básicos e aprofundados no estudo de redes neurais perceptron com aprendizado supervisionado, além de fazer uma revisão dos conceitos da matemática intervalar.

Este documento será organizado em 7 capítulos e um apêndice. A descrição do que contém cada capítulo e feita a seguir:
Capítulo 2: Este capítulo é uma introdução às redes neurais artificiais com destaque nas redes neurais Perceptron que foram implementadas para comparar seu comportamento com as redes neurais Perceptron Intervalares.
Capítulo 3: Neste capítulo é feita uma introdução da matemática intervalar. São descritos conceitos necessários da matemática intervalar para o desenvolvimento desta pesquisa.
Capítulo 4: Este capítulo da início às redes neurais intervalares, fazendo primeiro um estudo dos trabalhos até agora realizados na área, definindo em linhas gerais o

que cada trabalho propõe assim como uma análise das contribuições em comparação com o nosso, logo, são descritos conceitos e definições importantes das redes neurais intervalares. Dado que nosso trabalho é especificamente em redes neurais Perceptron Intervalar, elas merecem um destaque especial que são apresentadas no capítulo 5.

Capítulo 5: Este capítulo descreve o desenvolvimento da primeira parte de nossa dissertação: As redes neurais Perceptron Intervalar de uma camada. São apresentadas as estruturas, características e algoritmos que serão usados para o treinamento da mesma. Também são apresentados os testes feitos com este tipo de redes.

Capitulo 6: Este capitulo descreve a continuação de nossa pesquisa: As redes neurais perceptron intervalar de múltiplas camadas. Assim é apresentado o algoritmo de treinamento e um analise do comportamento desta rede junto com os testes feitos em dois conjunto de dados: O Iris e o problema do Xor.

Capítulo 7: Este capítulo contém as conclusões finais de nosso trabalho e alguns trabalhos futuros a ser realizados. Apresentaremos os diversos trabalhos que foram aceitos em revistas internacionais e congressos nacionais e regionais em decorrência do trabalho desta dissertação assim como os trabalhos que estão sendo preparados para futuras publicações.

Apêndice A: Neste apêndice é apresentada a prova de um teorema descrito no capítulo 5 que trata da separação das classes que o Perceptron Intervalar executa.

Para finalizar, são apresentadas todas as referências bibliográficas usadas em nossa pesquisa.

Capítulo 2

Redes Neurais Artificiais

2.1 Introdução

As Redes Neurais Artificiais (RNAs) inspiradas nas redes neurais biológicas são sistemas computacionais maciçamente paralelos, consistindo de um grande número de processadores simples com muitas interconexões. As redes neurais possibilitam um novo paradigma para a solução de problemas devido às suas características de aprendizado, adaptabilidade, robustez e tolerância a falha.[Hay94]

Os neurônios biológicos são bem mais lentos que os computadores digitais, mas a inferência humana é bem mais rápida. O cérebro compensa a operação relativamente lenta com o enorme número de neurônios, extremamente inter-conectados (cada neurônio tem entre 10^3 a 10^4 conexões), funcionando de maneira paralela e descentralizada.[Nas99]

Enquanto nos computadores o conhecimento é estritamente substituível, no cérebro o conhecimento é adaptável. O cérebro humano tem muitas características sofisticadas, entre elas: nova informação é adicionada pelo ajuste da força de conexão entre os neurônios biológicos, exibe características de tolerância a falhas (danos individuais em neurônios podem ocorrer sem maior degradação do desempenho geral), generalização para casos desconhecidos a partir de tarefas e exemplos conhecidos, etc.

Foi assim que as redes neurais artificiais foram modeladas, tentando imitar o comportamento do cérebro. Atualmente, as redes neurais artificiais são capazes de realizar tarefas que envolvem processamento simultâneo de grande quantidade de dados e onde respostas rápidas e acuradas são necessárias. Algumas funções das redes neurais artificiais são descritas a seguir:

1. Classificação: Capacidade de analisar dois objetos ou estados e de acordo com suas características, estimar suas similaridades e diferenças.
2. Segmentação: Capacidade de agrupar conceitos semelhantes.
3. Memória Associativa: Capacidade de associar objetos ou idéias com memórias relacionadas.
4. Modelagem: Capacidade de modelar relacionamentos entre exemplos, para possibilitar generalizações diante de casos novos.
5. Predição de Séries Temporais: Capacidade de prever acontecimentos em algum ponto do futuro, dado o acontecido no passado.
6. Satisfação de Restrições: Capacidade de solucionar problemas com restrições conflitantes.

Cada unidade de processamento atua como uma função reconhecedora simples de padrões. Ela recebe entradas de outros elementos, compara estas com sua memória e produz precisamente um sinal. As entradas passam por conexões com pesos (sua memória), que aumentam ou diminuem os sinais de saída. [dS00] Dentro de cada processador, as entradas são totalizadas, esse valor total de entrada é submetido a uma função para produzir a saída deste processador, geralmente variando de 0 a 1. A maneira como as unidades de processamento estão arranjadas numa rede neural está diretamente relacionada com a forma de aprendizado desta.

Este capítulo têm a seguinte organização:

- Na seção 2.2 é descrito o neurônio de McCulloch-Pitts.
- Na secão 2.3, apresenta-se as arquiteturas nas redes neurais artificiais.
- Na secão 2.4, apresenta-se os paradigmas de aprendizagem das redes neurais artificiais.
- Na seção 2.5, apresenta-se os modelos de redes neurais artificiais.
- Na seção 2.6 é descrito o processo de treinamento nas redes neurais artificiais em geral.
- Na seção 2.7 será apresentada a rede Perceptron Monocamada e suas características.
- Na seção 2.8 é apresentada a rede Perceptron com Múltiplas Camadas junto com suas principais características e seu algoritmo de aprendizagem.

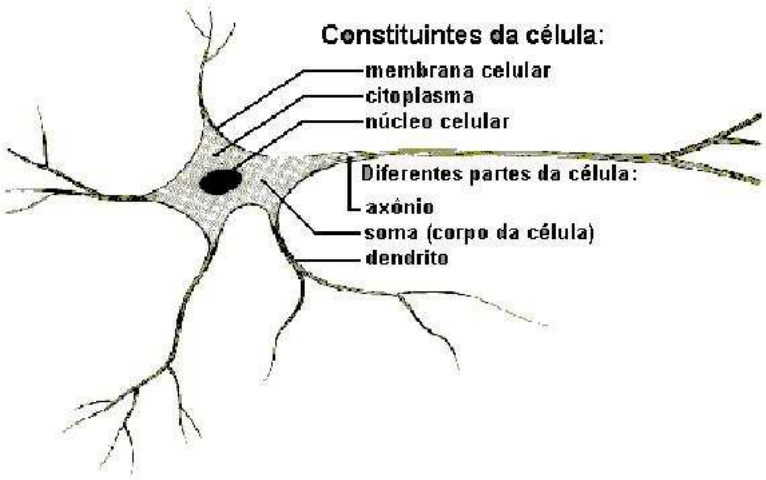

Figura 2.1: Modelo do Neurônio Biológico

Neste capítulo será aprofundado o estudo em duas redes neurais de aprendizagem supervisionada, Perceptron com uma camada e o Perceptron com múltiplas camadas porque nosso trabalho será focado nestas redes neurais.

2.2 Neurônios Artificiais: Modelo MCP

O modelo de neurônio proposto por McCulloch e Pitts [MP43] conhecido também por modelo MCP, é uma simplificação do que se conhecia a respeito do neurônio biológico naquela época. A sua descrição matemática resultou em um modelo com n terminais de entrada $x_1, x_2, ..., x_n$ (que representam os dentritos), e apenas um terminal de saída y (representando o axônio). Emula-se com os terminais de entrada e saída o comportamento das sinapses, assim, os terminais de entrada do neurônio têm pesos (resistores) acoplados $w_1, w_2, ..., w_n$, cujos valores podem ser positivos ou negativos. Dependendo das sinapses, o neurônio dispara quando a soma dos impulsos que ele recebe ultrapassa o seu limiar de excitação (threshold). A figura 2.1 apresenta o neurônio biológico e a figura 2.2 apresenta o modelo do neurônio de McCulloch-Pitts.

O neurônio artificial é a unidade básica de processamento de informação de uma rede neural artificial. A estrutura deste neurônio está formada por:

1. **Conjunto de Sinapses (elos de conexão):** Em uma rede neural cada um dos neurônios se caracteriza por seu peso ou força própria. Um sinal x_j na entrada j conectada a um neurônio k é multiplicado pelo peso sináptico w_{kj}, onde o índice k refere-se ao neurônio em questão e o índice j refere-se ao terminal de entrada.

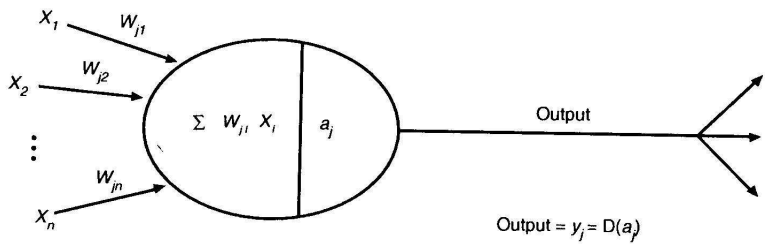

Figura 2.2: Modelo do Neurônio Artificial de McCulloch-Pitt

2. **Função de Junção:** serve para compor de forma linear os sinais de entrada já ponderados pelos seus respectivos pesos sinápticos. A função de junção também é conhecida como "combinador" linear ou simplesmente "somador". A saída desta função é chamada de "soma neta".

3. **Função de Ativação:** Seu uso faz-se necessário para restringir a amplitude de saída de um neurônio. Por convenção, as saídas dos neurônios devem estar restritas ao intervalo fechado $[0, 1]$ ou $[-1, 1]$, há casos ainda de saída binária zero ou um (função sinal). Assim, a função de ativação ou função restritiva tem o objetivo de adequar a saída do neurônio a um destes intervalos. Um fator que pode interferir na função de ativação é o "*bias*". Este fator tem o efeito de aumentar ou diminuir a entrada da função de ativação, dependendo se ele é positivo ou negativo, respectivamente. Desta forma, o *bias* interfere diretamente na sensibilidade do neurônio em análise às entradas.

   A partir do modelo proposto por McCulloch e Pitts foram derivados vários outros modelos que permitem a produção de uma saída qualquer, não necessariamente zero ou um, e com diferentes tipos de funções de ativação. A figura 2.3 mostra graficamente quatro tipos de funções de ativação diferentes: as funções lineares, as funções rampa, as funções degrau (*step*) e as funções sigmóide.

   Uma função de ativação é linear se pode ser definida pela equação (2.1), para algum número real α (y é a saída e x a entrada). Esta função pode ser restringida para produzir valores constantes em uma faixa $[-\gamma, +\gamma]$, e neste caso a função linear passa a ser uma função rampa como mostrada graficamente na figura 2.3.b e definida pela equação (2.2)

$$y = \alpha x \tag{2.1}$$

$$y = \begin{cases} +\gamma & \text{Se } x \geq +\gamma, \\ x & \text{Se } |x| < +\gamma, \\ -\gamma & \text{Se } x \leq -\gamma. \end{cases} \tag{2.2}$$

O valor máximo e mínimo da saída são $+\gamma$ e $-\gamma$, respectivamente. A função rampa é geralmente usada como uma função não linear simplificada.

A função degrau, ilustrada na figura 2.3.a é similar a uma função sinal, no sentido que a função produz a saída $+\gamma$ para os valores de x maiores que zero, caso contrário a função produz o valor $-\gamma$. A função degrau é definida pela equação 2.3

$$y = \begin{cases} +\gamma & \text{Se } x > 0, \\ -\gamma & \text{Se } x \leq 0. \end{cases} \tag{2.3}$$

A função sigmóide, conhecida também como *S-shape*, ilustrada na figura 2.3.c é uma função semi-linear, limitada e monotônica. É possível definir várias funções sigmoidais. As funções sigmoidais são encontradas na modelagem de diversos modelos nas mais variadas áreas. Uma das funções sigmoidais mais importantes é a função logística definida pela equação 2.4 onde T determina a suavidade da curva.

$$y = \frac{1}{1 + e^{-x/T}} \tag{2.4}$$

Em termos matemáticos, um neurônio k é descrito segundo as equações (2.5) e (2.6) onde $x_1, x_2, ..., x_m$ são sinais de entrada; $w_{k1}, w_{k2}, ..., w_{km}$ são pesos sinápticos do neurônio k; u_k é a saída da função de junção; b_k é o bias; $\phi(.)$ é a função de ativação e por fim y_k é o sinal de saída do neurônio. A equação (2.7) define o termo potencial de ativação v_k, que trata da entrada da função de ativação.

$$u_k = \sum_{j=1}^{m} w_{kj} x_j \tag{2.5}$$

$$y_k = \phi(u_k + b_k) = \phi(v_k) \tag{2.6}$$

$$v_k = u_k + b_k \tag{2.7}$$

2.3 Principais Arquiteturas de RNAs

A definição da arquitetura de uma RNA é um parâmetro importante na sua concepção uma vez que ela restringe o tipo de problema que pode ser tratado pela rede. As redes com camada única de neurônios chamadas de MCP, por exemplo, só conseguem resolver problemas linearmente separáveis. Redes recorrentes, por sua vez, são mais apropriadas para resolver problemas que envolvem processamento temporal. Fazem parte da definição da arquitetura os seguintes parâmetros: número de camadas da

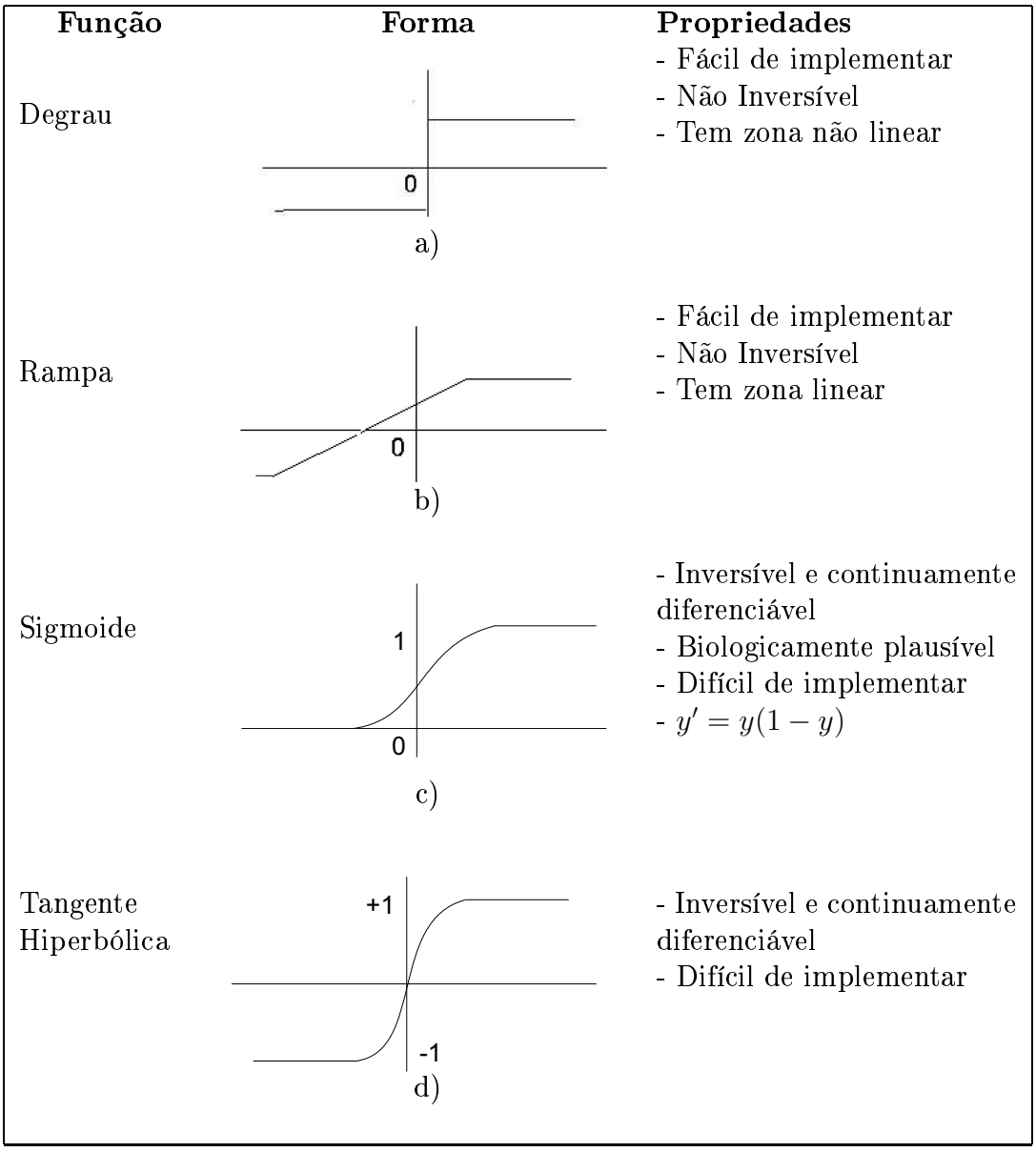

Figura 2.3: Formas e Propriedades Importantes das Funções de Ativação mais Usadas

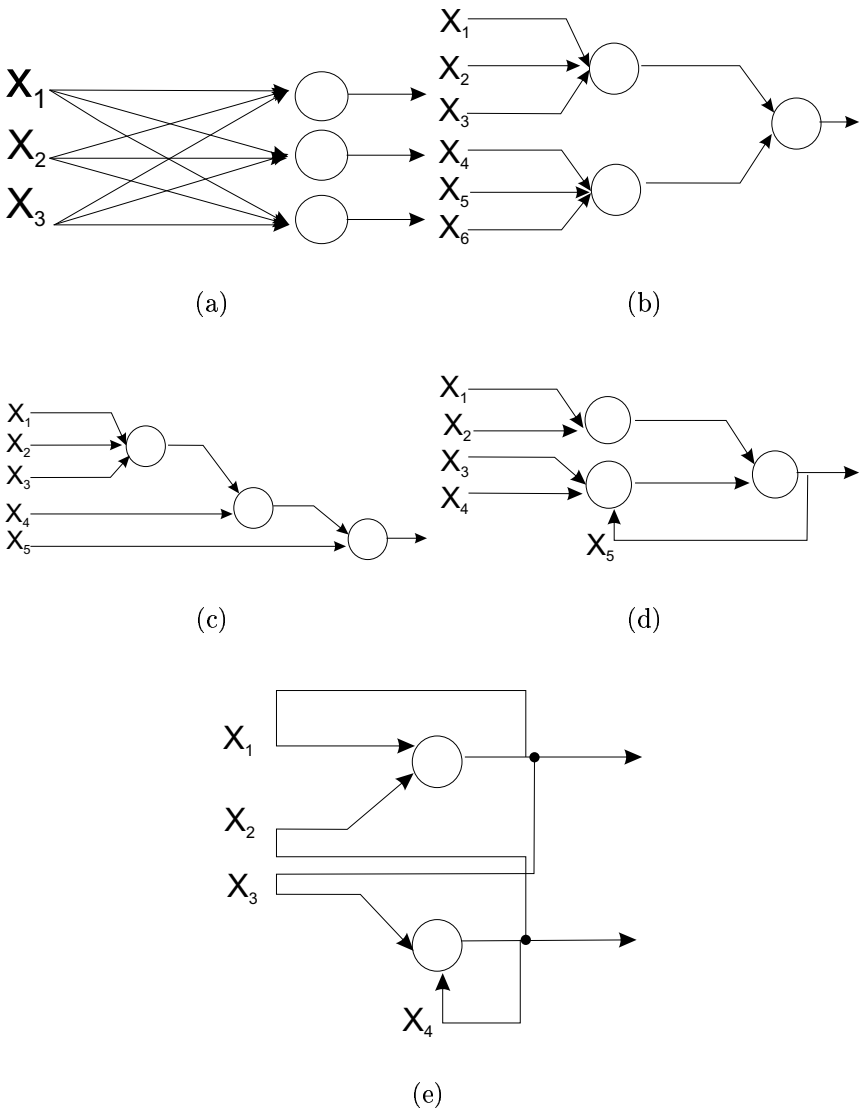

Figura 2.4: Principais Arquiteturas de RNA

rede, número de neurônios em cada camada, tipo de conexão entre os neurônios e topologia da rede. Alguns exemplos de arquiteturas de RNAs são apresentados na figura 2.4.

Quanto ao número de camadas, pode-se ter:

1. Redes de camada única, só existe um nó entre qualquer entrada e qualquer saída da rede.
2. Redes de múltiplas camadas, existe mais de um neurônio entre alguma entrada e alguma saída da rede.

Os neurônios podem ter conexões do tipo:

1. *Feedforward*: sem realimentação ou acíclica, a saída de um neurônio na i-ésima camada da rede não pode ser usada como entrada de neurônios em camadas de índice menor ou igual a i.

2. *Feedback*: com realimentação ou cíclica, a saída de algum neurônio na i-ésima camada da rede é usada como entrada de neurônios em camadas de índice menor ou igual a i.

    - Rede neural cuja saída final (única) é ligada às entradas comportam-se como autômatos reconhecedores de cadeias, onde a saída que é realimentada fornece o estado do autômato.
    - Se todas as ligações são cíclicas, a rede é denominada de auto-associativa. Estas redes associam um padrão de entrada com ela mesma, e são particularmente úteis para recuperação ou "regeneração" de um padrão de entrada.

Finalmente, as RNAs podem também ser classificadas quanto sua conectividade:

1. Rede fracamente (ou parcialmente) conectada.

2. Rede completamente conectada.

2.4 Paradigmas de Aprendizagem das Redes Neurais

A propriedade primordial das redes neurais é aprender conforme o ambiente onde estão inseridas e assim melhorar seu desempenho [Gon03].

No aprendizado conexionista não se procura obter regras como na abordagem simbólica de Inteligência Artificial (IA), mas determinar a intensidade de conexões entre neurônios. A utilização de uma RNA na solução de uma tarefa passa inicialmente por uma fase de aprendizagem, onde a rede extrai informações relevantes de padrões de informação apresentadas para a mesma, criando assim uma representação própria para o problema.

Na etapa de aprendizagem, o treinamento da rede neural consiste em um processo iterativo de ajuste de parâmetros da rede, os pesos das conexões entre as unidades de processamento, guardam ao final do processo, o conhecimento que a rede adquiriu do ambiente em que está operando. Uma definição geral do que vem a ser aprendizagem pode ser expressa da seguinte forma:

> "Aprendizagem é o processo pelo qual os parâmetros de uma rede neural são ajustados através de uma forma continuada de estímulo pelo ambiente no qual a rede está operando, sendo o tipo específico de aprendizagem realizada definido pela maneira particular como ocorrem os ajustes realizados nos parâmetros".[MM70]

Diversos métodos para treinamento de redes foram desenvolvidos, podendo estes serem agrupados em dois paradigmas principais: "aprendizado supervisionado" e "aprendizado não supervisionado", outros dois paradigmas bastante conhecidos são os de "aprendizado por reforço" e "aprendizado por competição".

2.4.1 Aprendizado Supervisionado

Este método é o mais comum no treinamento das RNAs, sendo chamado "aprendizado supervisionado" porque a entrada e saída desejadas para a rede são fornecidas por um supervisor externo.

Nesta abordagem, o caso é apresentado à rede neural, que faz sua predição. O algoritmo de aprendizado obtém a diferença entre a saída esperada e a realizada. Esta diferença é utilizada no ajuste dos pesos, de modo que na próxima vez a predição será mais próxima da esperada. Deste modo, a rede neural aprende a relação entre as entradas e as saídas.

Existem problemas reais não lineares, que possuem relações complexas entre as variáveis e para os quais nenhuma função matemática é conhecida nem pode ser facilmente derivada. Para estes casos e para aqueles em que o problema em si pode mudar com o tempo, as redes neurais com aprendizado supervisionado podem ser utilizadas. Os exemplos mais conhecidos de algoritmos para aprendizagem supervisionado são *regra delta* [WH60] e a sua generalização para redes de múltiplas camadas, o algoritmo *Back-Propagation* o regra delta generalizada [RHW86]. A figura 2.5 ilustra o mecanismo de aprendizado supervisionado.

2.4.2 Aprendizado Não Supervisionado ou AutoSupervisionado

Como o próprio nome sugere, não há um professor ou supervisor para acompanhar o processo de aprendizado, a figura 2.6 ilustra o processo.

É utilizado quando se possui grandes quantidades de dados e não se sabe a saída desejada. Neste caso deseja-se que a rede neural descubra o relacionamento dos dados,

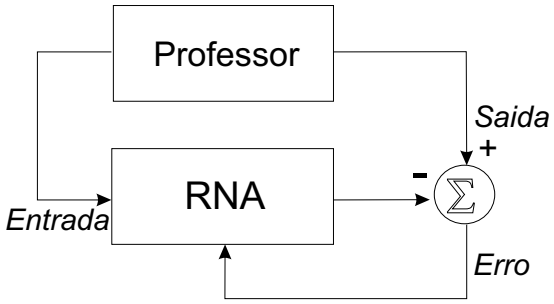

Figura 2.5: Mecanismo de Aprendizado Supervisionada

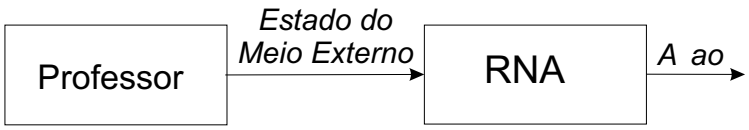

Figura 2.6: Mecanismo de Aprendizado Não Supervisionada

dividindo-os em grupos de elementos similares entre si. As redes neurais treinadas sob este paradigma possuem a capacidade de se organizar. Quando apresentados uma série de entradas, as unidades de saída se organizam inicialmente pela competição no reconhecimento do padrão e depois cooperando para o ajuste do peso de suas conexões.

Alguns métodos para implementação de aprendizado não supervisionado são: Aprendizado Hebbiano [Heb49] que motivou os primeiros métodos de aprendizado em RNAs, o Modelo de Linsker [Lin88] que foi proposto com o objetivo de modelar os primeiros estágios do sistema visual dos mamíferos. Aprendizagem por competição onde dado um padrão de entrada, faz com que as unidades de saída disputem entre si para serem ativadas [Fuk75] [Koh82] [Gro76].

2.4.3 Aprendizado por Reforço

Aprendizado por reforço é uma forma de aprendizado *on-line* obtido por um mapeamento de entrada e saída através de um processo de triagem e erro desenvolvido para maximizar o índice de desempenho escalar chamado de *sinal de reforço*. A figura 2.7 ilustra o procedimento. O termo aprendizagem por reforço foi inicialmente citado por Minsky [Min61] em seus estudos iniciais de *Inteligência Artificial - IA*.

A idéia básica que está por trás do termo *reforço* tem sua origem em estudos experimentais sobre aprendizado dos animais [Ham90]. Neste contexto, é interessante lembrar a *Lei do Efeito* [Tho11] que diz que "quanto maior a satisfação obtida com uma certa experiência em um animal, maiores as chances dele aprender". Sutton [SBW91] reformulou o dito por Thorndike na seguinte definição de aprendizado por reforço:

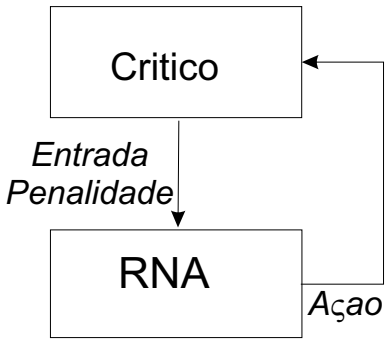

Figura 2.7: Mecanismo de Aprendizado por Reforço

> "Se uma ação tomada pelo sistema de aprendizagem é seguida de estados satisfatórios, então a tendência do sistema de produzir esta ação particular é reforçada. Se não for seguida de estados satisfatórios a tendência do sistema de produzir esta ação é enfraquecida" [SBW91]

2.5 Modelos de Redes Neurais

A combinação da arquitetura, do paradigma de aprendizado e de seu algoritmo definem um modelo de rede neural. Existem uma grande variedade de modelos. Por exemplo, alguns são otimizados para ter treinamento rápido e outros para ter memória de entradas passadas. Contudo, o melhor modelo para uma certa aplicação depende dos dados e das funcionalidades requeridas.

A tabela 2.1 relaciona modelos com seu paradigma de aprendizado, arquitetura e funções.

2.6 Treinamento de Redes Neurais

É na fase de treinamento que a rede neural adquire o conhecimento que vai permitir a realização de funções como classificação e predição de série temporais, entre outras. Este conhecimento é representado pelos pesos das conexões entre as unidades de processamento que são ajustados durante o treinamento.

O processo é extremamente dependente de fatores como a disponibilidade e característica dos dados, do modelo da rede neural e de sua função. Na maioria dos modelos, os pesos são inicializados com valores aleatórios. À medida que os exemplos são apresentados à rede neural, os pesos são ajustados de acordo com o algoritmo de

Modelo	**Paradigma de Aprendizado**	**Arquitetura**	**Funções Primárias**
BackPropagation	Supervisionado	Feedforward	Classificação
Recurrent Back-Propagation	Supervisionado	P. Recorrente	Modelagem, Séries Temporais
Adaptative Resonance Theory	Não supervisionado	P. Recorrente	Segmentação
ARTMAP	Supervisionado	P. Recorrente	Classificação
Kohonen Feature Maps	Não supervisionado	Feedforward	Segmentação
Radial Basis Function	Supervisionado	Feedforward	Classificação, Modelagem, Séries Temporais

Tabela 2.1: Modelos e Funções de Redes Neurais.

aprendizado. Este é dependente do modelo. Deve-se monitorar o treinamento da rede neural para determinar se ela está realmente aprendendo. Definido o problema, deve-se obter o banco de exemplos que será utilizado no treinamento.

É necessário determinar quais são os parâmetros importantes no processo de decisão. A etapa de seleção exige conhecimento profundo sobre o domínio do problema e dos dados envolvidos. Os dados devem ser tratados antes de serem submetidos à rede neural. Alguns parâmetros podem ser combinados em um só ou serem eliminados em caso de redundância. Freqüentemente, os dados devem ser transformados para uma forma que seja aceitável para a rede neural. A forma de representação dos dados é importante, se for feita a escolha errada, é possível que a rede neural não consiga aprender a relação entre os dados. Além disso a forma de representação afeta diretamente o tempo de treinamento e a precisão obtida. Muitos parâmetros determinam a velocidade do treinamento e o grau de generalização da rede neural, alguns desses parâmetros são gerais, como a função de ativação, e vários são específicos a cada modelo.

O coeficiente de aprendizado é um parâmetro geral que determina o grau das mudanças realizadas nos pesos visando a saída desejada. Quando se aplica um alto coeficiente de aprendizado, os pesos oscilam muito, mas o treinamento é mais rápido e quando o coeficiente de aprendizado é baixo, o treinamento é lento. A tabela 2.2 mostra alguns dos parâmetros mais importantes no treinamento das redes neurais.

Parâmetro	**Modelo**	**Descrição**
Coeficiente de Aprendizado	Todos	Controla o tamanho do passo para ajustes dos pesos. Diminui com o tempo em alguns modelos.
Função de Ativação	Todos	Seleciona a função de ativação utilizada na unidade de processamento
Momentum	*BackPropagation*	Suaviza os efeitos dos ajustes no peso em função do tempo
Error Tolerance	*Backpropagation*	Especifica quão próximo valor da saída deve estar antes que o erro seja considerado zero
Vigilance	ART	Especifica quão similares os padrões de entrada devem ser para serem classificados na mesma categoria.
Neighborhood	*kohonen Maps*	Define o tamanho da área ao redor da unidade de saída vencedora que será atualizada. Diminui com o tempo.
Número de Épocas	*kohonen Maps* e outros	Determina o número fixo de vezes que as redes neurais de alguns modelos passa pelos dados de treinamento.

Tabela 2.2: Parâmetros de Treinamento de Redes Neurais.

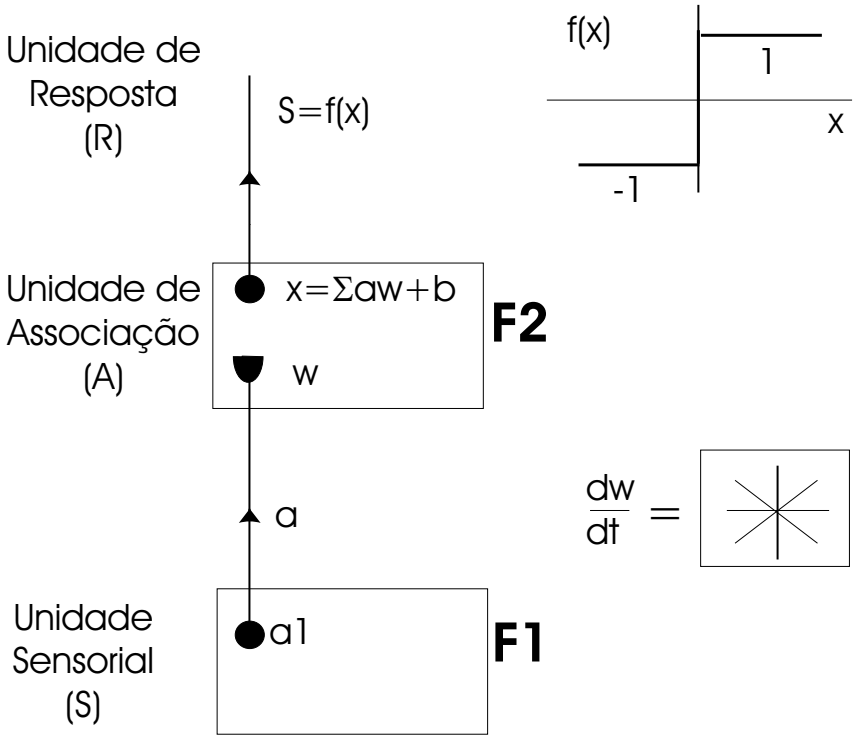

Figura 2.8: Neurônio de McCulloch-Pitts com Aprendizado

2.7 A Rede Perceptron com uma Camada

O modelo proposto por McCulloch-Pitts foi extraordinariamente não funcional, uma grande figura na seguinte década foi Frank Rosenblatt. Em 1958 Rosemblatt propôs a rede Perceptron [Ros58] como o primeiro modelo para aprendizagem de RNAs por meio de um tutor. A idéia principal do Perceptron é incorporar aprendizagem no neurônio de McCulloch-Pitts.

O Perceptron é a forma mais simples de uma RNA, consiste básicamente de um único neurônio com pesos sinápticos ajustáveis e uma polarização(bias). O algoritmo usado para ajustar os parâmetros livres desta RNA foi apresentado pela primeira vez no procedimento de aprendizagem desenvolvido por Rosenblatt, no qual depois de convergir posiciona uma superfície de decisão na forma de um hiperplano entre as classes. A prova de convergência do algoritmo é conhecida como Teorema de Convergência do Perceptron.

A figura 2.8 ilustra os principais elementos do Perceptron, incluídos na terminologia de Rosemblatt: a unidade sensorial (S); a unidade de Associação (A), onde é realizada a aprendizagem; e a unidade de resposta (R).

Apesar de ter causado grande euforia na comunidade científica da época, o Perceptron não teve vida muito longa, já que as duras críticas de Minsky e Paper [MP69] à

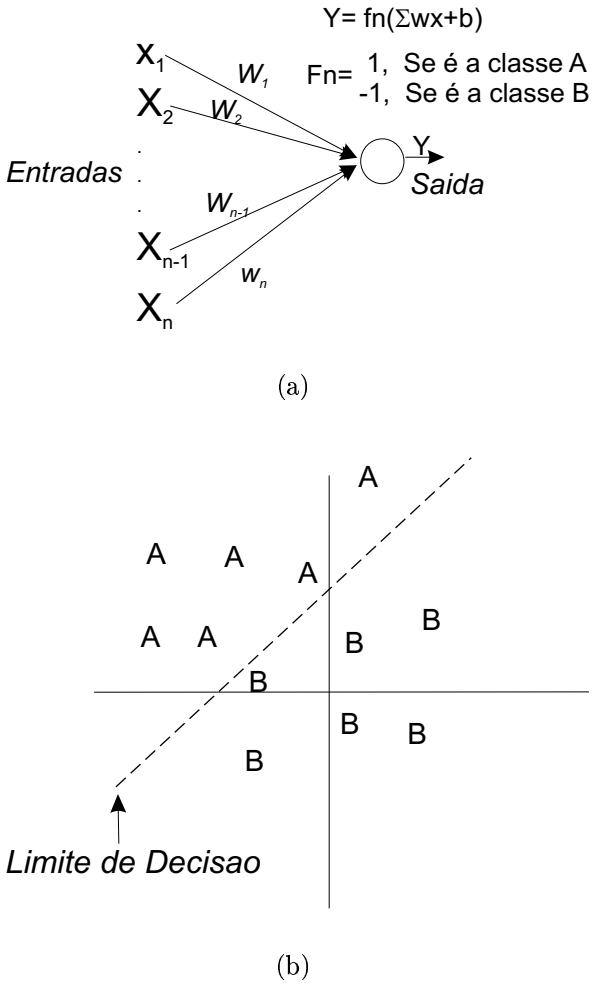

Figura 2.9: Rede Perceptron de uma Camada e a Separação das Classes

sua capacidade computacional causaram grande impacto sobre as pesquisas em RNAs, o que levou a um grande desinteresse pela área durante os anos 70 e início dos anos 80. Esta visão pessimista sobre a capacidade do Perceptron e das RNAs de uma maneira geral mudou com as descrições da rede Hopfield em 1982 [Hop82] e do algoritmo back-propagation em 1986. Foi em conseqüência destes trabalhos que as RNAs ganharam novo impulso, ocorrendo a partir do final dos anos 80, uma forte expansão no número de trabalhos de aplicação envolvendo RNAs e técnicas correlatas.

2.7.1 Algoritmo de Aprendizado do Perceptron

A rede Perceptron que decide quando uma entrada pertence a uma de duas classes (denotadas com A e B) é mostrada na figura 2.9. Um simples neurônio computa a soma dos pesos com os elementos de entrada e subtraem o bias, o resultado passa por um limitador não linear tal que a saída y pode ser 0 ou $+1$. A regra de decisão

responde 1 se é a classe A e 0 se é a classe B. O Perceptron forma duas regiões de decisão separadas por um hiperplano. Esta regiões são mostradas na figura 2.9 quando são 2 entradas, o hiperplano é uma reta.

Na tabela 2.3 é mostrado o procedimento para convergência do Perceptron. Os valores dos pesos e o bias são inicializados com valores pequenos aleatórios diferentes de zero. Então a uma nova entrada com N valores contínuos é aplicado na entrada da rede e a saída é computada. As conexões dos pesos são adaptadas só quando um erro acontece usando a equação no passo 4 na tabela 2.3. Esta equação inclui um ganho η no intervalo de 0.00 a 1.0 que é o coeficiente de aprendizado, o qual controla o índice de adaptação.

Passo 1	Inicialização dos pesos e bias Atribuir valores aleatórios pequenos para $w_i(0)$ $(0 \leq i \leq N-1)$ e b. Aqui $w_i(t)$ é o peso da entrada i no tempo t e b e o bias no neurônio de saída
Passo 2	Apresentar novos valores contínuos de entrada $x_0, x_1, ..., x_{n-1}$ com a saída desejada $d(t)$.
Passo 3	Calcular a saída atual $y(t) = f_n(\sum_{i=0}^{N-1} w_i(t)x_i(t) - b)$
Passo 4	Adaptar os Pesos $w_i(t+1) = w_i(t) + \eta[d(t) - y(t)]x_i(t)$, onde $0 \leq i \leq N-1$ $d(t) = \begin{cases} 1 & \text{Se a entrada é da classe A,} \\ 0 & \text{Se a entrada é da classe B.} \end{cases}$ nesta equação η é o ganho positivo menor que 1 e $d(t)$ é a saída correta desejada para a atual saída. Se a rede executa a decisão correta, então os pesos não mudam.
Passo 5	Repetir desde o passo 2

Tabela 2.3: Processo de Convergência da Rede Perceptron de uma Camada

A grande vantagem da implementação do algoritmo Perceptron é a simplicidade. São poucos parâmetros a ajustar e o padrão de entrada não necessita de um pré-processamento muito elaborado, dependendo da aplicação. Por outro lado, ele tem sua aplicação restrita a padrões não muito complexos, que sejam linearmente separáveis

2.7.2 Algoritmo de Aprendizado LMS

Ao mesmo tempo que Frank Rosenblatt trabalhava no modelo Perceptron, Bernard Widrow e seu estudante Marcian Hoff, introduziram o modelo Adaline e a regra de aprendizado chamado de algoritmo LMS *(Least Mean Square)* [WH60] [WS85] [DH73]. A rede Adaline é similar ao Perceptron, exceto que a função de ativação do Perceptron é do tipo degrau e da rede Adaline é uma função função de tipo linear.

A rede Adaline apresenta as mesmas limitações da rede Perceptron de Rosenblatt, só pode solucionar problemas linearmente separáveis. Mas o algoritmo LMS é mais potente que a regra de aprendizado do Perceptron, porque minimiza o erro médio quadrático tornando-o mais prático nas diferentes aplicações. Os pesos são assim corrigidos por um incremento que depende da diferença entre a saída desejada e a atual saída na direção do gradiente negativo.

Este algoritmo chamado também de "regra delta", teve uma maior importância porque deu origem ao algoritmo utilizado para treinar redes neurais de múltiplas camadas "regra delta generalizada" ou chamado também de "BackPropagation".

2.8 Rede Perceptron de Múltiplas Camadas

As redes de uma só camada resolvem apenas problemas linearmente separáveis. A solução de problemas não linearmente separáveis pode ser resolvido com redes com uma ou mais camadas intermediárias ou escondidas. O desconhecimento de algoritmos para treinar redes com uma ou mais camadas intermediárias foi uma das causas da redução das pesquisas em redes neurais artificiais na década de 70. As redes Perceptron de Múltiplas Camadas (MLP), apresentam um poder computacional muito maior do que aquele apresentado pelas redes sem camadas intermediárias.

As MLP são redes *feed-forward* com uma ou mais camadas de neurônios entre os neurônios de entrada e os neurônios de saída. Estas camadas adicionais contém unidades ocultas ou neurônios que não estão diretamente conectados aos neurônios de entrada e de saída. Em uma rede multi-camada, o processamento realizado por cada neurônio é definida pela combinação dos processamentos realizados pelos neurônios da camada anterior que estão conectados a ele. Quando se segue da primeira camada intermediária em direção à camada de saída, as funções implementadas se tornam cada vez mais complexas. Estas funções definem como é realizada a divisão do espaço de decisão.

Consideremos uma rede MLP com uma camada de unidades ocultas, esta rede tem

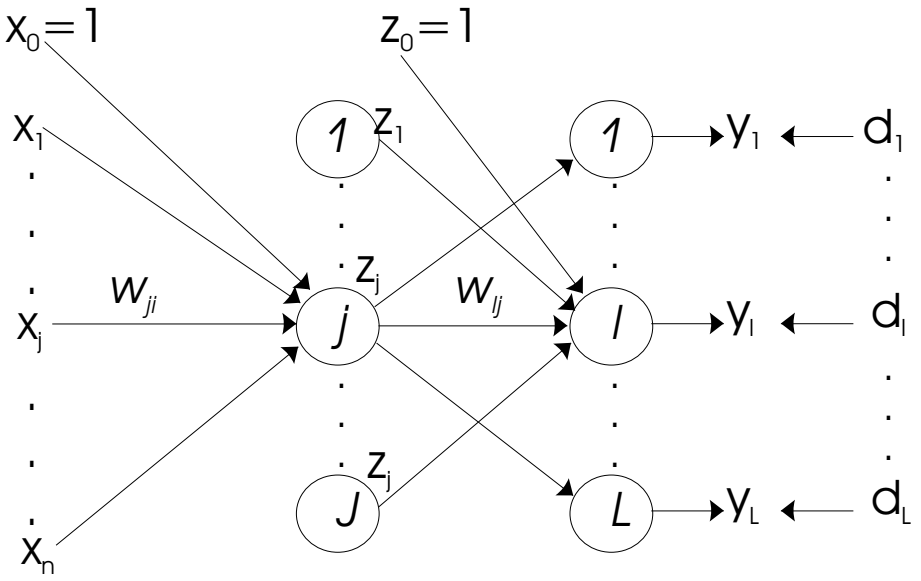

Figura 2.10: Camada Oculta com J neurônios de uma MLP

um conjunto de sinais de entrada $\{x_0, x_1, x_2, ..., x_n\}$ onde x_0 é o sinal *bias*. Neste conjunto de sinais o vetor de entrada $x \in \mathbb{R}^{n+1}$. A camada que recebe o sinal de entrada é chamada de camada oculta. A figura 2.10 mostra uma camada oculta tendo J unidades. A saída da camada oculta é o vetor $z = [z_0, z_1, ..., z_j]$ de valores reais $J+1$-dimensional. O vetor z é a fonte de entrada para a camada de saída de L unidades. A camada de saída gera um vetor L-dimensional y em resposta à entrada. Um dos aspectos relacionados as MLP diz respeito à função de ativação utilizada. Diversas funções de ativação têm sido propostas, estas funções são não lineares e diferenciáveis. As funções precisam ser diferenciáveis para que o gradiente possa ser calculado, direcionando o ajuste dos pesos, a maioria delas é também não decrescente.

Existem atualmente vários algoritmos para treinar redes MLP, [RM86], [Fah88], [Rie94], [Pea92], [Bat91], [HM94]. Estes algoritmos são geralmente do tipo supervisionado. De acordo com os parâmetros que eles atualizam, os algoritmos para treinamentos de redes do tipo MLP podem ser classificados em:

- Estáticos: não alteram a estrutura da rede, variam os valores de seus pesos para poder treinar. É utilizada a mesma regra de aprendizado para qualquer rede MLP sem importar o tamanho ou formato da rede.
- Dinâmicos: podem tanto reduzir quanto aumentar o tamanho da rede (número de camadas, número de neurônios nas camadas intermediárias e número de conexões)

O algoritmo de aprendizado mais conhecido para o treinamento destas redes é o algoritmo *back-propagation*[RM86]. A maioria dos métodos de aprendizado para RNAs do tipo MLP utilizam variações deste algoritmo.

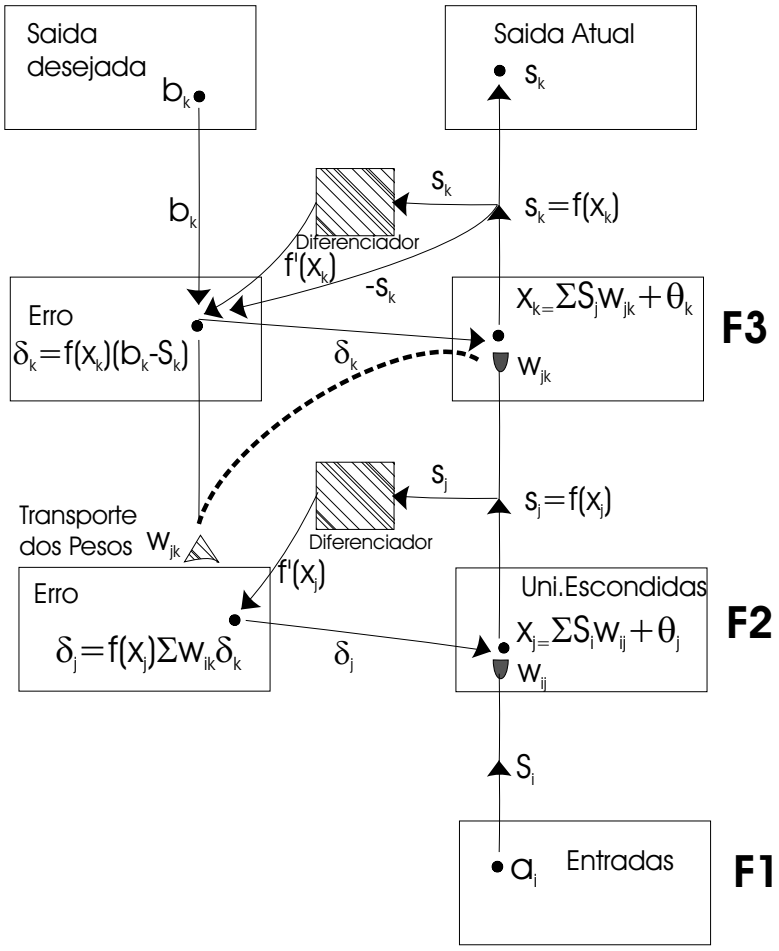

Figura 2.11: Diagrama de Blocos do Algoritmo BackPropagation

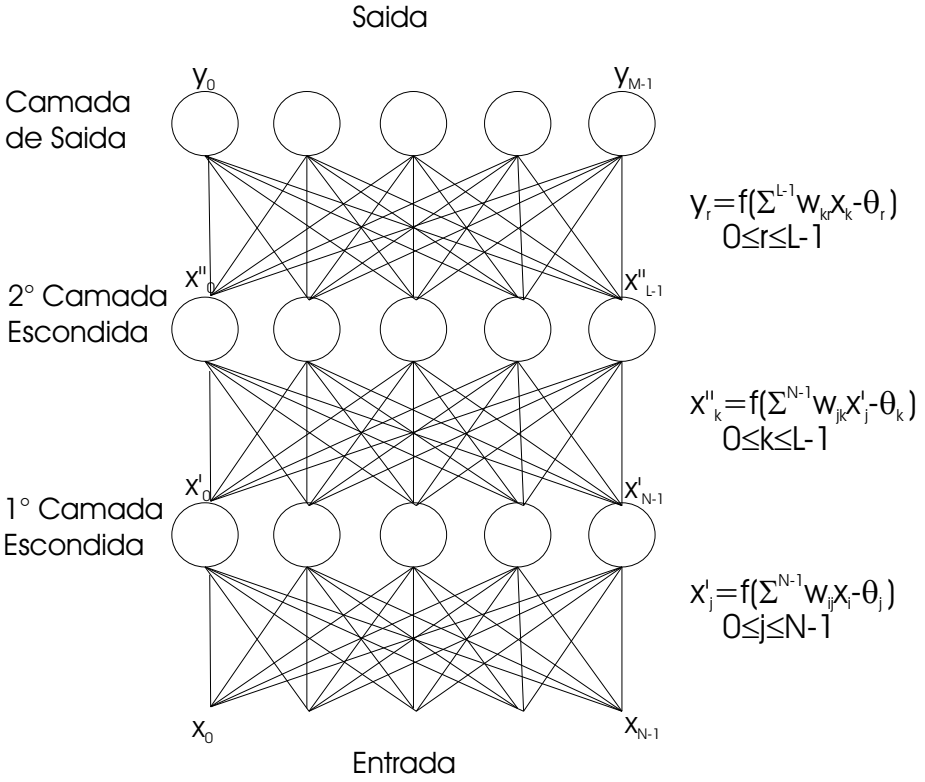

Figura 2.12: Rede Neural MLP com Duas Camadas

2.8.1 Algoritmo de Aprendizado - *BackPropagation*

O algoritmo *Back-Propagation* foi um dos principais responsáveis pelo ressurgimento das RNAs, por ocasião da publicação do livro *Parallel Distributed Processing*, mais conhecido por PDP, [RM86]. Embora a popularização deste algoritmo tenha surgido a partir de 1986, ele foi proposto muito antes, com diferentes propósitos, por diferentes pesquisadores, como: Bryson e Ho [BH69], Werbos [Wer74], Parker [Par85] e Le Cun [Cun85].

O algoritmo *BackPropagation* ou também chamado de retropropagação é uma generalização do algoritmo LMS [WH60]. Usa a técnica do gradiente para minimizar a função de custo que é o erro medio quadrado entre a saída desejada e a atual saída da rede. O *Backpropagation* é um poderoso algoritmo de aprendizado de propósito geral, mas também é caro em termos de requerimentos computacionais. Uma rede neural com uma simples camada oculta de elementos de pré-processamento treinada com este algoritmo modela qualquer função contínua com um alto grau de precisão (com neurônios suficientes na camada oculta) [Neg02].

O algoritmo básico *BackPropagation* consiste em três etapas. O padrão de entrada é apresentado à Rede Neural, estas entradas são propagadas até chegar à ultima camada da rede, as quais produzem uma saída que será a saída da rede. Como o algoritmo

BackPropagation é um aprendizado supervisionado, tem saídas desejadas as quais formam parte do processo de treinamento. A saída atual da rede é comparada com as saídas desejadas e um sinal de erro é produzido. Este sinal é a base para o passo de retro-propagação propriamente dito, onde o erro é passado desde a camada de saída até a camada de entrada. As conexões dos pesos são modificadas e a Rede Neural aprende a partir de uma nova experiência. A figura 2.11 descreve o diagrama de blocos do algoritmo de aprendizado para uma rede de 3 níveis.

A tabela 2.4 descreve o algoritmo de treinamento. O algoritmo assume que a função de ativação usada nas camadas da MLP é a função sigmóide, mas pode-se usar qualquer outra sempre que cumpra com as condições descritas anteriormente.

Tabela 2.4: Algoritmo *BackPropagation* para Treinar as MLP

Passo 1	Inicialização dos Pesos e Bias Os conjuntos de pesos de cada camada são iniciados com valores aleatórios pequenos. $w_{ij}(0)$ $(0 \leq i \leq N-1)$ e θ. Onde $w_{ij}(t)$ é o peso do neurônio i no tempo t ao neurônio j numa camada mais na frente, θ_j e o bias no neurônio j
Passo 2	Apresentar as entradas e as saídas desejadas Apresentar cada vetor de entrada de valores contínuos $x_0, x_1, ..., x_{n-1}$ e especificar as saídas desejadas $d_0, d_1, ..., d_{M-1}$. Se a rede é usada como um classificador então as saídas desejadas são conjuntos de zeros e uns. O conjunto de treinamento é apresentado ciclicamente até os pesos se estabilizarem.
Passo 3	Calcular as saídas atuais Usando as funções sigmoidais não lineares calculamos as saídas em cada camada até a camada de saída. As saídas de uma camada s são usadas como entradas na camada $s+1$, assim se a camada $s+1$ tem k neurônios e a camada s tem n neurônios, então as saídas em $s+1$ será: $y_k = f_n(\sum_{i=0}^{N-1} w_{ij}x_i - \theta_j)$ A figura 2.12 descreve uma rede MLP com 2 camadas ocultas e apresenta o cálculo da saída em cada camada.
Passo 4	Adaptação dos Pesos O algoritmo é recursivo, começa nos neurônios da camada de saída e trabalha até a camada de entrada. É esta a razão do nome *BackPropagation ou retropropagação*. O ajuste dos pesos usa a seguinte equação:

	$w_{ij}(t+1) = w_{ij}(t) - \eta\nabla e^2$ onde η é a taxa de aprendizado que varia entre 0 e 1 dependendo das características do problema e $-\nabla e^2$ é o gradiente negativo do erro médio quadrático para cada padrão de entrada. O gradiente é calculado como a derivada do erro com respeito a todos os pesos na rede. A atualização dos pesos é feita da seguinte forma: $w_{ij}(t+1) = w_{ij}(t) + \eta\delta_j x'_j$ Nesta equação $w_{ij}(t)$ é o peso desde o neurônio i ou desde uma entrada ao neurônio j no tempo t, x'_j é cada uma das saídas na camada do neurônio i ou é as entradas na rede e δ_j é o termo do erro para o neurônio j. Se o neurônio j está na camada de saída, então com a função de ativação sigmóide, temos que: $\delta_j = y_j(1-y_j)(d_j - y_j)$ onde d_j é a saída desejada do neurônio j e y_j é a saída atual. Se o neurônio j é um neurônio numa camada oculta, então: $\delta_j = x'_j(1-x'_j)\sum_k \delta_k w_{jk}$ onde k são todos os neurônios da camada $s+1$ relacionados com o neurônio j na camada s. As conexões dos *Bias* são atualizadas de forma similar aos pesos levando em conta a camada onde está.
Passo 5	Repetir desde o passo 2 até que $\sum_{p=1}^{r} e^2$ seja minimizado

Capítulo 3

Matemática Intervalar

3.1 Introdução

A Matemática Intervalar é uma teoria matemática originada na década de 60 [Moo66] com o objetivo de responder questões de exatidão e eficiência que surgem na prática da computação científica e na resolução de problemas numéricos.

A qualidade do resultado em computação científica depende do conhecimento e controle dos erros na computação. Algoritmos convencionais, chamados algoritmos pontuais, computam uma resposta exata sem o auxílio de uma análise rigorosa dos erros, o qual é extensa, dispendiosa e nem sempre viável. Desta forma, a obtenção de uma solução numérica para um problema real, aplicando os métodos tradicionais, geralmente conduz a resultados aproximados. Por outro lado, técnicas intervalares podem ser programadas em computadores de tal modo que a computação possua uma rigorosa e completa análise do erro durante a evolução dos cálculos numéricos, tornando-se necessário identificar qual a sua origem ou fonte.

Técnicas intervalares manipulam dados e parâmetros iniciais com intervalos, com o indicativo do erro máximo presente nestes valores, antes que os mesmos sejam introduzidos no computador. Desta forma, algoritmos intervalares, em contraste com os algoritmos pontuais, computam um intervalo como solução, com a garantia de que a resposta ótima pertence a este intervalo. Portanto, resultados intervalares carregam sempre consigo a segurança de sua qualidade e o grau de sua incerteza, pois o diâmetro de um intervalo solução é um indicativo da influência do erro do dado de entrada no erro do resultado final obtido.

Apresentaremos neste capítulo um estudo das definições básicas da matemática intervalar que serão necessárias para o desenvolvimento deste trabalho. Para se aprofundar mais nos conceitos de matemática intervalar ver [Moo62, Moo66, Moo79, Lyr03].

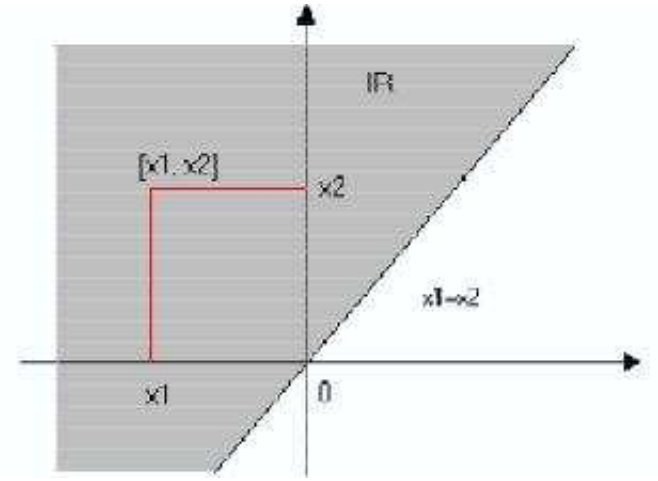

Figura 3.1: Representação Geométrica de $\mathbb{IR}$

Cabe ressaltar que neste capítulo e nos próximos, todos os números e funções que representem intervalos serão representados por letras maiúsculas e todos os números e funções reais serão representados por letras minúsculas. O extremo esquerdo ou ínfimo de cada intervalo será representado pela mesma letra que denota o intervalo em minúscula com uma barra abaixo e o extremo direito o supremo será representado pela letra que denota o intervalo em minúscula com uma barra abaixo.

3.2 Definições Básicas

Definição 3.2.1 *(Intervalos de Reais)*

Seja $\mathbb{R}$ *o conjunto dos números reais, então* $\forall \underline{x}\,\overline{x} \in \mathbb{R}$ *tal que* $\underline{x} < \overline{x}$*, o conjunto* $\{x \in \mathbb{R}/\underline{x} \leq x \leq \overline{x}\}$ *é um intervalo de reais ou simplesmente um intervalo e será denotado por* $X = [\underline{x}, \overline{x}]$*. Os pontos do conjunto dos intervalos de reais serão denotados por letras latinas maiúsculas.*

Exemplos: $[1;5], [0,2365; 0,6974], [-0,98742; 0,12455], [2;2]$

Definição 3.2.2 *(Conjunto* $\mathbb{IR}$*)*

Definimos e denotamos por $\mathbb{IR}$ *o conjunto de todos os intervalos de reais, isto é* $\mathbb{IR}=\{[\underline{x}, \overline{x}]/\underline{x}, \overline{x} \in \mathbb{R}, \underline{x} \leq \overline{x}\}$*. como é mostrado na figura 3.1*

Definição 3.2.3 *(Intervalos Degenerados)*

Um número $X \in \mathbb{IR}$*, é um intervalo degenerado se* $\underline{x} = \overline{x}$*.*

3.3 Operações e Propriedades da Aritmética de Moore

Segundo Moore [Moo79], uma operação aritmética binária $\oplus \in \{+, -, x, \div\}$ entre os intervalos X e Y é definida como $X \oplus Y = \{x \oplus y / x \in X, y \in Y\}$. Pode-se observar que $\mathbb{IR}$ constitui uma estrutura algébrica que generaliza a estrutura dos reais.

3.3.1 Igualdade entre Intervalos

Definição 3.3.1 *Sejam X e Y dois intervalos de $\mathbb{IR}$ Diz-se que $X = Y$ se, e somente se, $\underline{x} = \underline{y}$ e $\overline{x} = \overline{y}$.*

Logo, dois intervalos são considerados iguais se os conjuntos que eles representam forem os mesmos.

3.3.2 Adição em $\mathbb{IR}$

Proposição 3.3.2 *Sejam os intervalos X e Y então:*
$X + Y = [\underline{x}, \overline{x}] + [\underline{y}, \overline{y}] = [\underline{x} + \underline{y}, \overline{x} + \overline{y}]$.

Propriedades Algébricas da Adição em $\mathbb{IR}$: Sejam $X, Y, Z, \in \mathbb{IR}$, então:

- Associatividade: $X + (Y + Z) = (X + Y) + Z$;
- Comutatividade: $X + Y = Y + X$;
- Elemento Neutro: Seja $0 = [0; 0]$. Então $X + 0 = 0 + X = X$.
- Se $X + Y = X + Z$ então $Y = Z$
- Se $X + Y \subset X + Z$ então $Y \subset Z$.

Exemplo: Dado $X = [1; 4]$ e $Y = [2; 5]$ então:

$$\begin{aligned} X + Y &= [1; 4] + [2; 5] \\ &= [1 + 2; 4 + 5] \\ &= [3; 9]. \end{aligned}$$

3.3.3 Subtração em $\mathbb{IR}$

Proposição 3.3.3 *Sejam X e Y dois intervalos, então:*
$X - Y = X + (-Y) = [(\underline{x} - \overline{y}); (\overline{x} - \underline{y})]$

Propriedade Algébrica da subtração em $\mathbb{IR}$:

- Se $Y - X = Z - X$ então $Y = Z$

Exemplo: Dado $X = [-1; 1], Y = [0,236; 0,245]$, então:

$$\begin{aligned} X - Y &= [-1; 1] - [0,236; 0,245] \\ &= [-1; 1] + (-[0,236; 0,245]) \\ &= [-1; 1] + [-0,245; -0,236] \\ &= [-1 - 0,245; 1 - 0,236] \\ &= [-1,245; 0,764]. \end{aligned}$$

3.3.4 Multiplicação em $\mathbb{IR}$

Proposição 3.3.4 *Sejam X e Y dois intervalos, então:*
$X \cdot Y = [\underline{x}, \overline{x}] \cdot [\underline{y}, \overline{y}] = [min\{\underline{xy}, \overline{x}\underline{y}, \underline{x}\overline{y}, \overline{xy}\}, max\{\underline{xy}, \overline{x}\underline{y}, \underline{x}\overline{y}, \overline{xy}\}]$

Propriedades Algébricas da Multiplicação em $\mathbb{IR}$: Sejam $X, Y, Z \in \mathbb{IR}$ então:

- Associatividade: $X \cdot (Y \cdot Z) = (X \cdot Y) \cdot Z$;
- Comutatividade: $X \cdot Y = Y \cdot X$
- Elemento Neutro: Seja $1 = [1; 1]$, então $X \cdot 1 = 1 \cdot X = X$;
- Sub-distributividade: $X \cdot (Y + Z) \subset (X \cdot Y) + (X \cdot Z)$.
- $X \cdot 0 = 0 \cdot X = [0; 0]$.

Exemplo: Dado $X = [-2; -1], B = [2; 3]$, então:

$$\begin{aligned} X \cdot Y &= [-2; -1] \cdot [2; 3] \\ &= [min\{(-2) \cdot (2), (-2) \cdot 3, (-1) \cdot 2, (-1) \cdot 3\}; \\ &\quad max\{(-2) \cdot (2), (-2) \cdot 3, (-1) \cdot 2, (-1) \cdot 3\}] \\ &= [min\{-4, -6, -2, -3\}; max\{-4, -6, -2, -3\}] \\ &= [-6; -2]. \end{aligned}$$

3.3.5 Multiplicação por um Escalar

Definição 3.3.5 *O produto escalar a por ujm intervalo X denotado por $X.a$ é definido por $X.a = \{a.x : x \in X\}$*

Proposição 3.3.6 $X \cdot a = [min\{\underline{x} \cdot a, \overline{x} \cdot a\}; max\{\underline{x} \cdot a, \overline{x} \cdot a\}]$

Exemplo: Seja $X = [-0,024; 0]$ e $a = 0,2$, então:

$$\begin{aligned} X \cdot a &= [-0,024; 0] \cdot 0,2 \\ &= [min\{((-0,024) \cdot 0,2), (0 \cdot 0,2)\}; \\ &\quad max\{((-0,024) \cdot 0,2), (0 \cdot 0,2)\}] \\ &= [min\{-0,00480\}; max\{-0,00480\}] \\ &= [-0,0048; 0]. \end{aligned}$$

3.3.6 Divisão em $\mathbb{IR}$

Proposição 3.3.7 *Sejam $X = [\underline{x}; \overline{x}], Y = [\underline{y}; \overline{y}] \in \mathbb{IR}$ tal que $0 \notin Y$ então:*
$X \div Y = [min\{\underline{x} \div \overline{y}, \underline{x} \div \underline{y}, \overline{x} \div \overline{y}, \overline{x} \div \underline{y}\}; max\{\underline{x} \div \overline{y}, \underline{x} \div \underline{y}, \overline{x} \div \overline{y}, \overline{x} \div \underline{y}\}]$

Exemplo: Sejam $X = [1.2; 1.35], Y = [0.12; 0.26]$, então:

$$\begin{aligned} X \div Y &= [1.2; 1.35] \div [0.12; 0.26] \\ &= [min\{(1.2 \div 0.12), (1.2 \div 0.26), (1.35 \div 0.12), (1.35 \div 0.26)\}; \\ &\quad max\{(1.2 \div 0.12), (1.2 \div 0.26), (1.35 \div 0.12), (1.35 \div 0.26)\}] \\ &= [min\{10;\ 4.61538462;\ 11.25;\ 5.19230769\}; \\ &\quad max\{10;\ 4.61538462;\ 11.25;\ 5.19230769\}] \\ &= [4.61538462;\ 11.25]. \end{aligned}$$

3.3.7 Distância entre Intervalos

Definição 3.3.8 *Sejam X e Y dois intervalos, então a distância do intervalo X ao intervalo Y é definida por:*

$$dist(X, Y) = dist([\underline{x}, \overline{x}], [\underline{y}, \overline{y}]) = max\{|\underline{x} - \underline{y}|; |\overline{x} - \overline{y}|\}$$

Observe-se que, considerando que a distância usual nos reais é dada por $d(x, y) = |x-y|$, a distância entre dois intervalos é a maior distância (real) entre seus extremos. Esta definição é da distância de Moore [Moo66].

3.3.8 Ponto Médio de um Intervalo

Definição 3.3.9 *Seja $X = [\underline{x}, \overline{x}] \in \mathbb{IR}$. O ponto médio do intervalo X denotado por $\breve{x} = med(X)$ é o número real definido por:*

$$\breve{a} = med(A) = \frac{\overline{a} + \underline{a}}{2}$$

3.4 Funções Intervalares

3.4.1 Produto Cartesiano Intervalar

Definição 3.4.1 *Sejam X, Y conjuntos não vazios, formados por elementos intervalares. O produto cartesiano é o conjunto de todos os pares ordenados intervalares onde a primeira coordenada é constituída por elementos do conjunto X e a segunda coordenada é constituída por elementos do conjunto Y, ou seja: $X * Y = \{(x, y)/x \in X, y \in Y\}$*

Exemplo: Sejam $X = \{[1;4], [5;6], [-1;0]\}$ e $Y = \{[0;2], [8;9]\}$
$X \mathrm{x} Y = \{ \quad ([1;4], [0;2]), ([1;4], [8;9]), ([5;6], [0;2]),$
$([5;6], [8;9]), ([-1;0], [0;2]), ([-1;0], [8;9])\}$

3.4.2 Função Intervalar

Definição 3.4.2 *Seja $F : X \to Y$ uma função. Se $X = Dom(F) \subseteq \mathbb{IR}$ e $Cod(F) \subseteq \mathbb{IR}$, então dizemos que F é uma função intervalar de uma variável intervalar.*

Exemplo: $F : \mathbb{IR} \to \mathbb{IR}$ onde $F(X) = [0,23; 1,366] \cdot X + [0,2154; 0,21654]$

3.4.3 Imagem Intervalar de uma Função Real

Definição 3.4.3 *Sejam f uma função real de variável real, X um intervalo tal que $\forall x \in X$, $x \in Dom(f)$. Definimos a imagem da função f em X como o conjunto determinado por:*

$$\mathcal{I}(f; X) = \{f(x)/x \in X\}$$

Se f é continua para todo ponto $x \in X$, então $\mathcal{I}(f, X)$ é um intervalo [SB04] Esta é uma maneira natural de definir funções intervalares a partir de funções reais (contínuas). Ou seja se f é uma função real não assintótica $\mathcal{I}$ é uma função intervalar

Exemplo: Seja $f(x) = 1 - e^x$ e $X = [2;3] \subseteq \mathbb{IR} = \mathbb{D}\rtimes\succcurlyeq(\mho)$ então:

$$\begin{aligned}\mathcal{I}(f,[2;3]) &= \{f(x)/x \in [2;3]\} \\ &= \{1 - e^x/x \in [2;3]\} \\ &= [min\{1 - e^x/x \in [2;3]\}; max\{1 - e^x/x \in [2;3]\}] \\ &= [-19.085537, -6.389056].\end{aligned}$$

Proposição 3.4.4 *Se f é contínua no intervalo X então:*

$$\mathcal{I}(f;X) = [min\{f(x)/x \in X\}; max\{f(x)/x \in X\}]$$

Corolario 3.4.5 *Se $f : \mathbb{R} \to \mathbb{R}$ é contínua e monotônica crescente então:*

$$\mathcal{I}(f,X) = [f(\underline{x}); f(\overline{x})]$$

3.4.4 Função Projeção à Esquerda

Definição 3.4.6 *A função Π_1:$\mathbb{IR} \to \mathbb{R}$ definida por: $\Pi_1([a,b]) = a$ é chamada de projeção à esquerda.*

Exemplo: $\Pi_1([2;3]) = 2$

3.4.5 Função Projeção à Direita

Definição 3.4.7 *A função Π_2:$\mathbb{IR} \to \mathbb{R}$ definida por: $\Pi_2([a;b]) = b$ é chamada de projeção à direita.*

Exemplo: $\Pi_2([2;3]) = 3$

3.4.6 Função Semi-Intervalar

Definição 3.4.8 *Uma função F é chamada de Semi-intervalar se $F : \mathbb{R} \to \mathbb{IR}$ ou $F : \mathbb{IR} \to \mathbb{R}$. no primeiro caso é chamada de função semi-intervalar à direita e no segundo de função semi-intervalar à esquerda.*

Exemplo: Função semi-intervalar à esquerda

$$F(X) = \begin{cases} +1 & \text{Se } \underline{x} + \overline{x} > 0, X_i = [\underline{x}; \overline{x}] \in \mathbb{IR}, \\ 0 & \text{em caso contrario.} \end{cases}$$

Exemplo: Seja Sign:$\mathbb{IR} \to \mathbb{R}$ a função definida por :

$$sign(X) = \begin{cases} +1 & \text{Se } \underline{x} > 0, \\ 0 & \text{Se } \underline{x} \leq 0 \leq \overline{x}, \\ \text{-1} & \text{Se } \overline{x} < 0. \end{cases}$$

Sign é chamada de função sinal e é uma função semi-intervalar a direita

3.4.7 Função Semi-Intervalar Mínima

Definição 3.4.9 *Seja a função F:*$\mathbb{IR} \to \mathbb{IR}$. *A função* $f_{min} : \mathbb{IR} \to \mathbb{R}$ *definida por:*$f_{min}(X) = \Pi_1(F(X))$ *é chamada de função semi-intervalar mínima de F.*

3.4.8 Função Semi-Intervalar Máxima

Definição 3.4.10 *Seja a função F:*$\mathbb{IR} \to \mathbb{IR}$. *A função* $f_{max} : \mathbb{IR} \to \mathbb{R}$ *definida por:*$f_{max}(X) = \Pi_2(F(X))$ *é chamada de função semi-intervalar máxima de F.*

3.4.9 Função Real Mínima

Definição 3.4.11 *Seja F:*$\mathbb{IR} \to \mathbb{IR}$. *A função* $f_{Rmin} : \mathbb{R} \to \mathbb{R}$ *definida por:* $f_{Rmin}(x) = \Pi_1(F[x; x])$, *é chamada de função real mínima de F.*

3.4.10 Função Real Máxima

Definição 3.4.12 *Seja F:*$\mathbb{IR} \to \mathbb{IR}$. *A função* $f_{Rmax} : \mathbb{R} \to \mathbb{R}$ *definida por:* $f_{Rmax}(x) = \Pi_2(F[x; x])$, *é chamada de função real máxima de F.*

3.5 Continuidade

Teorema 3.5.1 *Seja uma função intervalar F:*$\mathbb{IR} \to \mathbb{IR}$ *é continua (Na topologia de Moore) se e somente se existem funções contínuas* $f_i : \mathbb{IR} \to \mathbb{R}$ *e* $f_s : \mathbb{IR} \to \mathbb{R}$ *tais que*

$\forall X \in \mathbb{IR}, F(X) = [f_i(X); f_s(X)]$.

Portanto $f_i(X) \leqq f_s(X)$ *para cada* $X \in \mathbb{IR}$.

Teorema 3.5.2 *Seja F uma função semi-intervalar à direita. F é contínua se e somente se existem funções contínuas $f_i : \mathbb{R} \to \mathbb{R}$ tais que $\forall x \in \mathbb{R}, F(x) = [f_i(x); f_s(x)]$.*

Prova: Em [Sil02]

3.6 Derivadas de Funções Intervalares

Definição 3.6.1 *(Derivada Intervalar) Seja $F : \mathbb{IR} \to \mathbb{IR}$, tal que as funções reais mínima e máxima são deriváveis. Dizemos que a derivada de $\mathbb{F}$ em relação da variável $X \in \mathbb{IR}$ é a função intervalar $F' : \mathbb{IR} \to \mathbb{IR}$ definida por:*

$$F'(X) = [min\{\{f'_{Rmin}(x)/x \in X\} \cup \{f'_{Rmax}(x)/x \in X\}\};$$
$$max\{\{f'_{Rmin}(x)/x \in X\} \cup \{f'_{Rmax}(x)/x \in X\}\}]$$

Definição 3.6.2 *(Função Intervalar Derivável)*

Seja $F : \mathbb{IR} \to \mathbb{IR}$ é dita derivável no intervalo $X = [\underline{x}; \overline{x}]$, se $\forall x \in X$ a derivada intervalar for definida.

Definição 3.6.3 *(Derivada da Função Semi-Intervalar Esquerda)*

Seja F uma função semi-intervalar a esquerda. Dizemos que a derivada de F em relação a uma variável $X \in \mathbb{IR}$ é a soma das derivadas reais em relação a todos os $x \in X$ que são significantes no cálculo da função $F(X)$.

Exemplo:

$$F(X) = \frac{\overline{x} + \underline{x}}{2}$$
$$F'(X) = f'(\overline{x}) + f'(\underline{x})$$

Capítulo 4

Redes Neurais Intervalares

4.1 Introdução

O usuário não pode afirmar a exatidão da resposta estimada sem o auxílio de uma análise de erro, que é extensa, dispendiosa e nem sempre viável. No entanto, a matemática intervalar busca dar suporte a estes problemas. A importância das redes neurais adicionando a matemática intervalar é a necessidade atual de existirem ferramentas que estejam preparadas para trabalhar com dados intervalares além de garantir uma minimização dos erros nos processos computacionais das RNA. Este fato inspirou vários trabalhos nesta linha de pesquisa.

Serão apresentados na primeira parte deste capítulo alguns trabalhos pesquisados na área de redes neurais intervalares como uma revisão bibliográfica que deu início ao nosso trabalho. Na segunda parte deste capítulo é apresentada uma definição para as redes neurais intervalares e é definida uma estrutura para o neurônio intervalar que usaram as redes neurais intervalares propostas. Sendo nosso trabalho focado em redes neurais Perceptron Intervalares, será dedicado o capítulo seguinte para a explicação delas.

4.2 Revisão Bibliográfica

Utilizar a computação intervalar com as redes neurais artificiais não é uma abordagem nova, foram apresentados vários outros trabalhos além do nosso. A seguir são discutidos alguns deles.

4.2.1 Intervalos para Representação do Conhecimento

No ano 1991, foi apresentado um trabalho na área de redes neurais intervalares intitulado *An extension of the BP-algorithm to interval input vectors-learning from numerical data and expert's knowledge* [Tan91].

O objetivo deste trabalho foi a extensão do algoritmo BackPropagation para uma rede neural que tenha como entrada vetores com dados intervalares. A rede proposta faz um mapeamento de entradas intervalares a uma saída intervalar.

Neste artigo foi proposta a representação do conhecimento, de um especialista, com intervalos formando assim o conjunto de dados para treinar a rede neural. O algoritmo proposto pode ser visto como uma extensão do algoritmo *BackPropagation* para o treinamento de redes intervalares.

Foi definida uma função de custo, a qual é minimizada usando a idéia do gradiente para conseguir treinar a rede intervalar de múltiplas camadas. Também é estabelecida uma função sigmoidal intervalar que é usada como função de ativação.

A principal contribuição deste artigo foi a definição de um método de aprendizado desde o conhecimento do especialista representado em intervalos. É feita uma comparação da rede neural intervalar proposta com a rede pontual aplicadas em dois exemplos. Para poder treinar as redes neurais tradicionais com o conjunto de dados intervalares, o autor utiliza os extremos de cada intervalo e forma a base deles o conjunto de treinamento para a rede neural tradicional. Foi mostrado que a rede neural intervalar para o mesmo conjunto de dados é melhor na separação das classes.

A principal diferença deste trabalho com o nosso, é que as redes neurais intervalares que serão propostas nesta dissertação utilizam pesos intervalares, detalhe que faz com que o algoritmo de treinamento seja diferente.

Um segundo trabalho nesta linha foi o intitulado *Interval-Arithmetic-Based Neural Networks*, [IN01]. Nele foram abordados os vetores de dados intervalares como entradas para redes neurais. Cada intervalo da entrada gerado pelo conhecimento de um especialista, então os intervalos são construídos com base nas regras e é obtido o conjunto de treinamento com dados intervalares e dados reais, onde cada dado real é visto como um intervalo degenerado. Os intervalos são também utilizados para completar padrões de dados que faltam onde são substituídos por seu domínio intervalar. Segundo o autor, o teste na rede gera bons resultados e todos os padrões são bem classificados.

Neste trabalho é apresentado o algoritmo de aprendizado para as redes MLP, mas não considera uma estrutura intervalar da rede, além disso, o domínio de uma variável pode ser todo o conjunto $\mathbb{R}$ sendo impossível a entrada desse "tipo"de intervalos.

4.2.2 Intervalos para a Representação de Dados incompletos

Outra contribuição na área de redes neurais intervalares é o trabalho intitulado *Handling uncertainty in neural networks: An Interval Approach* [Sim96].

Neste trabalho, os dados incompletos de um conjunto de dados são representados com intervalos. É proposto uma estrutura para uma unidade de processamento nas redes neurais chamada de célula intervalar incluindo a definição de sua função de ativação e propriedades dinâmicas do modelo. Em resumo, este trabalho tem dois objetivos: o algoritmo deve decrementar o total de incerteza da informação na rede o qual significa decrementar a largura dos intervalos dos pesos e a combinação dos intervalos para o qual introduze-se relações de ordem em $\mathbb{IR}$.

Embora neste trabalho seja definida a estrutura de uma rede neural intervalar de várias camadas, não existe uma definição específica de como esta rede poderia ser treinada. Isto é importante ressaltar porque o maior problema ao fazer uma reconstrução das redes neurais para intervalos é a definição dos algoritmos de aprendizado que podem ser aplicados.

4.2.3 Métodos de Trabalho com Dados Intervalares

Dado que as redes neurais tradicionais não têm a característica de trabalhar diretamente com dados intervalares, foram propostos métodos para dar um pré-processamento aos dados intervalares de tal forma que as redes neurais tradicionais consigam manipulá-los, um exemplo destes trabalhos é o artigo intitulado *Multilayer Perceptron on Interval Data* [FB02].

O objetivo principal deste artigo foi propor alternativas de solução quando se tem um problema que possui entradas intervalares, assim, são propostos vários métodos para processar estes dados e logo são comparados. O primeiro método é chamado de "Método de Valores Extremos", o qual é a forma mais simples onde cada intervalo de entrada na rede é transformado em um par de números reais tomando o ínfimo e o supremo de cada intervalo ou o valor médio e a distância entre os limites do intervalo, assim com n intervalos de entrada o conjunto de entradas agora tem $2n$ entradas de tipo real. Neste método não é considerado a filosofia dos intervalos, o que significa que o novo conjunto de entrada não necessariamente vai representar o problema a ser solucionado. Além da perda de informação no conjunto de dados não existe controle dos erros nos processos computacionais. Mas o autor comenta que uma melhor solução pode ser encontrada aplicando a matemática intervalar de Moore.

O segundo método é chamado de "Método Probabilístico", onde cada intervalo é visto como um conjunto de números onde cada um deles têm a mesma probabilidade de ser escolhido, então existe uma probabilidade uniforme dentro do intervalo, assim é escolhido um ponto qualquer ou um subconjunto de números de cada intervalo. Neste segundo método, os pontos que formam o conjunto de dados para treinar a rede neural não necessariamente representam o problema a ser solucionado e não existe garantia de que a solução encontrada seja correta além de multiplicar o tamanho do conjunto de treinamento sem ganho algum. Na comparação dos métodos é concluído que este tipo de dados podem reduzir a qualidade do resultado nas redes MLP. O problema pode ser ocasionado pela falta de representatividade do conjunto de treinamento usado.

4.2.4 Outros Algoritmos Reconstruídos para Intervalos

Nas redes neurais tradicionais existem vários métodos para treiná-las, um deles é o método de Newton o qual foi reconstruído para trabalhar com intervalos no artigo intitulado *On Interval Weighted Three-layer Neural Networks* [BBdK$^+$98].

Neste trabalho, igualmente aos demais foi apresentado o problema de se ter dados intervalares na entrada, mas aqui também existe uma preocupação pelos problemas ocasionados pelos erros no processamento computacional. Baseado no trabalho apresentado em [KS93] onde explica que os valores de entrada e saída de uma rede neural não necessariamente coincidem com a função sigmóide o problema é encontrar os valores dos pesos nas redes neurais como um sistema de equações da forma $F(X^j, W, V) = T^j$. Então é proposto o algoritmo *Interval Newton/Generalized bisection Algorithm.*

4.3 Redes Neurais Intervalares

Definição 4.3.1 *Toda rede neural é chamada de intervalar se alguns ou todos os seus conjuntos de entradas, saídas e pesos são valores intervalares.*

Assim uma rede neural intervalar é um modelo conexionista formado por unidades de processamento unidas por enlaces representados por intervalos. Portanto como nas redes neurais tradicionais, as redes neurais intervalares também podem ser classificadas pelo número de camadas, pela forma das conexões de seus neurônios, etc.

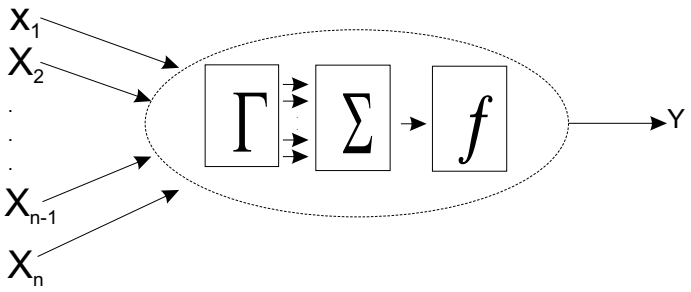

Figura 4.1: Neurônio Intervalar

4.3.1 Neurônio Intervalar

O neurônio intervalar é baseado no modelo do neurônio de McCulloch e Pitts proposto em [MP43]. O neurônio intervalar é a unidade de processamento das redes neurais artificiais intervalares (RNAI). Este neurônio intervalar recebe entradas intervalares ou reais e as processa obtendo uma saída. Este processamento é feito através de 3 funções:

- Função Normalizadora: Esta função analisa a natureza da entrada e normaliza os dados para intervalos.
- Função de Junção: Tem a mesma função de junção nos neurônios das redes tradicionais, compõe de forma linear os sinais de entrada pelos seus respectivos pesos sinápticos.
- Função de Ativação: São funções $\mathbb{IR} \rightarrow \mathbb{R}$, contínuas e deriváveis intervalarmente.

Um neurônio intervalar junta-se a outro neurônio para construir uma rede neural intervalar. O peso das conexões entre neurônios está definido por um número $W \in \mathbb{IR}$. A figura 4.1 apresenta a estrutura do neurônio intervalar baseada no neurônio de McCulloc-Pitts.

Capítulo 5

Redes Neurais Perceptron Intervalares Monocamada

5.1 Introdução

Neste capítulo apresentaremos as Redes Neurais Perceptron Intervalares (RPI) de uma camada com aprendizado Supervisionado. A importância deste tipo de redes é a necessidade atual de existirem ferramentas que estejam preparadas para trabalhar com dados intervalares além de garantir uma minimização dos erros nos processos computacionais das RNA.

Os trabalhos apresentados no capítulo anterior são uma amostra da preocupação de muitos pesquisadores em tentar solucionar através das redes neurais problemas que contenham dados intervalares garantindo computações com tolerância ao erro. Assim, esta dissertação tenta contribuir com a solução de parte dos problemas e limitações das redes neurais tradicionais para com dados intervalares. São propostas definições, estruturas e algoritmos para o que chamamos *Redes Neurais Intervalares* no caso específico redes neurais intervalares com aprendizado supervisionado baseadas nas redes neurais Perceptron.

5.2 Rede Neural Perceptron Intervalar de uma Camada

A Rede Perceptron de uma camada é a forma mais simples de uma RNA usada para classificação de padrões linearmente separáveis, ou seja padrões que estão em lados

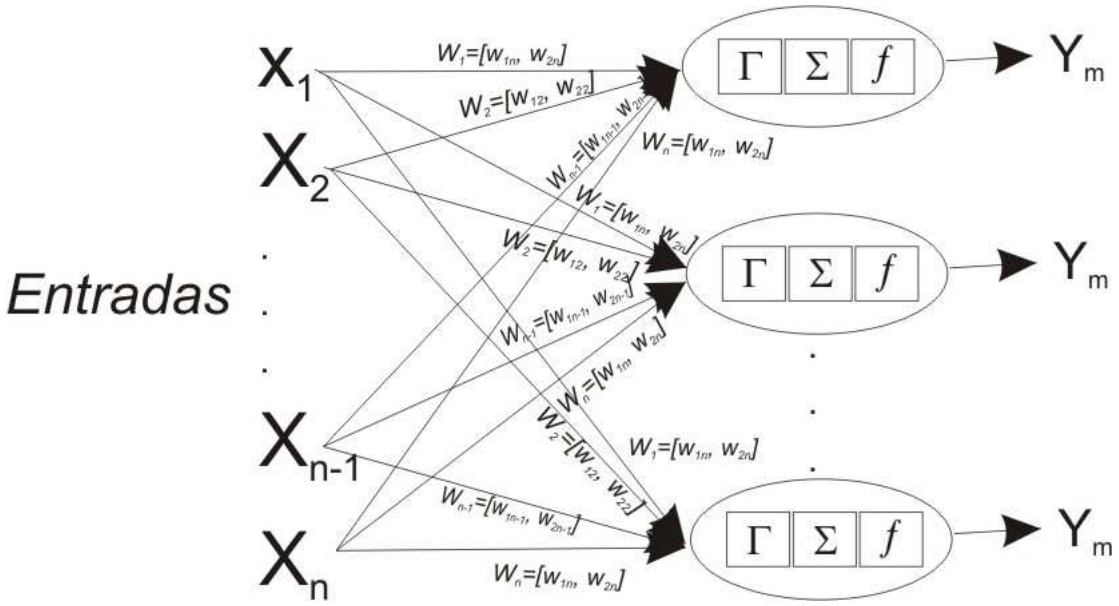

Figura 5.1: Rede Neural Perceptron Intervalar

opostos de um hiperplano. A forma mais básica consiste de um único neurônio com pesos sinápticos ajustáveis e uma polarização(*bias*).

A *Rede Neural Perceptron Intervalar* de uma camada é baseada na Perceptron de Rosemblatt e da mesma forma é a maneira mais simples de uma RNAI. Está formado por neurônios intervalares organizados em uma simples camada que processa as n entradas em m saídas, onde m é o número de neurônios na sua camada.

Uma rede Perceptron Intervalar com um único neurônio é limitado ao reconhecimento de duas classes, que podem ser codificadas como 0 e 1. Para este neurônio pode-se usar a função de ativação da equação 5.1.

O processamento de aprendizagem no Perceptron Intervalar é feito com continuas modificações de seus pesos e bias para se ajustar ao conjunto de treinamento. Assim o espaço dos intervalos é dividido pela separação das classes e cada intervalo pertence à classe onde está a maior área dele segundo a equação 5.1.

$$y = \begin{cases} +1 & \text{Se } \underline{x} + \overline{x} > 0, X_i = [\underline{x}, \overline{x}] \in \mathbb{IR}, \\ 0 & \text{Caso contrário.} \end{cases} \tag{5.1}$$

A rede Perceptron intervalar é apresentada na figura 5.1. Os neurônios estão interconectados, cada uma das interconexões entre neurônios é representado por um intervalo que contém o peso com que esses neurônios se relacionam.

5.3 Algoritmos de Aprendizagem na Rede Perceptron Intervalar de uma Camada

5.3.1 Algoritmo Base

A rede Perceptron Intervalar tem que passar por um treinamento para extrair conhecimento através de um conjunto de dados. Cada neurônio computa e envia sua saída que indicará qual neurônio será ativado.

Utilizando a matemática intervalar apresentada no capítulo 3 são obtidas as saídas e é feita a correção dos pesos baseado nos erros obtidos comparando a saída desejada com a saída da rede. O processo de aprendizado na rede Perceptron Intervalar termina quando o erro em todos os padrões é menor que um limiar. Na tabela 5.1 é descrito o algoritmo usado para treinar as redes Perceptron Intervalar de uma camada.

5.3.2 Algoritmo da Regra Delta Intervalar

A regra delta ou chamada também de algoritmo LMS(*Least Mean Square*) junto com o modelo Adaline foram propostos por Widroff e Hoff[WH60]. Este algoritmo é importante porque é a base do algoritmo chamado de regra delta generalizada usado para a correção dos pesos em redes neurais de múltiplas camada.

Igual às redes neurais tradicionais, as redes neurais intervalares de uma camada têm limitações em relação ao tipo de problemas que podem resolver, assim se faz importante reconstruir outros algoritmos para outras arquiteturas de redes intervalares. Um passo fundamental para chegar ao algoritmo de aprendizado das redes neurais de múltiplas camadas é a reconstrução do algoritmo da regra delta que será apresentado nesta seção.

A regra delta intervalar é aplicada a uma rede neural intervalar com uma arquitetura com uma única camada de neurônios. Este modelo mapea do espaço dos intervalos $\mathbb{IR}$ ao espaço dos reais $\mathbb{R}$ usando uma função de ativação como mostra a equação 5.2

$$y = f_n(V) = \breve{v} \tag{5.2}$$

O objetivo deste algoritmo é a minimização de uma função de custo dada pelo erro médio quadrático cometido no reconhecimento dos padrões intervalares apresentados à rede modificando os pesos das conexões com o método do gradiente.

Passo 1	Iniciar dos Pesos e Bias Iniciar o conjunto de pesos e bias com intervalos aleatórios pequenos para W_{ij} $(0 \leq i \leq N-1)$, θ_i e $(\overline{w}_{ij} - \underline{w}_{ij})$ é mínimo.
Passo 2	Apresentar um padrão de entrada $[X, d]$ X é o conjunto de dados de entrada que representa o padrão e d é um vetor binário que codifica a classe à qual pertence X.
Passo 3	Normalizar o conjunto de dados de entrada X para intervalos, o qual será: $X_i = ([\underline{x}_0, \overline{x}_0], [\underline{x}_1, \overline{x}_1], ..., [\underline{x}_{n-1}, \overline{x}_{n-1}])$.
Passo 4	Calcular a saída da rede para a entrada X $y(t) = f_n(\sum_{i=0}^{N-1} W_{ij}(t)X_i(t) - \theta_j)$ f_n é a função de ativação intervalar, o cálculo é feito pela equação 5.1
Passo 5	Adaptação dos Pesos $W_{ij}(t+1) = W_{ij}(t) + \eta[d(t) - y(t)]X_i$, onde $0 \leq i \leq N-1$ $d(t) = \begin{cases} +1 & \text{Se a entrada é da classe A,} \\ 0 & \text{Se a entrada é da classe B.} \end{cases}$ η é o coeficiente de aprendizado menor que 1 e $d(t)$ é a saída desejada para a atual saída. Se a rede faz a decisão correta, então os pesos não mudam.
Passo 6	Repetir desde o passo 2

Tabela 5.1: Algoritmo de Aprendizagem na Rede Perceptron Intervalar

A regra Delta intervalar é semelhante ao algoritmo do Perceptron Intervalar, esta regra também minimiza o erro cometido na saída da rede em relação aos valores desejados d_j pertencentes ao conjunto de treinamento. Assim a função de custo a ser minimizada é definida pela equação 5.3 onde p representa o número de neurônios na camada de saída, y_j é a saída da função de ativação em cada um dos neurônios dada pela equação 5.2. A função de ativação esta definida em base a V_j que representa a saída da função junção intervalar de cada um dos neurônios na camada de saída dada pela equação 5.4.

$$e = \frac{1}{2}\sum_{j=1}^{p}(d_j - y_j)^2 \tag{5.3}$$

$$V_j = \sum_{i=1}^{n} W_{ij}X_i + B_j \tag{5.4}$$

O gradiente negativo que indica a direção onde está o ponto mínimo da função de custo, é calculado usando a definição 3.6.3 do capítulo 3. Assim pela equação 5.3 a derivada em relação ao W_{ij} é a seguinte:

$$\Delta e = \frac{\partial e}{\partial W_{ij}} = \frac{\partial e}{\partial y_j}\frac{\partial y_j}{\partial W_{ij}}$$

então:

$$\frac{\partial e}{\partial y_j} = -(d_j - y_j)$$

logo, a equação 5.2 poderia ser reescrita como:

$$y_j = \frac{1}{2}\left[sup\left(\sum_{i=1}^{n} W_{ij}X_k + B_j\right) + inf\left(\sum_{i=1}^{n} W_{ij}X_i + B_j\right)\right]$$

Para o cálculo da função V_j em relação a um W_{ij} específico existem quatro possibilidades para o ínfimo do intervalo resultante e quatro para o supremo do intervalo resultante que são:

1. Para o ínfimo

$$\underline{v}_j = inf\left(\sum_{k=1}^{i-1} W_{kj}X_k + B_j\right) + (\underline{w}_{ij}\underline{x}_i + \underline{b}_j) + inf\left(\sum_{m=i+1}^{n} W_{mj}X_m + B_j\right), \tag{5.5}$$

$$\underline{v}_j = inf\left(\sum_{k=1}^{i-1} W_{kj}X_k + B_j\right) + (\overline{w}_{ij}\underline{x}_i + \underline{b}_j) + inf\left(\sum_{m=i+1}^{n} W_{mj}X_m + B_j\right), \tag{5.6}$$

$$\underline{v}_j = inf\left(\sum_{k=1}^{i-1} W_{kj}X_k + B_j\right) + (\underline{w}_{ij}\overline{x}_i + \underline{b}_j) + inf\left(\sum_{m=i+1}^{n} W_{mj}X_m + B_j\right), \quad (5.7)$$

$$\underline{v}_j = inf\left(\sum_{k=1}^{i-1} W_{kj}X_k + B_j\right) + (\overline{w}_{ij}\overline{x}_i + \underline{b}_j) + inf\left(\sum_{m=i+1}^{n} W_{mj}X_m + B_j\right). \quad (5.8)$$

2. Para o supremo

$$\overline{v}_j = sup\left(\sum_{k=1}^{i-1} W_{kj}X_k + B_j\right) + (\underline{w}_{ij}\underline{x}_i + \overline{b}_j) + sup\left(\sum_{m=i+1}^{n} W_{mj}X_m + B_j\right), \quad (5.9)$$

$$\overline{v}_j = sup\left(\sum_{k=1}^{i-1} W_{kj}X_k + B_j\right) + (\overline{w}_{ij}\underline{x}_i + \overline{b}_j) + sup\left(\sum_{m=i+1}^{n} W_{mj}X_m + B_j\right), \quad (5.10)$$

$$\overline{v}_j = sup\left(\sum_{k=1}^{i-1} W_{kj}X_k + B_j\right) + (\underline{w}_{ij}\overline{x}_i + \overline{b}_j) + sup\left(\sum_{m=i+1}^{n} W_{mj}X_m + B_j\right), \quad (5.11)$$

$$\overline{v}_j = sup\left(\sum_{k=1}^{i-1} W_{kj}X_k + B_j\right) + (\overline{w}_{ij}\overline{x}_i + \overline{b}_j) + sup\left(\sum_{m=i+1}^{n} W_{mj}X_m + B_j\right). \quad (5.12)$$

Assim, pela definição 3.6.3 o cálculo da derivada de y_j em relação a W_{ij} é dado pela equação 5.13.

$$\frac{\partial y_j}{\partial W_{ij}} = \frac{1}{2}\left(\frac{\partial \overline{v}_j}{\partial w} + \frac{\partial \underline{v}_j}{\partial w}\right) \quad (5.13)$$

onde $w \in W_{ij}$, $\underline{v}$ é escolhida entre as equações 5.5, 5.6, 5.7, 5.8 e $\overline{v}_j$ é dado por uma das equações 5.9, 5.10, 5.11 e 5.12 dependendo do sinal de W_{ij} e X_i.

Logo

$$\Delta e = -\frac{\partial e}{\partial W_{ij}} = -(d_j - y_j)\frac{\underline{v}'_j(w) + \overline{v}'_j(w)}{2} \quad (5.14)$$

O algoritmo para treinar uma rede neural intervalar de uma camada pelo algoritmo da regra delta intervalar é apresentado na tabela 5.2

5.4 Análise Experimental do Desempenho da Rede Perceptron Intervalar com o Algoritmo Base

Conforme a seção 2.6 do capítulo 2 o processo do treinamento de uma rede neural é extremamente dependente de diferentes fatores como a disponibilidade das características nos dados, parâmetros significativos no processo, etc. No processo de treinamento

1.	Inicializar todos os pesos W_{ij} e bias B_j com intervalos aleatórios pequenos para $(\overline{w}_{ij} - \underline{w}_{ij})$ é mínimo.
2.	Repetir - Apresentar um padrão de entrada $[X_k, d_k]$ X_k é um vetor de intervalos que representam o padrão k. d_k é a saída desejada e está formada por números reais - Calcular a saída da rede para a entrada X_k $y_j = f_n\left(\sum_{i=0}^{N-1} W_{ij}(t)X_i(t) + B_j\right)$ f_n é a função de ativação intervalar e o cálculo é feito pela equação 5.2. X_i é um elemento do padrão de entrada X_k e n é o número de entradas para cada padrão. - Calcular o erro cometido e_k para o padrão k com a equação 5.3 - Correção dos Pesos e bias $W_{ij}(t+1) = W_{ij}(t) - \eta\Delta e$, onde Δe é dado pela equação 5.14. η é o coeficiente de aprendizado menor que 1 e maior que 0.
3.	Até que $\frac{1}{M}\sum_{k=1}^{M} e_k$ seja minimizado.

Tabela 5.2: Algoritmo da Regra Delta Intervalar para Redes Neurais Intervalares de uma Camada

1.	Gerar n números reais aleatórios entre 0 e 1 que serão a base dos novos n números intervalares.
2.	Para cada número x gerado anteriormente, gerar 2 números aleatórios entre 0.1 e 0.111 que chamaremos de incrementos
3.	Logo, ao número x somar um dos incrementos e subtrair o outro incremento, obtendo assim um número intervalar.
4.	Repetir o processo para os n números.

Tabela 5.3: Processo para Gerar um conjunto de dados com n números Intervalares

da rede Perceptron Intervalar, o coeficiente de aprendizado e a tolerância do erro são alguns dos parâmetros que vão influenciar diretamente no desempenho da rede. Assim, o objetivo desta seção é fazer uma análise da influência destes parâmetros na rede Perceptron Intervalar comparando-os com a influência destes parâmetros na rede Perceptron pontual.

Para poder fazer uma análise, foram criados (tabela 5.3) 80 conjuntos de dados intervalares experimentais linearmente separáveis para treinar uma rede neural intervalar com um só neurônio de saída. Dado que a rede Perceptron tradicional não consegue trabalhar com estes dados intervalares, foi usado o método probabilístico proposto em [FB02] (onde explica que para trabalhar com as redes neurais tradicionais com dados intervalares, escolhe-se um representante de cada intervalo então é formado um conjunto de números reais em base ao conjunto de números intervalares). Assim o conjunto de dados reais é igualmente separável e será usado para treinar uma rede neural pontual com uma estrutura equivalente a estrutura da rede Perceptron Intervalar.

Os conjuntos de dados (intervalar e real) utilizados para o analise, tem 90 padrões que foram distribuídos em sessenta padrões para o treinamento e trinta para o teste. Em cada conjunto de dados, um padrão será formado por um vetor com duas entradas, e uma saída que pode ser 1 ou 0 tanto para os dados intervalares como para os dados reais.

A rede Perceptron tradicional foi treinada e testada 100 vezes (modificando a inicialização dos pesos) com cada um dos conjuntos de dados reais para uma tolerância do erro $\epsilon = 0$ e coeficiente de aprendizado $\eta = 0.2$ modificando a inicialização dos pesos dado que é aleatória. A figura 5.2 apresenta a distribuição do número de épocas para a rede Perceptron intervalar e para a Perceptron tradicional. Logo, foi visto que para esta configuração e com $\eta = 0.2$, em média o número de épocas para a rede perceptron intervalar foi menor em comparação com a media do número de épocas da rede perceptron tradicional. Para continuar com o estudo foram treinadas e testadas as duas redes modificando o η desde 0.3 até 0.7.

Pelas experiências feitas, a rede neural perceptron intervalar precisou de menor número de épocas e o erro no reconhecimento foi similar com o erro na rede perceptron tradicional. Assim, o melhor resultado foi obtido com o coeficiente de aprendizado de $\eta = 0.3$, onde a média do número de épocas necessário para treinar a rede Perceptron Intervalar foi de 17.2870 enquanto que para a rede Perceptron tradicional foi 26.7704. O desvio padrão para a rede Perceptron Intervalar foi de 50.1692 e para a rede pontual foi de 80.7051. A figura 5.3 apresenta a distribuição do número de épocas para a rede Perceptron intervalar e para a Perceptron tradicional com $\eta = 0.3$.

Usando os conjuntos de teste, a media do percentagem do erro na classificação da

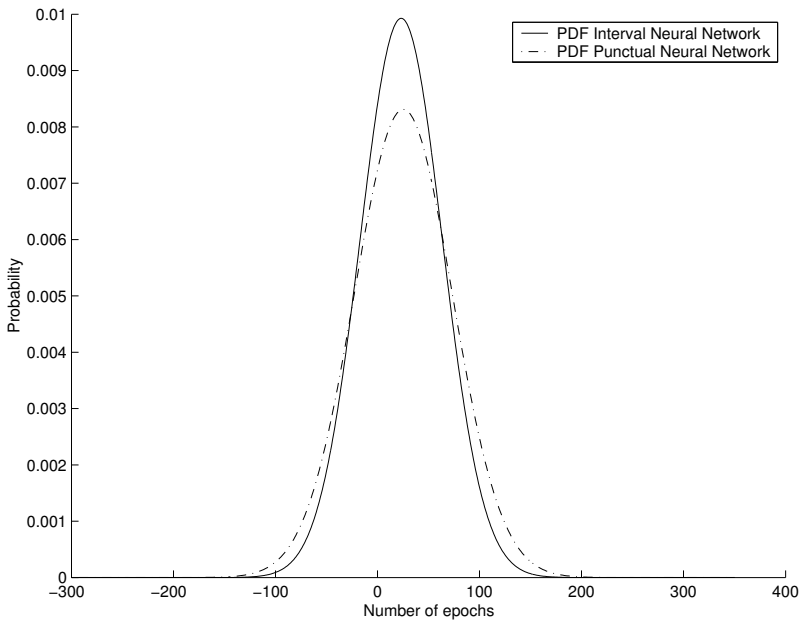

Figura 5.2: Função de Densidade de Probabilidade para o Número de Épocas com $\eta = 0.2$

rede Perceptron Intervalar com os 80 conjuntos de dados intervalares foi 2.3062% com $\eta = 0.3$ e para a rede Perceptron tradicional com seus respectivos conjuntos de dados foi de 2.3004% com o mesmo coeficiente, assim é visto que o número de épocas é menor na rede perceptron intervalar e o erro no reconhecimento é semelhante ao erro na rede perceptron tradicional.

Na tabela 5.4, são apresentados os resultados comparativos da média do número de épocas que a rede neural intervalar e a rede pontual precisam para solucionar um problema com os dados experimentais junto com a percentagem do erro.

As experiências anteriores foram feitas variando o coeficiente de aprendizado (η) até 0.7. Com um $\eta = 0.8$ a rede Perceptron Intervalar e a rede Perceptron pontual não conseguiam convergir.

As figuras 5.3, 5.4 5.5, 5.6 e 5.7 mostram a função de densidade de probabilidade para o número de épocas requeridos pela rede Perceptron Intervalar em comparação com a rede Perceptron pontual para o coeficiente de aprendizado $\eta = 0.3$, $\eta = 0.4$, $\eta = 0.5$, $\eta = 0.6$, $\eta = 0.7$ e com uma tolerância ao erro $\epsilon = 0$.

O coeficiente de aprendizado (η) faz com que as RNAI venham a convergir mais rápido ou mais devagar, dependendo do valor. O erro mínimo aceitável também vai influenciar no treinamento da rede para decidir quando deve parar o processo. Por isto, estes parâmetros são importantes para avaliar o comportamento da rede. Novas experiências foram repetidas mudando a tolerância do erro e fixando o coeficiente de

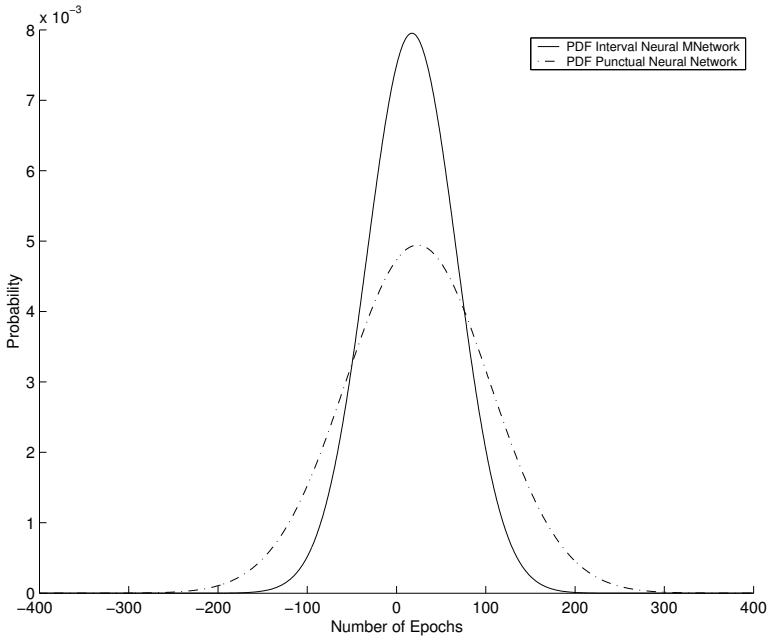

Figura 5.3: Função de Densidade de Probabilidade para o Número de Épocas com $\eta = 0.3$

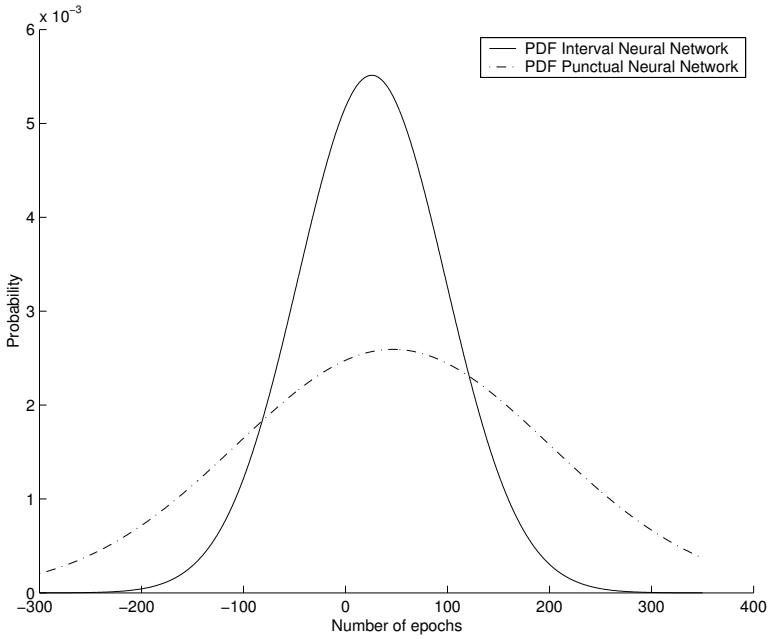

Figura 5.4: Função de Densidade de Probabilidade para o Número de Épocas com $\eta = 0.4$

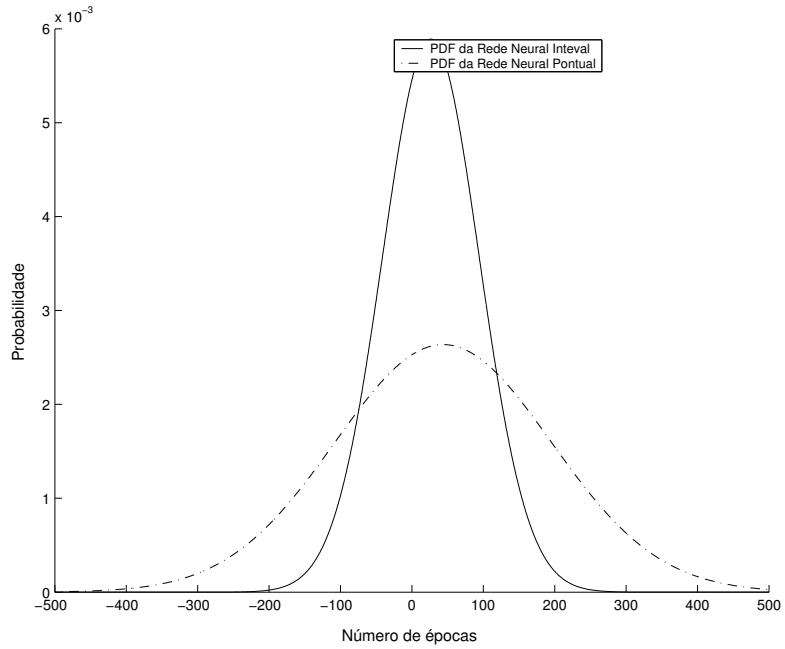

Figura 5.5: Função de Densidade de Probabilidade para o Número de Épocas com $\eta = 0.5$

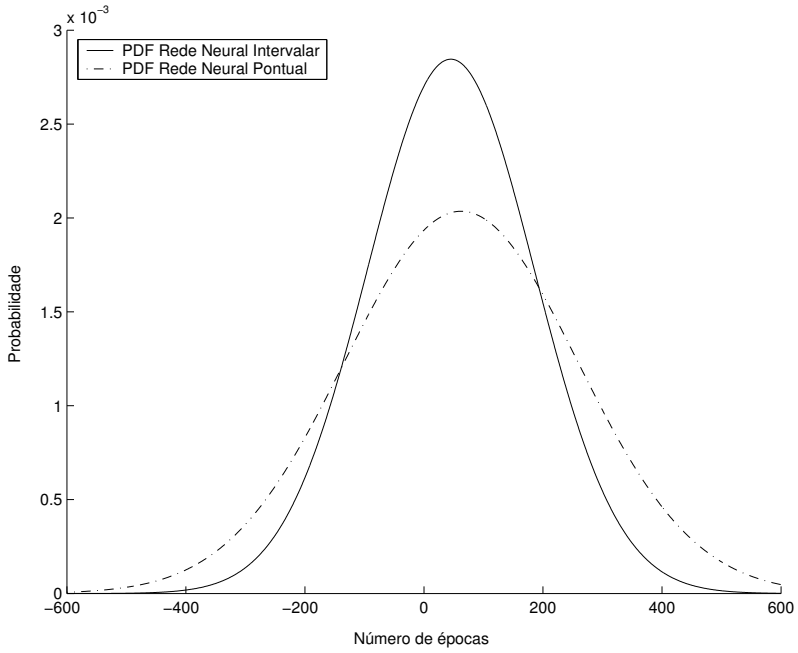

Figura 5.6: Função de Densidade de Probabilidade para o Número de Épocas com $\eta = 0.6$

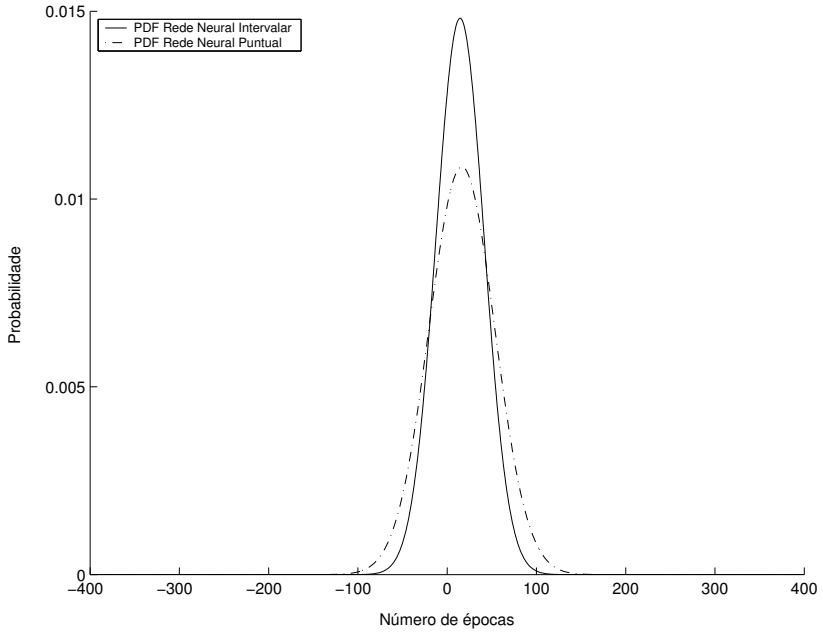

Figura 5.7: Função de Densidade de Probabilidade para o Número de Épocas com $\eta = 0.7$

η	Percep Intervalar.		Perceptron Tradicional	
	Núm. Épocas	% de Erro	Núm. Épocas	% de Erro
0.2	23.2479	1.1012	25.298	1.0208
0.3	17.2870	2.3062	26.7704	2.3004
0.4	25.7477	2.4551	46.6102	2.4309
0.5	26.5243	2.5572	43.781	2.4938
0.6	45.1708	2.5325	62.4529	2.6423
0.7	31.8649	2.4478	46.2234	2.4684

Tabela 5.4: Rede Perceptron Intervalar vs. Rede Perceptron Pontual com η Variável e $\epsilon = 0$.

		$\epsilon = 0$	$\epsilon = 1$	$\epsilon = 2$
Intervalar	Número de Épocas	17.2870	36.2355	8.3851
	Percentagem de Erro	2.3062%	2.0991%	4.3795%
Pontual	Número de Épocas	26.7704	46.1286	8.3179
	Percentagem de Erro	2.3004%	2.1431%	4.42%

Tabela 5.5: Rede Perceptron Intervalar vs. Rede Perceptron Pontual com ϵ Variável e $\eta = 0.3$.

aprendizado. Na tabela 5.5 são mostrados os resultados do treinamento da RNAI de uma camada comparando com uma rede pontual com a mesma arquitetura variando o erro mínimo e com $\eta = 0.3$ fixo.

Pelas experiências mostradas nas figuras 5.2, 5.3, 5.4, 5.5, 5.6 e 5.7 é visto que a influência do coeficiente de aprendizado(η) nas redes Perceptron Intervalar é maior que nas redes Perceptron pontuais. Assim, o número de épocas necessário para que a rede Perceptron Intervalar convergia é menor que nas redes Perceptron pontuais com similar erro no reconhecimento dos padrões.

Para uma visualização melhor dos testes feitos até este ponto, na figura 5.8 é mostrado o resumo da tabela 5.4, onde vemos que existe uma tendência do crescimento no número de épocas que as redes neurais Perceptron pontual e intervalar precisam para convergir, mas o número de épocas da rede neural Perceptron Intervalar é sempre menor que nas redes pontuais com uma percentagem de efetividade similar. Na figura 5.9 apresenta a tendência da percentagem do erro no reconhecimento variando o coeficiente de aprendizado.

5.5 Análise Experimental com o Algoritmo da Regra Delta Intervalar

Para testar o algoritmo proposto da regra delta intervalar, foram feitos testes similares aos testes feitos com o algoritmo de treinamento da Perceptron Intervalar. Assim novamente foram gerados 80 conjuntos aleatórios de dados intervalares para os quais testou-se o algoritmo. Cada conjunto tem 90 padrões de dados, dos quais foram usados 65% para treinamento e 35% para teste. Para poder treinar uma rede neural com o algoritmo da regra delta tradicional, foram obtidos 80 conjuntos de dados reais desde os conjuntos de dados intervalares com o método probabilístico.

Os resultados dos testes feitos com este algoritmo da regra delta intervalar são mostrados na tabela 5.6. A rede neural intervalar foi treinada com o algoritmo delta intervalar e com os conjuntos de dados propostos, logo a rede neural pontual de similar

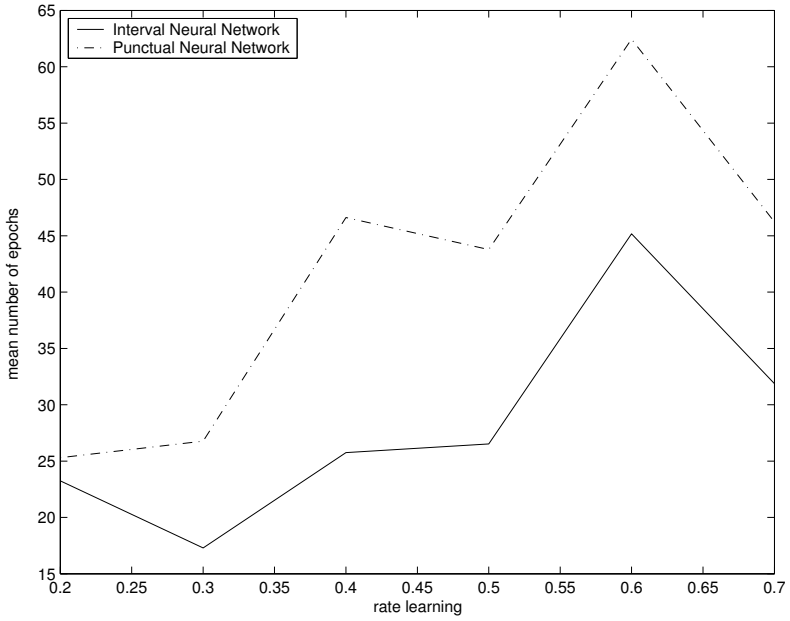

Figura 5.8: Estimativa do Número de Épocas com Respeito ao Coeficiente de Aprendizado

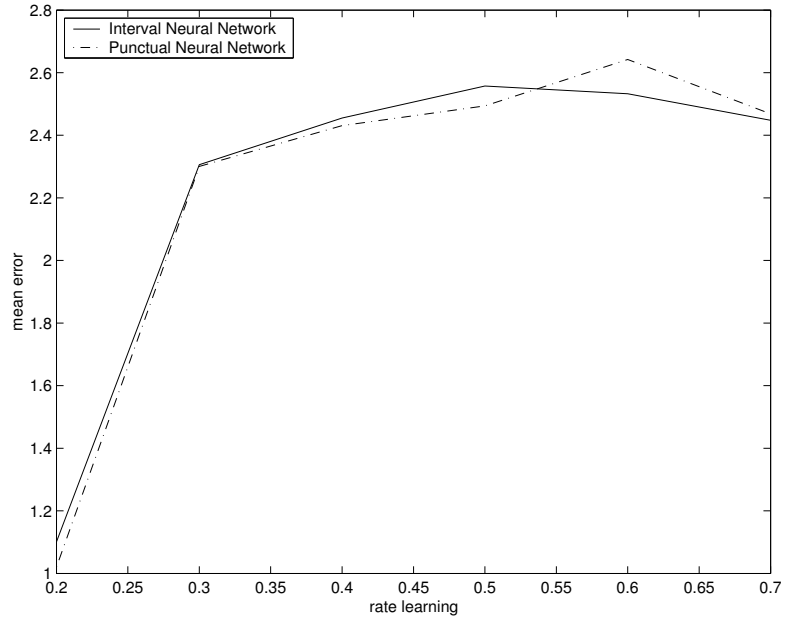

Figura 5.9: Estimativa da Percentagem de Erro para o Coeficiente Variável

η	R. Delta Intervalar.		R. Delta Pontual	
	Núm. Épocas	% de Erro	Núm. Épocas	% de Erro
0.2	4.3477	0.0048	4.3071	0.0041
0.3	3.1708	0.0028	3.1369	0.0018
0.4	3.2063	0.0027	3.1714	0.0018
0.5	2.4986	0.0015	2.4996	0.0030
0.6	2.2544	0.0010	2.2553	0.0020
0.7	3.6574	0.0020	3.6501	0.0010

Tabela 5.6: Regra Delta Intervalar vs. Regra Delta Pontual com η Variável e $\epsilon = 0.2$.

arquitetura foi treinada com a regra delta pontual também foi treinada com os conjuntos de dados reais propostos. As redes foram treinadas cem vezes com cada conjunto de dados (real ou intervalar, dependendo da rede) com um coeficiente de aprendizado de $\eta = 0.2$, $\eta = 0.3$, $\eta = 0.4$, $\eta = 0.5$, $\eta = 0.6$ e $\eta = 0.7$. A figura 5.10 é um resumo dos testes feitos.

Dado um conjunto de pontos, quando uma rede neural de uma camada é treinada como o algoritmo LMS tradicional, ela tenta traçar uma reta que contenha o maior número de pontos dados. Esta idéia não é simples quando se tem um conjunto de dados intervalares porque cada intervalo é um grupo de muitos pontos então com certeza o algoritmo LMS intervalar sempre vai deixar alguns pontos intervalares de fora. Mesmo assim, pelos testes feitos, o número de épocas para o treinamento da rede neural intervalar é igual ao número de épocas da rede neural tradicional mas o erro cometido na rede neural intervalar é maior. Este fato é intuitivo porque não existe como desenhar uma reta entre intervalos sem deixar pelo menos um ponto dentro de algum intervalo de fora. Embora a rede neural intervalar treinada com o algoritmo LMS apresente um erro maior nos testes, este algoritmo é importante porque a rede neural intervalar pode extrair conhecimento do conjunto de dados, resultado que será importante para a generalização deste algoritmo para redes neurais de múltiplas camadas.

A figura 5.10 apresenta um resumo do número de épocas que precisa uma rede neural intervalar de uma camada treinada com o algoritmo da regra delta intervalar junto com o número de épocas que precisa uma rede neural tradicional treinada com a regra delta e a figura 5.11 apresenta a média da percentagem dos erros cometidos nos conjuntos de teste apresentados à rede neural intervalar e a rede neural tradicional com diferentes coeficientes de aprendizado.

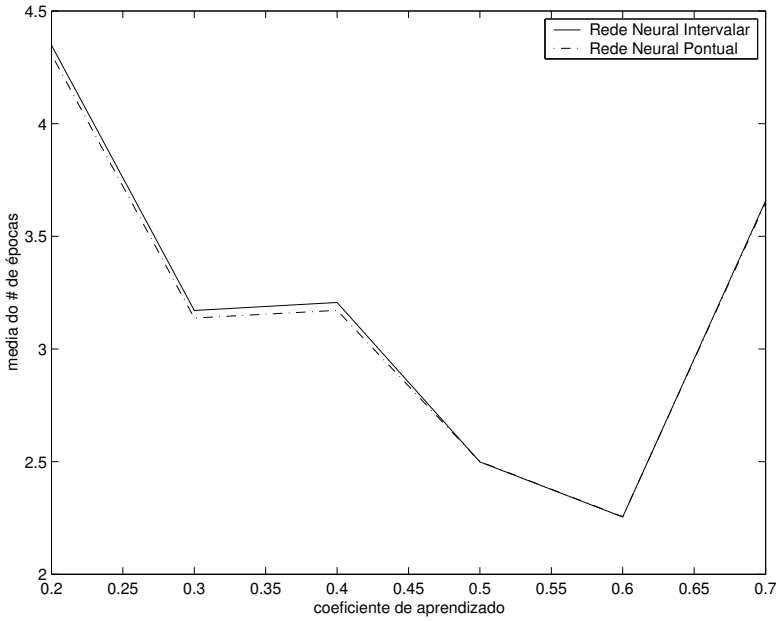

Figura 5.10: Estimativa do Número de Épocas com Respeito ao Coeficiente de Aprendizado

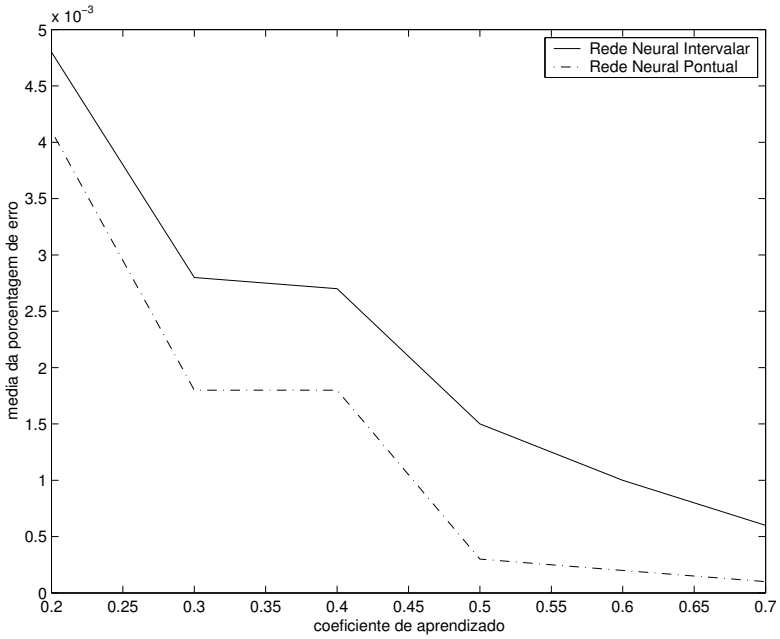

Figura 5.11: Estimativa do Percentagem de Erro para o Coeficiente Variável

5.6 Propriedades de Aprendizagem

Pelas experiências feitas com a rede neural Perceptron Intervalar de uma camada podemos supor que a inclusão da matemática intervalar no processo de aprendizagem da rede neural, traz consigo vantagens além do controle dos erros.

A rede Perceptron Intervalar avaliada em vários conjuntos de dados, variando os parâmetros do treinamento precisou, em média, de um número menor de épocas para treinar, comparando com uma rede Perceptron pontual com a mesma arquitetura. É visto também, que ela não perde o poder de generalização nos dados.

5.7 Separação das Classes pelo Perceptron Intervalar

Depois da convergência do algoritmo de aprendizado da rede Perceptron tradicional, a rede posicionará uma superfície de decisão na forma de um hiperplano entre as duas classes, se fossem só duas entradas este hiperplano seria uma reta.

Depois da convergência do Perceptron Intervalar, espera-se estabelecer uma forma de separação de classes similar à rede perceptron tradicional, assim podemos formular duas perguntas, a primeira é: Onde esta a separação destas classes intervalares?. Se os pesos são quem definem o hiperplano de separação de classes e os pesos na RPI são intervalos, pode-se esperar que a rede intervalar com duas entradas forme uma região de decisão e não uma reta, mas quem define esta região?.

Uma resposta intuitiva para as perguntas formuladas pode ser que a rede Perceptron Intervalar faça uma divisão das classes por uma reta formada pelos ínfimos dos pesos e outra formada pelos supremos dos pesos, tendo assim a região procurada.

Para responder às duas perguntas anteriores foi feita uma análise desta situação. Será apresentado um problema de classificação de duas classes à rede Perceptron Intervalar com duas entradas e uma saída. O conjunto de dados tem 90 padrões, que serão divididos 65% para o treinamento e 35% para o teste e com uma taxa de aprendizado ($\eta = 0.3$). Logo que foi treinada a rede, foi apresentado o conjunto de teste, onde todos os padrões foram reconhecidos corretamente então a rede conseguiu posicionar a região que separa estas classes. Se nossa resposta intuitiva anterior estivesse certa poderíamos fazer o gráfico da reta formada pelos ínfimos e a reta formada pelos supremos para mostrar os resultados da classificação. A figura 5.12 apresenta o conjunto de dados com o qual foi treinada e testada a rede junto com as retas dos ínfimos e supremos dos pesos que teriam que formar a região de decisão da rede e separar corretamente as classes.

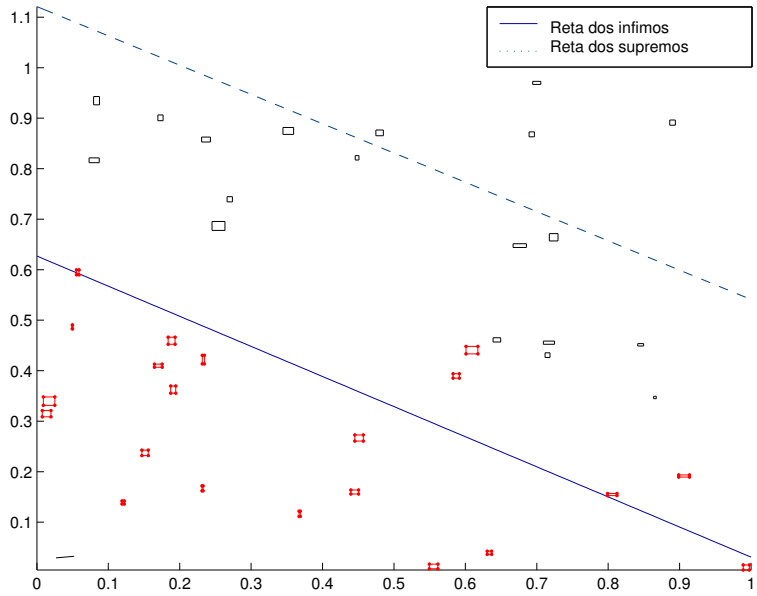

Figura 5.12: Conjunto de Dados e Retas dos Ínfimos e Supremos dos Pesos

Analisando a figura 5.12 vemos que se a rede classificou bem todos os padrões do conjunto de teste, então estas duas retas não poderiam ser as que formam a região de decisão porque pela posição das retas, a rede não conseguiu separar as classes.

Logo, para responder as perguntas anteriormente formuladas foram feitos vários testes com diferentes conjuntos de dados aleatórios sempre linearmente separáveis. Foram apresentados 80 conjuntos de dados diferentes onde os intervalos que formam o conjunto de dados não são necessariamente simétricos e treinada a rede Perceptron Intervalar com a mesma configuração do primer conjunto de dados proposto. Depois deste teste foi visto que a reta formada pelos pontos médios dos pesos e bias é uma referência de onde está a região de separação das classes na rede Perceptron Intervalar. A figura 5.13 mostra um dos conjuntos que foi usado para treinar e testar a rede junto com a reta formada pelos pontos médios dos pesos e bias.

Na figura 5.13 vemos que a reta dos pontos médios dos intervalos representaria uma divisão das classes mas ainda não sabemos se só esta reta é que define a separação ou se é uma região perto desta reta. Para isto foram feitos novos testes com os conjuntos de dados anteriores. Depois de treinada a rede com um dos conjuntos anteriormente descritos e situada a reta dos pontos médios, foi gerado um novo conjunto de testes com intervalos situados perto da reta dos pontos médios.

Cada padrão no novo conjunto de teste é da forma: (X, Y) onde $X = [x, x]$ um intervalo degenerado e $Y = [\underline{y}, \overline{y}]$. Assim a representação de cada um destes intervalos pode ser vista em $\mathbb{R}^2$ como uma pequena linha vertical (poderia também se escolher Y como um degenerado ou que nenhum dos intervalos X, Y sejam degenerados). A

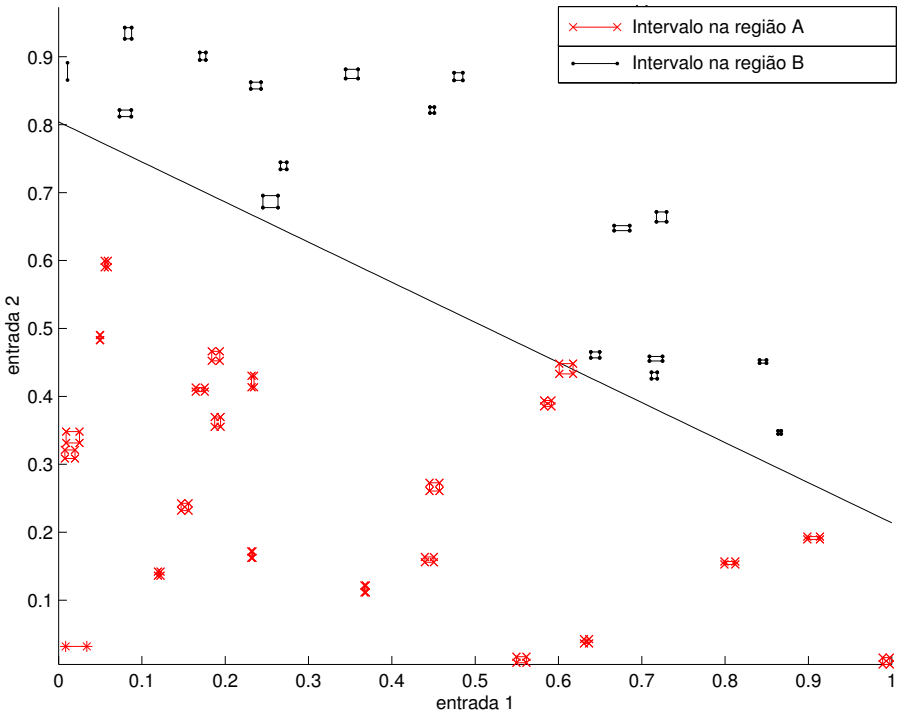

Figura 5.13: Conjunto de Dados Intervalar e Reta dos Pontos Médios dos Pesos

idéia é ter algum intervalo que situado próximo da reta de separação das classes ou que intercepte a reta. A figura 5.15 e a figura 5.14 apresentam duas situações observadas nos novos testes: a primeira situação é que nenhum dos intervalos do conjunto de teste intercepta a reta, a segunda situação é que alguns dos intervalos do conjunto de teste intercepta a reta dos pontos médios.

Na situação da figura 5.14 vemos que todos os padrões do novo conjunto de teste foram classificados corretamente então poderíamos inferir que a separação das classes é dada pela reta dos pontos médios dos pesos. Porém, observando a figura 5.15 vemos que alguns intervalos que segundo a reta tinham que ser classificados de uma classe, foram classificados da outra classe então esta reta não necessariamente representa a separação das classes, pelo que concluímos que existe algum fator mais que decide na classificação ou em caso contrário que esta reta não mostra a separação de classes da rede Perceptron Intervalar. Para abordar este novo questionamento foi feito uma análise matemática do processo da rede com o que podemos enunciar o teorema a seguir.

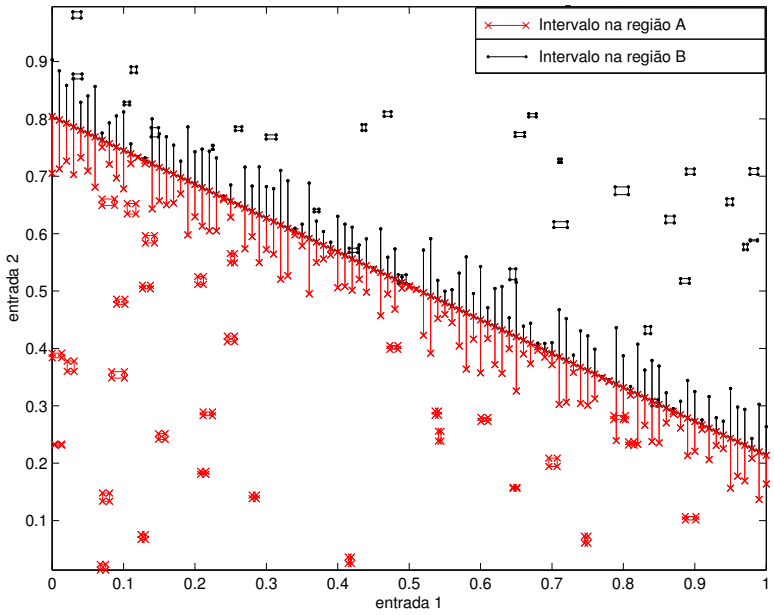

Figura 5.14: Conjunto de Teste próximo da Reta dos Pontos Médios dos Pesos onde Nenhum Intervalo Intercepta a Reta

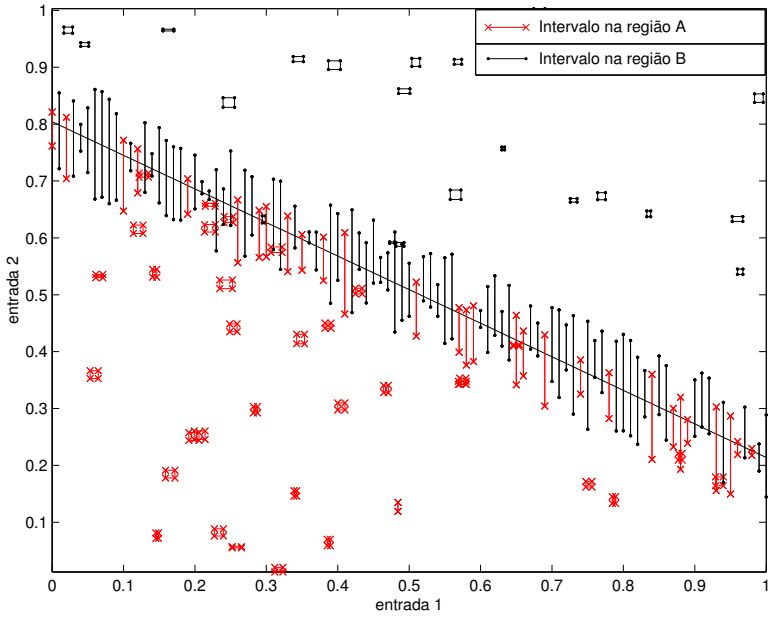

Figura 5.15: Conjunto de Teste próximo da Reta dos Pontos Médios dos Pesos onde Alguns Intervalos Interceptam a Reta

Teorema 5.7.1 *Separação das Classes*

Dado uma rede Perceptron Intervalar com um único neurônio e duas entradas. W_1, W_2 *são pesos intervalares,* B *é o bias intervalar,* S *é a saída da função junção do único neurônio e* X_1, X_2 *as entradas para a rede Perceptron intervalar. Depois que o algoritmo de aprendizado para a rede Perceptron Intervalar converge, é posicionada uma região de decisão a qual tem esta situada perto da reta formada pelos pontos médios das conexões intervalares (pesos) e das entradas intervalares. Assim pode-se dizer que a reta é uma referência da posição da região de decisão. A equação desta reta é dada por:*

$$\breve{w}_1\breve{x}_1 + \breve{w}_2\breve{x}_2 + \breve{b} + C$$

onde para qualquer intervalo K, $\breve{k}$ *representa o ponto médio do intervalo e* C *é uma variação que depende do sinal dos intervalos. Esta variação pode ser uma das seguintes opções:*

$$\pm\left(\frac{\Delta_1(\underline{x}_1 - \overline{x}_1)}{2} + \frac{\Delta_2(\underline{x}_2 - \overline{x}_2)}{2}\right), \tag{5.15}$$

$$\pm\left(\frac{\Delta_2(\underline{x}_2 - \overline{x}_2)}{2} - \frac{\Delta_1(\underline{x}_1 - \overline{x}_1)}{2}\right), \tag{5.16}$$

$$\pm\left(\Delta_1\breve{x}_1 - \frac{\Delta_2(\underline{x}_2 - \overline{x}_2)}{2}\right), \tag{5.17}$$

$$\pm\left(\Delta_1\breve{x}_1 + \frac{\Delta_2(\underline{x}_2 - \overline{x}_2)}{2}\right), \tag{5.18}$$

$$\pm\left(\frac{\Delta_1(\underline{x}_1 - \overline{x}_1)}{2} + \Delta_2\breve{x}_2\right), \tag{5.19}$$

$$\pm\left(\frac{\Delta_1(\underline{x}_1 - \overline{x}_1)}{2} - \Delta_2\breve{x}_2\right), \tag{5.20}$$

$$\pm(\Delta_1\breve{x}_1 + \Delta_2\breve{x}_2), \tag{5.21}$$

$$\pm(\Delta_1\breve{x}_1 - \Delta_2\breve{x}_2). \tag{5.22}$$

Se $sign(W_1) = 0$ *então uma das quatro equações a seguir será escolhida como referencia da separação das classes segundo* $sign(X_1)$ *é 1, -1 ou 0.*

$$\breve{w}_1\overline{x}_1 + \breve{w}_2\breve{x}_2 + \breve{b} \pm \left(\frac{\Delta_2(\underline{x}_2 - \overline{x}_2)}{2}\right), \tag{5.23}$$

$$\breve{w}_1\underline{x}_1 + \breve{w}_2\breve{x}_2 + \breve{b} \pm \left(\frac{\Delta_2(\underline{x}_2 - \overline{x}_2)}{2}\right), \tag{5.24}$$

$$\breve{w}_1\overline{x}_1 + \breve{w}_2\breve{x}_2 + \breve{b} \pm \Delta_2\breve{x}_2, \tag{5.25}$$

$$\breve{w}_1\underline{x}_1 + \breve{w}_2\breve{x}_2 + \breve{b} \pm \Delta_2\breve{x}_2. \tag{5.26}$$

Se $sign(W_2) = 0$ *então a reta da separação das classes poderá ser:*

$$\breve{w}_1\breve{x}_1 + \breve{w}_2\underline{x}_2 + \breve{b} \pm \left(\frac{\Delta_1(\underline{x}_1 - \overline{x}_1)}{2}\right), \tag{5.27}$$

$$\breve{w}_1\breve{x}_1 + \breve{w}_2\overline{x}_2 + \breve{b} \pm \left(\frac{\Delta_1(\underline{x}_1 - \overline{x}_1)}{2}\right), \tag{5.28}$$

$$\breve{w}_1\breve{x}_1 + \breve{w}_2\underline{x}_2 + \breve{b} \pm \Delta_1\breve{x}_1, \tag{5.29}$$

$$\breve{w}_1\breve{x}_1 + \breve{w}_2\overline{x}_2 + \breve{b} \pm \Delta_1\breve{x}_1. \tag{5.30}$$

Se $sign(W_1) = 0$ *e* $sign(W_2) = 0$ *as equações anteriores vão se combinar.*
Proof: A prova é apresentada no apêndice A

Assim, nesta seção podemos concluir que a rede Perceptron Intervalar situa uma região de separação das classes próxima à reta formada pelos pontos médios dos pesos intervalares. Além disso, esta região vai depender muito da largura dos intervalos das entradas à rede. Dado que a largura dos intervalos representa imprecisão, a rede perceptron intervalar é um reflexo dessa imprecisão dado que em quanto maior imprecisão nos dados, maior será a separação das classes intervalares.

Capítulo 6

Redes Neurais Perceptron Intervalares de Múltiplas camadas

6.1 Introdução

Igual como nas redes neurais tradicionais, as redes neurais intervalares de uma camada estão limitadas em quanto ao tipo de problemas que podem solucionar, assim, se faz necessário propor as redes neurais intervalares de múltiplas camadas. Neste capítulo apresentaremos as Redes Neurais Perceptron Intervalares (RPI) de múltiplas camadas com aprendizado Supervisionado.

Em alguns dos trabalhos apresentados no capítulo 4 já são vistas as redes intervalares de múltiplas camadas. Neste capitulo tenta se propor uma rede perceptron intervalar de múltiplas camadas diferente das propostas até agora existentes e se fará uma analise do treinamento e comportamento desta rede.

6.2 Rede Neural Intervalar Perceptron de Múltiplas Camadas

Da mesma forma que nas redes neurais tradicionais Perceptron de uma camada as redes neurais Perceptron intervalar de uma camada só conseguem modelar problemas linearmente separáveis, esta foi uma limitação nas redes neurais tradicionais e é uma limitação nas redes neurais intervalares. Assim, é necessário reconstruir o modelo tradicional para as *Redes Neurais Perceptron Intervalar* de múltiplas camadas.

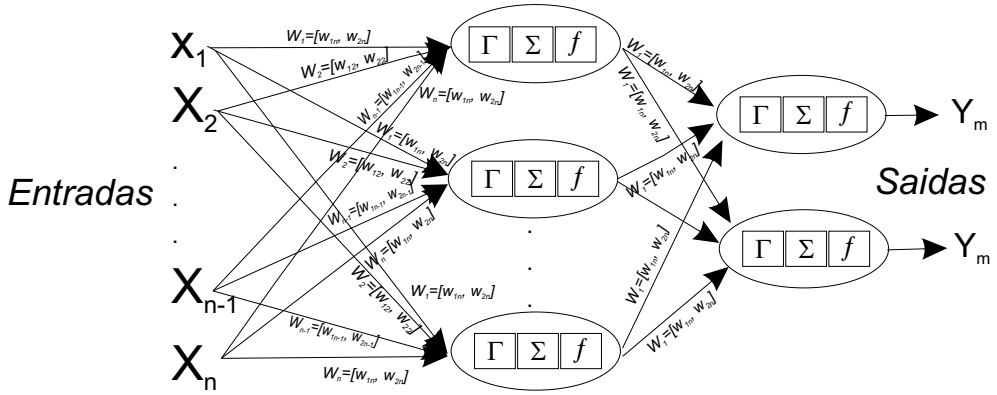

Figura 6.1: Rede Neural Perceptron Intervalar com uma Camada Escondida

A Rede Neural Perceptron Intervalar de Múltiplas Camadas tem como base a rede Perceptron Intervalar de uma camada. Assim, esta rede está formada por várias camadas onde são ordenados os neurônios intervalares e conectados aos neurônios da seguinte camada. Os neurônios de uma camada processam a informação dada pelas suas entradas e enviam uma resposta que será a entrada para a camada seguinte ou será a saída da rede dependendo se estamos na última camada ou em camadas prévias. A figura 6.1 é um exemplo da arquitetura da rede Perceptron Intervalar com uma camada oculta.

A rede neural intervalar Perceptron de múltiplas camadas tem nas suas entradas um conjunto de dados intervalares ou não intervalares que representam características de algum objeto e na sua saída um conjunto de números reais correspondentes aos dados de entrada. Esta rede está formada por neurônios intervalares os quais fazem um mapeamento camada a camada de $\mathbb{IR} \rightarrow \mathbb{R}$, assim alguns exemplos de funções de ativação que podem ser usadas nestes neurônios são função descrita pela equação 5.2 que usa saídas reais (vista como uma função linear) e a função dada pela equação 6.1 que limita as saídas da rede entre 0 e 1.

$$y_k(V_k) = \frac{1}{1 + e^{-\check{v}_k}} \tag{6.1}$$

O treinamento desta rede intervalar de múltiplas camadas é feito com a modificação dos pesos por sucessivas atualizações a fim de minimizar uma superfície do erro. Assim, o algoritmo reconstruído para o treinamento desta rede neural intervalar é a regra delta intervalar generalizada que é descrita na seção seguinte.

6.3 Regra Delta Intervalar Generalizada

A regra delta generalizada, também conhecida por algoritmo de retro-propagação [dABdO00] foi usada para treinar redes neurais de múltiplas camadas, baseada em este algoritmo, outros algoritmos foram derivados tentando superar algumas deficiências do algoritmo base. A regra delta tradicional foi reconstruida para poder treinar uma rede neural intervalar de uma camada, assim em base a este algoritmo, também foi reconstruido o algoritmo de regra delta generalizada para poder treinar uma rede neural intervalar de múltiplas camadas com aprendizado supervisionado, desta forma a regra delta generalizada intervalar tenta minimizar uma função de custo definida com base nas saídas da rede neural Perceptron Intervalar comparando-as com as saídas desejadas dado que estas redes têm um aprendizado supervisionado.

A equação da função de custo a minimizar é dada pela equação 6.2.

$$E = \sum_k e_k = \frac{1}{2}\sum_k (d_k - y_k)^2 \tag{6.2}$$

onde d_k representa a saída desejada para o padrão apresentado, y_k representa a saída do neurônio k na camada de saída. O gradiente negativo que indica a direção onde está o ponto mínimo da função de custo, é calculado usando a definição 3.6.3 do capítulo 3. Então pela equação 6.2 a derivada em relação ao peso W_{ij} que é o peso da conexão entre o neurônio i na camada n e o neurônio j na camada $n+1$ é:

$$\Delta e = \frac{\partial e}{\partial \breve{v}_j}\frac{\partial \breve{v}_j}{\partial W_{ij}} \tag{6.3}$$

Logo, para calcular o segundo fator da equação 6.3, temos que:

$$\breve{v}_j = \frac{1}{2}\left[sup\left(\sum_{i=1}^{n} W_{ij}X_i + B_j\right) + inf\left(\sum_{i=1}^{n} W_{ij}X_i + B_j\right)\right]$$

Para identificar o $\breve{v}_j$ é necessário ver o cálculo de V_j em relação a um W_{ij}, então V_j será o intervalo onde o ínfimo é escolhido entre as equações 6.4, 6.5, 6.6 e 6.7, da mesma forma o supremo do intervalo é escolhido entre as equações 6.8, 6.9, 6.10 e 6.11.

1. Para o ínfimo

$$\underline{v}_j = inf\left(\sum_{k=1}^{i-1} W_{kj}X_k + B_j\right) + (\underline{w}_{ij}\underline{x}_i + \underline{b}_j) + inf\left(\sum_{m=i+1}^{n} W_{mj}X_m + B_j\right) \tag{6.4}$$

$$\underline{v}_j = inf\left(\sum_{k=1}^{i-1} W_{kj}X_k + B_j\right) + (\overline{w}_{ij}\underline{x}_i + \underline{b}_j) + inf\left(\sum_{m=i+1}^{n} W_{mj}X_m + B_j\right) \tag{6.5}$$

$$\underline{v}_j = inf\left(\sum_{k=1}^{i-1} W_{kj}X_k + B_j\right) + (\underline{w}_{ij}\overline{x}_i + \underline{b}_j) + inf\left(\sum_{m=i+1}^{n} W_{mj}X_m + B_j\right) \quad (6.6)$$

$$\underline{v}_j = inf\left(\sum_{k=1}^{i-1} W_{kj}X_k + B_j\right) + (\overline{w}_{ij}\overline{x}_i + \underline{b}_j) + inf\left(\sum_{m=i+1}^{n} W_{mj}X_m + B_j\right) \quad (6.7)$$

2. Para o supremo

$$\overline{v}_j = sup\left(\sum_{k=1}^{i-1} W_{kj}X_k + B_j\right) + (\underline{w}_{ij}\underline{x}_i + \overline{b}_j) + sup\left(\sum_{m=i+1}^{n} W_{mj}X_m + B_j\right) \quad (6.8)$$

$$\overline{v}_j = sup\left(\sum_{k=1}^{i-1} W_{kj}X_k + B_j\right) + (\overline{w}_{ij}\underline{x}_i + \overline{b}_j) + sup\left(\sum_{m=i+1}^{n} W_{mj}X_m + B_j\right) \quad (6.9)$$

$$\overline{v}_j = sup\left(\sum_{k=1}^{i-1} W_{kj}X_k + B_j\right) + (\underline{w}_{ij}\overline{x}_i + \overline{b}_j) + sup\left(\sum_{m=i+1}^{n} W_{mj}X_m + B_j\right) \quad (6.10)$$

$$\overline{v}_j = sup\left(\sum_{k=1}^{i-1} W_{kj}X_k + B_j\right) + (\overline{w}_{ij}\overline{x}_i + \overline{b}_j) + sup\left(\sum_{m=i+1}^{n} W_{mj}X_m + B_j\right) \quad (6.11)$$

Assim, pela definição 3.6.3, a derivada de $\breve{v}_j$ em relação a W_{ij} é dada pela equação 6.12.

$$\frac{\partial \breve{v}_j}{\partial W_{ij}} = \frac{1}{2}\left(\frac{\partial \overline{v}_j}{\partial w} + \frac{\partial \underline{v}_j}{\partial w}\right) \quad (6.12)$$

onde $w \in W_{ij}$, $\underline{v}_j$ é escolhida entre as equações 6.4, 6.5, 6.6, 6.7 e $\overline{v}_j$ é uma das equações 6.8, 6.9, 6.10 ou 6.11 dependendo do $sign(W_{ij})$ e do $sign(X_i)$.

Para o cálculo do primeiro fator da equação 6.3 temos que:

$$\delta_j = \frac{\partial e}{\partial \breve{v}_j} = \frac{\partial e}{\partial y_j}\frac{\partial y_j}{\partial \breve{v}_j} \quad (6.13)$$

Então é calculada a derivada para poder mudar os pesos em função do gradiente descendente de forma que o peso será atualizado de acordo com:

$$\Delta_e = \delta_j \frac{\partial \breve{v}_j}{\partial W_{ij}} \quad (6.14)$$

Logo, o cálculo do segundo fator da equação 6.13 é uma derivada de uns função real que vai depender da função de ativação que o neurônio j utilize, então a derivada desta função vai se denotar por:

$$\frac{\partial y_j}{\partial \breve{v}_j} = \lambda'(\breve{v}_j) \quad (6.15)$$

Para computar o primeiro fator da equação 6.13 deve se considerar dois casos.

Primeiro para quando o neurônio j está na saída $j = k$ da rede, então:

$$\frac{\partial e}{\partial y_k} = -(d_k - y_k) \tag{6.16}$$

Assim pelas equações 6.15, 6.16, a equação 6.13 será:

$$\delta_k = -(d_k - y_k)\lambda'(\breve{v}_j) \tag{6.17}$$

Logo a equação 6.3 para qualquer neurônio k na camada de saída será:

$$\Delta e = -(d_k - y_k)\lambda'(\breve{v}_k)\frac{1}{2}\left(\frac{\partial \overline{v}_k}{\partial w} + \frac{\partial \underline{v}_k}{\partial w}\right) \tag{6.18}$$

O segundo caso é quando j é um neurônio em alguma camada escondida. Logo da mesma forma como acontece nas redes neurais pontuais, não so conhecemos a contribuição do neurônio j no erro da saída é por isso que é usada a media do erro cometido, então:

$$\frac{\partial e}{\partial Y_j} = \sum_h \frac{\partial e}{\partial \breve{v}_h}\frac{\partial \breve{v}_h}{\partial Y_j} \tag{6.19}$$

Para o cálculo do segundo fator tem-se que:

$$\breve{v}_h = \frac{1}{2}\left[sup\left(\sum_{k=1}^{n} W_{kh}Y_k + B_h\right) + inf\left(\sum_{k=1}^{n} W_{kh}Y_k + B_h\right)\right]$$

Para identificar $\breve{v}_h$ é necessário ver o cálculo de V_h em relação a um y_j, então V_h será o intervalo onde o ínfimo é escolhido entre as equações 6.20, 6.21, 6.22 e 6.23, da mesma forma o supremo do intervalo é escolhido entre as equações 6.24, 6.25, 6.26 e 6.27.

1. Para o ínfimo

$$\underline{v}_h = inf\left(\sum_{m=1}^{j-1} W_{mh}Y_m + B_h\right) + (\underline{w}_{jh}\underline{y}_j + \underline{b}_h) + inf\left(\sum_{n=j+1}^{p} W_{nh}Y_n + B_h\right) \tag{6.20}$$

$$\underline{v}_h = inf\left(\sum_{m=1}^{j-1} W_{mh}Y_m + B_h\right) + (\overline{w}_{jh}\underline{y}_j + \underline{b}_h) + inf\left(\sum_{n=j+1}^{p} W_{nh}Y_n + B_h\right) \tag{6.21}$$

$$\underline{v}_h = inf\left(\sum_{m=1}^{j-1} W_{mh}Y_m + B_h\right) + (\underline{w}_{jh}\overline{y}_j + \underline{b}_h) + inf\left(\sum_{n=j+1}^{p} W_{nh}Y_n + B_h\right) \tag{6.22}$$

$$\underline{v}_h = inf\left(\sum_{m=1}^{j-1} W_{mh}Y_m + B_h\right) + (\overline{w}_{jh}\overline{y}_j + \underline{b}_h) + inf\left(\sum_{n=j+1}^{p} W_{nh}Y_n + B_h\right) \quad (6.23)$$

2. Para o supremo

$$\overline{v}_h = sup\left(\sum_{m=1}^{j-1} W_{mh}Y_m + B_h\right) + (\underline{w}_{jh}\underline{y}_j + \overline{b}_h) + sup\left(\sum_{n=j+1}^{p} W_{nh}Y_n + B_h\right) \quad (6.24)$$

$$\overline{v}_h = sup\left(\sum_{m=1}^{j-1} W_{mh}Y_m + B_h\right) + (\overline{w}_{jh}\underline{y}_j + \overline{b}_h) + sup\left(\sum_{n=j+1}^{p} W_{nh}Y_n + B_h\right) \quad (6.25)$$

$$\overline{v}_h = sup\left(\sum_{m=1}^{j-1} W_{mh}Y_m + B_h\right) + (\underline{w}_{jh}\overline{y}_j + \overline{b}_h) + sup\left(\sum_{n=j+1}^{p} W_{nh}Y_n + B_h\right) \quad (6.26)$$

$$\overline{v}_h = sup\left(\sum_{m=1}^{j-1} W_{mh}Y_m + B_h\right) + (\overline{w}_{jh}\overline{y}_j + \overline{b}_h) + sup\left(\sum_{n=j+1}^{p} W_{nh}Y_n + B_h\right) \quad (6.27)$$

Dado que Y_j é um intervalo degenerado, então a equação 6.20 é equivalente à equação 6.22 e 6.21 é equivalente à equação 6.23. O mesmo acontece com as equações para o supremo, isto é as equações 6.24 e 6.26 são equivalentes às equações 6.25 e 6.27, respectivamente.

Assim, pela definição 3.6.3, a derivada de $\breve{v}_h$ em relação a Y_j é dada pela equação 6.12.

$$\frac{\partial \breve{v}_h}{\partial Y_j} = \frac{1}{2}\left(\frac{\partial \overline{v}_h}{\partial y_j} + \frac{\partial \underline{v}_h}{\partial y_j}\right) \quad (6.28)$$

onde $y_j \in Y_j$, $\underline{v}_h$ é escolhida entre as equações 6.20 e 6.22, e $\overline{Y}_j$ é uma das equações 6.24, 6.10 dependendo do $sign(W_{ij})$ e do $sign(X_i)$.

Então, a equação 6.19 é re-escrita como:

$$\frac{\partial e}{\partial Y_j} = \sum_h \frac{\partial e}{\partial \breve{v}_h}\frac{(\overline{v}'_h + \underline{v}'_h)}{2}$$

$$\frac{\partial e}{\partial Y_j} = \sum_h \delta_h \frac{(\overline{v}'_h + \underline{v}'_h)}{2}$$

$$\delta_j = \lambda'(\breve{v}_j)\sum_h \delta_h \frac{(\overline{v}'_h + \underline{v}'_h)}{2} \quad (6.29)$$

Se a função de ativação é a função da equação 6.1 então:

$$\lambda'(\breve{v}_j) = y_j(1 - y_j)$$

1.	Inicializa os pesos W_{ij} e bias B_j de todas as camadas com intervalos aleatórios pequenos para $(\overline{w}_{ij} - \underline{w}_{ij})$ é mínimo.
2.	Repetir - Apresentar um padrão de entrada $[X_k, d_k]$ X_k é um vetor de intervalos que representam o padrão k. d_k é a saída desejada. - Propagar a ativação dos neurônios da camada de entrada para a primeira camada oculta para X_k $y_j = f_n\left(med\left(\sum_{i=0}^{N-1} W_{ij}(t)X_i(t) + B_j\right)\right)$ - Propagar as saídas da primeira camada oculta através de todas as camadas até a camada de saída. Dada uma camada r, a saída dela é normalizada pelos neurônios na camada $r+1$ depois calculada a saída nessa camada até a camada de saída. $y_k = f_n\left(med\left(\sum_{j=0}^{M-1} W_{jk}(t)Y_j(t) + B_k\right)\right)$ $med(.)$ é a função ponto médio nos intervalos e f_n é a função de ativação intervalar a qual pode ser a equação 6.1. - Calcular o erro cometido e_k para o padrão k com a equação 6.2 - Correção dos Pesos e bias $W_{ij}(t+1) = W_{ij}(t) - \eta\Delta e$, onde Δe é dado pela equação 6.18 quando j é um neurônio em alguma camada oculta e a equação 6.30 quando j é um neurônio na camada de saída. η é o coeficiente de aprendizado menor que 1 e maior que 0.
3.	Até que $\frac{1}{M}\sum_{k=1}^{M} e_k$ seja minimizado.

Tabela 6.1: Algoritmo de Treinamento para uma Rede Perceptron Intervalar de várias Camadas

Logo a equação 6.3 para algum neurônio j em alguma camada oculta será:

$$\Delta e = \lambda'(\breve{v}_j)\left[\sum_h \delta_w \frac{(\overline{v}'_h + \underline{v}'_h)}{2}\right]\left[\frac{\partial \overline{v}_k}{\partial w} + \frac{\partial \underline{v}_k}{\partial w}\right] \tag{6.30}$$

Onde $w \in W_{ij}$. A tabela 6.1 descreve o algoritmo para o treinamento de uma rede neural Perceptron intervalar de múltiplas camadas que usa a regra delta intervalar generalizada para a modificação dos pesos.

Entrada	Descrição
1	Longitude do Sepalo
2	Largura do Sepalo
3	Longitude do Pétalo
4	Largura do pétalo

Tabela 6.2: Entradas do conjunto de dados IRIS

6.4 Classificação de Padrões com a Rede Perceptron Intervalar de Múltiplas Camadas

Para verificar a convergência do algoritmo proposto para uma rede neural Perceptron Intervalar de múltiplas camadas, foi utilizado um banco de dados reais chamado de *Iris* o qual foi obtido do repositório de banco de dados para aprendizado de máquina [SHM98]. Este banco de dados contém 3 classes, cada uma com 50 exemplos, aqui cada uma refere a um tipo de planta Iris, então:

1. Iris setosa: Classe 1
2. Iris Versicolor: Classe 2
3. Iris Virginica: Classe 3.

Cada padrão de entrada consiste em quatro atributos descritos na tabela 6.2. Logo, este banco de dados foi transformado em intervalos, onde para cada uma das entradas em cada padrão foi gerado dois valores aleatórios entre 0 e 0.01, logo um desses valores foi somado ao valor da entrada gerando o supremo do intervalo e foi subtraído o segundo valor gerado obtendo assim o ínfimo do intervalo de entrada. Assim, a rede Perceptron intervalar de múltiplas camadas foi treinada com um conjunto de dados intervalar. Para o treinamento da rede foi utilizado o 65% dos dados e 35% foi utilizado para o teste.

A rede Perceptron Intervalar foi treinada com uma camada oculta com 10 neurônios e 2 neurônios na camada de saída. Utilizando um coeficiente de aprendizado $\eta = 0.3$, o algoritmo convergiu depois de 980 épocas. A matriz de confusão da classificação é mostrada na tabela 6.3

As figuras 6.2-6.7 mostram a classificação feita com a rede Perceptron intervalar de múltiplas camadas levando em consideração cada par de entradas. Assim a figura 6.2 representa a entrada 1 relacionada com a entrada 2, 6.3 representa a entrada 1 relacionada com a entrada 3, 6.4 representa a entrada 1 relacionada com a entrada

	Classe 1	Classe 2	Classe 3
Classe 1	15	-	-
Classe 2	-	14	1
Classe 3	-	1	14

Tabela 6.3: Matriz de confusão da classificação para o IRIS

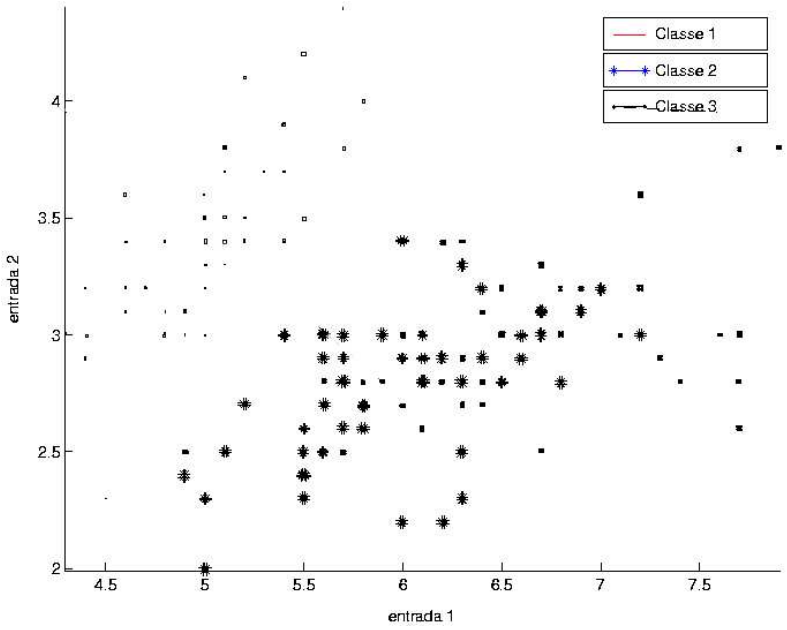

Figura 6.2: Entrada 1 vs Entrada 2: Classificação do IRIS por RPI de Múltiplas Camadas

4, 6.5 representa a entrada 2 relacionada com a entrada 3, 6.6 representa a entrada 2 relacionada com a entrada 4 e 6.7 representa a entrada 3 relacionada com a entrada 4. Pela tabela 6.3 é visto que os resultados obtidos com a rede Perceptron Intervalar foi ótima. A figura 6.8 mostra o progresso do erro no processo de treinamento.

6.5 Propriedades de Aprendizagem

Pelos testes feitos com o banco de dados IRIS, podemos ver que o algoritmo proposto conseguiu convergir. Assim, a Perceptron Intervalares de múltiplas camadas resolve problemas não linearmente separáveis intervalares. É observado que o processo de convergência é oscilatório mas ele consegue convergir. Analisando o fato da oscilação foi apresentado o problema da XOR(com intervalos degenerados) para a rede Perceptron Intervalar de múltiplas camadas variando o coeficiente de aprendizado de $\eta = 0.1$ até $\eta = 0.7$. As figuras 6.9-6.15 mostram o processo de convergência da rede, logo é visto que da mesma forma como foi explicado na Perceptron Intervalar de uma camada, esta

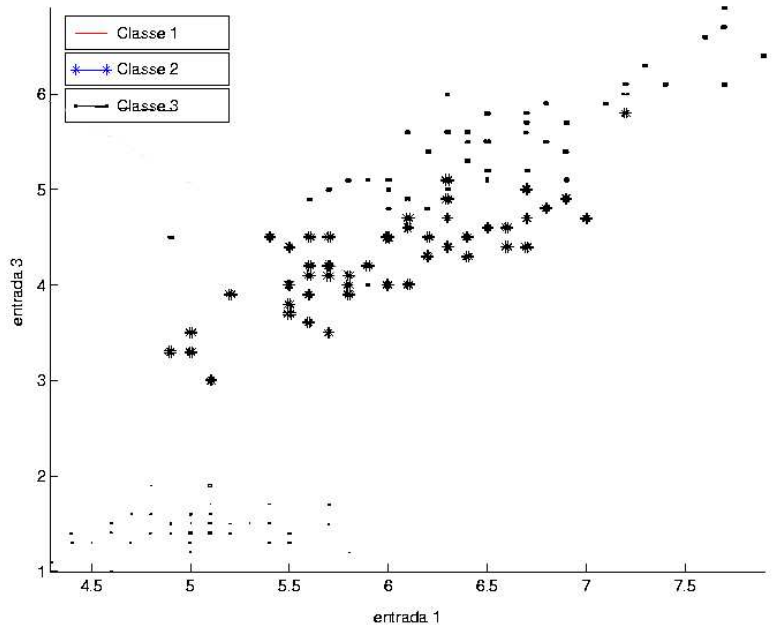

Figura 6.3: Entrada 1 vs Entrada 3: Classificação do IRIS por RPI de Múltiplas Camadas

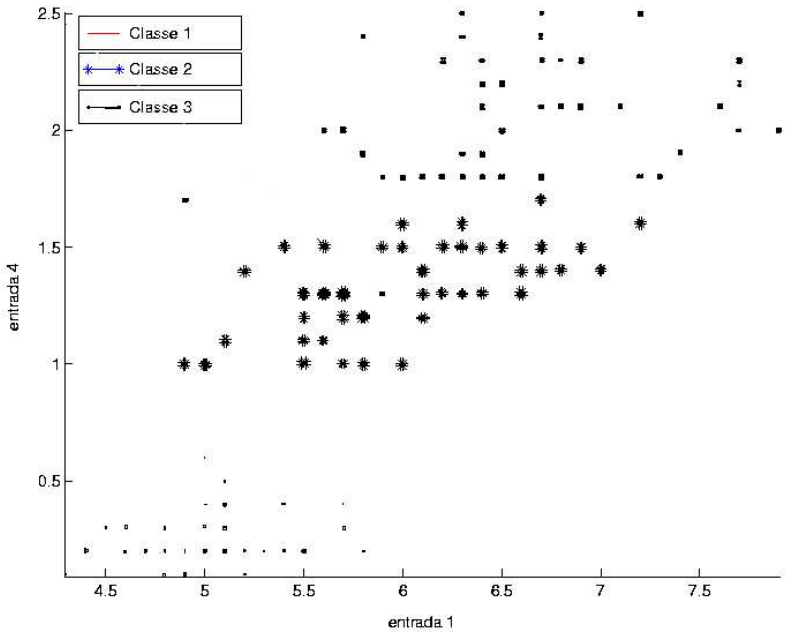

Figura 6.4: Entrada 1 vs Entrada 4: Classificação do IRIS por RPI de Múltiplas Camadas

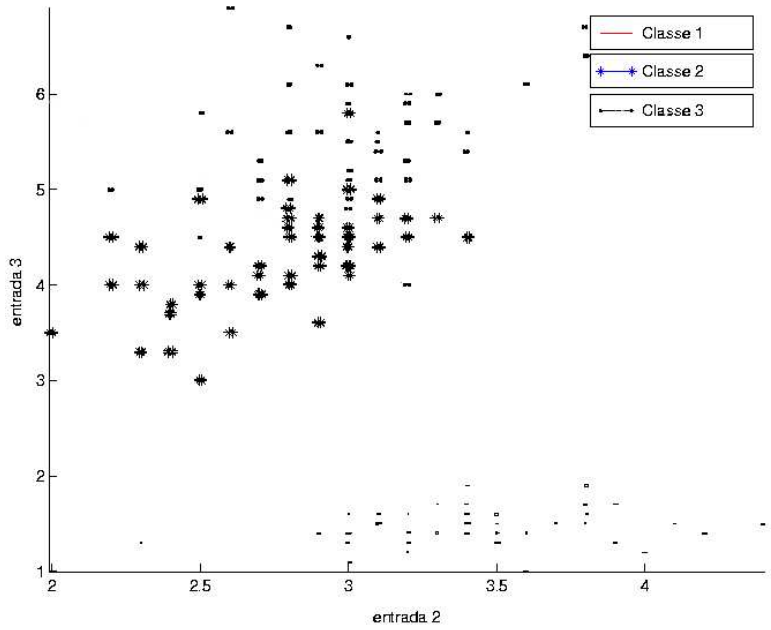

Figura 6.5: Entrada 2 vs Entrada 3: Classificação do IRIS por RPI de Múltiplas Camadas

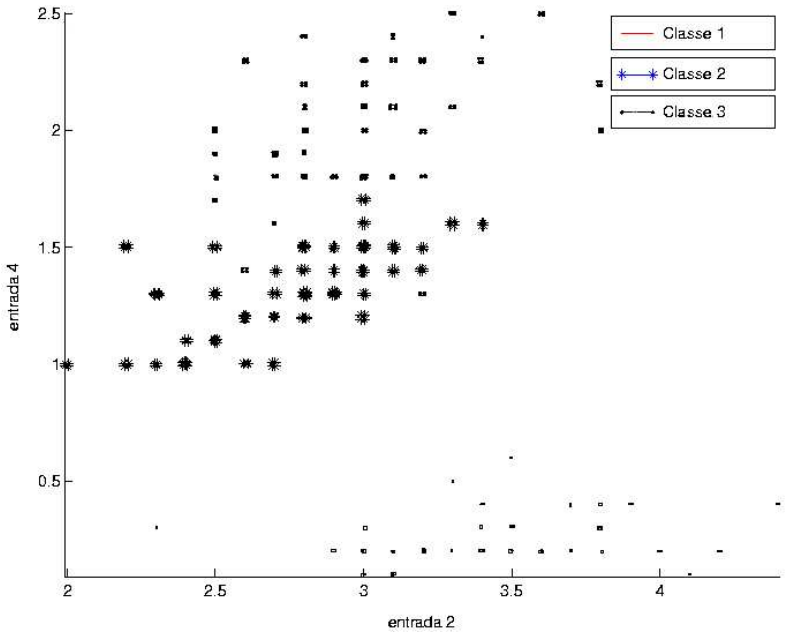

Figura 6.6: Entrada 2 vs Entrada 4: Classificação do IRIS por RPI de Múltiplas Camadas

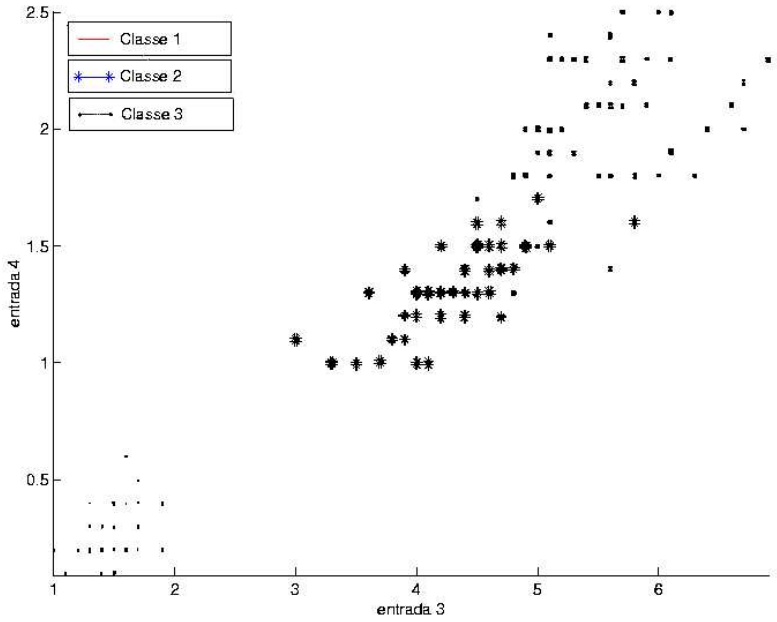

Figura 6.7: Entrada 3 vs Entrada 4: Classificação do IRIS por RPI de Múltiplas Camadas

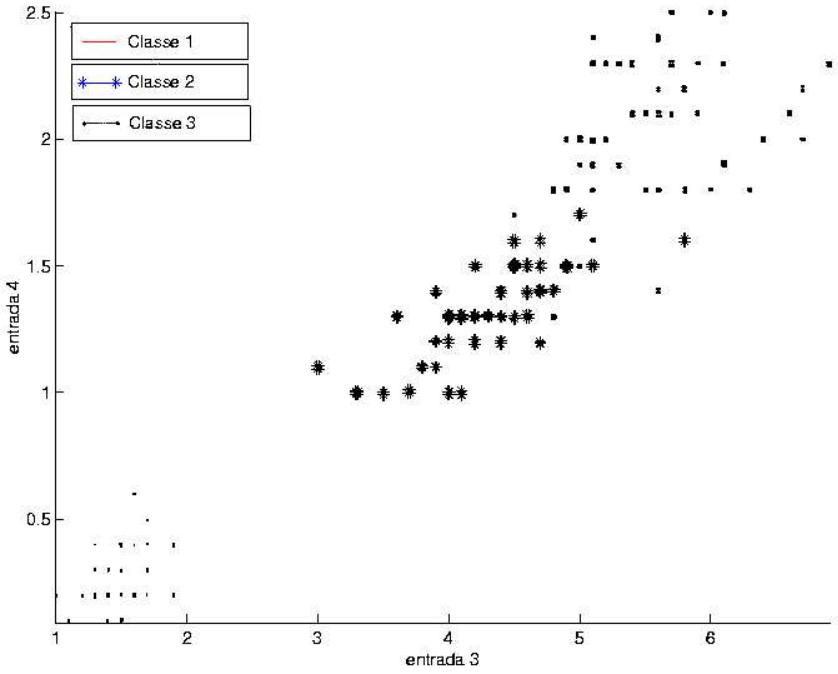

Figura 6.8: Erro no Treinamento para a Classificação do IRIS por a RPI de Múltiplas Camadas

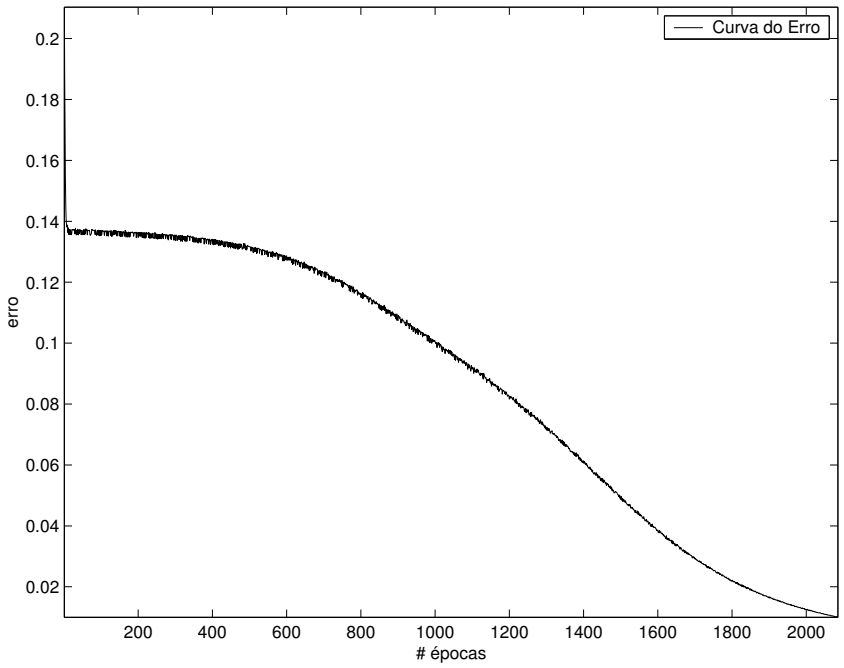

Figura 6.9: Erro no Treinamento para a Classificação do XOR com $\eta = 0.1$

rede Perceptron Intervalar é mais sensível ao coeficiente de aprendizado em comparação com a rede Perceptron tradicional. Observando as figuras do progresso do erro no treinamento da rede Perceptron Intervalar de múltiplas camadas é visto que quando o coeficiente de aprendizado η é menor, a oscilação no treinamento é menor e quando o coeficiente de aprendizado η é mais alto, a oscilação é maior.

Embora o processo de convergência da rede Perceptron Intervalar é oscilante, ela faz uma classificação bem definida. Mas a maior vantagem desde tipo de redes é o controle da informação representada em intervalos.

6.6 Conclusões

As redes neurais tradicionais são partes de uma enorme área de pesquisa. Elas são usadas em diferentes aplicações. Assim, os pesquisadores estão sempre tentando melhorar o comportamento ou alguma característica nestas redes. As redes neurais perceptron intervalar tentam melhorar o fato da precisão nos dados processados por elas dando uma melhor representação aos dados:"dados intervalares"e fazendo um processamento destes dados mais seguros:"Matemática Intervalar". É assim que são propostas estruturas bem definidas para este tipo de redes neurais junto com seus algoritmos de treinamento, os quais foram testados com diferentes dados intervalares.

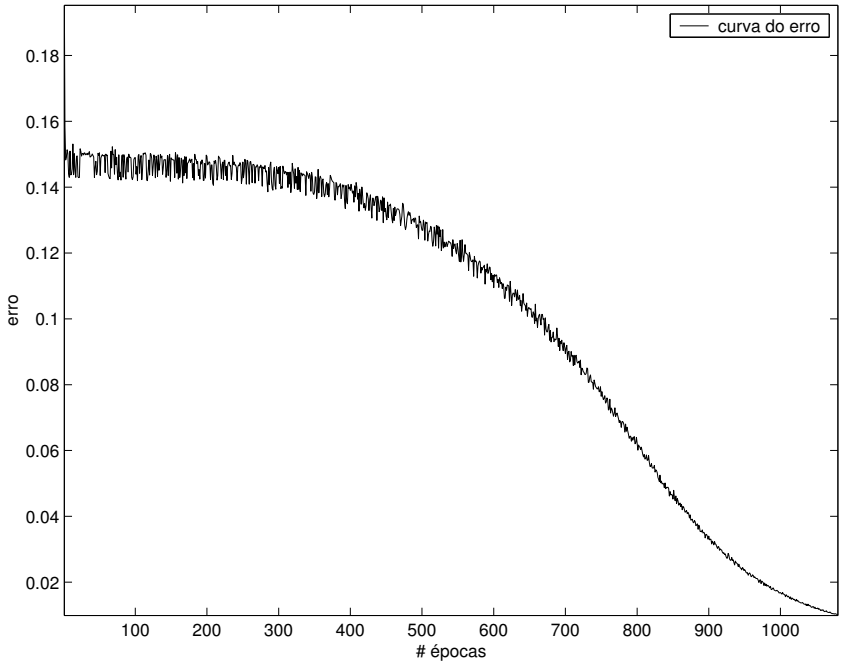

Figura 6.10: Erro no Treinamento para a Classificação do XOR com $\eta = 0.2$

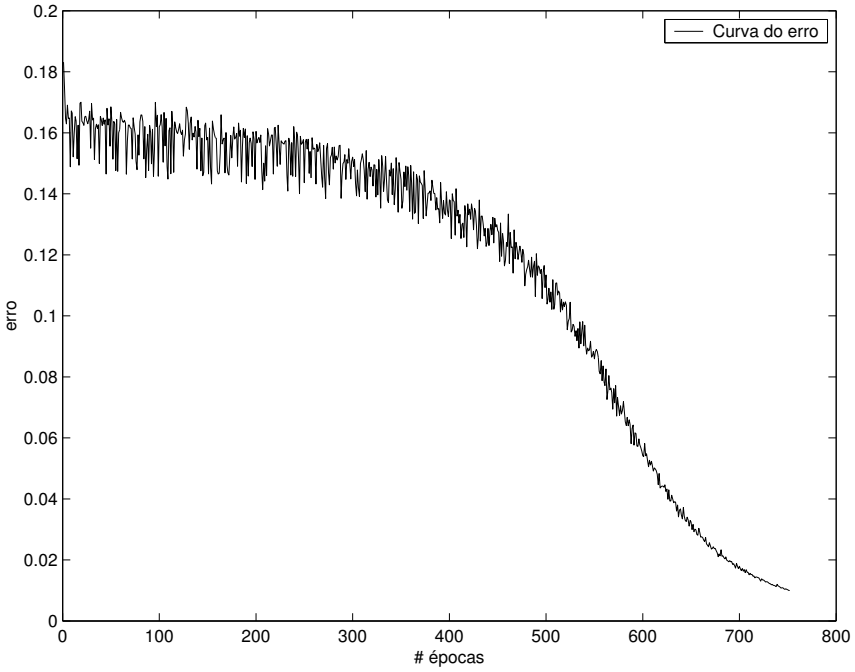

Figura 6.11: Erro no Treinamento para a Classificação do XOR com $\eta = 0.3$

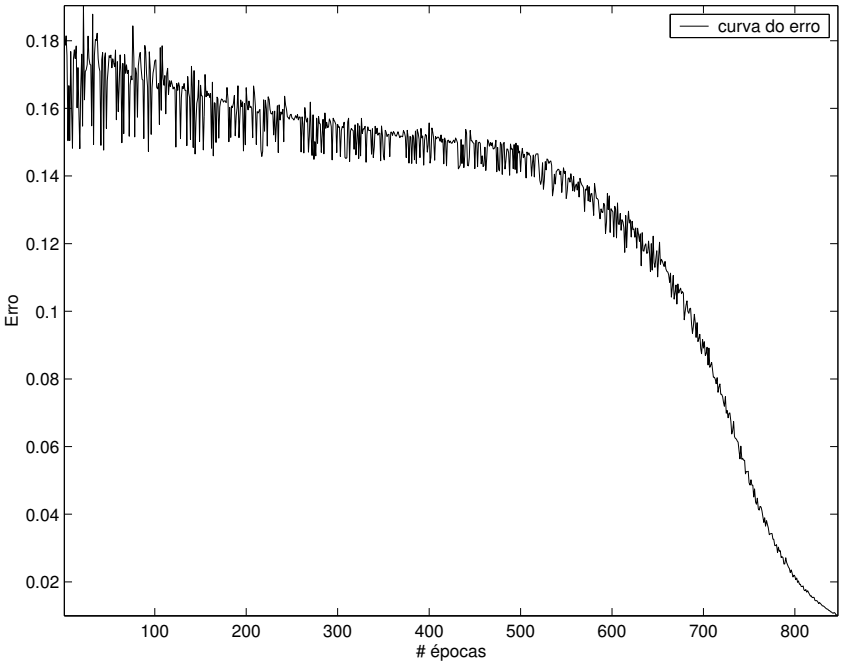

Figura 6.12: Erro no Treinamento para a Classificação do XOR com $\eta = 0.4$

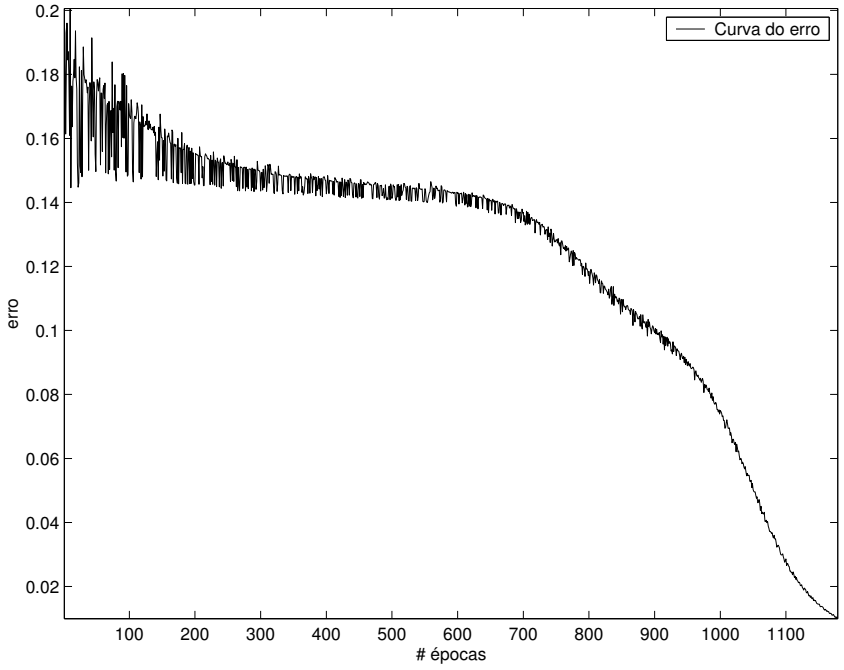

Figura 6.13: Erro no Treinamento para a Classificação do XOR com $\eta = 0.5$

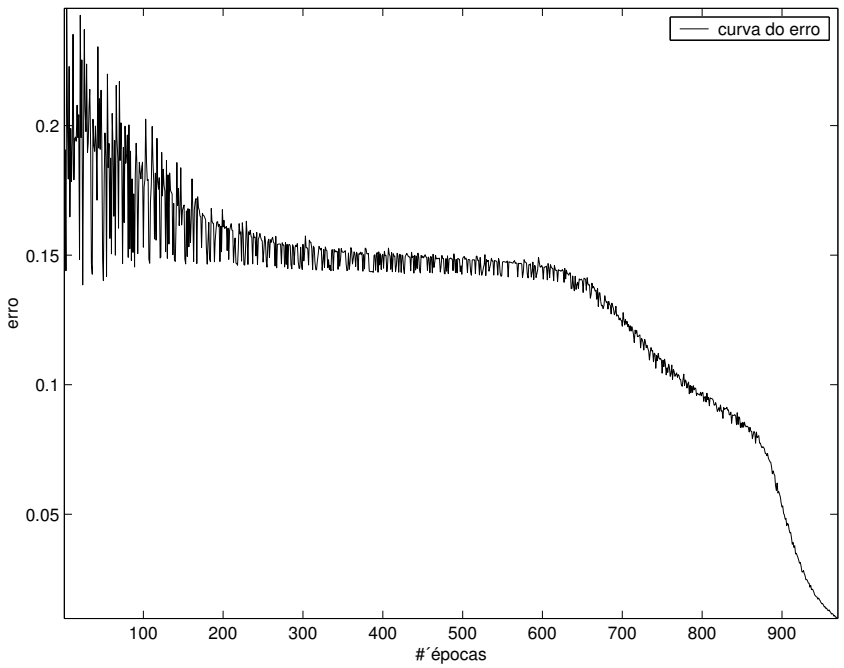

Figura 6.14: Erro no Treinamento para a Classificação do XOR com $\eta = 0.6$

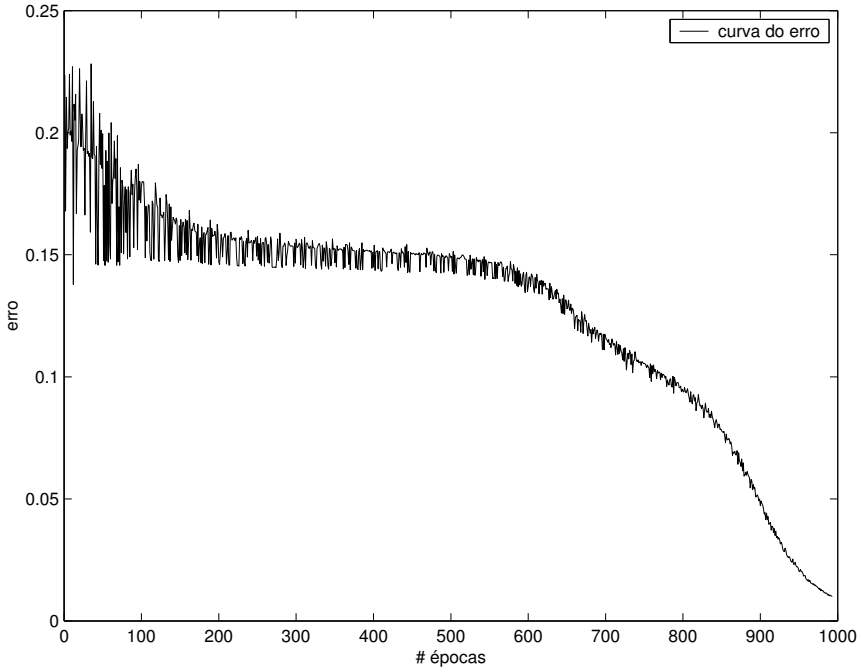

Figura 6.15: Erro no Treinamento para a Classificação do XOR com $\eta = 0.7$

Capítulo 7

Considerações Finais

Não há ramo da matemática, por abstrato que seja, que não possa um dia vir a ser aplicado aos fenômenos do mundo real!

Nicolai I. Lobachevsky

7.1 Conclusões

A matemática intervalar é uma alternativa para solucionar problemas onde existam erros computacionais e erros de exatidão que se apresentam cotidianamente. Embora os intervalos são uma alternativa de solução deste tipo de problemas, eles também podem ser visto como uma alternativa para lidar com incerteza na representação de alguma característica de algum objeto como o peso, a largura ou fenômenos da natureza, por exemplo, temperatura, umidade. Assim os intervalos levam consigo muito mais informações que um valor real. Assim uma das maiores importâncias dos intervalos é a quantidade de informação que contém e que conseguem controlar. Atualmente, a Matemática Intervalar é muito utilizada para aplicações em Soft Computing [Nol04], no tratamento e modelagem da incerteza em computação, no processamento de informações fuzzy, na teoria de controle, na inteligência artificial, representação do conhecimento, redes, computação gráfica, e diversas outras aplicações em Ciência e Tecnologia [Kre] que lidam com dados incertos.

Dado o avanço das aplicações da matemática intervalar, é visto que precisa-se de construir algoritmos que consigam trabalhar diretamente com os intervalos, então os algoritmo tradicionais são reconstruídos para ser usados nos casos intervalares.

As redes neurais artificiais perceptron intervalar, são uma área nova dentro da pesquisa de redes neurais artificiais e tenta ser uma proposta para o tratamento de

dados intervalares. Até agora alguns trabalhos foram realizados, nesta dissertação apresentamos uma contribuição nesta linha.

Nosso trabalho apresenta estruturas definidas para as redes neurais intervalares propostas (RPI de uma camada e de múltiplas camadas) e a construção dos seus algoritmos de treinamento, os quais foram testados para analisar o comportamento e medir as vantagens e desvantagens em relação aos algoritmos tradicionais.

Assim, foi proposta uma definição das redes neurais intervalares e seguidamente foram propostas as redes neurais Perceptron Intervalares. A rede Perceptron Intervalar de uma camada classifica padrões linearmente separáveis. A medida que o algoritmo de treinamento desta rede convergi, um neurônio na camada de saída posiciona uma região de decisão entre as classes, a qual tem como ponto de referência a reta dos pontos médios dos parâmetros que a formam (pesos e entradas). Com um conjunto de dados com uma distribuição aleatória uniforme, existe uma probabilidade maior que o número de épocas necessário para a rede Perceptron Intervalar de uma camada convergia mais rápido em comparação com a rede Perceptron tradicional. A rede Perceptron com Múltiplas Camadas vai conseguir classificar padrões não necessariamente separáveis linearmente. O algoritmo que utiliza para o treinamento é baseado no algoritmo *backpropagation*, assim pelos testes feitos é visto que é recomendável utilizar um coeficiente de aprendizado pequeno.

Uma das desvantagens da computação intervalar, é o tempo que levam os algoritmos em executar os processos, dado que tem que lidar com um número maior de cálculos, mas os intervalos levam consigo informação que precisa ser controlada para ter uma computação segura.

No desenvolvimento deste trabalho foram feitas algumas publicações [PEBL04a], [PEBL04b] e foi submetido um artigo na revista TEMA [PEBL]. Além disto estamos no processo da preparação de outros artigos da segunda parte do trabalho que são as redes neurais Perceptron Intervalares com Múltiplas Camadas para serem submetidos a revistas e congressos da área.

Assim podemos concluir que o trabalho aqui apresentado, é uma proposta de como utilizar as redes neurais artificiais perceptron com dados intervalares, os quais podem representar incerteza, conhecimento de especialistas e controle nos erros computacionais.

7.2 Trabalhos Futuros

Dado que esta linha de pesquisa é relativamente nova. Pretendemos futuramente desenvolver conceitos da matemática intervalar e conseqüentemente definir a rede neural totalmente intervalar. Assim deixamos este ponto para trabalhos futuros como novas dissertações de mestrado ou teses de doutorado.

Neste trabalho foi mostrado que as redes neurais intervalares conseguem resolver problemas com dados intervalares e com dados pontuais. No entanto, pretendemos futuramente aplicar estas redes em problemas reais que podem ser matéria de trabalhos futuros.

Embora neste documento são propostas as redes neurais Perceptron Intervalares com algoritmos de aprendizado como a regra delta intervalar e sua generalização, existem outros algoritmos que podem ser reconstruídos para poder treinar estas redes que visem a aceleração do processo de treinamento, os quais são matéria de futuros estudos.

Outro ponto importante para estudos futuros é o estudo da classificação de padrões intervalares com outros métodos os quais poderiam ser reconstruídos desde os métodos tradicionais.

Apêndice A

Prova do Teorema da Separação Linear das Classes com uma Perceptron Intervalar

Para provar este teorema é importante analisar o sinal dos intervalos X_1, X_2, W_1, W_2, B e as combinações entre eles na equação A.1:

$$S = W_1 X_1 + W_2 X_2 + B \tag{A.1}$$

Se $\breve{w}_1, \breve{w}_2$, $\breve{b}$ são os pontos médios dos W_1, W_2, B respectivamente, pela equação A.1 temos que:

$$\begin{aligned}[\underline{s}, \overline{s}] =&[\breve{w}_1 - \Delta_1, \breve{w}_1 + \Delta_1][\underline{x}_1, \overline{x}_1] + [\breve{w}_2 - \Delta_2, \breve{w}_2 + \Delta_2][\underline{x}_2, \overline{x}_2] + \\ &[\breve{b} - \Delta_b, \breve{b} + \Delta_b]\end{aligned} \tag{A.2}$$

O sinal dos intervalos pode ser: +1, -1 ou 0. O sinal é importante porque vai influenciar nos cálculos feitos com os intervalos, assim temos que analisar as combinações dos intervalos na equação A.1

1. Se $sign(W_1) = sign(W_2) = sign(B) = sign(X_1) = sign(X_2) = 1$, pela equação A.2. A saída da rede é dada por:

$$\begin{aligned}[\underline{s}, \overline{s}] =&[(\breve{w}_1 - \Delta_1)\underline{x}_1, (\breve{w}_1 + \Delta_1)\overline{x}_1] + [(\breve{w}_2 - \Delta_2)\underline{x}_2, (\breve{w}_2 + \Delta_2)\overline{x}_2] + \\ &[\breve{b} - \Delta_b, \breve{b} + \Delta_b] \\ [\underline{s}, \overline{s}] =&[(\breve{w}_1 - \Delta_1)\underline{x}_1 + (\breve{w}_2 - \Delta_2)\underline{x}_2 + (\breve{b} - \Delta_b), (\breve{w}_1 + \Delta_1)\overline{x}_1 + \\ &(\breve{w}_2 + \Delta_2)\overline{x}_2 + (\breve{b} + \Delta_b)]\end{aligned}$$

Como $\breve{s} = \frac{\underline{s} + \overline{s}}{2}$, então:

$$\begin{aligned}\breve{s} =& \frac{(\breve{w}_1 + \Delta_1)\overline{x}_1 + (\breve{w}_2 + \Delta_2)\overline{x}_2 + \breve{b} + \Delta_b +}{2} \\ & \frac{(\breve{w}_1 - \Delta_1)\underline{x}_1 + (\breve{w}_2 - \Delta_2)\underline{x}_2 + \breve{b} - \Delta_b}{2} \\ \breve{s} =& \frac{\breve{w}_1(\underline{x}_1 + \overline{x}_1) + \breve{w}_2(\underline{x}_2 + \overline{x}_2) + 2\breve{b} +}{2} \\ & \frac{\Delta_1(\overline{x}_1 - \underline{x}_1) + \Delta_2(\overline{x}_2 - \underline{x}_2)}{2}\end{aligned}$$

$$\breve{s} = \breve{w}_1\breve{x}_1 + \breve{w}_2\breve{x}_2 + \breve{b} - \frac{\Delta_1(\underline{x}_1 - \overline{x}_1)}{2} - \frac{\Delta_2(\underline{x}_2 - \overline{x}_2)}{2} \tag{A.3}$$

Para os seguintes casos, o resultado é igual:

- Se $sign(W_1) = sign(W_2) = sign(B) = sign(X_1) = sign(X_2) = -1$
- Se $sign(W_2) = 1$ e $sign(W_1) = sign(X_2) = sign(X_1) = -1$
- Se $sign(W_1) = sign(X_1) = 1$ e $sign(W_2) = sign(X_2) = -1$.
- Se $sign(W_1) - 1$ e $sign(W_2) = sign(X_2) = sign(X_1) = 1$.
- Se $sign(W_2) = sign(X_2) = 1$ e $sign(X_1) = sign(W_1) = -1$.

2. Se $sign(W_1) = sign(W_2) = sign(B) = sign(X_1) = -1$ e $sign(X_2) = 1$
   A saída da rede é dada por:

$$\begin{aligned}[\underline{s}, \overline{s}] =& [(\breve{w}_1 + \Delta_1)\overline{x}_1, (\breve{w}_1 - \Delta_1)\underline{x}_1] + [(\breve{w}_2 - \Delta_2)\overline{x}_2, (\breve{w}_2 + \Delta_2)\underline{x}_2] + \\ & [\breve{b} - \Delta_b, \breve{b} + \Delta_b] \\ [\underline{s}, \overline{s}] =& [(\breve{w}_1 + \Delta_1)\overline{x}_1 + (\breve{w}_2 - \Delta_2)\overline{x}_2 + (\breve{b} - \Delta_b), \\ & (\breve{w}_1 - \Delta_1)\underline{x}_1 + (\breve{w}_2 + \Delta_2)\underline{x}_2 + (\breve{b} + \Delta_b)]\end{aligned}$$

Como $\breve{s} = \frac{\underline{s} + \overline{s}}{2}$, então:

$$\begin{aligned}\breve{s} =& \frac{(\breve{w}_1 - \Delta_1)\underline{x}_1 + (\breve{w}_2 + \Delta_2)\underline{x}_2 + \breve{b} + \Delta_b +}{2} \\ & \frac{(\breve{w}_1 + \Delta_1)\overline{x}_1 + (\breve{w}_2 - \Delta_2)\overline{x}_2 + \breve{b} - \Delta_b}{2} \\ \breve{s} =& \frac{\breve{w}_1(\overline{x}_1 + \underline{x}_1) + \breve{w}_2(\overline{x}_2 + \underline{x}_2) + 2\breve{b} -}{2} \\ & \frac{\Delta_1(\underline{x}_1 - \overline{x}_1) + \Delta_2(\underline{x}_2 - \overline{x}_2)}{2}\end{aligned}$$

$$\breve{s} = \breve{w}_1\breve{x}_1 + \breve{w}_2\breve{x}_2 + \breve{b} - \frac{\Delta_1(\underline{x}_1 - \overline{x}_1)}{2} + \frac{\Delta_2(\underline{x}_2 - \overline{x}_2)}{2} \tag{A.4}$$

Para os seguintes casos, o resultado é igual:

- Se $sign(W_1) = sign(W_2) = sign(X_1) = 1$ e $sign(X_2) = -1$.
- Se $sign(W_1) = sign(X_1) = sign(X_2) = 1$ e $sign(W_2) = -1$.
- Se $sign(W_1) = sign(X_2) = -1$ e $sign(X_1) = sign(W_2) = 1$.

3. Se $sign(W_1) = sign(W_2) = sign(X_2) = -1$ e $sign(X_1) = 1$
   A saída da rede é dada por:

$$\begin{aligned}[\underline{s}, \overline{s}] =& [(\breve{w}_1 - \Delta_1)\overline{x}_1, (\breve{w}_1 + \Delta_1)\underline{x}_1] + [(\breve{w}_2 + \Delta_2)\overline{x}_2, (\breve{w}_2 + \Delta_2)\underline{x}_2] + \\ & [\breve{b} - \Delta_b, \breve{b} + \Delta_b] \\ [\underline{s}, \overline{s}] =& [(\breve{w}_1 - \Delta_1)\overline{x}_1 + (\breve{w}_2 + \Delta_2)\overline{x}_2 + (\breve{b} - \Delta_b), \\ & (\breve{w}_1 + \Delta_1)\underline{x}_1 + (\breve{w}_2 - \Delta_2)\underline{x}_2 + (\breve{b} + \Delta_b)]\end{aligned}$$

   Como $\breve{s} = \dfrac{\underline{s} + \overline{s}}{2}$, então:

$$\begin{aligned}\breve{s} =& \frac{(\breve{w}_1 + \Delta_1)\underline{x}_1 + (\breve{w}_2 - \Delta_2)\underline{x}_2 + \breve{b} + \Delta_b + }{2} \\ & \frac{(\breve{w}_1 - \Delta_1)\overline{x}_1 + (\breve{w}_2 + \Delta_2)\overline{x}_2 + \breve{b} - \Delta_b}{2} \\ \breve{s} =& \frac{\breve{w}_1(\overline{x}_1 + \underline{x}_1) + \breve{w}_2(\overline{x}_2 + \underline{x}_2) + 2\breve{b} + }{2} \\ & \frac{\Delta_1(\underline{x}_1 - \overline{x}_1) - \Delta_2(\underline{x}_2 - \overline{x}_2)}{2}\end{aligned}$$

$$\breve{s} = \breve{w}_1\breve{x}_1 + \breve{w}_2\breve{x}_2 + \breve{b} + \frac{\Delta_1(\underline{x}_1 - \overline{x}_1)}{2} - \frac{\Delta_2(\underline{x}_2 - \overline{x}_2)}{2} \tag{A.5}$$

   Para os seguintes casos, o resultado é igual:

- Se $sign(W_1) = 1$ e $sign(W_2) = sign(X_2) = sign(X_1) = -1$
- Se $sign(W_1) = sign(W_2) = sign(X_2) = 1$ e $sign(X_1) = -1$.
- Se $sign(W_1) = sign(X_2) = 1$, $sign(W_2) = sign(X_1) = -1$.
- Se $sign(W_1) = 1$, $sign(W_2) = sign(X_1) = sign(X_2) = -1$.

4. Se $sign(X_1) = 0$ e $sign(W_1) = sign(W_2) = sign(X_2) = -1$
   A saída da rede é dada por:

$$\begin{aligned}[\underline{s}, \overline{s}] =& [(\breve{w}_1 - \Delta_1)\overline{x}_1, (\breve{w}_1 - \Delta_1)\underline{x}_1] + [(\breve{w}_2 + \Delta_2)\overline{x}_2, (\breve{w}_2 - \Delta_2)\underline{x}_2] + \\ & [\breve{b} - \Delta_b, \breve{b} + \Delta_b] \\ [\underline{s}, \overline{s}] =& [(\breve{w}_1 - \Delta_1)\overline{x}_1 + (\breve{w}_2 + \Delta_2)\overline{x}_2 + (\breve{b} - \Delta_b), \\ & (\breve{w}_1 - \Delta_1)\underline{x}_1 + (\breve{w}_2 - \Delta_2)\underline{x}_2 + (\breve{b} + \Delta_b)]\end{aligned}$$

Como $\breve{s} = \frac{\underline{s} + \overline{s}}{2}$, então:

$$\begin{aligned}
\breve{s} = & \frac{(\breve{w}_1 - \Delta_1)\underline{x}_1 + (\breve{w}_2 - \Delta_2)\underline{x}_2 + \breve{b} + \Delta_b +}{2} \\
& \frac{(\breve{w}_1 - \Delta_1)\overline{x}_1 + (\breve{w}_2 + \Delta_2)\overline{x}_2 + \breve{b} - \Delta_b}{2} \\
\breve{s} = & \frac{\breve{w}_1(\overline{x}_1 + \underline{x}_1) + \breve{w}_2(\overline{x}_2 + \underline{x}_2) + 2\breve{b} -}{2} \\
& \frac{\Delta_1(\underline{x}_1 + \overline{x}_1) - \Delta_2(\underline{x}_2 - \overline{x}_2)}{2}
\end{aligned}$$

$$\breve{s} = \breve{w}_1\breve{x}_1 + \breve{w}_2\breve{x}_2 + \breve{b} + \Delta_1\breve{x}_1 - \frac{\Delta_2(\underline{x}_2 - \overline{x}_2)}{2} \tag{A.6}$$

Para os seguintes casos, o resultado é igual:

- Se $sign(W_1) = sign(W_2) = sign(X_2) = 1$ e $sign(X_1) = 0$.
- Se $sign(W_1) = 1$, $sign(W_2) = sign(X_2) = -1$ e $sign(X_1) = 0$
- Se $sign(X_1) = 0$, $sign(W_2) = sign(X_2) = 1$ e $sign(W_1) = -1$.

5. Se $sign(X_2) = 0$ e $sign(W_1) = sign(W_2) = sign(X_1) = -1$
   A saída da rede é dada por:

$$\begin{aligned}
[\underline{s}, \overline{s}] = & [(\breve{w}_1 + \Delta_1)\overline{x}_1, (\breve{w}_1 - \Delta_1)\underline{x}_1] + [(\breve{w}_2 - \Delta_2)\overline{x}_2, (\breve{w}_2 - \Delta_2)\underline{x}_2] + \\
& [\breve{b} - \Delta_b, \breve{b} + \Delta_b] \\
[\underline{s}, \overline{s}] = & [(\breve{w}_1 + \Delta_1)\overline{x}_1 + (\breve{w}_2 - \Delta_2)\overline{x}_2 + (\breve{b} - \Delta_b), \\
& (\breve{w}_1 - \Delta_1)\underline{x}_1 + (\breve{w}_2 - \Delta_2)\underline{x}_2 + (\breve{b} + \Delta_b)]
\end{aligned}$$

Como $\breve{s} = \frac{\underline{s} + \overline{s}}{2}$, então:

$$\begin{aligned}
\breve{s} = & \frac{(\breve{w}_1 - \Delta_1)\underline{x}_1 + (\breve{w}_2 - \Delta_2)\underline{x}_2 + \breve{b} + \Delta_b +}{2} \\
& \frac{(\breve{w}_1 + \Delta_1)\overline{x}_1 + (\breve{w}_2 - \Delta_2)\overline{x}_2 + \breve{b} - \Delta_b}{2} \\
\breve{s} = & \frac{\breve{w}_1(\overline{x}_1 + \underline{x}_1) + \breve{w}_2(\overline{x}_2 + \underline{x}_2) + 2\breve{b} -}{2} \\
& \frac{\Delta_1(\underline{x}_1 - \overline{x}_1) - \Delta_2(\overline{x}_2 + \underline{x}_2)}{2}
\end{aligned}$$

$$\breve{s} = \breve{w}_1\breve{x}_1 + \breve{w}_2\breve{x}_2 + \breve{b} + \frac{\Delta_1(\underline{x}_1 - \overline{x}_1)}{2} - \Delta_2\breve{x}_2 \tag{A.7}$$

Para os seguintes casos, o resultado é igual:

- Se $sign(X_2) = 0$,$sign(W_1) = sign(W_2) = -1$ e $sign(X_1) = 1$

- Se $sign(W_1) = 1$, $sign(W_2) = sign(X_1) = -1$ e $sign(X_2) = -1$.

6. Se $sign(X_1) = sign(X_2) = 0$ e $sign(W_1) = sign(W_2) = -1$
   A saída da rede é dada por:

$$\begin{aligned}[\underline{s}, \overline{s}] =& [(\breve{w}_1 - \Delta_1)\overline{x}_1, (\breve{w}_1 - \Delta_1)\underline{x}_1] + [(\breve{w}_2 - \Delta_2)\overline{x}_2, (\breve{w}_2 - \Delta_2)\underline{x}_2] + \\ & [\breve{b} - \Delta_b, \breve{b} + \Delta_b] \\ [\underline{s}, \overline{s}] =& [(\breve{w}_1 - \Delta_1)\overline{x}_1 + (\breve{w}_2 - \Delta_2)\overline{x}_2 + (\breve{b} - \Delta_b), \\ & (\breve{w}_1 - \Delta_1)\underline{x}_1 + (\breve{w}_2 - \Delta_2)\underline{x}_2 + (\breve{b} + \Delta_b)]\end{aligned}$$

   Como $\breve{s} = \dfrac{\underline{s} + \overline{s}}{2}$, então:

$$\begin{aligned}\breve{s} =& \frac{(\breve{w}_1 - \Delta_1)\underline{x}_1 + (\breve{w}_2 - \Delta_2)\underline{x}_2 + \breve{b} + \Delta_b + }{2} \\ & \frac{(\breve{w}_1 - \Delta_1)\overline{x}_1 + (\breve{w}_2 - \Delta_2)\overline{x}_2 + \breve{b} - \Delta_b}{2} \\ \breve{s} =& \frac{\breve{w}_1(\overline{x}_1 + \underline{x}_1) + \breve{w}_2(\overline{x}_2 + \underline{x}_2) + 2\breve{b} - }{2} \\ & \frac{\Delta_1(\overline{x}_1 + \underline{x}_1) - \Delta_2(\overline{x}_2 + \underline{x}_2)}{2}\end{aligned}$$

$$\breve{s} = \breve{w}_1\breve{x}_1 + \breve{w}_2\breve{x}_2 + \breve{b} + \Delta_1\breve{x}_1 - \Delta_2\breve{x}_2 \tag{A.8}$$

   Para os seguintes casos, o resultado é igual:

- Se $sign(W_1) = 1$, $sign(W_2) = -1$ e $sign(X_1) = sign(X_2) = 0$

7. Se $sign(X_1) = 0$,$sign(W_1) = sign(W_2) = -1$ e $sign(X_2) = 1$
   A saída da rede é dada por:

$$\begin{aligned}[\underline{s}, \overline{s}] =& [(\breve{w}_1 - \Delta_1)\overline{x}_1, (\breve{w}_1 - \Delta_1)\underline{x}_1] + [(\breve{w}_2 - \Delta_2)\overline{x}_2, (\breve{w}_2 + \Delta_2)\underline{x}_2] + \\ & [\breve{b} - \Delta_b, \breve{b} + \Delta_b] \\ [\underline{s}, \overline{s}] =& [(\breve{w}_1 - \Delta_1)\overline{x}_1 + (\breve{w}_2 - \Delta_2)\overline{x}_2 + (\breve{b} - \Delta_b), \\ & (\breve{w}_1 - \Delta_1)\underline{x}_1 + (\breve{w}_2 + \Delta_2)\underline{x}_2 + (\breve{b} + \Delta_b)]\end{aligned}$$

Como $\breve{s} = \frac{\underline{s} + \overline{s}}{2}$, então:

$$\begin{aligned}
\breve{s} =& \frac{(\breve{w}_1 - \Delta_1)\underline{x}_1 + (\breve{w}_2 + \Delta_2)\underline{x}_2 + \breve{b} + \Delta_b +}{2} \\
& \frac{(\breve{w}_1 - \Delta_1)\overline{x}_1 + (\breve{w}_2 - \Delta_2)\overline{x}_2 + \breve{b} - \Delta_b}{2} \\
\breve{s} =& \frac{\breve{w}_1(\overline{x}_1 + \underline{x}_1) + \breve{w}_2(\overline{x}_2 + \underline{x}_2) + 2\breve{b} -}{2} \\
& \frac{\Delta_1(\overline{x}_1 + \underline{x}_1) + \Delta_2(\underline{x_2} - \overline{x}_2)}{2}
\end{aligned}$$

$$\breve{s} = \breve{w}_1\breve{x}_1 + \breve{w}_2\breve{x}_2 + \breve{b} - \Delta_1\breve{x}_1 + \frac{\Delta_2(\underline{x}_2 - \overline{x}_2}{2} \tag{A.9}$$

Para os seguintes casos, o resultado é igual:

- Se $sign(X_1) = 0$, $sign(W_1) = sign(X_2) = -1$ e $sign(W_2) = 1$.

8. Se $sign(W_1) = sign(W_2) = 1$, $sign(X_1) = 0$ e $sign(X_2) = -1$
   A saída da rede é dada por:

$$\begin{aligned}
[\underline{s}, \overline{s}] =& [(\breve{w}_1 + \Delta_1)\underline{x}_1, (\breve{w}_1 + \Delta_1)\overline{x}_1] + [(\breve{w}_2 + \Delta_2)\underline{x}_2, (\breve{w}_2 - \Delta_2)\overline{x}_2] + \\
& [\breve{b} - \Delta_b, \breve{b} + \Delta_b] \\
[\underline{s}, \overline{s}] =& [(\breve{w}_1 + \Delta_1)\underline{x}_1 + (\breve{w}_2 + \Delta_2)\underline{x}_2 + (\breve{b} - \Delta_b), \\
& (\breve{w}_1 + \Delta_1)\overline{x}_1 + (\breve{w}_2 - \Delta_2)\overline{x}_2 + (\breve{b} + \Delta_b)]
\end{aligned}$$

   Como $\breve{s} = \frac{\underline{s} + \overline{s}}{2}$, então:

$$\begin{aligned}
\breve{s} =& \frac{(\breve{w}_1 + \Delta_1)\overline{x}_1 + (\breve{w}_2 - \Delta_2)\overline{x}_2 + \breve{b} + \Delta_b +}{2} \\
& \frac{(\breve{w}_1 + \Delta_1)\underline{x}_1 + (\breve{w}_2 + \Delta_2)\underline{x}_2 + \breve{b} - \Delta_b}{2} \\
\breve{s} =& \frac{\breve{w}_1(\overline{x}_1 + \underline{x}_1) + \breve{w}_2(\overline{x}_2 + \underline{x}_2) + 2\breve{b} +}{2} \\
& \frac{\Delta_1(\overline{x}_1 + \underline{x}_1) + \Delta_2(\underline{x}_2 - \overline{x}_2)}{2}
\end{aligned}$$

$$\breve{s} = \breve{w}_1\breve{x}_1 + \breve{w}_2\breve{x}_2 + \breve{b} + \Delta_1\breve{x}_1 + \frac{\Delta_2(\underline{x}_2 - \overline{x}_2)}{2} \tag{A.10}$$

   Para os seguintes casos, o resultado é igual:

   - Se $sign(W_1) = sign(X_2) = 1$, $sign(W_2) = -1$ e $sign(X_1) = 0$

9. Se $sign(W_1) = sign(W_2) = sign(X_1) = 1$ e $sign(X_2) = 0$

A saída da rede é dada por:

$$\begin{aligned}[\underline{s},\overline{s}] =&[(\breve{w}_1-\Delta_1)\underline{x}_1,(\breve{w}_1+\Delta_1)\overline{x}_1]+[(\breve{w}_2+\Delta_2)\underline{x}_2,(\breve{w}_2+\Delta_2)\overline{x}_2]+\\ &[\breve{b}-\Delta_b,\breve{b}+\Delta_b]\\ [\underline{s},\overline{s}] =&[(\breve{w}_1-\Delta_1)\underline{x}_1+(\breve{w}_2+\Delta_2)\underline{x}_2+(\breve{b}-\Delta_b),\\ &(\breve{w}_1+\Delta_1)\overline{x}_1+(\breve{w}_2+\Delta_2)\overline{x}_2+(\breve{b}+\Delta_b)]\end{aligned}$$

Como $\breve{s}=\dfrac{\underline{s}+\overline{s}}{2}$, então:

$$\begin{aligned}\breve{s} =&\frac{(\breve{w}_1+\Delta_1)\overline{x}_1+(\breve{w}_2+\Delta_2)\overline{x}_2+\breve{b}+\Delta_b+}{2}\\ &\frac{(\breve{w}_1-\Delta_1)\underline{x}_1+(\breve{w}_2+\Delta_2)\underline{x}_2+\breve{b}-\Delta_b}{2}\\ \breve{s} =&\frac{\breve{w}_1(\overline{x}_1+\underline{x}_1)+\breve{w}_2(\overline{x}_2+\underline{x}_2)+2\breve{b}-}{2}\\ &\frac{\Delta_1(\underline{x}_1+\overline{x}_1)+\Delta_2(\overline{x}_2+\underline{x}_2)}{2}\end{aligned}$$

$$\breve{s}=\breve{w}_1\breve{x}_1+\breve{w}_2\breve{x}_2+\breve{b}-\frac{\Delta_1(\underline{x}_1-\overline{x}_1)}{2}+\Delta_2\breve{x}_2 \tag{A.11}$$

Para os seguintes casos, o resultado é igual:

- Se $sign(X_2)=0$, $sign(W_2)=sign(X_1)=1$ e $sign(W_1)=-1$.
- Se $sign(X_2)=0$, $sign(W_1)=sign(X_1)=-1$ e $sign(W_2)=1$.

10. Se $sign(W_1)=sign(W_2)=1$ e $sign(X_1)=sign(X_2)=-1$.
    A saída da rede é dada por:

$$\begin{aligned}[\underline{s},\overline{s}] =&[(\breve{w}_1+\Delta_1)\underline{x}_1,(\breve{w}_1-\Delta_1)\overline{x}_1]+[(\breve{w}_2+\Delta_2)\underline{x}_2,(\breve{w}_2-\Delta_2)\overline{x}_2]+\\ &[\breve{b}-\Delta_b,\breve{b}+\Delta_b]\\ [\underline{s},\overline{s}] =&[(\breve{w}_1+\Delta_1)\underline{x}_1+(\breve{w}_2+\Delta_2)\underline{x}_2+(\breve{b}-\Delta_b),\\ &(\breve{w}_1-\Delta_1)\overline{x}_1+(\breve{w}_2-\Delta_2)\overline{x}_2+(\breve{b}+\Delta_b)]\end{aligned}$$

Como $\breve{s} = \frac{\underline{s} + \overline{s}}{2}$, então:

$$\begin{aligned}\breve{s} =& \frac{(\breve{w}_1 - \Delta_1)\overline{x}_1 + (\breve{w}_2 - \Delta_2)\overline{x}_2 + \breve{b} + \Delta_b +}{2} \\ & \frac{(\breve{w}_1 + \Delta_1)\underline{x}_1 + (\breve{w}_2 + \Delta_2)\underline{x}_2 + \breve{b} - \Delta_b}{2} \\ \breve{s} =& \frac{\breve{w}_1(\overline{x}_1 + \underline{x}_1) + \breve{w}_2(\overline{x}_2 + \underline{x}_2) + 2\breve{b} +}{2} \\ & \frac{\Delta_1(\underline{x}_1 - \overline{x}_1) + \Delta_2(\underline{x}_2 - \overline{x}_2)}{2}\end{aligned}$$

$$\breve{s} = \breve{w}_1\breve{x}_1 + \breve{w}_2\breve{x}_2 + \breve{b} + \frac{\Delta_1(\underline{x}_1 - \overline{x}_1)}{2} + \frac{\Delta(\underline{x}_2 - \overline{x}_2)}{2} \tag{A.12}$$

Para os seguintes casos, o resultado é igual:

- Se $sign(W_1) = sign(W_2) = -1$ e $sign(X_1) = sign(X_2) = 1$

11. Se $sign(W_1) = sign(X_1) = sign(X_2) = 1$ e $sign(W_2) = 0$.
A saída da rede é dada por:

$$\begin{aligned}[\underline{s}, \overline{s}] =& [(\breve{w}_1 - \Delta_1)\underline{x}_1, (\breve{w}_1 + \Delta_1)\overline{x}_1] + [(\breve{w}_2 - \Delta_2)\overline{x}_2, (\breve{w}_2 + \Delta_2)\overline{x}_2] + \\ & [\breve{b} - \Delta_b, \breve{b} + \Delta_b] \\ [\underline{s}, \overline{s}] =& [(\breve{w}_1 - \Delta_1)\underline{x}_1 + (\breve{w}_2 - \Delta_2)\overline{x}_2 + (\breve{b} - \Delta_b), \\ & (\breve{w}_1 + \Delta_1)\overline{x}_1 + (\breve{w}_2 + \Delta_2)\overline{x}_2 + (\breve{b} + \Delta_b)]\end{aligned}$$

Como $\breve{s} = \frac{\underline{s} + \overline{s}}{2}$, então:

$$\begin{aligned}\breve{s} =& \frac{(\breve{w}_1 + \Delta_1)\overline{x}_1 + (\breve{w}_2 + \Delta_2)\overline{x}_2 + \breve{b} + \Delta_b +}{2} \\ & \frac{(\breve{w}_1 - \Delta_1)\underline{x}_1 + (\breve{w}_2 - \Delta_2)\overline{x}_2 + \breve{b} - \Delta_b}{2} \\ \breve{s} =& \frac{\breve{w}_1(\overline{x}_1 + \underline{x}_1) + 2\breve{w}_2\overline{x}_2 + 2\breve{b} -}{2} \\ & \frac{\Delta_1(\underline{x}_1 - \overline{x}_1) + \Delta_2(\overline{x}_2 - \overline{x}_2)}{2}\end{aligned}$$

$$\breve{s} = \breve{w}_1\breve{x}_1 + \breve{w}_2\overline{x}_2 + \breve{b} - \frac{\Delta_1(\underline{x}_1 - \overline{x}_1)}{2} \tag{A.13}$$

Para os seguintes casos, o resultado é igual:

- Se $sign(W_1) = sign(X_2) = 1$, $sign(X_1) = 0$ e $sign(W_2) = 0$.
- Se $sign(W_2) = 0$, $sign(X_1) = sign(X_2) = 1$ e $sign(W_1) = -1$.
- Se $sign(W_2) = 0$, $sign(W_1) = sign(X_1) = -1$ e $sign(X_2) = 1$.

12. Se $sign(W_1) = sign(X_1) = 1$, $sign(W_2) = 0$ e $sign(X_2) = -1$.
A saída da rede é dada por:

$$\begin{aligned}[\underline{s}, \overline{s}] =& [(\breve{w}_1 - \Delta_1)\underline{x}_1, (\breve{w}_1 + \Delta_1)\overline{x}_1] + [(\breve{w}_2 + \Delta_2)\underline{x}_2, (\breve{w}_2 - \Delta_2)\underline{x}_2] + \\ & [\breve{b} - \Delta_b, \breve{b} + \Delta_b] \\ [\underline{s}, \overline{s}] =& [(\breve{w}_1 - \Delta_1)\underline{x}_1 + (\breve{w}_2 + \Delta_2)\underline{x}_2 + (\breve{b} - \Delta_b), \\ & (\breve{w}_1 + \Delta_1)\overline{x}_1 + (\breve{w}_2 - \Delta_2)\underline{x}_2 + (\breve{b} + \Delta_b)]\end{aligned}$$

Como $\breve{s} = \dfrac{\underline{s} + \overline{s}}{2}$, então:

$$\begin{aligned}\breve{s} =& \frac{(\breve{w}_1 + \Delta_1)\overline{x}_1 + (\breve{w}_2 - \Delta_2)\underline{x}_2 + \breve{b} + \Delta_b + }{2} \\ & \frac{(\breve{w}_1 - \Delta_1)\underline{x}_1 + (\breve{w}_2 + \Delta_2)\underline{x}_2 + \breve{b} - \Delta_b}{2} \\ \breve{s} =& \frac{\breve{w}_1(\overline{x}_1 + \underline{x}_1) + 2\breve{w}_2\underline{x}_2 + 2\breve{b} - }{2} \\ & \frac{\Delta_1(\underline{x}_1 - \overline{x}_1) + \Delta_2(\underline{x}_2 - \underline{x}_2)}{2}\end{aligned}$$

$$\breve{s} = \breve{w}_1\breve{x}_1 + \breve{w}_2\underline{x}_2 + \breve{b} - \frac{\Delta_1(\underline{x}_1 - \overline{x}_1)}{2} \tag{A.14}$$

Para os seguintes casos, o resultado é igual:

- Se $sign(W_2) = 0$ e $sign(W_1) = sign(X_1) = sign(X_2) = -1$.

13. Se $sign(W_1) = 1$, $sign(X_1) = sign(X_2) = -1$ e $sign(W_2) = 0$.
A saída da rede é dada por:

$$\begin{aligned}[\underline{s}, \overline{s}] =& [(\breve{w}_1 + \Delta_1)\underline{x}_1, (\breve{w}_1 - \Delta_1)\overline{x}_1] + [(\breve{w}_2 + \Delta_2)\underline{x}_2, (\breve{w}_2 - \Delta_2)\underline{x}_2] + \\ & [\breve{b} - \Delta_b, \breve{b} + \Delta_b] \\ [\underline{s}, \overline{s}] =& [(\breve{w}_1 + \Delta_1)\underline{x}_1 + (\breve{w}_2 + \Delta_2)\underline{x}_2 + (\breve{b} - \Delta_b), \\ & (\breve{w}_1 - \Delta_1)\overline{x}_1 + (\breve{w}_2 - \Delta_2)\underline{x}_2 + (\breve{b} + \Delta_b)]\end{aligned}$$

Como $\breve{s} = \dfrac{\underline{s} + \overline{s}}{2}$, então:

$$\begin{aligned}\breve{s} =& \frac{(\breve{w}_1 - \Delta_1)\overline{x}_1 + (\breve{w}_2 - \Delta_2)\underline{x}_2 + \breve{b} + \Delta_b + }{2} \\ & \frac{(\breve{w}_1 + \Delta_1)\underline{x}_1 + (\breve{w}_2 + \Delta_2)\underline{x}_2 + \breve{b} - \Delta_b}{2} \\ \breve{s} =& \frac{\breve{w}_1(\overline{x}_1 + \underline{x}_1) + 2\breve{w}_2\underline{x}_2 + 2\breve{b} + \Delta_1(\underline{x}_1 - \overline{x}_1)}{2}\end{aligned}$$

$$\breve{s} = \breve{w}_1\breve{x}_1 + \breve{w}_2\underline{x}_2 + \breve{b} + \frac{\Delta_1(\underline{x}_1 - \overline{x}_1)}{2} \tag{A.15}$$

Para os seguintes casos, o resultado é igual:

- Se $sign(W_2) = 0$, $sign(W_1) = sign(X_2) = -1$ e $sign(X_1) = 1$.

14. Se $sign(W_1) = 0$, $sign(W_2) = sign(X_1) = sign(X_2) = 1$
A saída da rede é dada por:

$$\begin{aligned}
[\underline{s}, \overline{s}] =& [(\breve{w}_1 - \Delta_1)\overline{x}_1, (\breve{w}_1 + \Delta_1)\overline{x}_1] + [(\breve{w}_2 - \Delta_2)\underline{x}_2, (\breve{w}_2 + \Delta_2)\overline{x}_2] + \\
& [\breve{b} - \Delta_b, \breve{b} + \Delta_b] \\
[\underline{s}, \overline{s}] =& [(\breve{w}_1 - \Delta_1)\overline{x}_1 + (\breve{w}_2 - \Delta_2)\underline{x}_2 + (\breve{b} - \Delta_b), \\
& (\breve{w}_1 + \Delta_1)\overline{x}_1 + (\breve{w}_2 + \Delta_2)\overline{x}_2 + (\breve{b} + \Delta_b)]
\end{aligned}$$

Como $\breve{s} = \dfrac{\underline{s} + \overline{s}}{2}$, então:

$$\begin{aligned}
\breve{s} =& \frac{(\breve{w}_1 + \Delta_1)\overline{x}_1 + (\breve{w}_2 + \Delta_2)\overline{x}_2 + \breve{b} + \Delta_b + }{2} \\
& \frac{(\breve{w}_1 - \Delta_1)\overline{x}_1 + (\breve{w}_2 - \Delta_2)\underline{x}_2 + \breve{b} - \Delta_b}{2} \\
\breve{s} =& \frac{\breve{w}_1(\overline{x}_1 + \overline{x}_1) + \breve{w}_2(\overline{x}_2 + \underline{x}_2) + 2\breve{b} - }{2} \\
& \frac{\Delta_2(\underline{x}_2 - \overline{x}_2)}{2}
\end{aligned}$$

$$\breve{s} = \breve{w}_1\overline{x}_1 + \breve{w}_2\breve{x}_2 + \breve{b} - \frac{\Delta_2(\underline{x}_2 - \overline{x}_2)}{2} \tag{A.16}$$

Para os seguintes casos, o resultado é igual:

- Se $sign(W_2) = sign(X_1) = 1$, e $sign(W_1) = sign(X_2) = 0$
- Se $sign(W_1) = 0$, $sign(W_2) = sign(X_2) = -1$ e $sign(X_1) = 1$.

15. Se $sign(W_1) = sign(X_2) = 0$, $sign(W_2) = 1$ e $sign(X_1) = -1$.
A saída da rede é dada por:

$$\begin{aligned}
[\underline{s}, \overline{s}] =& [(\breve{w}_1 + \Delta_1)\underline{x}_1, (\breve{w}_1 - \Delta_1)\underline{x}_1] + [(\breve{w}_2 + \Delta_2)\underline{x}_2, (\breve{w}_2 + \Delta_2)\overline{x}_2] + \\
& [\breve{b} - \Delta_b, \breve{b} + \Delta_b] \\
[\underline{s}, \overline{s}] =& [(\breve{w}_1 + \Delta_1)\underline{x}_1 + (\breve{w}_2 + \Delta_2)\underline{x}_2 + (\breve{b} - \Delta_b), \\
& (\breve{w}_1 - \Delta_1)\underline{x}_1 + (\breve{w}_2 + \Delta_2)\overline{x}_2 + (\breve{b} + \Delta_b)]
\end{aligned}$$

Como $\breve{s} = \frac{\underline{s} + \overline{s}}{2}$, então:

$$\begin{aligned}\breve{s} =& \frac{(\breve{w}_1 - \Delta_1)\underline{x}_1 + (\breve{w}_2 + \Delta_2)\overline{x}_2 + \breve{b} + \Delta_b +}{2} \\ & \frac{(\breve{w}_1 + \Delta_1)\underline{x}_1 + (\breve{w}_2 + \Delta_2)\underline{x}_2 + \breve{b} - \Delta_b}{2} \\ \breve{s} =& \frac{\breve{w}_1(\overline{x}_1 + \underline{x}_1) + \breve{w}_2(\overline{x}_2 + \underline{x}_2) + 2\breve{b} +}{2} \\ & \frac{\Delta_2(\overline{x}_2 + \underline{x}_2)}{2}\end{aligned}$$

$$\breve{s} = \breve{w}_1\underline{x}_1 + \breve{w}_2\breve{x}_2 + \breve{b} + \Delta_2\breve{x}_2 \tag{A.17}$$

Para os seguintes casos, o resultado é igual:

- Se $sign(W_1) = 0$, $sign(W_2) = 1$ e $sign(X_1) = sign(X_2) = -1$.
- Se $sign(W_1) = -1$, $sign(W_2) = sign(X_1) = -1$ e $sign(X_2) = 1$.
- Se $sign(W_1) = sign(X_2) = 0$ e $sign(W_2) = sign(X_1) = -1$.
- Se $sign(W_1) = 0$ e $sign(W_2) = sign(X_1) = sign(X_2) = -1$.

Para os seguintes casos a derivação pode ser feita analogamente:

1. Se $sign(X_2) = 0$,$sign(W_1) = sign(W_2) = 1$ e $sign(X_1) = -1$, então:

$$\breve{s} = \breve{w}_1\breve{x}_1 + \breve{w}_2\breve{x}_2 + \breve{b} + \frac{\Delta_1(\underline{x}_1 - \overline{x}_1)}{2} + \Delta_2\breve{x}_2 \tag{A.18}$$

2. Se $sign(W_1) = sign(W_2) = 1$ e $sign(X_1) = sign(X_2) = 0$, então:

$$\breve{s} = \breve{w}_1\breve{x}_1 + \breve{w}_2\breve{x}_2 + \breve{b} + \Delta_1\breve{x}_1 + \Delta_2\breve{x}_2 \tag{A.19}$$

3. Se $sign(W_1) = sign(X_2) = 1$, $sign(W_2) = 0$ e $sign(X_1) = 0$.

$$\breve{s} = \breve{w}_1\breve{x}_1 + \breve{w}_2\overline{x}_2 + \breve{b} + \Delta_1\breve{x}_1 \tag{A.20}$$

4. Se $sign(W_1) = 1$, $sign(W_2) = sign(X_1) = 0$ e $sign(X_2) = -1$, então:

$$\breve{s} = \breve{w}_1\breve{x}_1 + \breve{w}_2\underline{x}_2 + \breve{b} + \Delta_1\breve{x}_1 \tag{A.21}$$

5. Se $sign(W_1) = sign(X_1) = 1$, $sign(W_2) = -1$ e $sign(X_2) = 0$, então:

$$\breve{s} = \breve{w}_1\breve{x}_1 + \breve{w}_2\breve{x}_2 + \breve{b} - \frac{\Delta_1(\underline{x}_1 - \overline{x}_1)}{2} - \Delta_2\breve{x}_2 \tag{A.22}$$

6. Se $sign(W_1) = 0$, $sign(X_1) = sign(W_2) = 1$ e $sign(X_2) = -1$, então:

$$\breve{s} = \breve{w}_1\overline{x}_1 + \breve{w}_2\breve{x}_2 + \breve{b} + \frac{\Delta_2(\underline{x}_2 - \overline{x}_2)}{2} \tag{A.23}$$

7. Se $sign(W_1) = 0$, $sign(W_2) = 1$ e $sign(X_1) = sign(X_2) = -1$, então:

$$\breve{s} = \breve{w}_1\underline{x}_1 + \breve{w}_2\breve{x}_2 + \breve{b} + \frac{\Delta_2(\underline{x}_2 - \overline{x}_2)}{2} \tag{A.24}$$

8. Se $sign(W_1) = sign(W_2) = 0$ e $sign(X_1) = sign(X_2) = 1$, então:

$$\breve{s} = \breve{w}_1\overline{x}_1 + \breve{w}_2\overline{x}_2 + \breve{b} \tag{A.25}$$

9. Se $sign(W_1) = sign(W_2) = 0$, $sign(X_1) = 1$ e $sign(X_2) = -1$, então:

$$\breve{s} = \breve{w}_1\overline{x}_1 + \breve{w}_2\underline{x}_2 + \breve{b} + \Delta_1\breve{x}_1 \tag{A.26}$$

10. Se $sign(W_1) = sign(W_2) = 0$, $sign(X_1) = -1$ e $sign(X_2) = 1$, então:

$$\breve{s} = \breve{w}_1\underline{x}_1 + \breve{w}_2\overline{x}_2 + \breve{b} \tag{A.27}$$

11. Se $sign(W_1) = sign(W_2) = 0$ e $sign(X_1) = sign(X_2) = -1$, então:

$$\breve{s} = \breve{w}_1\underline{x}_1 + \breve{w}_2\underline{x}_2 + \breve{b} \tag{A.28}$$

12. Se $sign(W_1) = 0$, $sign(W_2) = -1$ e $sign(X_1) = sign(X_2) = 1$, então:

$$\breve{s} = \breve{w}_1\overline{x}_1 + \breve{w}_2\breve{x}_2 + \breve{b} + \Delta_2\breve{x}_2 \tag{A.29}$$

13. Se $sign(W_1) = sign(X_2) = 0$, $sign(W_2) = -1$ e $sign(X_1) = 1$, então:

$$\breve{s} = \breve{w}_1\overline{x}_1 + \breve{w}_2\breve{x}_2 + \breve{b} - \Delta_2\breve{x}_2 \tag{A.30}$$

14. Se $sign(X_1) = sign(X_2) = 0$, $sign(W_1) = -1$ e $sign(W_2) = 1$, então:

$$\breve{s} = \breve{w}_1\breve{x}_1 + \breve{w}_2\breve{x}_2 + \breve{b} - \Delta_1\breve{x}_1 + \Delta_2\breve{x}_2 \tag{A.31}$$

15. Se $sign(W_2) = sign(X_1) = 0$, $sign(X_2) = 1$ e $sign(W_1) = -1$, então:

$$\breve{s} = \breve{w}_1\breve{x}_1 + \breve{w}_2\overline{x}_2 + \breve{b} - \Delta_1\breve{x}_1 \tag{A.32}$$

16. Se $sign(W_1) = sign(X_2) = -1$ e $sign(X_1) = sign(W_2) = 0$, então:

$$\breve{s} = \breve{w}_1\breve{x}_1 + \breve{w}_2\underline{x}_2 + \breve{b} - \Delta_1\breve{x}_1 \tag{A.33}$$

Para os seguintes casos, a largura dos intervalos vai influenciar na derivação da equação junto com o sinal dos intervalos, o resultado para cada caso pode ser feito analogamente como nos casos anteriores:

1. $sign(W_1) = sign(X_1) = -1$ e $sign(X_2) = sign(W_2) = 0$.
2. $sign(W_1) = sign(X_1) = 1$ e $sign(W_2) = sign(X_2) = 0$.
3. $sign(W_1) = 1$ e $sign(W_2) = sign(X_1) = sign(X_2) = 0$.
4. $sign(W_1) = 1$, $sign(W_2) = sign(X_2) = 0$ e $sign(X_1) = -1$.
5. $sign(W_1) = sign(X_1) = 0$ e $sign(W_2) = sign(X_2) = 1$.
6. $sign(W_2) = 1$ e $sign(X_1) = sign(X_2) = sign(W_1) = 0$.
7. $sign(W_1) = sign(X_1) = 0$, $sign(W_2) = 1$ e $sign(X_2) = -1$.
8. $sign(W_1) = sign(W_2) = sign(X_2) = 0$ e $sign(X_1) = 1$.
9. $sign(W_1) = sign(W_2) = sign(X_1) = 0$ e $sign(X_2) = 1$.
10. $sign(W_1) = sign(W_2) = sign(X_1) = sign(X_2) = 0$.
11. $sign(W_1) = sign(W_2) = sign(X_1) = 0$ e $sign(X_2) = -1$.
12. $sign(W_1) = sign(W_2) = sign(X_2) = 0$ e $sign(X_1) = -1$.
13. $sign(W_1) = sign(X_1) = 0$, $sign(W_2) = -1$ e $sign(X_2) = 1$.
14. $sign(W_1) = sign(X_1) = sign(X_2) = 0$ e $sign(W_2) = -1$
15. $sign(W_1) = sign(X_1) = 0$ e $sign(W_2) = sign(X_2) = -1$.
16. $sign(W_2) = sign(X_2) = 0$, $sign(X_1) = 1$ e $sign(W_1) = -1$.
17. $sign(W_2) = sign(X_2) = sign(X_1) = 0$ e $sign(W_1) = -1$

Referências Bibliográficas

[Bat91] R. Battiti. First and second-order methods for learning: between steepest descent and newton's method. Technical report, University of Trento, 1991.

[BBdK+98] M. Beheshti, A. Berrached, A. de Korvin, C. Hu, and O. Sirisaengtaksin. On interval weightted three-layer neural networks. *The 31st annual Simulation Symposium*, 1998.

[BH69] A. E. Bryson and Y. Ho. *Applied optimal Control.* Blaisdell, 1969.

[Cun85] Y. Le. Cun. A learning procedure for assymetric threshold network. In *Cognitiva 85*, pages 599–604, 1985.

[dABdO00] F. M. de Azevedo, L. Brasil, and R. L. de Oliveira. *Redes Neurais com aplicações em Controle e em Sistemas Especialistas*, chapter 7. Visual Books Editora, 2000.

[DH73] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis.* New York, 1973.

[dS00] F. P. C. de Souza. Localização e leitura automática de caracteres alfanuméricos - uma aplicação na identificação de veículos. Master's thesis, Universidade Federal do Rio Grande do Sul, 2000.

[Fah88] S. E. Fahlman. An empirical study of learning speed in backpropagation networks. Technical report, Carnegie-Mellow University, 1988.

[FB02] R. Fabrice and C. Brieuc. Multilayer perceptron on interval data. In A. Sokolowski K. Jajuga and H.-H. Bock, editors, *Classification, Clustering, and Data Analysis (IFCS 2002)*, pages 427–434, Cracow, Poland, July 2002. Springer. http://apiacoa.org/publications/2002/ifcs02.pdf.

[Fuk75] K. Fukushima. Cognitron: a selforganising multilayered neural network. *Biological. Cybernetic*, 23:121–134, 1975.

[Gon03] C. H. R. Gonçalves. Utilizando redes neurais artificiais para predição de falhas em links de redes Ópticas. Master's thesis, Universidade Federal do Ceará, 2003.

[Gro76] S. Grossberg. Adaptive pattern recognition and universal recoding: Parallel development and coding of neural feature detectors. *Biological Cybernetics*, 24:121–134, 1976.

[Ham90] S. E. Hampson. *Connectionistic Problem Solving: computational Aspects of Biological Learning*. Birkhauser, 1990.

[Hay94] S. Haykin. *Neural Network A comprehensive foundation*. Macmillian College Publishing Company Inc., 1994.

[Heb49] D. O. Hebb. *The Organization of Behavior*. Wiley-Interscience, New York, 1949.

[HM94] M. Hagan and M. Menhaj. Training feedforward networks with the marquardt algorithm. In *IEEE transactions on Neural Networks*, volume 5, pages 989–993, 1994.

[Hop82] J. J. Hopfield. Neural networks and physical systems with emergent collective properties. In *National Academic Sciencie*, volume 79, pages 2554–2558, 1982.

[IN01] H. Ishibuchi. and M. Nii. Interval-arithmetic-based neural networks. In *Hybrid Methods in Pattern Recognition*, chapter 1, pages 22–28. H. Bunke - University of Bern and A. Kandel - University of South Florida, 2001.

[KLRK98] V. Kreinovich, A. Lakeyev, J. Rohn, and P. Kahl. *Computational Complexity and Feasibility of Data Processing and Interval computations*, pages 1–21. Kluwer Academic publishers, 1998.

[Koh82] T. Kohonen. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 1982.

[Kre] V. Kreinovich. Applications of interval computations. http://www.cs.utep.edu/interval-comp/appl.html. Último acesso em 2004, Outubro,28.

[KS93] V. Kreinovich and O. Sirisaengtaksin. Three layer neural networks are universal approximators for functionals and for control strategies. *Neural Parallel and Scientific Computations*, 1:325–346, 1993.

[Lin88] R. Linsker. Self organization in perceptual network. *Computer*, 21(3):105–117, 1988.

[Lyr03] A. Lyra. *Uma Fundamentação Matemática para o Processamento de Imagens Digitais intervalares.* PhD thesis, Universidade do Rio Grande do Norte UFRN- LECA - PPGEE, 2003.

[Min61] M. L. Minsky. Steps towards artificial intelligence. In *the institute of Radio Engineers*, volume 49, pages 8–32, 1961.

[MM70] J. M. Mendel and R. W. McLaren. *Adaptative, Learning, and Pattern Recognition Systems; Theory and Applications*, chapter Reinforcement-learning control and pattern recognition systems, pages 287–318. Academic Press, New York, 1970.

[Moo62] R. E. Moore. *Interval Arithmetic and Automatic error Analysis in digital Computing.* PhD thesis, Standford University, 1962.

[Moo66] R. E. Moore. *Interval Analysis.* Printice Hall, New Jersey, 1966.

[Moo79] R. E. Moore. *Methods and Applications of Interval Analysis*, page 190. SIAM - Philadelphia, 1979.

[MP43] W. S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115–133, 1943.

[MP69] M. Minsky and S. Papert. *Perceptrons: an introduction to computational geometry.* Massachusetts, 1969.

[Nas99] A. S. Nascimento. Desenvolvendo agentes inteligentes para gerência proactiva de redes atm. Master's thesis, Universidad Federal de Ceará, 1999.

[Neg02] M. Negnewitsky. *Artificial Intelligence a Guide to Intelligent Systems.* Addiso-Wesley, 2002.

[Nol04] A. Di Nola, editor. *Soft Computing - A Fusion of Foundations Methodologies and Applications.* Springer-Verlag Heidelberg, 2004. http://link.springer.de/link/service/journals/00500/index.htm.

[Par85] D. Parker. Learning logic: Casting the cortex of the human brain in silicon. Technical report, Center for Computational Research in Economics and Management Science. MIT, 1985.

[Pea92] B. Pearlmutter. Gradient descent: second order momentum and saturation error. In J. E. Moody, S. Hanson, and R. Lippmann, editors, *Advances in*

Neural Information Processing Systems 2, pages 887–894. Morgan Kaufmann, 1992.

[PEBL] R. E. Patino-Escarcina, B R. C. Bedregal, and A. Lyra. Interval neural network of one layer with supervisioned learning. submetido á TEMA: Tendências em Matemática Aplicada e Computacional.

[PEBL04a] R. E. Patino-Escarcina, B. R. C. Bedregal, and A. Lyra. Interval computing in neural networks:one layer interval neural networks. In *7th International Conference on Information Technology In LNCS Springer-Verlag*, volume 3356, Hyderabad, India Decembro 2004.

[PEBL04b] R. E. Patino-Escarcina, B. R. C. Bedregal, and A. Lyra. Redes neurais intervalares de uma camada. In *Annais do XXVII CNMAC - Congresso Nacional de Matemática Aplicada e Computacional*, page 491, Setembro 2004. Resumo.

[RHW86] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagation errors. *Nature*, 323:533–536, 1986.

[Rie94] M. Riedmiller. Rprop - description and implementation details. Technical report, University of Karlsrube, 1994.

[RM86] D. E. Rumelhart. and J. L. McClelland. *Parallel distribuited processing.* The MIT Press, 1986.

[Ros58] F. Rosenblatt. the perceptron: A probabilistic model for information storage and organization in the brain. *Psychol Rev.*, 65:386–408, 1958.

[SAD94] C. D. Santos, M. S. Aguiar, and G. P. Dimuro. As técnicas intervalares em substituição ao método clássico de estimativa de erro. In *Congresso Nacional de Matemática aplicada e computacional SBMAC*, volume 1, 1994.

[SB04] Benedito A.and R. Santigo and B. Bedregal. Formal aspect of connectivess and optimality in interval computation: Formal aspect of computing. In *submited*, 2004.

[SBW91] R. S. Sutton, A. G. Barto, and R. J. Willians. Reinforcement learning is direct adaptative optimal control. In *the American control conference*, pages 2143–2146, 1991.

[SHM98] C.L. Blake S. Hettich and C.J. Merz. UCI repository of machine learning databases, 1998.

[Sil02] M. M. Macedo Torres Silveira. Teoria fuzzy intervalar: Uma proposta de integração da matemática intervalar à teoria fuzzy. Master's thesis, Universidade Federal do Rio Grande do Norte-UFRN - DIMAP - PPGSC, 2002.

[Sim96] S. J. Simoff. Handling uncertainty in neural networks:an interval approach. In *IEEE International Conference on Neural Networks*, volume 1, pages 606 –610, Junho 1996.

[Tan91] H. Ishibuchi.and H. Tanaka. An extension of the bp-algorithm to interval input vectors-learning from numerical data and expert's knowledge. In *IEEE International Joint Conference on Neural Networks*, volume 2, pages 1588 – 1593, 1991.

[Tho11] E. L. Thorndike. *Animal Intelligence.* Darien, 1911.

[Wer74] P. Werbos. *Beyond Regression: New tools for Prediction and Analysis in the Behavioral Sciences.* PhD thesis, Harvard University, 1974.

[WH60] B. Widrow and M. E. Hoff. Adaptative switching circuits. Technical report, Institute of Radio Engineers, Western Electronic Show and convention, 1960.

[WS85] B. Widrow and S. D. Stearns. *Adaptative Signal Processing.* New Jersey, 1985.

[Zem02] J. Zemke. *b4m A free interval arithmetic toolbox for Matlab based on BIAS version 1.02.004.* Eibendorfer Strabe 38 - 21071 Hamburg, 2002. www.ti3.tu-harburg.de/zemke/b4m/index.html.