UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
CENTRO DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Uma Investigação sobre a Utilização de Processamento Paralelo no Projeto de Máquinas de Comitê

Rafael Marrocos Magalhães

Orientador: Prof. Dr. Jorge Dantas de Melo

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Engenharia Elétrica da UFRN (área de concentração: Engenharia de Computação) como parte dos requisitos para obtenção do título de Mestre em Ciências.

Número de ordem PPgEE: M188
Natal, RN, janeiro de 2007

Divisão de Serviços Técnicos

Catalogação da publicação na fonte. UFRN / Biblioteca Central Zila Mamede

Magalhães, Rafael Marrocos.
Uma Investigação sobre a utilização de processamento paralelo no projeto de máquinas de comitê / Rafael Marrocos Magalhães. - Natal[RN], 2007.
49 f.

Orientador: Jorge Dantas de Melo.

Dissertação (Mestrado) - Universidade Federal do Rio Grande do Norte. Centro de Tecnologia. Programa de Pós-Graduação em Engenharia Elétrica.

1. Redes neurais artificiais - Dissertação. 2. Processamento paralelo - Dissertação. 3. Aprendizagem de máquina - Dissertação. 4. Máquinas de comitê - Dissertação. I. Melo, Jorge Dantas de. II. Universidade Federal do Rio Grande do Norte. III. Título.

RN/UF/BCZM CDU 004.032.26(043.3)

Uma Investigação sobre a Utilização de Processamento Paralelo no Projeto de Máquinas de Comitê

Rafael Marrocos Magalhães

Dissertação de Mestrado aprovada em 5 de janeiro de 2007 pela banca examinadora composta pelos seguintes membros:

Prof. Dr. Jorge Dantas de Melo (orientador) . . . . . DCA/UFRN

Prof. Dr. Adrião Duarte Dória Neto (examinador) . . . . . DCA/UFRN

Prof. Dr. Paulo Sérgio da Motta Pires (examinador) . . . . . DCA/UFRN

Prof. Dr. Guilherme de Alencar Barreto (examinador) . . . . . UFC

Agradecimentos

Agredecimento especial aos meus pais pelo amor e incentivo incondicionais nas minhas escolhas.

Aos meus orientadores, professores Jorge Dantas e Adrião Duarte, pela oportunidade de orientação oferecida, por acreditarem na capacidade de realização do trabalho e pelo acompanhamento durante todo o desenvolvimento do mesmo.

A três amigos com quem contarei durante o resto dessa passagem mundana (fui longe aqui), com os quais aprendi, compartilhei, enfim, vivi durante esse mestrado: Danilo Lima, Márcio Passos e Patric Lacouth.

A todos os professores das disciplinas que cursei, alunos e colegas do LECA com quem convivi, e neste ponto prefiro não citá-los individualmente, para não correr o risco de esquecer alguém :)

À CAPES, pelo apoio financeiro.

Resumo

Este trabalho apresenta uma investigação sobre a utilização de processamento paralelo na construção de redes neurais artificiais com estruturas do tipo máquinas de comitê. Para tanto, realiza-se uma revisão bibliográfica dos principais conceitos em aprendizagem de máquinas, máquinas de comitê e processamento paralelo, bem como alguns trabalhos desenvolvidos na mesma linha de pesquisa. É feito a definição de uma estrutura do modelo paralelo, sua arquitetura, metodologia de desenvolvimento e possibilidades de extensão. Apresenta-se o desenvolvimento de algumas estruturas e os resultados obtidos a partir da aplicação em problemas de aproximação de funções e classificação de padrões.

Palavras-chave: Processamento Paralelo, Aprendizagem de Máquina, Redes Neurais Artificiais, Máquinas de Comitê, Aprendizagem Supervisionada.

Abstract

This work presents an investigation on the use of parallel processing in construction of artificial neural nets with committee machines structures. For this purpose, a bibliographical research are done in main concepts of learning of machines, committee machines and parallel processing, as well as some works developed in the same line. The definition of a parallel structure model, its architecture and development methodology is made, also the possibilities of extensions. The development of some structures and the results gotten from applications in function approximation and pattern classification problems are presented.

Keywords: Parallel Processing, Machine Learnign, Artificial Neural Networks, Ensembles, Supervised Learning.

Sumário

Lista de Figuras

Lista de Tabelas

Nomenclatura

AM:	Aprendizagem de Máquina
AR:	Aprendizagem por Reforço
FBR:	Função de Base Radial
PMC:	Perceptron de Múltiplas Camadas
RNA:	Redes Neurais Aritificiais

Capítulo 1

Introdução

A ciência apresenta a cada dia uma face de seu desenvolvimento e evolução através de resultados que aparecem em todos os campos do saber científico. Dentre muitos destes esforços estão os direcionados ao entendimento do funcionamento do corpo humano e mais especificamente no entendimento do cérebro. Aliados a esses avanços existem linhas de pesquisas, combinadas e proporcionando a prática da interdisciplinaridade, que possibilitam a exploração do estudo sobre a aprendizagem de máquina.

A aprendizagem de máquina pode ser encarada como uma sub-área da inteligência artificial que tem entre seus objetivos a construção de sistemas capazes de adquirir conhecimento de forma automática [Rezende 2005]. A área evoluiu com a criação de diversos métodos e técnicas, vários destes com percepção e inspiração das observações do funcionamento do próprio ser humano. Entre elas, as redes neurais artificiais (bioinspiradas pelo cérebro) e a aprendizagem por reforço (inspirada em comportamento e aprendizagem animal).

Os avanços em campos da engenharia foram igualmente excepcionais. As engenharias elétrica e de computação apresentaram esforços significativos no desenvolvimento de sistemas de comunicação e de processamento da informação. Estes esforços permitiram a criação de computadores e redes de informação igualmente velozes. O surgimento de supercomputadores a partir de redes de estações de trabalho (*clusters*) foi fundamental na exploração de problemas cada vez maiores, permitindo que o conhecimento sobre fenômenos até então desconhecidos fossem aos poucos passíveis de observação e entendimento.

Há pouco menos de uma década a literatura provisionou que quatro ramos de desenvolvimentos em aprendizagem de máquina estariam em destaque no cenário de pesquisas, são eles [Dietterich 1998]:

1. Ensembles de classificadores.
2. Métodos de escalonamento dos algoritmos de aprendizagem supervisionada.
3. Aprendizagem por reforço.
4. Aprendizagem de modelos complexos.

Essa previsão vem se confirmando com a apresentação cada vez maior de publicações e estudos nas áreas mencionadas. Estudos em comitês de máquinas (ensembles) tiveram esforços consideráveis com dados empíricos, formalizações e novos métodos [Optiz &

Maclin 1999, Valentini & Masulli 2002, Estévez et al. 2002, Kuncheva 2004, Dimitrakakis & Bengio 2005]. Novos métodos de treinamento em aprendizagem supervisionada foram desenvolvidos e avaliados [Alves 2002, Batista 2003]. Avanços em arquiteturas e algoritmos de aprendizagem por reforço apareceram com diversas aplicações [Konda & Tsitsiklis 2001, Park & Venayagamoorthy 2003]. Em todos os casos houve avanços atingíveis em grande parte devido ao desenvolvimento em estruturas dotadas de alto poder computacional [Alves 2002, Estévez et al. 2002, Moore et al. 2005].

1.1 O Problema

A literatura proporciona a verificação de alguns métodos utilizados nas tarefas de aprendizagem de máquinas computacionais inteligentes. Os métodos abordam diferentes paradigmas, entre eles a aprendizagem supervisionada e a aprendizagem não supervisionada. Para cada paradigma existe uma variedade de algoritmos e técnicas passíveis de utilização, cada uma com suas características próprias e com considerações favoráveis e desfavoráveis que podem variar de acordo com a natureza do problema, poder computacional ou qualidade dos dados.

Técnicas que foram utilizadas inicialmente de forma separada e em contextos bem distintos, devido a abordagem histórica de seus desenvolvimentos, começaram a ser utilizadas em conjunto na solução de problemas com maior complexidade.

A combinação de métodos de diferentes paradigmas, caso das redes neurais e aprendizagem por reforço, começaram com redes simples com estruturas reduzidas e em seguida com utilização de redes mais complexas e de maior capacidade [Anderson & Hong 1994]. Observa-se que os algoritmos exigem elevados recursos computacionais para tratamento de problemas mais complexos. Isso sugere o estudo da implementação das aplicações a um ambiente de alto poder de processamento (*cluster*).

Petridis e seu grupo de pesquisa [Petridis et al. 1993] utilizam o processamento paralelo no treinamento de redes neurais com arquitetura do tipo perceptrons de múltiplas camadas (PMC). Em outro nível de experimentação, Estevèz et al [Estévez et al. 2002], utilizam o paralelismo em estruturas maiores, a mistura hierárquica de especialistas. Anderson e Hong [Anderson & Hong 1994] por sua vez utilizam a aprendizagem de reforço no treinamento de máquinas de comitê.

As lacunas observadas pelos pesquisadores permitem a realização dos seguintes questionamentos:

- Seria possível a aplicação do paralelismo em diferentes níveis das redes do tipo máquinas de comitê, ou seja, poderia existir o paralelismo nas redes especialistas e ao mesmo tempo na arquitetura como um todo?
- Seria possível paralelizar o algoritmo de aprendizagem formulado por Jacobs e Jordan [Jacobs & Jordan 1991] para uma rede modular, ou ainda, para uma arquitetura semelhante em um ambiente paralelo?

1.2 Motivação

Três condições motivam a elaboração desta proposta de trabalho: o acesso facilitado à máquinas e equipamentos de comunicação de alta velocidade, a existência de referências sobre a construção e manutenção de *clusters* de computadores e a utilização de técnicas e sistemas de programação paralelo.

Redes de computadores formadas por máquinas com processamento razoável são encontradas em diversos centros de pesquisas e laboratórios universitários em vários países. Referências sobre a utilização de redes como estas podem ser encontrados com facilidade na rede mundial de computadores ou na literatura especializada [Pitanga 2004, Sloan 2004]. Métodos e técnicas de programação em sistemas paralelos são apresentados igualmente em diversos meios como a internet, livros, revistas e jornais especializados [Foster 1995, Moore et al. 2005].

Estas três premissas agregadas ao surgimento de novas ferramentas de programação [Collobert et al. 2002, Neumann 2005], vêm possibilitando resultados significativos na área de aprendizagem de máquinas. Desta forma, fazendo uso desses meios, torna-se possível a criação de pesquisas no desenvolvimento de novas ferramentas de programação e novas abordagens na aprendizagem de máquina.

1.3 Objetivos

Este trabalho visa a elaboração de um estudo em arquiteturas de agrupamentos de redes neurais artificiais (máquinas de comitê e redes modulares) fazendo uso de outros paradigmas e algoritmos de aprendizagem de máquina em um ambiente de alto poder de processamento (*cluster*).

É objetivo do trabalho que o estudo desenvolvido pelo mesmo, possa gerar resultados relevantes na área de aprendizagem de máquina. E que este estudo e seus *softwares* possam ser aplicados em áreas correlacionadas como a robótica, controle, classificação, entre outras.

1.4 Organização do texto

O trabalho segue dividido em cinco partes. No segundo capítulo encontra-se a revisão bibliográfica que aborda os fundamentos necessários à apresentação do trabalho desenvolvido, entre eles a aprendizagem de máquina, seus paradigmas e as máquinas de comitê. O capítulo 3 avança com a revisão e apresenta o processamento paralelo e os trabalhos relacionados ao tema abordado neste. O capítulo 4 apresenta o trabalho, inclui a descrição e metodologia de desenvolvimento utilizados. O quinto capítulo apresenta a aplicação do trabalho desenvolvido na resolução de problemas de classificação de padrões e aproximação de funções. O trabalho finaliza-se então no capítulo 6 onde são feitas considerações do que foi desenvolvido e algumas expectativas com relação a possibilidades de extensão.

Capítulo 2

Métodos de Aprendizagem e Redes Neurais

Antes de descrever o trabalho desenvolvido, sua arquitetura e métodos, faz-se necessário a apresentação breve e concisa dos principais pontos e características básicas das técnicas e métodos envolvidos e necessários à apresentação da mesma. Entre as áreas de interesse, estão a aprendizagem de máquina em duas de suas vertentes, aprendizagem supervisionada e a associação entre diversas máquinas denominada de máquinas de comitê.

A Aprendizagem de Máquina, tema em que este trabalho é inserido, é apresentada inicialmente com a seção 2.1, onde se faz uma definição e contextualização da mesma. Na seção 2.2 são apresentadas algumas das técnicas e arquiteturas comumente utilizadas e presentes no estudo da aprendizagem supervisionada, em especial, as redes neurais artificiais. O capítulo segue com a seção 2.3, onde uma segunda vertente da aprendizagem de máquina é abordada: a aprendizagem por reforço, sua arquitetura e modelo de atuação são comentados, assim como alguns de seus algoritmos mais utilizados.

O capítulo finaliza apresentando as máquinas de comitê, que tratam de um estudo sobre diversas maneiras de fazer agrupamentos de máquinas menos complexas. A seção 2.4 comenta sobre as técnicas, vantagens e limitações dessas máquinas.

2.1 Aprendizagem de Máquina

A Aprendizagem de Máquina (AM) trata de um campo do saber científico preocupado em projetar e criar modelos computacionais capazes de mimetizar, aprender e/ou criar conhecimento, técnicas e habilidades encontradas no ser humano e na natureza. Isto é feito através da observação e do modelamento da capacidade desejada. O estudo das técnicas da Aprendizagem de Máquina é feito através da utilização de uma junção interdisciplinar de diversas áreas do conhecimento humano, dentre elas podem ser citadas: estatística, matemática, física, engenharia, inteligência artificial, filosofia, teoria da informação, biologia, ciências cognitivas, complexidade computacional e teoria de controle.

Considera-se que a aprendizagem de máquina concentra-se na construção de programas computacionais que sejam capazes de aprender, e ainda além, melhorar o desempenho de uma habilidade desejada, de forma automática ao passar de um período. Este período deve ser visto como uma série de realizações e experimentações da técnica ou habilidade desejada, também denominado de *experiência*. Desta forma o programa computacional gerado para tal fim pode ser visto como um conjunto pré-estabelecido de regras bem-definidas desenvolvido para solução de um problema de aprendizagem, sendo denominado de *algoritmo de aprendizagem* [Haykin 2001].

Para Mitchel [Mitchell 1997], o algoritmo de aprendizagem é passível de aprendizado se for capaz de aprender uma determinada tarefa $\boldsymbol{t}$ através de uma período de experimentação $\boldsymbol{e}$ desempenhando esta função com um determinado grau de precisão desejado $\boldsymbol{p}$. Assim, o algoritmo passa por um período de experimentação $\boldsymbol{e}$, ou seja, de aprendizado (comumente chamado de treinamento), antes que possa ser utilizado para realizar a tarefa, técnica ou habilidade desejada.

Kecman [Kecman 2001] faz duas observações de importância fundamental ao estudo das técnicas e algoritmos de aprendizagem de máquina. São elas:

- O mundo real é impreciso e incerto.
- Exatidão exige custos.

A primeira observação enfoca a incerteza das realizações das técnicas e habilidades do ser humano e da natureza que deverão ser aprendidas pelos algoritmos de aprendizagem e também sobre a imprecisão das mesmas. De forma a exemplificar, considerando-se a tarefa de reconhecer caracteres manuscritos de uma carta, pode-se observar a imprecisão percebendo-se que o ser humano ao escrever diversas vezes uma mesma palavra agrega pequenas ou grandes diferenças em todas as suas execuções, ou seja, uma letra de uma palavra pode estar em tamanho, intensidade ou deslocada de forma desigual nas variadas vezes em que a palavra é escrita.

A segunda remete à preocupação acerca dos requisitos computacionais exigidos para realização da aprendizagem de máquina. Para que uma determinada técnica de aprendizagem seja bem sucedida, ou seja, que forneça resultados satisfatórios com relação à precisão, um custo elevado será exigido. Estes custos incluem também, mas não somente, a velocidade de processamento, capacidade de memória disponível, complexidade de codificação e até mesmo a qualidade dos dados utilizados para o treinamento desses algoritmos [Batista 2003].

Existe uma variedade de algoritmos criados para a tarefa de aprendizagem de máquina, sendo mais comumente encontrados na literatura: Algoritmos Genéticos, Aprendizagem Bayesiana, Aprendizagem por Reforço, Árvores de Decisão, Máquinas de Vetor de Suporte, Redes Neurais Artificiais [Mitchell 1997, Sutton & Barto 1998, Kecman 2001]. Existem também algumas formas de se fazer interagir o ambiente contendo a habilidade, técnica ou tarefa que se deseja aprender e a máquina de aprendizagem durante a etapa de aprendizado propriamente dito, ou seja a etapa de treinamento. Costuma-se dividir essas formas de interação em dois paradigmas de aprendizagem: aprendizagem com um professor e aprendizagem sem um professor.

A aprendizagem com um professor, designada com mais freqüência como aprendizagem supervisionada, faz uso de informações previamente conhecidas e rotuladas pro-

venientes do ambiente. A aprendizagem sem um professor por sua vez não faz uso de exemplos rotulados, entretanto pode fazer uso de um crítico que guia o desenvolvimento durante o treinamento, caso da Aprendizagem por Reforço, ou apenas utilizando heurísticas e medidas pré-definidas para que o algoritmo se ajuste automaticamente, caso dos métodos não supervisionados.

2.2 Aprendizagem Supervisionada

Neste paradigma de aprendizagem de máquina, os algoritmos responsáveis pelo treinamento da máquina dispõem de conhecimento sobre o ambiente. Este conhecimento é apresentado em tuplas de informações. Cada tupla desta, conhecido como exemplo rotulado, pode ser utilizada durante o treinamento de uma máquina de aprendizagem. Estes exemplos são formadas por um conjunto de informações sobre o dados de entrada, ou seja, atributos acerca do dado utilizado pela máquina e também por uma saída, alvo ou ainda resposta que deverá ser oferecida pela máquina, após o treinamento. Neste trabalho são utilizadas técnicas de Redes Neurais Artificiais como paradigmas de aprendizagem supervisionados.

2.2.1 Redes Neurais Artificiais

As Redes Neurais Artificiais (RNA) podem ser definidas como conjuntos de estruturas de dados e algoritmos que têm suas inspirações em ciências como a biologia, criadas para reter, comparar ou gerar dados relacionados a um sistema. Conforme Haykin [Haykin 2001], as Redes Neurais Artificiais tiveram seu início com o trabalho inicial de dois cientistas, MacCulloch, um psiquiatra e neuroanatomista e Pitts, um matemático. Os dois definiriam o que provavelmente seria uma representação matemática de um neurônio e como funcionaria de forma rudimentar o processamento da informações a partir destes neurônios. Haykin avança com um histórico sobre a evolução das RNA e suas aplicações.

A Figura 2.1 demonstra a representação de um modelo não linear de neurônio artificial. As variáveis x_i representam os dados de entrada do neurônio, os pesos sinápticos são representados pelos w_{ki}, onde o índice i refere-se a entrada i e o índice k ao neurônio k, b_k representa o valor do bias, $\phi(\cdot)$ a função de ativação do neurônio, e a saída do neurônio é representada por y_k.

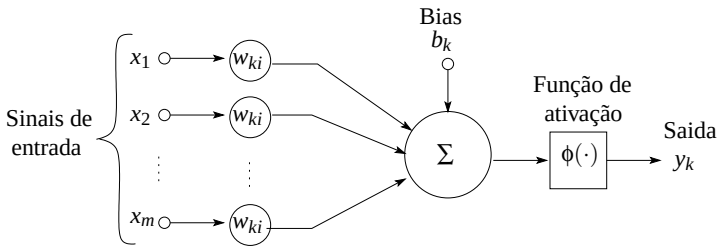

Figura 2.1: Modelo não-linear de um neurônio artificial.

Os dados de entrada, ou seja, o vetor de entrada representado por $\mathbf{x_i} = [x_1, ..., x_m]^T$,

geralmente com $\mathbf{x_i} \in \mathbb{R}$, formam uma representação do estado ou da informação a ser processada pela rede neural. O vetor de entrada é ponderado pelos respectivos pesos sinápticos w_{ki} e então acumulado pelo combinador central representado pela função aditiva juntamente a um bias que tem a função de aumentar ou reduzir o valor do acumulador. A expressão a seguir representa o valor do acumulador:

$$v_k = b_k + \sum_{i=1}^{m} w_{ki} x_i \tag{2.1}$$

O valor do acumulador v_k, também conhecido como campo local induzido ou ainda potencial de ativação, é utilizado pela função de ativação $\phi(\cdot)$ para gerar a saída y_k do neurônio:

$$y_k = \phi(v_k) \tag{2.2}$$

Geralmente a função de ativação $\phi(\cdot)$ é do tipo não linear. Entre as mais comuns estão a função de limiar, função linear por partes e função sigmóide. As funções sigmóides são as mais freqüentemente utilizadas no desenvolvimento das redes neurais.

Existe uma variação nas arquiteturas de algoritmos que fazem uso da estrutura do neurônio artificial, como por exemplo, o perceptron de uma única camada, as redes compostas por múltiplos perceptrons que fazem uso de várias camadas de neurônios artificiais e as redes de base radial, que fazem uso de uma variação do neurônio artificial. Neste trabalho é feito uso dos perceptrons de múltiplas camadas.

Perceptron de Múltiplas Camadas

Redes do tipo Perceptron de Múltiplas Camadas (PMC) também conhecidas como MLP (do inglês *Multi-layer Perceptrons*) são um dos tipos mais difundidos de arquitetura de Redes Neurais Artificiais. A arquitetura de uma rede PMC pode ser observada a partir da Figura 2.2. Os dados fluem da camada mais a esquerda, denominada camada de entrada, atravessam cada uma das camadas seguintes, conhecidas como camadas ocultas, e seu fluxo termina na última camada à direita, a camada de saída. Em cada passagem por entre as camadas, a entrada é ponderada por um peso sináptico e acumulada juntamente ao bias formando o campo local induzido que é então utilizado pela função de ativação em cada neurônio das várias camadas da rede.

As redes neurais de perceptrons de múltiplas camadas costumam interagir com o ambiente, durante sua fase de treinamento, da maneira como é mostrada na Figura 2.3. O ambiente fornece à rede um valor de entrada que o descreve, este valor é utilizado pela rede e pelo professor, por isso aprendizagem supervisionada. A rede fornece então uma saída, denominada saída real, e o professor gera uma saída que seria a resposta desejada, ou resposta ideal, dada a entrada fornecida. As duas saídas são combinadas de forma a gerar um valor de diferença entre as duas. Este valor, denominado sinal de erro, é utilizado então pelo algoritmo de treinamento da rede para reajustar seus parâmetros de forma a convergir para o resultado ideal.

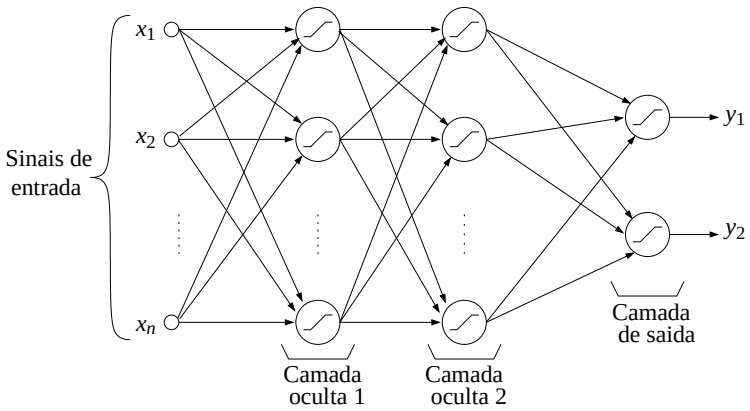

Figura 2.2: Arquitetura genérica de uma rede neural artificial tipo MLP com duas camadas ocultas.

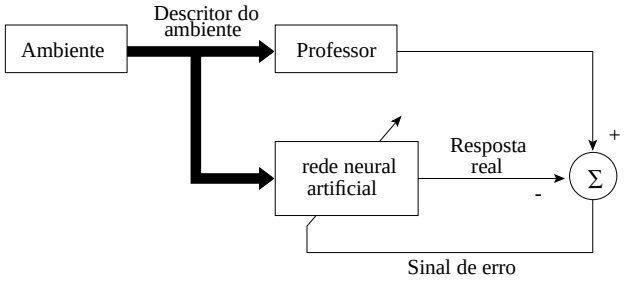

Figura 2.3: Interação entre a rede neural e o ambiente.

Para que uma rede desse tipo possa ser útil, assim como as demais, ela deve prever um algoritmo de treinamento eficiente. Um algoritmo comumente utilizado e já bem definido na literatura é o *algoritmo de retropropagação do erro*. O algoritmo da retropropagação consiste em ciclos de treinamentos compostos por: apresentações aleatórias de vetores de entradas na RNA, obtenção da diferença entre a resposta fornecida pela rede e a resposta desejada, utilização dessa diferença como o argumento de uma alteração realizada sob o conjunto dos pesos sinápticos da rede, sendo esta alteração baseada no gradiente descendente.

Existem outros algoritmos de treinamento de redes do tipo MLP, alguns oriundos do algoritmo da retropropagação, diferentes na forma de convergência ou em como alterar dinamicamente os parâmetros. Os ciclos de treinamento podem fazer a atualização dos pesos sinápticos de forma interativa (*online*), ou seja a cada passo do algoritmo, ou em forma de lote (*batch* ou *offline*), após determinada quantidade de passos.

Redes de Base Radial

Redes Neurais de Funções de Base Radial (FBR) também denominadas de RBF (do inglês, *Radial-basis Function*) são redes estruturadas com três camadas apenas, cada uma com funções bem específicas. A camada de entrada, que se conecta ao ambiente a ser trabalhado, um segunda camada, a única oculta, responsável por transformações não line-

ares dos dados de entrada e a terceira e última camada com saída linear fornece a resposta da rede ao padrão de entrada. Um exemplo de FBR é apresentado na Figura 2.4.

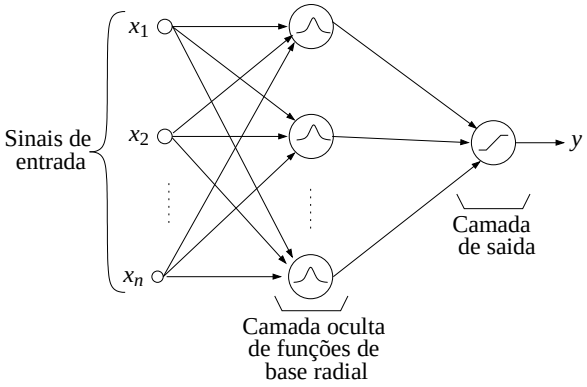

Figura 2.4: Arquitetura de uma rede neural artificial tipo RBF.

A idéia da RBF é a de elevar o conjunto de entrada a um espaço de dimensão superior ao do problema, pois a partir do teorema de Cover [Cover 1965] demonstra-se que a probabilidade de haver uma separação linear de um dado conjunto de valores em alta dimensão é maior. Esta é a função da camada oculta da RBF que é composta por um conjunto de funções que comportam as características necessárias para essas redes. Entre essas funções pode-se citar as: múltiquádricas inversas e gaussianas.

2.3 Aprendizagem por Reforço

A aprendizagem por reforço é inspirada na forma como os seres humanos costumam aprender durante a maior parte de sua vida, isto é, através da interação direta com o meio ambiente. Ao contrário da aprendizagem supervisionada, onde existe um professor, o qual apresenta para a máquina de aprendizagem diversos exemplos e a avalia durante os passos de treinamento, as técnicas de aprendizagem por reforço extraem conhecimento do que deve ser aprendido apenas interagindo com o ambiente.

O termo aprendizagem por reforço apareceu primeiramente em trabalhos de Minsky em 1961, quando várias idéias de inteligência artificial surgiram, e onde a aprendizagem supervisionada motivou a maior quantidade de cientistas, talvez devido aos resultados apresentados até então. A aprendizagem por reforço (AR) tem sua origem em duas linhas de pesquisa. A primeira concentra-se no aprendizado a partir dos métodos de tentativa e erro, que teve início com o estudo psicológico da aprendizagem animal. Já a segunda linha é oriunda dos estudos dos problemas de otimização da teoria de controle e suas soluções através do uso de funções de valores e programação dinâmica [Sutton & Barto 1998].

A Figura 2.5 ilustra um diagrama de blocos de como se dá o funcionamento e o fluxo de informação em agentes de aprendizado que utilizam a aprendizagem por reforço. Esta é uma estrutura básica contendo os principais elementos da arquitetura de AR.

O agente de aprendizagem atua no ambiente através de uma ação. A ação tomada pelo agente depende da política de ações do agente. Em termos de aplicação, a ação remete a

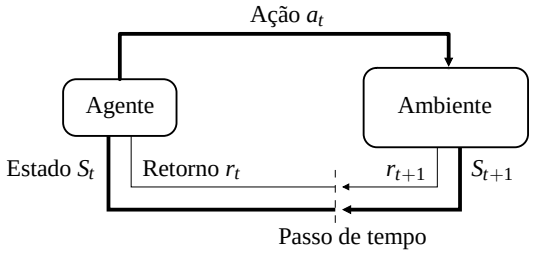

Figura 2.5: Diagrama em blocos de uma arquitetura utilizando aprendizagem por reforço.

que tipo de ato o agente pode tomar no ambiente do problema. Tendo o agente realizado a ação no ambiente, este retorna ao primeiro uma recompensa, este retorno é um valor escalar que representa os efeitos da ação realizada. O estado, ou seja, a percepção que o agente tem sobre o ambiente, poderá ser alterado quando há a realização de uma ação.

O objetivo deste modelo é fazer com que o agente consiga aprender a tomar ações ao longo do tempo de forma a obter os melhores retornos (recompensas) possíveis. Os algoritmos de aprendizagem por reforço não recebem instruções de qual ação ele deve tomar enquanto está aprendendo, este seria o caso dos algoritmos supervisionados. O agente deve aprender a mapear um conjunto de estados e ações que maximize(ou minimize, dependendo do problema) o retorno total fornecido pelo ambiente. Isto deve ser feito de forma que o resultado a longo prazo seja o melhor possível ao agente. Isto implica que nem sempre a ação que retorne a melhor recompensa imediata deve ser tomada.

Uma questão abordada pelos algoritmos de aprendizagem por reforço diz respeito à forma de como o ambiente se comporta, ou seja, se ele é um ambiente dinâmico ou estático. Enquanto os algoritmos supervisionados são utilizados, de forma geral, em ambientes estáticos, os algoritmos de AR estão em constante atualização e devido a isto surge o dilema da diversificação/intensificação. Para garantir que a máquina de aprendizado esteja sempre atualizada sobre as melhores ações, os algoritmos devem estar sempre explorando as inúmeras possibilidades do ambiente. Entretanto é sabido que os algoritmos devem visar também sempre as melhores recompensas (a longo prazo), e isto é feito escolhendo um conjunto de ações previamente estabelecidas como ótimas. Assim, deve existir uma forma de ponderar o quanto a máquina irá diversificar por novas ações e estados dentro do ambiente e quando ela irá intensificar nas ações já definidas e que retornam as melhores recompensas para o agente.

Elementos da aprendizagem por reforço

Existem quatro elementos principais na abordagem de aprendizagem de máquina utilizando-se de algoritmos de aprendizagem por reforço, que são: uma política, uma função de retorno, uma função de valor e o modelo do ambiente [Sutton & Barto 1998].

- **Política:** A política define a forma como o agente vai escolher suas ações dado o estado em que o ambiente se encontra. A política faz um mapeamento estado/ação que será utilizado pelo agente. Ela pode ser implementada, por exemplo, utilizando-se redes neurais artificiais, árvores de decisões, k-vizinhos ou ainda tabelas de asso-

ciação. A política é o elemento primordial do agente de aprendizagem por reforço, pois nela está concentrado o conhecimento sobre o ambiente em que o mesmo atua.

- **Função de Retorno:** A função de retorno trata do objetivo fim da aprendizagem por reforço. Assim como a política, a função de retorno (ou recompensa), também faz um mapeamento, entretanto o mapeamento feito por esta refere-se a atribuição à cada par estado/ação de um valor escalar, contido em $\mathbb{R}$. A função de recompensa não é modificável pelo agente, ao contrário, ela é intrínseca ao ambiente. O agente objetiva obter sempre os maiores valores (considerando que os maiores valores representam as escolhas desejadas). Logo a função de recompensa é utilizada para se estabelecer as melhores políticas.
- **Função de valor:** Como mencionado anteriormente, os algoritmos de aprendizagem por reforço são aplicados de modo a obter um grande retorno ao passar do tempo e não somente de forma imediata. A função de valor caracteriza-se por avaliar o quão bom, para o agente, é estar em um determinado estado (estado/ação) e seguir uma determinada política a partir de então. Neste sentido a função de valor é dependente da política utilizada pelo agente e representa uma estimativa de retorno total esperado (soma de todos os retornos obtido).
- **Modelo de ambiente:** O modelo do ambiente define como o mesmo se modifica e responde às ações realizadas. Pode ser representado pelo 4-uplo $(S, A, P^a_{ss'}, R^a_{ss'})$, onde S representa o conjunto de estados, A o conjunto das ações realizáveis pelo agente, $P^a_{ss'}$ as probabilidades de transições entre o estado s e o estado s' pela realização da ação a, e $R^a_{ss'}$ o retorno obtido na transição do estado s para o estado s' devido a ação a. Sutton apresenta em seu trabalho [Sutton 1992] algumas arquiteturas e modelos de ambientes para máquinas de AR.

2.3.1 Algoritmos de aprendizagem por reforço

Existe três algoritmos básicos para o tratamento de problemas de aprendizagem por reforço: Programação Dinâmica, métodos de Monte Carlo e aprendizagem por Diferenças Temporais. Outros algoritmos são oriundos desses três fundamentais. Todos esses algoritmos são baseados em processos iterativos de aproximação das funções de valor, ou seja, avaliação de uma dada política e posterior melhoria da mesma.

A Programação Dinâmica faz uso de um modelo preciso do ambiente e é baseada no princípio da otimalidade de Bellman. Costuma ser aplicada em problemas de pequenas dimensões, não sendo apropriada para problemas muito complexos ou com domínios de grandes dimensões devido ao elevado custo computacional agregado.

Os métodos de Monte Carlo não requerem um completo modelamento do problema. Torna-se possível a aprendizagem a partir de simulações do ambiente, sem precisar explorar por todos os estados de uma vez. Geralmente é recomendado para tarefas episódicas, isto é, para tarefas que tenham um estado inicial seguido de um número finito de ações até um estado final. Isto é uma grande vantagem se comparado aos métodos de Programação Dinâmica pois além de aprender com simulações, sua aplicação é viável em ambientes onde não é possível ter um modelo completo e bem definido.

Métodos de Diferenças Temporais (DT) são algoritmos que não requerem um modelo

do ambiente, assim como os de Monte Carlo, recomendado também para tarefas episódicas, entretanto não há a necessidade que se complete um ciclo ou um episódio para que sejam incrementados os seus parâmetros. A cada passo o agente toma uma decisão na escolha de uma ação e na obtenção de uma recompensa. Seus parâmetros são atualizados de forma a obter melhores resultados de forma *online*.

Sutton e Barto [Sutton & Barto 1998]apresenta algumas técnicas de utilização dos potenciais dos algoritmos básicos de forma conjunta, usando por exemplo, algoritmos de n-passos para as Diferenças Temporais, ou ainda utilizando-se de traços de elegibilidade. Ambos fazem uso de uma média entre o modelo de Monte Carlo e da DT.

2.4 Máquinas de Comitê

Máquinas de Comitê são estruturas que fazem uso de um conceito comumente utilizado: dividir para conquistar. Este conceito visa dividir uma tarefa grande e complexa em um conjunto reduzido de sub-tarefas que sejam mais fáceis de serem resolvidas e então reagrupadas novamente [Haykin 2001]. A partir desse conceito, as máquinas de comitê podem ser definidas de forma sintética como um conjunto de máquinas de aprendizagem, também denominadas especialistas, cujas decisões são combinadas para uma resposta teoricamente superior a alcançada individualmente, ou seja, uma máquina com desempenho melhor.

Nos últimos anos uma das principais áreas do aprendizado de máquinas diz respeito a caracterização de métodos capazes de construir tais máquinas de comitê [Valentini & Masulli 2002]. Termos como: comitê de máquinas, ensemble de máquinas, redes modulares, fusão de classificadores, combinadores, agregadores, entre outros, têm sido empregados na atribuição de um grupo de máquinas de aprendizado que trabalham juntas na resolução de problemas de AM. Para Valentini e Masulli [Valentini & Masulli 2002], esta variedade de denominações reflete a ausência de uma teoria unificada sobre métodos de comitê e também ao curto tempo de pesquisa da área dado seu surgimento recente comparado ao tempo de estudo em aprendizagem de máquinas.

O interesse nesta área teve um grande impulso devido a disponibilidade cada vez maior de computadores mais velozes, e mais recentemente aos clusters de computadores com preços mais baixos, permitindo dessa forma a implementação e experimentação de máquinas diversificadas.

Conforme Haykin [Haykin 2001] as máquinas de comitê podem ser sub-divididas em duas grandes categorias:

1. Estruturas estáticas. São denominadas máquinas de comitê estáticas todas as formas de se agrupar a resposta proveniente de várias máquinas especialistas por meio de um mecanismo que não utilize diretamente os dados de entrada do problema abordado. Entre os métodos estáticos encontram-se:
    - Média de ensemble, que utiliza diferentes formas de combinações lineares entre as saídas dos especialistas, e
    - Reforço, em que algoritmos de baixa precisão são utilizados de forma a alcançar respostas com precisão elevada.

2. Estruturas dinâmicas. Quando no desenvolvimento de máquinas de comitê o sinal de entrada é utilizado pela rede de passagem (elemento agrupador ou integrador) na construção da resposta global da máquina, então esta é uma máquina dinâmica, pois sua resposta é gerada dinamicamente com relação a entrada da rede. Entre os tipos de redes de comitê dinâmica encontram-se:
   - Mistura de especialistas, onde as repostas produzidas pelos especialistas são combinadas por outra rede, essa última denominada rede de passagem, e
   - Mistura hierárquica de especialistas, extensão do modelo anterior. A diferença está na forma de agrupar o resultado produzido pelos vários especialistas, que pode ser feito por uma série de redes de passagem agrupadas hierarquicamente.

   Enquanto que o princípio de dividir para conquistar é aplicado apenas uma vez na mistura de especialistas, ele é feito diversas vezes na mistura hierárquica de especialistas. Uma denominação muito comum atribuída a esta categoria de máquinas de comitê é o termo "redes modulares", que será utilizado neste trabalho.

2.4.1 Máquinas Estáticas

As máquinas de comitê do tipo média de ensemble, representada na Figura 2.6, são estruturas do tipo estática, onde a saída produzida pela rede é obtida a partir de uma combinação linear das entradas y_i do combinador, que gera então o resultado final $\mathbf{y}$. Estas redes foram motivadas pela tentativa de redução do custo computacional e redução do risco de ajuste excessivo. Supondo que todos os especialistas formam uma única rede neural, logo todas as suas conexões estariam feitas, neste caso o custo de processamento se elevaria e a convergência se tornaria mais difícil de ser atingida em função da grande escala da rede.

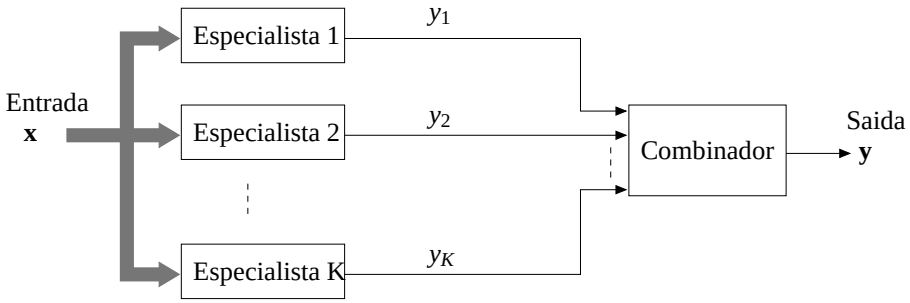

Figura 2.6: Diagrama em blocos da arquitetura de uma rede média de ensemble.

A expectativa das médias de ensembles é que os especialistas alcancem diferentes mínimos locais durante a fase de treinamento. E então forneçam uma saída global melhorada quando combinadas suas respostas individuais. Utilizando-se do dilema bias / variância, discutido por Haykin [Haykin 2001] que uma rede única sofre durante sua aprendizagem, as médias de ensemble tentam reduzir inicialmente o bias treinando excessivamente cada especialista que a compõe, e em seguida a variância agrupando os especialistas, já que cada uma é treinada com inicializações de seus pesos de forma diferenciada.

Uma importante característica no projeto de máquinas de comitê é que a eficiência dessas está relacionada à independência dos erros de cada componente base do todo, ou seja, de cada máquina que constitui a máquina de comitê. De uma maneira geral, para que um ensemble seja eficiente, os especialistas devem atuar em espaços de domínio diversificados, assim poderão apresentar respostas coerentes e não coerentes de maneira diferente uma das outras [Optiz & Maclin 1999].

A abordagem de máquinas estáticas por reforço (do inglês *boosting*, termo mais reconhecido na literatura) difere-se fundamentalmente das médias de ensemble. Enquanto que nas médias de ensemble os especialistas aprendem de um mesmo conjunto de treinamento, podendo entretanto variar as condições de inicialização, na abordagem por reforço o conjunto de treinamento atribuído a um especialista deve ser ao máximo diferente em sua distribuição com relação ao atribuído a outro especialista. A literatura apresenta três técnicas básicas de utilização de máquinas desse tipo:

- Reforço por filtragem, utiliza algoritmos de treinamento fracos, filtrando em cada especialista o conjunto de treinamento, e selecionando os bons resultados e descartando os resultados com baixo desempenho. Esta abordagem exige uma grande quantidade de dados no conjunto de treinamento. Sua vantagem está na baixa utilização de memória.
- Reforço por subamostragem, tenta ultrapassar a necessidade de um conjunto elevado de treinamento fazendo uso de um conjunto fixo de dados em que os exemplos são amostrados novamente de acordo com determinada distribuição de probabilidade.
- Reforço por ponderação, assim como o reforço por subamostragem, o conjunto de treinamento é fixo, a diferença está em ponderar os exemplos de treinamento, assim o erro é calculado em relação a estas ponderações.

Encontram-se entre as técnicas de agregação mais comuns para este tipo de comitê os algoritmos *Bagging* e *Boosting* [Kuncheva 2004, Bittencourt 2005]. *Bagging* é um acrônimo originado da expressão em inglês *Bootstrap Aggregating*. Nessa metodologia de agregação são gerados amostras aleatórias de dados para o treinamento de cada especialista a partir de um grande conjunto de dados, assim pode ocorrer ou não repetição ou ausência de exemplos no treinamento geral. Na combinação dos resultados é utilizado um votação simples onde é atribuído a resposta com maior freqüência.

Nas técnicas de *Boosting* o conjunto de treinamento de seus especialistas depende do desempenho de cada um individualmente. Uma parte do conjunto de treinamento é exposto a um especialista que experimenta-o (treina e classifica). Os resultados que forem menos significativos são então encaminhados a outro especialista que repete o procedimento aos demais. Dessa forma é garantido que o erro gerado pelos especialistas da máquina de comitê seja o mais independente possível, premissa de que o comitê tenha um bom desempenho.

Os modelos de máquinas de comitê apresentadas permitem a criação a partir de máquinas homogêneas. Para criação de comitês fazendo uso de especialistas heterogêneos encontram-se as técnicas de *Voting* e *Stacking*[Bittencourt 2005]. O método de *Voting* (Votação) obtém a resposta de cada especialista heterogêneo e realiza uma votação. Esse

método permite variações em seu sistema de votação, como exemplos, a votação ponderada, votação com confiança diferenciada e votação com predição de probabilidade. Os métodos de *Stacking* fazem uso de um metaclassificador além dos especialistas. Ele utiliza as respostas geradas pelos especialistas para gerar sua própria resposta, pode utilizar esquemas de votação ou ponderação na seleção de qual especialista escolher. Não é exigido ao metaclassificador um algoritmo próprio de treinamento.

2.4.2 Máquinas Dinâmicas

O segundo tipo de máquinas de comitê são as máquinas dinâmicas, que definem-se conforme Osherson (apud [Haykin 2001], p.386):

> *Um rede neural é chamada de modular se a computação realizada pela rede pode ser decomposta em dois ou mais módulos (subsistemas) que operam sobre entradas distintas sem comunicação entre eles. As saídas dos módulos são mediadas por uma unidade integradora que não pode alimentar a informação de volta para os módulos. Em particular, a unidade integradora (1) decide como as saídas dos módulos devem ser combinadas para formar a saída final do sistema e (2) decide quais módulos devem aprender que padrões de treinamento.*

Nas redes modulares, observa-se a utilização de dois paradigmas de aprendizagem, aprendizagem supervisionada e aprendizagem não supervisionada. A aprendizagem supervisionada é observada na utilização de um professor no treinamento da rede, apresentando os exemplos e as respostas desejadas, sem entretanto se preocupar com o que deve ser aprendido especificamente por cada especialista. A aprendizagem não supervisionada por outro lado, é expressada pela competição exercida pelos especialistas em saber o que cada um deve aprender, em que será feita a especialização, em qual espaço do conjunto de dados apresentados pela máquina maior. Esta competição ocorre de forma natural, em que cada máquina torna-se especialista no conjunto em que produzir os melhores resultados.

Algumas vantagens das redes modulares em relação as redes simples dizem respeito a velocidade de aprendizagem. A aprendizagem é acelerada em problemas em que existe uma decomposição natural dos dados em funções mais simples, isto é devido a capacidade intrínseca da rede modular em dividir o espaço de entrada, diferente por exemplo das redes de perceptrons de múltiplas camadas onde o conjunto como um todo deve ser aprendido.

A Figura 2.7 apresenta graficamente um diagrama de blocos do modelo de mistura de especialistas, proposto por Jacobs e Jordan [Jacobs & Jordan 1991]. Na Figura, são apresentados K módulos, cada um representando uma rede chamada de *rede especialista*, e uma uma rede que faz a integração entre elas, denominada *rede de passagem*. Seja a variável $\mathbf{x}$, de dimensão p, a representação do conjunto de exemplos de treinamento e a variável $\mathbf{d}$, de dimensão q, suas respectivas respostas desejadas. A entrada $\mathbf{x}$ é aplicada às redes especialistas e à rede de passagem simultâneamente. Seja $\mathbf{y_i}$ a saída do i-ésimo especialista e g_i a ativação da i-ésima saída da rede de passagem, e por fim $\mathbf{y}$ a saída da rede modular, então pode-se escrever que:

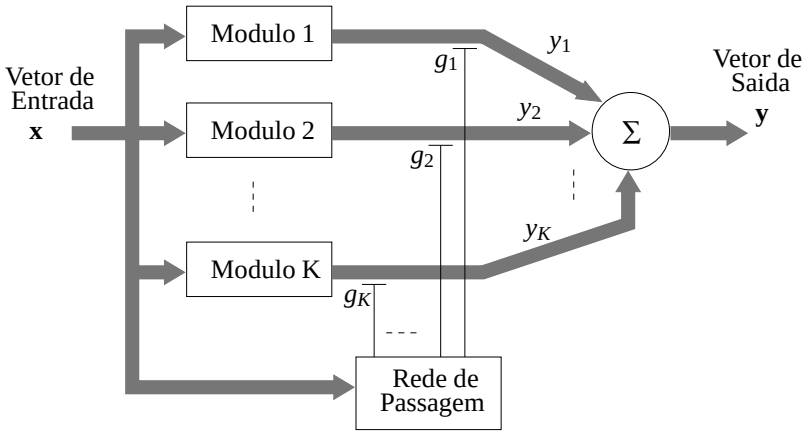

Figura 2.7: Diagrama em blocos da arquitetura de uma rede modular.

$$\mathbf{y} = \sum_{i=1}^{K} g_i \mathbf{y_i} \tag{2.3}$$

O objetivo do algoritmo de treinamento da rede modular deve ser o de modelar a distribuição de probabilidade dos padrões do conjunto de treinamento utilizando sua estrutura modular. A rede de passagem consiste de uma única camada de K neurônios, atribuídos respectivamete a cada especialista da rede. Ao contrário do que ocorre nos especialistas a saída da rede de passagem deve gerar resultados com duas restrições:

$$0 \leq g_i \leq 1 \qquad \text{para todo } i \tag{2.4}$$

e

$$\sum_{i=1}^{K} g_i = 1 \tag{2.5}$$

Considerando que u_i seja o produto interno do vetor de entrada $\mathbf{x}$ pelo vetor de pesos sinápticos a_k da rede de passagem, então a saída g_i da rede pode ser obtida a partir da expressão:

$$g_i = \frac{exp(u_i)}{\sum_{j=1}^{K} exp(u_j)}, \quad i = 1, 2, ..., K \tag{2.6}$$

A Equação 2.6 apresenta uma transformação exponencial normalizada, e pode ser vista como uma generalização da função logística no caso de múltiplas entradas, e é comum ser referida por *função softmax*. A derivação na obtenção do algoritmo pode ser encontrada no trabalho de [Haykin 2001]. O algoritmo sintetizado é apresentado a seguir:

Algoritmo de aprendizagem do modelo de mistura de especialistas

1 *Inicialização*. Atribuir valores iniciais aos pesos das diferentes redes especialistas e a rede de passagem usando valores pequenos e uniformemente distribuidos.

2 *Adaptando os especialistas e a rede de passagem*. Apresenta-se a rede um exemplo de entrada do vetor **x** e sua resposta desejada respectiva **d**, computa-se para a iteração $n = 0, 1, 2, ...$, a saída $i = 1, 2, ..., K$, e o neurônio $m = 1, 2, ..., q$:

$$
\begin{aligned}
u_i(n) &= \mathbf{x}^T(n)\mathbf{a}_i(n) \\
g_i(n) &= \frac{exp(u_i)}{\sum_{j=1}^{K} exp(u_j)} \\
y_i^{(m)}(n) &= \mathbf{x}^T\mathbf{w}_i^{(m)}(n) \\
\mathbf{y}_i(n) &= [y_i^{(1)}(n), y_i^{(2)}(n), ..., y_i^{(q)}(n)]^T \\
h_i(n) &= \frac{g_i(n)exp(-\frac{1}{2}\| \mathbf{d}(n) - \mathbf{y}_i(n) \|^2)}{\sum_{j=1}^{K} g_j(n)exp(-\frac{1}{2}\| \mathbf{d}(n) - \mathbf{y}_i(n) \|^2)} \\
e_i^{(m)}(n) &= d^{(m)}(n) - y_i^{(m)}(n) \\
\mathbf{w}_i^{(m)}(n+1) &= \mathbf{w}_i^{(m)}(n) + \eta h_i(n) e_i^{(m)}(n)\mathbf{x}(n) \\
\mathbf{a}_i(n+1) &= \mathbf{a}_i(n) + \eta[h_i(n) - g_i(n)]\mathbf{x}(n)
\end{aligned}
$$

3 Repetir o passo 2 para todos os exemplos de treinamento disponíveis

4 Repetir os passos 2 e 3 enquanto a rede não chegue na condição de parada.

Uma extensão natural ao modelo de mistura de especialista é o modelo de mistura hierárquica de especialistas. Neste modelo os módulos presentes na Figura 2.7 podem ser vistos como redes modulares completas e a rede de passagem como uma rede de passagem de hierarquia superior. A diferença agora está na divisão do espaço de entradas que é divido em subespaços, e suas informações são distribuídas entre os vários níveis de redes de passagem, para que então sejam utilizados pelas especialistas.

2.5 Conclusão

Apresentou-se uma breve explanação sobre a Aprendizagem de Máquinas, sua fundamentação e concepção além de algumas de suas linhas de pesquisas, dentre as quais as redes neurais artificais, aprendizagem por reforço e as máquinas de comitê. Além disso, foram expostos paradigmas de treinamento e exemplos de arquiteturas e algoritmos de treinamentos. O texto segue adiante com uma descrição das técnicas de processamento paralelo e a exposição de alguns trabalhos que fazem uso destas na aprendizagem de máquinas.

Capítulo 3

Processamento Paralelo e Aplicações

Este capítulo dá continuidade a apresentação de métodos envolvidos e necessários ao desenvolvimento do trabalho. São abordadas as técnicas de processamento paralelo e alguns trabalhos que fazem uso da mesma.

A seção 3.1 é dedicada ao assunto de paralelismo. Seu surgimento, viabilidade e seus desenvolvimentos mais recentes são comentados, além de algumas linguagens e metodologias de programação que também são apresentadas. Finalizando o capítulo, na seção 3.2, são apresentados sínteses de alguns trabalhos em que este se motiva, isto é feito na forma de um histórico minimalista.

3.1 Processamento paralelo

A demanda por processamento de dados, incluindo dados científicos, ao longo dos anos tem sido sempre crescente. A computação de grandes massas de dados exigiu uma transformação na forma como os mesmos deveriam ser processados. Para superar as limitações das tecnologias tradicionais de processamento serializado da informação, tais como a velocidade máxima de processamento ou ainda a quantidade máxima de dados processados em um mesmo instante, foram desenvolvidos equipamentos capazes de processar concomitantemente diversos dados. Intuitivamente o nome dado a este novo paradigma foi *processamento paralelo*.

O processamento paralelo é motivado pela maximização das capacidades de processamento. Esta motivação levou a indústria da computação a desenvolver diversos computadores com múltiplos processadores, em uma diversidade de maneiras de interligação e utilização dos mesmos. Este modelo de computação de grande escala foi utilizado predominantemente por grandes empresas, governos e universidades até o fim da década de 80 [Pitanga 2004].

No início da década de noventa, dada toda a infra-estrutura de redes de microcomputadores, a melhoria do desempenho desses micros, a maior acessibilidade para esse modelo de computação, e ainda os elevados custos de máquinas de grande porte, tem-se que algumas universidades e centros de computação foram levados ao desenvolvimento e estudos de tecnologias de alto desempenho com baixos custos. Assim surgiram os *clusters* de computadores de baixo custo.

A pesquisa em *clusters* de computadores teve início em 1962 em estudos feitos pela

IBM para Defesa Aérea Norte-Americana, entretanto somente em 1993 foi apresentado pela NASA o primeiro *cluster* de computadores de baixo custo desenvolvido com equipamentos comercialmente disponíveis, recebendo o nome de Beowulf [Pitanga 2004, Sloan 2004]. Esta classe de computadores de alto poder computacional pode ser projetada em qualquer ambiente onde se tenha uma rede de microcomputadores, incluindo arquiteturas de redes como a Ethernet e os protocolos de comunicação como o TCP/IP.

Um ambiente de computação paralelo é criado para dois fins específicos: prover alta disponibilidade e prover alto poder de processamento. No provimento de disponibilidade o ambiente paralelo tem o intuito de fazer com que uma aplicação não pare de ser executada, para tanto são adicionados recursos em redundância, seja *hardware* ou *software*. No provimento de capacidade de processamento a estrutura paralela é intensamente utilizada para a função de processar uma grande quantidade de computação.

A programação em ambientes paralelos é realizada de acordo com a arquitetura de *hardware* na qual está implantada. Dependendo da forma como é feita a comunicação entre processador e memória atribui-se um modelo de programação. Os principais modelos de programação incluem a programação por compartilhamento de memória e a programação por passagem de mensagem.

No primeiro caso, os processadores da arquitetura paralela compartilham de uma única memória, logo qualquer variável pode ser acessada/modificada por todos os processadores. Este modelo exige um controle de acesso a memória. Para arquiteturas deste tipo existem linguagens de programação e compiladores adequados que geram códigos automaticamente de forma otimizada, entre os quais: Fortran M, High Performance Fortran - HPF e CC++[Foster 1995].

A metodologia de programação por passagem de mensagem foi desenvolvida e implementada em grande escala na década de 90. Esta metodologia faz uso de um conjunto bem definido de funções capazes de fazer comunicação entre processos durante a execução dos mesmos. Assim mesmos os processos estando em máquinas diferentes em uma rede de computadores eles podem compartilhar informações.

Foster apresenta em seu livro [Foster 1995] uma grande e detalhada metodologia de desenvolvimento de *softwares* em paralelo, desde a decomposição do problema(em dados ou funcional), criação do modelo de comunicação, estudo da granularidade e refinamento. Apresenta ainda, de forma exemplificada, a utilização de linguagens de programação paralela, dentre as quais CC++, Fortran M, HPF e MPI, e metodologias e ferramentas de análise de desempenho.

3.1.1 Noções sobre a programação paralela

Alves [Alves 2002] faz uma síntese de forma didática dos principais pontos a serem observados no desenvolvimento de programas em paralelo. De forma a elucidar esses aspectos, é feita a suposição do seguinte problema de obtenção do valor da função

$$f(x_1, x_2, x_3) = x_1x_2 + x_1x_3 + x_2x_3 + x_2^2 \tag{3.1}$$

A forma natural de explorar o paralelismo seria dividir os produtos em processadores diferentes para então poder somá-los. A Figura 3.1 apresenta como poderia ser executado

esse algoritmo inicial.

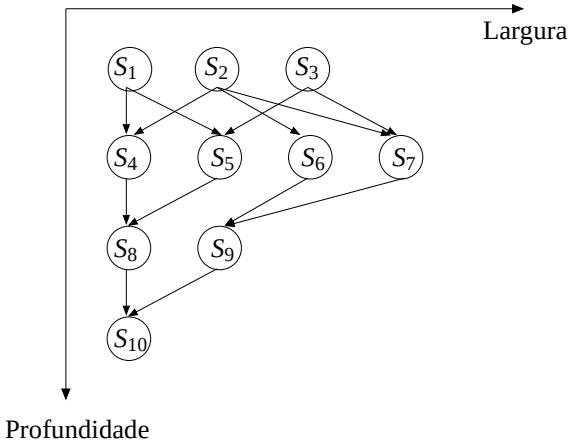

Figura 3.1: Divisão de processamento e comunicação do algoritmo inicial.

No grafo, cada nó representado por S_i constitui um passo de processamento, cada coluna um processador e cada linha um passo de tempo. As arestas entre os nós representam a passagem de informação entre um processador e outro. Dessa forma tem-se que S_1, S_2 *e* S_3 representam a atribuição dos valores x_1, x_2 *e* x_3 respectivamente as variáveis do programa. Na segunda linha, que representa o segundo passo de tempo, S_4 representa o produto entre x_1x_2 , $S_5 = x_1x_3$, $S_6 = x_2^2$ e $S_7 = x_2x_3$, seguindo o raciocínio, $S_8 = S_4 + S_5$, $S_9 = S_6 + S_7$ e por fim o resultado final obtido no último passo de tempo no primeiro processador em que $S_{10} = S_8 + S_9$.

Do grafo podem ser observadas algumas características do algoritmo. A primeira se refere a largura, ou seja o número de colunas, ela diz que quanto maior for este valor de largura maior também será o número de processadores, logo maior possibilidade de aplicar paralelismo. No outro vértice está a profundidade, as linhas, que informam a quantidade de passos de tempo necessários a conclusão do algoritmo, neste caso quanto maior for a quantidade de passos maior o tempo, logo menor desempenho.

Observa-se que esta primeira implementação necessitou de quatro processadores e quatro passos de tempo. Entretanto se o problema fosse analisado antes de uma implementação, poderia ser formulado da seguinte maneira:

$$f(x_1, x_2, x_3) = (x_1 + x_2)(x_2 + x_3) \tag{3.2}$$

esta representa a mesma função inicial apresentada na Equação 3.1 apenas trabalhada algebricamente. Posto desta forma o problema poderia ser resolvido utilizando-se do algoritmo representado na Figura 3.2. Onde S_1, S_2 *e* S_3 representam respectivamente os valores x_1, x_2 *e* x_3. S_4 representa a adição entre x_1 e x_2, $S_5 = x_2 + x_3$, e S_8 o resultado final obtido a partir do produto entre S_4 e S_5.

Comparando os grafos das representações dos dois algoritmos possíveis, percebe-se que o segundo apresenta uma quantidade de colunas inferior, logo menor possibilidade de paralelismo, posto dessa forma seria mais lento. Entretanto o primeiro grafo apresenta uma quantidade de linhas superior a do segundo, ou seja maior tempo para a execução

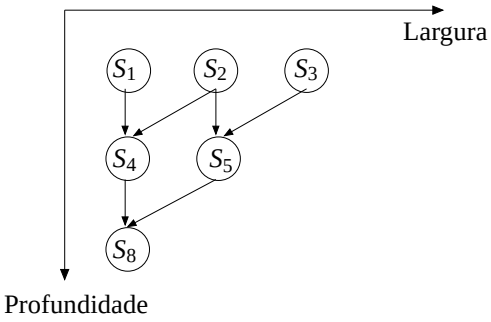

Figura 3.2: Divisão de processamento e comunicação do algoritmo alterado.

do procedimento, logo menor desempenho. Isto leva a necessidade de uma pré análise do problema antes de sua implementação em um algoritmo paralelo.

3.1.2 Desenvolvimento de programas paralelos

O objetivo ao se desenvolver soluções paralelas é explorar a concorrência, a escalabilidade e a localidade. A escalabilidade trata da possibilidade do algoritmo crescer conforme o tamanho do problema e ao número de processadores da arquitetura enquanto que a localidade trata da distribuição da informação entre os processadores. Conforme Foster [Foster 1995] o desenvolvimento de uma aplicação utilizando os princípios da programação paralela envolve as seguintes etapas: particionamento, comunicação, aglomeração e mapeamento (daí gera-se a sigla PCAM). Alves [Alves 2002] adiciona ao método a análise de desempenho, pois dependendo do resultado obtido nesta, torna-se necessário o retorno a um passo anterior, assim a arquitetura pode chamar-se a partir do acrônimo PCAMA. Os passos são descritos a seguir:

- Particionamento. No primeiro passo o problema é observado e procura-se a melhor forma de particionar o problema maior em sub-tarefas menores, de forma a aproveitar a maior possibilidade possível de exploração do paralelismo. Neste momento não são preocupações onde cada tarefa será executada nem por quantos processadores.
- Comunicação. É estabelecido entre as tarefas todas as ligações de passagem de informações necessárias ao algoritmo. Todo o fluxo de informação deve ser analisado quanto à sua permanência, alteração e alcance.
- Aglomeração. Avalia a divisão das tarefas e comunicação. Este processo viabiliza o agrupamento de tarefas de forma a ponderar processamento e comunicação, tentando-se estabelecer o melhor custo/benefício entre ambos. Aqui é feito um estudo de granularidade, que diz respeito a o tamanho das tarefas, alta granularidade significa muitas tarefas, logo muita comunicação.
- Mapeamento. Etapa em que cada tarefa é atribuída a um processador na arquitetura da máquina paralela, tendo o cuidado de acomodar de forma adequada tarefas mais pesadas em processadores de maior porte e comunicações mais intensas entre processadores mais próximos.

- Análise de desempenho. Ponto em que é feito a análise de todo o processo. Dois parâmetros devem ser observados, o tempo de execução e a eficiência.

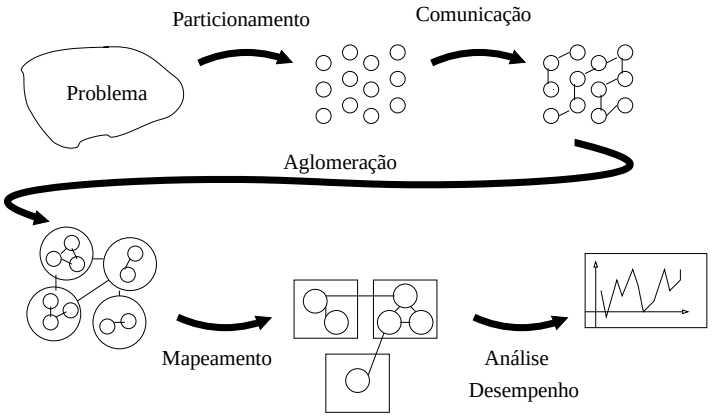

Figura 3.3: Etapas do desenvolvimento de aplicações paralelas a partir do modelo PCAMA.

3.1.3 Análise de desempenho

Sobre a análise de desempenho devem ser observados o ganho da aplicação, também denominado *speed-up* (proveniente da Lei de Amdahl [Addahl 1967]), e a eficiência, medida obtida a partir do ganho com relação a capacidade de processamento, que podem ser obtidos a partir das expressões:

$$\text{Ganho}_P(n) = \frac{T_{es}}{T_{ep}} \tag{3.3}$$

e

$$\text{Eficiência}_P(n) = \frac{Ganho_P(n)}{P} \tag{3.4}$$

onde T_{ep} representa o tempo de execução paralela e é obtido a partir da Equação 3.5 adiante e T_{es} o tempo obtido pela melhor implementação do algoritmo em uma máquina seqüencial, que nem sempre é igual a implementação paralela. P refere-se ao número de processadores e n ao tamanho da instância do problema em questão, pois um mesmo algoritmo paralelo pode apresentar diferentes ganhos para instâncias diferentes de um mesmo problema. T_{ep} pode ser obtido a partir de

$$T_{ep} = \frac{1}{T}(\textit{tempo de execução} + \textit{tempo de comunicação} + \textit{tempo ocioso}) \tag{3.5}$$

onde T representa o número de tarefas da aplicação, o *tempo de execução* é o tempo gasto por cada processador para executar a tarefa, o *tempo de comunicação* é o tempo gasto entre as passagens de informação entre um processador e outro, por fim, o *tempo ocioso*

que é o tempo em que uma tarefa fica esperando pela informação necessária proveniente de outro processador para prosseguir.

3.1.4 Arquitetura *Beowulf*

Os *clusters* de computadores do tipo Beowulf surgiram no início da década de 90. O Beowulf foi reconhecido pela comunidade de desenvolvimento em processamento de alto desempenho como um gênero de arquitetura [Merkey 2004], isto se deve a grande quantidade de centros de processamento de alto desempenho estarem utilizando este tipo de máquina.

Beowulf é o nome dado a um sistema específico de estações de trabalhos conectadas através de uma rede para o provimento de computação paralela e de alto desempenho. Seu desenvolvimento data do ano de 1993, quando Donald Becker e Thomas Sterling, então pesquisadores do Centro de Excelência em Dados Espaciais e Ciências da Informação (CESDIS do inglês *Center of Excellence in Space Data and Information Sciences*), foram incumbidos de desenvolver um sistema de computação de alto desempenho de baixo custo. A concepção baseou-se na utilização de computadores legados, ou seja, fora de uso.

O projeto teve como um de seus protótipos uma máquina formada por 16 computadores com processador Intel 80486 DX4 de 100MHz, 16MB de memória RAM e 512MB de armazenamento. Os micro-computadores foram interligados através de duas placas de rede de 10Mbps, formando dois barramentos independentes, duplicando teoricamente suas capacidades de transmissão e recepção.

A partir deste protótipo diversos outros foram desenvolvidos, validados e aperfeiçoados por inúmeros outros centros de computação. A arquitetura Beowulf está dividida em duas classes principais: I e II. A classe I é definida como uma arquitetura composta *exclusivamente* de máquinas (*hardware*) depreciadas ligados através de uma rede própria, geralmente Ethernet, e utilizando *software* livre, geralmente GNU/Linux.

A Classe II é construída a partir de máquinas escolhidas especificamente para o projeto e não sofrem restrições quanto ao preço ou arquitetura. Hoje os grandes centros de computação de alto desempenho que utilizam *clusters* realizam projetos milionários na elaboração de *beowulfs*. Algumas estruturas de redes comuns nesta classe incluem Myrinet, Gigabit Ethernet e/ou INfiniband. Máquinas mais robustas são incorporadas como placas com mais de um processador, processadores com múltiplos núcleos e com maior quantidade de memória de execução e armazenamento.

Na arquitetura *beowulf* os computadores não podem ser usados para outras aplicações, isto é, eles têm uso restrito à computação de alto desempenho, logo não necessitam de *hardware* de interação como teclado, mouse e mídias diversas. É conveniente distinguir *cluster* de alto desempenho como o *beowulf* de arquiteturas de alta disponibilidade também desenvolvidas com *software* livre. A última não é dedicada ao processamento intensivo mas sim a manutenção de serviços em operação.

3.1.5 Programação por passagem de mensagem

A programação em ambiente de alto desempenho atrai desenvolvedores com idéias das mais diversas. Uma delas, porém, tem sido utilizada de forma mais intensa. Trata-se da programação por passagem de mensagem. A partir do princípio de que o *cluster* é construído de múltiplas estações de trabalho independentes, e que cada uma destas estações é constituída de componentes independentes como, memória, processador e discos de armazenamento, então a programação deve ser realizada pensando-se nessas características.

O programa a ser desenvolvido em um ambiente paralelo é distribuído por uma gama de máquinas independentes e espalhadas entre si. Cada função, etapa ou rotina de um programa em um determinado local e momento pode fazer uso de informações que estejam localizadas em locais diferentes e em momentos diferentes. Observa-se então a necessidade de padronização na maneira de fazer uso de espaço em memória e distribuição de informação.

A programação por passagem de mensagem vem ao encontro da resolução de parte dessas dificuldades. A proposta se baseia em criar em cada processo, etapa ou parte do programa um espaço de memória próprio. Para que seja possível a utilização de uma variável de um processo por outro processo deve existir uma comunicação daquele com este através de funções específicas.

De forma sintética a programação se baseia em simular a existência de inúmero programas diferentes trocando informações por um meio comum. A programação por passagem de mensagem, após ter sido estabelecida, passou por um grande e bem estabelecido processo de aprovação internacional conhecido como fórum MPI[1] (do inglês *Message-Passing Interface*) de padronização. O fórum foi realizado por pesquisadores da comunidade científica, governantes e representantes da indústria de *software* e *hardware*. A primeira realização do Fórum foi entre os anos de 1992 e 1994 a segunda entre 1995 e 1997, em cada uma delas foram realizadas padronizações sobre o funcionamento e formas de utilização em determinadas linguagens de programação computacional.

No primeiro fórum foram estabelecidos e padronizados entre outros: a interface de programação para aplicação (API); permitir a comunicação de forma eficiente em múltiplas plataformas e sistemas heterogêneos; definir a API para as linguagens de programação C padrão ANSI e Fortran77; Prover transparência as falhas ocorridas devido ao meio de comunicação; Permitir a ocorrência de processos para execução em múltiplos processadores.

Já no segundo fórum os esforços foram dedicados ao aperfeiçoamento e extensão das potencialidades da API do MPI. Entre os objetivos alcançados estão: definição para as linguagens de programação C++ ANSI e Fortran90; Obtenção de dados em diferentes localidades; criação, inicialização e termino de processos em tempo de execução, ou seja de forma dinâmica.

Algumas observações podem ser feitas a partir dos fóruns de padronização. Entre as considerações a favor destacam-se as seguintes:

[1] MPI, sigla da expressão em inglês *Message Passing Interface*, com tradução literal *Interface de Passagem de Mensagem*. O Sítio oficial do Fórum está hospedado sob o domínio http://www.mpi-forum.org/

- Portabilidade. Com a padronização permitiu-se o desenvolvimento de aplicações e que estas possam ser portadas tanto em linguagem como em sistema ou ainda em *hardware*. O padrão define de forma universal o modo de chamadas as funções e rotinas do MPI.
- Compiladores. Não exige-se um compilador específico na criação do binário, pois as definições são estabelecidas a partir de chamadas presentes em linguagens de programação padrões e não em compiladores específicos.
- Paralelismo. A padronização permite uma programação flexível com relação ao modo de realizar o paralelismo do programa, seja por função ou por dados.
- Memória. O padrão não faz uso de memória compartilhada, preservando de forma forçada a consistência dos dados durante todo o processamento.
- Acesso. Como a definição foi realizada por membros da comunidade científica, governo e indústria, o padrão tornou-se aberto, desta forma, livre de cobranças intelectuais e mercantis.
- Abordagem. Devido à necessidade de definição do que cada processo deve fazer e o que deve ser transmitido, há a exigência de tratamento adequado da decomposição do problema durante o desenvolvimento paralelo.

Entre as considerações contras destacam-se as seguintes:

- Imposição. O modelo proposto exige a re-escrita dos códigos previamente existentes, pois o modelo não automatiza o paralelismo de programas seriais, gerando assim um paralelismo tudo ou nada.
- Memória. Por não utilizar memória compartilhada o modelo requer uma distribuição dos dados, o que atrasa o processamento quando a informação desejada não se encontra num meio comum.
- Códigos. Como cada processo, função ou tarefa do programa todo deve ser escrito de forma a realizar suas etapas individualmente, os códigos dos programas tornam-se maiores se comparados as suas versões serializadas.
- Complexidade. Dado o modelo de distribuição das operações, torna-se mais complicado saber onde cada informação deve estar em determinado momento.

As definições e características do modelo de comunicação por passagem de mensagem encontram divididas em quatro classes [Alves 2002]: chamadas de gerenciamento e controle, comunicação entre pares, comunicações e operações entre grupos e, por último, as arbitrariedades.

As chamadas de gerenciamento tratam da inicialização e identificação de processos, criação de grupos e controle dos processos. Geralmente são utilizadas no início e fim dos códigos gerados. A segunda classe estabelece as comunicações entre dois processos, ou seja, comunicações ponto-a-ponto. Essas comunicações podem ser síncronas, assíncronas e fazer uso ou não de opções bloqueantes.

As chamadas da terceira classe são responsáveis pela comunicações entre grupos de processos, comunicações *multicasting* e *broadcasting*; ou seja, ponto-multiponto, multiponto-ponto e multiponto-multiponto. Essa classe permite também a realização de operações

entre múltiplas tarefas como, por exemplo, a realização de operações como soma, produto, divisão, subtração, operações lógicas, entre variáveis contidas em tarefas diferentes, agilizando assim o processo de programação.

A última classe trata das arbitrariedades do modelo. Permite a criação de tipos de dados novos, como por exemplos, a criação de estruturas compostas por mais de um tipo primitivo (real, inteiro, caractere) e a possibilidade de comunicação dessas estruturas entre as tarefas.

Após a padronização foram criadas inúmeras implementações das definições do MPI, entre as principais encontram-se, de forma gratuita e com livre acesso as implementações, LAM-MPI, OpenMPI, PICL, PVM, PARMACS, P4 e MPICH. Versões privadas e com licenças podem ser obtidas também, algumas provêm acesso a suporte e manuais, entre elas a MPL, NX e CMMD.

3.2 Trabalhos relacionados

Nesta seção sintetizam-se alguns trabalhos que motivaram o desenvolvimento deste. Utilizando a ordenação cronológica, o primeiro, o terceiro e o último trabalho tratam da aplicação de algoritmos de treinamento de redes neurais em máquinas de processamento paralelo, enquanto que o segundo trabalho trata da aplicação de algoritmos de aprendizagem por reforço no treinamento de máquinas de comitê.

3.2.1 Simulação paralela de redes neurais artificiais

Petridis realizou um trabalho [Petridis et al. 1993] onde comenta sobre o problema que surgiu desde o estabelecimento das redes neurais artificiais quanto à sua implementação. Os métodos utilizados até então incluíam os seguintes:

1. Uso de computadores convencionais e programação utilizando algoritmo seqüenciais.
2. Uso de sistemas com processamento paralelo e técnicas de programação para distribuição da carga computacional entre os processadores.
3. Uso de circuitos VLSI como aceleradores co-processadores para fins específicos.
4. Uso de técnicas ópticas ou híbridas eletro-ópticas.

Para implementação em ambientes de processamento paralelo, foco de seu trabalho, três possibilidades foram elencadas: (a) paralelizar o próprio algoritmo, (b) distribuir partes da rede neural sobre os processadores disponíveis ou, (c)distribuir a informação de aprendizagem.

O sistema utilizado para o processamento paralelo foi o Transputer. As máquinas baseadas na tecnologia Transputer podem ser definidas como uma estrutura que implementa o conceito de MIMD[2] associado a um modelo de memória distribuída, e também implementam diretamente o conceito de computação paralela a partir do modelo de passagem de mensagem.

[2]MIMD é um acrônimo em Inglês que se refere a *Multiple Intruction Multiple Data* conceito de realizar múltiplas instruções simultaneamente em um conjunto de dados distintos.

A rede neural foi implementada da seguinte forma. Em cada processador do computador foi inserido um programa de uma rede, utilizando-se para tanto o conceito de SPMD[3], um processador controlador *mestre* envia para os processadores *escravos* a arquitetura da rede neural que será trabalhada, juntamente com o seu conjunto de pesos iniciais e uma parte do conjunto de treinamento. Cada *escravo* cria a rede de acordo com os parâmetros e é utilizado o algoritmo da retropropagação para a atualização dos pesos. Após cada *escravo* realizar o procedimento sobre todos os dados do conjunto de treinamento que lhe foi atribuído, esse passa o valor de atualização dos pesos juntamente ao erro médio obtido durante o treinamento para o processador controlador. O controlador recebe as informações do *escravo* e lhe envia um novo conjunto de treinamento, e isto se repete até que cada *escravo* tenha realizado o treinamento sobre todo o conjunto de dados, quando então o controlador calcula o valor da atualização média e o do erro médio e repassa-os para os processadores *escravos* para prosseguirem. Este procedimento se repete até que o valor do erro médio consiga atingir o valor desejado ou outra condição de parada estabelecida.

Sobre a medida de desempenho observa-se que o ganho obtido (*speed-up*) é consideravelmente inferior ao valor teórico, conforme aumenta-se a quantidade de processadores na divisão da carga de processamento. Isto ocorre devido a quantidade de iterações necessárias com o acréscimo de processos *escravos*. O mesmo pode ser observado quando o tamanho dos blocos do conjunto de treinamento que é entregue a cada *escravo* tem seu tamanho reduzido. Os piores desempenhos são obtidos com valores elevados ou reduzidos excessivamente para o tamanho deste bloco. Para valores elevados o desempenho cai devido a sobrecarga na comunicação, ao passo que para valores reduzidos o desempenho torna-se inferior ao atraso excessivo na atualização dos pesos sinápticos da rede.

Foram realizados testes com 28 processadores, na aproximação da função $f(x) = sen(x)$ definida $x \in [0,360]$ com 450 exemplos no conjunto de treinamento. A rede tinha arquitetura com uma entrada, duas camadas ocultas, com 10 e 5 neurônios respectivamente, e uma saída, e taxa de aprendizagem $\eta = 0,4$. Sendo realizado em apenas um processador o algoritmo convergiu em $1200,37$ segundos e foram necessários 138 iterações (épocas). Executado em paralelo, Petridis fornece dois gráficos em que são apresentados resultados com desempenho superiores. O primeiro variando o número de iterações necessárias à convergência em relação ao tamanho do bloco do conjunto de treinamento. O segundo variando o tempo necessário em relação novamente ao tamanho do bloco. Observam-se resultados convergindo em torno de 75 épocas e aproximadamente 150 segundos.

Petridis realiza ainda alterações na atualização e faz algumas observações com respeito aos resultados obtidos. O ganho obtido com a paralelização do algoritmo ficou entre $1,88$ e $1,98$, dependendo do tamanho da rede e o número de padrões do conjunto de treinamento.

[3]SPMD é um acrônimo em inglês de *Single Program Multiple Data*, isto significa que o mesmo programa é distribuído sobre todos os processadores e o que os faz diferentes está no conjunto de dados que serão processados.

3.2.2 Aprendizagem por reforço e máquinas de comitê

Anderson e Hong apresentam um trabalho [Anderson & Hong 1994] sobre a utilização de máquinas de comitês em problemas de aprendizagem com reforço. Revisam alguns trabalhos que costumam utilizar redes neurais na solução de problemas de controle, entretanto usualmente utilizando redes neurais simples. Logo, o desempenho do sistema depende de vários fatores como a estrutura da rede, seu tamanho, o tamanho do problema que será aplicado, a quantidade de dados para o treinamento, o tipo de neurônio das redes entre outras. Comentam que geralmente redes maiores são capazes de aprender tarefas mais complexas, exigindo porém um tempo de treinamento muito superior.

A solução para problemas maiores faz uso de redes modulares (máquinas de comitê), que podem aprender de forma mais rápida problemas que podem ser decompostos. Comentam que as redes neurais foram utilizadas inicialmente apenas em problemas de aprendizagem supervisionada. Propõem então a utilização de redes modulares em problemas de aprendizagem por reforço utilizando o algoritmo Q-Learning para o treinamento.

Os autores sintetizam os conceitos básicos da aprendizagem por reforço, do algoritmo Q-Learning [Sutton & Barto 1998], e das redes modulares. No contexto do trabalho as redes modulares assumem duas possibilidades de agrupamento, uma com uma rede de passagem que apenas pondera as saídas provenientes das redes especialistas a partir de um conhecimento *a priori* inserido pelo utilizador, e outra em que a rede de passagem aprende a escolher a melhor rede, logo também participa do treinamento, não necessitando desta forma de um conhecimento *a priori*.

A contribuição aparece na reformulação de como treinar a rede modular a partir de técnicas de aprendizagem por reforço. Os pesos das redes são treinados utilizando o algoritmo Q-Learning de forma a maximizar a função de valor.

Quando a rede de passagem não é do tipo fixa, a rede modular está apta a aprender tanto a decomposição de todo o problema como o controle de cada sub-problema. Neste caso os pesos das redes especialistas e da rede de passagem são atualizados ao mesmo tempo. A rede utilizada nos especialistas é do tipo RBF, para que fosse possível a comparação com resultados contidos na literatura.

São apresentados como resultados experimentais a realização do experimento do problema clássico do pêndulo invertido. Para uma rede modular com rede de passagem fixa, foram testados dois e quatro especialistas, além de quatro métodos de particionamento: particionamento do espaço de estados de acordo com a posição do carro(do problema do pêndulo), a velocidade do carro, a posição do pêndulo, e a velocidade do pêndulo. Os resultados obtidos utilizando-se a rede modular não ultrapassam os que são obtidos com redes simples, devido provavelmente ao tipo do problema e da sua simplicidade. Enfatizam que foi possível adaptar um método de aprendizagem por reforço em redes modulares, e que devem ser realizados testes com outros tipos de redes neurais e com variação dos parâmetros de sua configuração.

3.2.3 Um algoritmo paralelo escalonável para o treinamento de misturas hierárquicas de especialistas

Estevèz et al [Estévez et al. 2002], apresentam uma proposta de um algoritmo paralelo e escalonável de aprendizagem de redes modulares do tipo mistura hierárquica de especialistas. O estudo é dividido em duas etapas, a primeira consiste no estudo do algoritmo e a sua paralelização, enquanto que a segunda parte estuda a comunicação como forma de melhorar o desempenho.

Comentam que apesar da característica inerente às redes neurais de que são modelos massivamente paralelos, sua implementação em um ambiente de processamento paralelo não se demonstra com facilidade. Diversas formas de paralelizar as redes se apresentam levando vantagem em sua estrutura, ou da sua forma matricial de operacionalização e ainda sobre a paralelização da apresentação dos dados. Três são os níveis de paralelismo dos algoritmos: paralelismo das conexões, paralelismo dos neurônios e paralelismo do conjunto de treinamento. Define-se uma taxonomia para os esquemas de paralelização do algoritmo da retropropagação das redes PMC.

Para estruturas complexas de redes modulares, em que é feito a utilização de especialistas mais complexos com redes PMC e vários níveis de hierarquia, o tempo necessário para treiná-la cresce excessivamente, incentivo pelo qual promovem o trabalho. A estrutura das redes modulares sugere uma estratégia de paralelização de baixa granularidade, isto é, grandes partes das tarefas do algoritmo são agrupados em blocos maiores, evitando-se dessa forma excesso em comunicação.

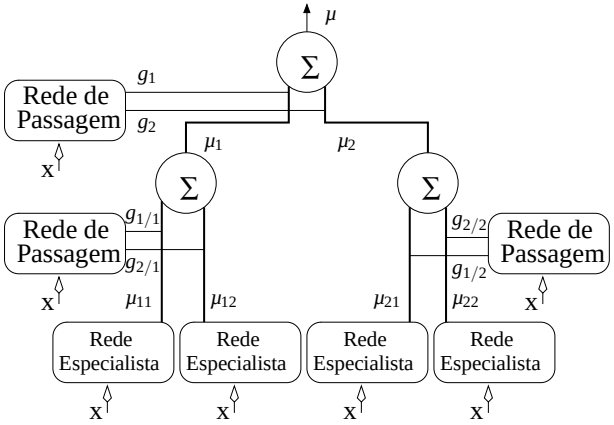

Figura 3.4: Modelo de rede modular com mistura hierárquica de especialistas.

A Figura 3.4 apresenta o diagrama de blocos da arquitetura MHE com a terminologia utilizada pelos autores. A arquitetura deve ser semelhante a uma árvore em que as redes de passagem estão nos nós não terminais, enquanto que os especialistas estão nas folhas das árvores. A tarefa de cada especialista é de aproximar uma função sobre uma região do espaço de entrada. São apresentados quatro especialistas, onde **x** representa o vetor de entradas, μ_{ij} a saída do ij-ésimo especialista, $g_i(x)$ a saída da rede de passagem de maior nível e denota a probabilidade de que a resposta virá da sua ramificação esquerda ou direita e $g_{j/i}(x)$ a saída da rede de menor nível hierárquico, que denota a probabilidade

de qual especialista fornecerá a resposta desejada correta.

Os autores apresentam o algoritmo de aprendizagem seqüencial para o modelo MHE, que é uma extensão do apresentado na seção 2.4.2, onde são feitas divisões conforme forem os níveis hierárquicos. A implementação eficiente de algoritmos paralelos de redes neurais são geralmente baseados na divisão do conjunto de treinamento, em que partes do conjunto são treinadas em diferentes processadores, alterando desta forma o algoritmo seqüencial do treinamento. A implementação paralela proposta, faz uso da modularidade da própria arquitetura da rede. Usa-se a baixa granularidade na programação, isto significa que é assumido que entre duas comunicações consecutivas haverá um processamento excessivo.

A decomposição funcional das tarefas ocorreu de dois tipos de módulos: módulo especialista (contendo toda a rede especialista) e módulo de passagem (que agrega a rede de passagem e toda a computação realizada naquele nó da árvore). Para uma árvore binária com n_e especialistas nas folhas, são necessários $2 \times n_e - 1$ processadores para a implementação de um módulo por processador.

O trabalho segue apresentando cada etapa do algoritmo e também um diagrama dos tempos gastos com comunicação, processamento e ociosidade de cada etapa módulo. Dois experimentos são realizados em uma arquitetura MIMD, utilizando a biblioteca de passagem de mensagens MPI e a linguagem de programação C para que fosse possível fazer a portabilidade do programa para outros *clusters*. O primeiro experimento avalia duas arquiteturas de MHE onde a primeira tem dois especialistas e uma rede de passagem, e a segunda quatro especialistas e três redes de passagem totalizando sete processadores. Um conjunto com 10.000 exemplos é utilizado para o treinamento, onde cada rede trabalha com uma parte do conjunto. Observa-se neste experimento que o ganho (speed-up) com relação ao tamanho do conjunto de treinamento satura rapidamente para o valor máximo de $2,6$ no caso de 2 especialistas (três processadores) e $4,5$ para quatro especialistas (sete processadores).

O segundo experimento utiliza um conjunto fixo de exemplos contendo 8.000 valores e o número de processadores varia entre três e onze. Um resultado interessante foi observado, o ganho obtido utilizando-se o algoritmo proposto conseguiu um elevado e surpreendente valor quando comparado com outros algoritmos de treinamento de redes neurais. Para 11 processadores o valor atingiu 6,6 e cresceu de três processadores a onze de forma linear.

Os autores prosseguem definindo um modelo teórico para o tempo de processamento necessário para execução de arquiteturas maiores de MHE baseado no algoritmo paralelo e observam certa coerência em suas proposições, apresentadas em gráficos. Finalizam o trabalho concluindo que a arquitetura de redes MHE são passíveis de paralelização, que o ganho obtido com o paralelismo é aproximadamente linear, dependendo do número de especialistas e de redes de passagem.

3.2.4 Algoritmo Paralelo Competitivo

Alves [Alves 2002] desenvolve um trabalho de aplicação do paralelismo diferente da maneira clássica. Não utiliza-se do paralelismo sobre o conjunto de dados ou sobre a

estrutura, e sim através de uma forma exploratória, competitiva e colaborativa.

Para a utilização do paralelismo competitivo faz-se necessário um algoritmo adequado. Alves desenvolveu um algoritmo eficiente e escalonável para o treinamento de perceptrons de múltiplas camadas. O algoritmo foi desenvolvido de forma a aproveitar ao máximo as capacidades dos nós componentes da máquina paralela. O algoritmo é baseado no algoritmo da retropropagação e nas técnicas de processamento paralelo aplicadas ao treinamento de redes neurais já existentes. Algumas características clássicas foram mantidas, tais como:

- Cada tarefa paralela executa o treinamento de uma cópia individual da rede.
- Todos os exemplos pertencentes ao conjunto de treinamento da rede são utilizados durante a aprendizagem.

Entre as novas características adicionadas ao treinamento das redes estão:

- Utilização de matriz de ganhos sinápticos próprias em cada tarefa.
- Todos os exemplos do conjunto de treinamento são apresentados em sua totalidade em cada uma das cópias da rede.
- Após um número pré-determinado de iterações do algoritmo de treinamento, as tarefas devem trocar informações sobre o andamento do processo de aprendizagem, verificando a continuidade do treinamento.
- Estruturação simplificada da comunicação para evitar comunicações excessivas inviabilizando o paralelismo. Comunicações do tipo *expansão-e-contração*[Foster 1995] devem ser evitadas.

O fato de que cada rede (tarefa) inicializa com diferentes parâmetros (matrizes de ganho) individuais, permite que cada tarefa faça a busca pelo objetivo em um ponto diferente do espaço de buscas, cobrindo dessa forma uma área superior. Para que cada tarefa seja eficiente em seu treinamento todo o conjunto de padrões deve ser apresentado à rede. Porém, isto inviabilizaria o tempo de processamento, pois tornaria-o igual ao do procedimento serializado. A solução proposta por Alves foi a de dividir o conjunto de treinamento em K partes iguais e utilizá-las de maneira alternada em cada época de treinamento.

O algoritmo utiliza o método da retropropagação da maneira convencional. A cada iteração as matrizes de ganhos são atualizadas localmente em cada tarefa. Com o passar de um número pré-definido de épocas, as tarefas trocam informações sobre as trajetórias tomadas individualmente. Para definição do tempo necessário antes da troca de informação podem ser utilizados o comportamento do erro ao decorrer do tempo ou a definição em um valor determinado de épocas.

Sobre a forma de utilização das informações provenientes das tarefas, três heurísticas são definidas e utilizadas:

- Heurística 1. A cada nova iteração ou quantidade de épocas pré-definidas verifica-se o melhor conjunto de pesos através do menor erro médio obtido sobre o conjunto de treinamento. Esse conjunto escolhido é então difundido sobre todas as outras estruturas. Uma forma de ampliar essa utilização é utilizar dois ou mais conjuntos de pesos melhores.

- Heurística 2. Numa nova iteração uma tarefa determinada utiliza a média geral de todos os conjuntos de pesos sinápticos, dessa forma é estabelecido um caminho médio, aparentemente mais seguro e estável.
- Heurística 3. Em uma iteração verifica-se os melhores conjuntos de pesos sinápticos. Modifica-se esses conjuntos de pesos através de duas formas: acréscimo de uma porção do conjunto de ganhos locais de outra rede; e/ou adiciona-se uma perturbação aleatória de valor reduzido (semelhante a aplicação de mutação na terminologia de aprendizagem de Algoritmos Genéticos) com o objetivo de deslocar o conjunto de pesos sobre o seu universo dimensional, na tentativa de fugir de mínimos locais.

A comunicação utilizada por Alves dá-se através da forma em anel, conforme apresentado na Figura 3.5. A escolha desse tipo de comunicação baseou-se na redução do tempo de comunicação, evitando uma comunicação *fork-and-join*, e a necessidade de sincronismo. Como o conjunto de partições é dividido igualmente entre o número de tarefas, cada tarefa transmite informações referentes ao treinamento aplicado a cada partição. Assim, após a passagem por todos os subconjuntos (partições) dos exemplos de treinamento a rede tem informação sobre o que fazer para atualizar sua matriz de ganhos sinápticos.

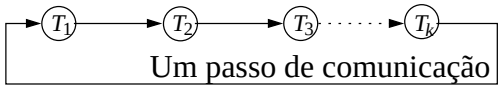

Figura 3.5: Modelo de comunicação em anel.

Alves aplica o novo algoritmo a problemas de aproximação e compressão de imagens. Compara resultados com a arquitetura PMC e o algoritmo da retropropagação convencionais e demonstra melhores resultados. Para a aproximação de funções obtém ganhos de até 5,04 utilizando 8 máquinas paralelas com eficiência de 0,63. Nas aplicações de compressão de imagens são alcançados ganhos de até 7,2 com diferentes taxas de compressão. São analisados também as relações sinal-ruído e outros parâmetros estatísticos de qualidade sobre as imagens.

3.3 Conclusão

Este capítulo apresentou a origem e técnicas do processamento paralelo bem como um modelo de desenvolvimento de software utilizando tal metodologia. Foram apresentados quatro trabalhos onde o processamento paralelo foi utilizado, e também, a utilização de diferentes paradigmas de aprendizagem de máquinas. O capítulo seguinte elenca algumas possiblidades de aplicação do processamento em aprendizagem de redes neurais artificiais e, ainda, apresenta o desenvolvimento das aplicações propostas em um abiente paralelo.

f

Capítulo 4

Algoritmos Paralelos Desenvolvidos

4.1 Introdução

Neste capítulo, serão avaliadas as possibilidades de utilização do processamento paralelo para o projeto de máquinas de comitê, bem como os algoritmos desenvolvidos para este fim. Mais especificamente, busca-se paralelizar os algoritmos de treinamento apresentados na seção 2.4.1 e na seção 2.4.2, verificando-se pontos onde o paralelismo pode ser aplicado, ou seja, na definição de seus especialistas e na estrutura como um todo.

Com esses objetivos em mente, desenvolveu-se um ambiente de programação em que fosse possível avaliar as proposições elencadas, permitindo-se a avaliação do desempenho paralelo não apenas em termos dos parâmetros ganho e eficiência, mas também em termos da qualidade dos resultados.

A Figura 4.1 apresenta uma visão do ambiente computacional paralelo, onde se mostra uma arquitetura de máquina de comitê composta por vários especialistas e um elemento avaliador. Os especialistas são identificados por $Esp_i, i = 1,...,k$, os processadores do ambiente pelos retângulos de linha tracejada nomeados por $p_i, i = 1,...,k$. Observa-se que cada especialista pode estar diretamente associado a um processador (caso, por exemplo, do especialista 1), ou vários especialistas estarem associados a apenas um processador (caso dos especialistas 2 e 3) ou, ainda, um especialista estar associado a vários processadores (caso do especialista k). Da mesma forma, pode-se designar um processador para o elemento avaliador.

O avaliador poderá representar uma rede de passagem (designação do elemento avaliador nas redes modulares) ou um crítico (designação do elemento avaliador nos comitês de especialistas), sendo responsável por como será gerada a resposta final da máquina. Torna-se possível avaliar diversas forma de implementação deste elemento, que pode fazer uso da entrada e/ou da resposta desejada conforme representado pelas linhas pontilhadas da figura.

4.2 Metodologia

A metodologia inclui a organização e utilização de equipamentos computacionais em arquiteturas que promovem a possibilidade de processamento paralelo e implementação de algoritmos em programas que façam uso desta arquitetura.

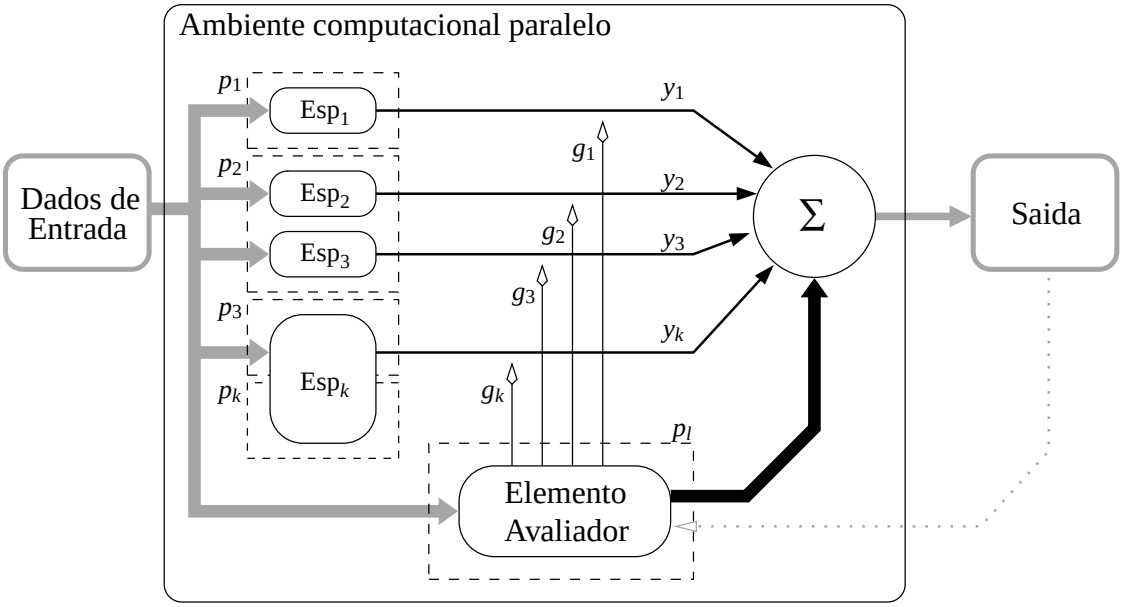

Figura 4.1: Diagrama em blocos da arquitetura proposta genérica.

Para a arquitetura computacional foi utilizada uma rede dedicada de computadores, que formam um sistema de *cluster* do tipo Beowulf. A arquitetura é composta por nove computadores com a seguinte configuração: processadores Pentium 3 com clock de 800 MHz, sendo 1 computador com 256 MB de memória e os demais com 128 MB, uma placa de rede padrão Ethernet e velocidade de 100 Mbps, disco rígido de 20 GB. A rede é conectada por um switch 3Comm de 100 Mbps, e contém sistema de proteção elétrica mantido por 3 no-breaks.

Este *cluster* está dotado de sistema operacional e de um conjunto de bibliotecas necessárias ao desenvolvimento de aplicações paralelas. O sistema operacional que as máquinas utilizam é o GNU/Linux distribuído pelo grupo de desenvolvimento Fedora em sua versão Core 3. Foi utilizado o conjunto de aplicações OSCAR 4.2 para instalação e manutenção dos *softwares*. O OSCAR contém, dentre outros programas: Bibliotecas de passagem de mensagem MPI (versão LAM/MPI e MPICH) e PVM, gerenciamento de desempenho Ganglia (com gráficos de análises) e execução de comandos distribuídos C3.

Para implementação dos algoritmos propostos, foi desenvolvido um *software* com a linguagem de programação C++, devido à orientação a objetos e velocidade. Foi criada uma documentação de utilização do sistema para que seja possível a sua reutilização e aperfeiçoamento por outros usuários desenvolvedores.

O *software* foi elaborado com estruturas de representação, execução, fluxo e controle dos diversos algoritmos de aprendizagem, entre eles a retropropagação, mistura gaussiana associativa e um novo algoritmo proposto, tudo em um ambiente computacional de alto desempenho. Isto possibilita o avanço do software no objetivo de atingir a forma como foi desenvolvido o *framework*[1] de Collobert [Collobert et al. 2002] na aprendiza-

[1]No contexto de desenvolvimento de software o *framework* é uma estrutura de suporte que pode incluir, entre outros, programas de apoio, bibliotecas de códigos, linguagens de *scripts* que permitam sua utilização

gem supervisionada ou ainda o trabalho de Neumann[Neumann 2005] na aprendizagem por reforço.

A avaliação da proposta fez-se com base em resultados de experimentos previamente escolhidos. Cada experimento foi repetido um certo número de vezes para aferição de estatísticas e percepção de estabilidade do comportamento durante a execução.

4.3 Paralelismo nas redes neurais

O estudo da aplicação das técnicas e métodos do processamento paralelo ao problema de treinamento das redes neurais artificiais é observado há mais de uma década através de trabalhos como o de Petridis et al [Petridis et al. 1993], Anderson e Hong [Anderson & Hong 1994], Prechelt [Prechelt 1995], Torresen [Torresen 1996], Provost e Hennessy [Provost & Hennessy 1996], Topping et al [Topping et al. 1998], Li e Silva [Li & da Silva 1999], Mahapatra et al [Mahapatra et al. 1999], Harmamani [Harmanani 2002], Seiffert [Seiffer 2002], Estevèz et al [Estévez et al. 2002], Alves [Alves 2002], Albuquerque et al [Albuquerque et al. 2004] e Smith [Wesley-Smith 2006], Kontár [Kontár 2006]. É possível observar várias possibilidades de exploração do paralelismo em diferentes arquiteturas de redes neurais, onde é possível destacar três formas básicas do paralelismo: decomposição em domínio, decomposição em arquitetura e paralelismo competitivo.

4.3.1 Decomposição em domínio

A decomposição em domínio trata do paralelismo aplicado sobre o conjunto de dados a ser utilizado durante o treinamento da arquitetura, para exemplificar o funcionamento deste modelo de paralelismo será feito o uso do algoritmo da retropropagação.

Supondo que se deseja treinar um rede PMC com um conjunto de treinamento grande, então inicializa-se em K processadores K tarefas, cada uma contendo uma rede de igual arquitetura e todas inicializadas de maneira igual, isto é com o mesmo número de camadas, neurônios, entradas e saídas, com o mesmo valor para os pesos sinápticos e taxas de aprendizagem também. Após a inicialização o conjunto é dividido em K partes aproximadamente iguais e distribuídas entre as tarefas.

Como o algoritmo pode ser executado em modo lote (*batch*), para cada exemplo em cada tarefa é armazenado e acumulado o valor do erro que será utilizado para atualização dos pesos sinápticos da rede. Após a passagem de todos os exemplos daquele subconjunto naquela época, o valor acumulado do erro em cada uma das tarefas é transmitido para uma tarefa mestre que se encarrega de fazer o somatório total e então redistribuir o valor calculado a todas as tarefas novamente, para que elas atualizem seus pesos. Assim todas as redes devem permanecer com o mesmo valor de pesos.

Esta metodologia torna-se interessante quando o conjunto de treinamento é demasiado grande, pois o tempo de processamento tende a reduzir-se bastante. Uma das desvantagens deste modelo é que ele pressupõe a utilização de um modelo de treinamento em lote.

---

na criação e desenvolvimento de um *software* maior. Uma definição mais detalhada pode ser obtida na Wikipédia, sob o domínio http://pt.wikipedia.org/wiki/Framework

Outra desvantagem está associada a forma como se dá a comunicação entre as tarefas do algoritmo, que se centraliza em uma única delas.

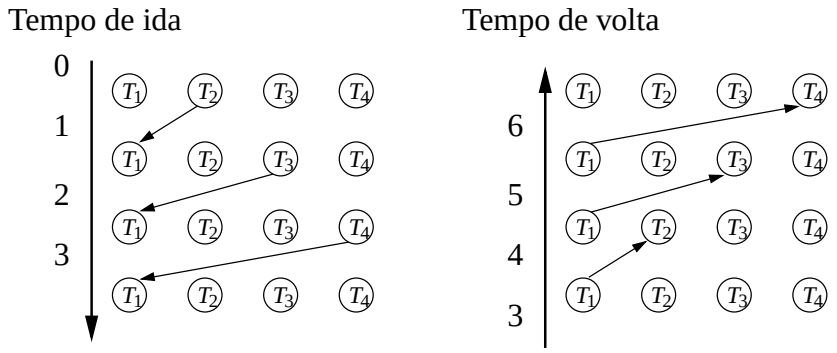

Figura 4.2: Comunicação padrão realizada no paralelismo por decomposição.

A Figura 4.2 apresenta um diagrama da comunicação padrão realizada para um exemplo de 4 tarefas. Deve ser observado que é necessário acumular em uma tarefa e depois distribuir de volta para todas as outras tarefas. Para o caso de 4 tarefas, são necessários 6 etapas de comunicação. No caso de 8 tarefas esse tempo sobe para 14 etapas de comunicação. Este problema de comunicação pode ser reduzido aplicando-se estratégias de comunicação como a de dividir-para-conquistar e a comunicação do tipo *butterfly* [Foster 1995].

A comunicação através da técnica de dividir para conquistar é obtida aplicando a concorrência nas comunicações. Para exemplificar, no caso de 4 tarefas seria possível transmitir informações entre as tarefas T_1 e T_2 e entre as tarefas T_3 e T_4 em uma única etapa, para em seguida realizar-se a comunicação entre T_1 eT_3, o que reduz o número de etapas de comunicação de 6 para 4. No caso de 8 tarefas o número de etapas se reduziria de 14 para 6, bastante significativa.

A decomposição em domínio apresenta uma boa adaptação a arquiteturas com máquinas heterogêneas, isto porque sendo a divisão do processamento em função da quantidade de exemplos pertencentes ao conjunto de treinamento, é possível atribuir uma quantidade reduzida de exemplos para nós de baixo desempenho e, em contrapartida, uma quantidade elevada de exemplos para nós com um alto desempenho de processamento.

4.3.2 Decomposição em arquitetura

A decomposição em arquitetura trata do paralelismo aplicado sobre a estrutura da própria rede. Logo uma infinidade de possibilidades surgem a partir das características próprias de cada rede. No caso de uma rede do tipo PMC o paralelismo poderia ser aplicado entre as camadas, de diferentes formas.

Na primeira forma de decomposição em camadas, cada tarefa seria associada a computação realizada por todos os neurônios de um mesma camada da rede, assim para uma rede com quatro camadas por exemplo, seria possível criar 4 tarefas. Isto é uma restrição que dificulta a escalabilidade do algoritmo, uma vez que o mesmo não poderia ser automaticamente adaptado a uma mudança no número de processadores da máquina paralela.

Se, ao invés de se considerar em cada tarefa a computação associada a todos os neurônios da camada, mas sim a apenas uma parcela deles, as possibilidades de paralelização aumentam, embora a estrutura das comunicações se torne muito mais complexa.

No caso de uma rede de dimensão homogênea, com todas as camadas possuindo um número de neurônios muito próximo, essa estratégia é eficiente em *clusters* homogêneos e ineficientes em *clusters* heterogêneos.

Outra forma de decomposição em arquitetura consiste em considerar agrupamentos de neurônios pertencente a várias camadas (decomposição horizontal), porém a quantidade de comunicação cresce demasiadamente e o sincronismo exigido dificulta sua implementação e aumenta a complexidade algorítmica. No caso limite da decomposição em neurônios individuais, além dos problemas de comunicação inerentes, tem-se ainda a ociosidade de tarefas devido ao sincronismo exigido na obtenção das saídas de cada neurônio.

4.3.3 Paralelismo competitivo

Um novo paradigma de exploração do paralelismo foi apresentado nos trabalhos de Alves [Alves 2002] e de Albuquerque [Albuquerque et al. 2004]. Em ambos os casos, o processamento paralelo não é utilizado apenas para a redução do tempo de processamento, ou seja, na execução de mais computação em menos tempo. A idéia do paralelismo competitivo é de utilizar a computação paralela de forma cooperativa e colaborativa. Em vez de ser feito o treinamento de uma única estrutura da rede, como nos paradigmas anteriores, a computação competitiva realiza o treinamento em inúmeras arquiteturas diferentes ao mesmo tempo.

A idéia é baseada na exploração otimizada dos parâmetros que compõem a rede neural. Alves aplica o paralelismo da seguinte maneira: são geradas K tarefas para o treinamento de uma determinada arquitetura de rede, dado um conjunto de treinamento. Cada tarefa atualiza seus pesos sinápticos durante um certo número de épocas utilizando uma partição do conjunto de treinamento. Em seguida, a partição utilizada é repassada a outra tarefa e uma nova é recebida. Se existem K partições, o processo de treinamento é encerrado quando todas forem utilizadas por cada uma das tarefas. Associada a esta forma de decomposição do conjunto de treinamento, são utilizados diferentes valores dos parâmetros de treinamento em cada tarefa, como a taxa de aprendizagem, por exemplo.

Assim, diferentes regiões do espaço de definição dos pesos podem ser exploradas de forma concorrente, acelerando o treinamento e reduzindo a possibilidade de se cair em mínimos/máximos locais, aumentando a possibilidade de encontrar mínimos/máximos globais.

O trabalho de Albuquerque et al [Albuquerque et al. 2004], trata da utilização do paralelismo e de algoritmos genéticos na definição da arquitetura da rede e na definição de seus pesos sinápticos. Cada tarefa fica incumbida de avaliar uma determinada arquitetura. Logo, quão maior for a quantidade de tarefas e o número de processadores, mais rápida e mais eficiente pode ser a busca no universo de definição dos parâmetros da rede. Em ambos os casos, a aplicação da competitividade apresentou resultados superiores não apenas em tempo como também em qualidade comparado a outros tipos de paralelismo.

4.4 Análise do paralelismo no projeto das máquinas de comitê

Apresenta-se a seguir os algoritmos associados as duas arquiteturas consideradas no ambiente paralelo: arquitetura de comitê de especialistas e arquitetura de rede modular com especialistas de múltiplas camadas. Também são detalhadas as características do treinamento implementado para cada uma delas.

4.4.1 Comitê de especialistas

Arquitetura

Para o desenvolvimento da arquitetura de comitê foi selecionada a rede do tipo perceptron de múltiplas camadas, apresentada na seção 2.2.1, para implementar os especialistas da máquina. A seleção baseou-se na já comprovada eficiência e estabilidade desse tipo de rede quando da aplicação a diversos problemas, tais como: aproximação de funções, classificação de padrões e sistemas de controle. A arquitetura se encontrada mostrada na Figura 4.3.

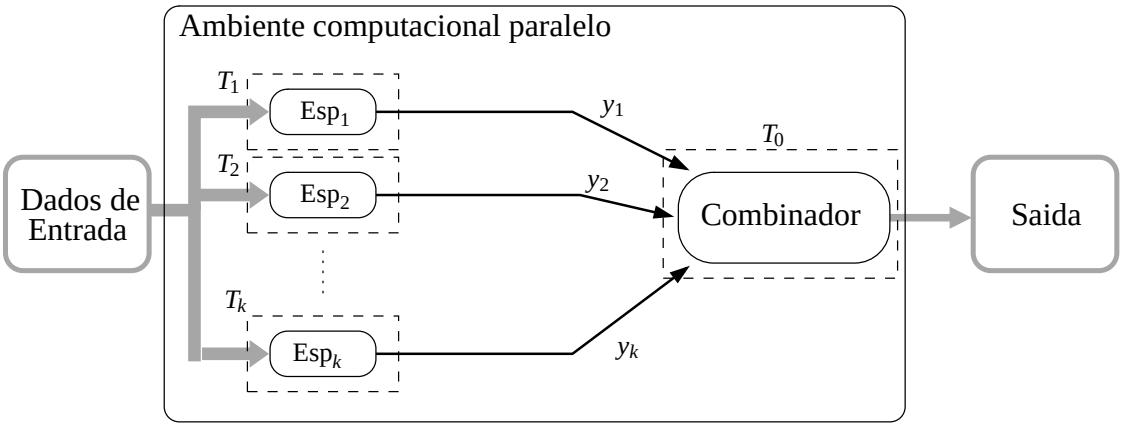

Figura 4.3: Diagrama em blocos da arquitetura paralela para máquinas de comitê.

Do ponto de vista do paralelismo, estratégia adotada foi de associar um especialista a cada tarefa, assim cada rede de perceptrons de múltiplas camadas estará sendo tratada por um processador exclusivo. Pode-se observar na própria Figura 4.3 que a estrutura das comunicações é bastante simples, do tipo um-para-todos (na distribuição dos dados de entrada entre os especialistas) ou todos-para-um (no envio das saídas dos especialistas ao combinador).

Esta estratégia se aplica bem ao modelo de redes do tipo média de ensemble. Cada rede é inicializada com um conjunto de pesos aleatórios e diferentes. A intenção como descrito na seção 2.4.1, é que, durante o treinamento, cada uma das redes busque atingir um mínimo local da superfície de erro diferente das outras.

Uma extensão natural da arquitetura proposta seria a ampliação da mesma através da aplicação de níveis hierárquicos. A Figura 4.4 apresenta um diagrama com uma possível

hierarquia de máquinas de ensemble. Cada especialista de ordem superior é composto por outros dois especialistas de ordem inferior e um combinador local. Observa-se pela equação 4.1 que a saída total da rede combina as saídas locais dos especialistas que por sua vez combinam suas saídas locais internas.

O paralelismo também pode ser explorado nessa arquitetura, seja entre cada um dos especialistas de ordem inferior, seja entre os especialista de ordem superior. Como na arquitetura anterior, pode-se explorar a superfície do erro de forma independente durante o treinamento e manter a mesma estrutura das comunicações, acrescida aqui de mais um nível hierárquico.

$$\text{Saída Global} = \overline{\Sigma}_i \underline{\Sigma}_j \phi_{i,j}(x_i) \tag{4.1}$$

onde $\phi_{i,j}$ representa a saída fornecida pela rede especialista i,j correspondente à entrada x_i, $\underline{\Sigma}_j$ representa a saída do combinador associado ao especialista de ordem inferior j e $\overline{\Sigma}_i$ representa a saída do combinador associado aos especialistas superior i. A arquitetura hierárquica pode fornecer uma quantidade maior de mínimos locais diferentes, melhorando desta forma o resultado final.

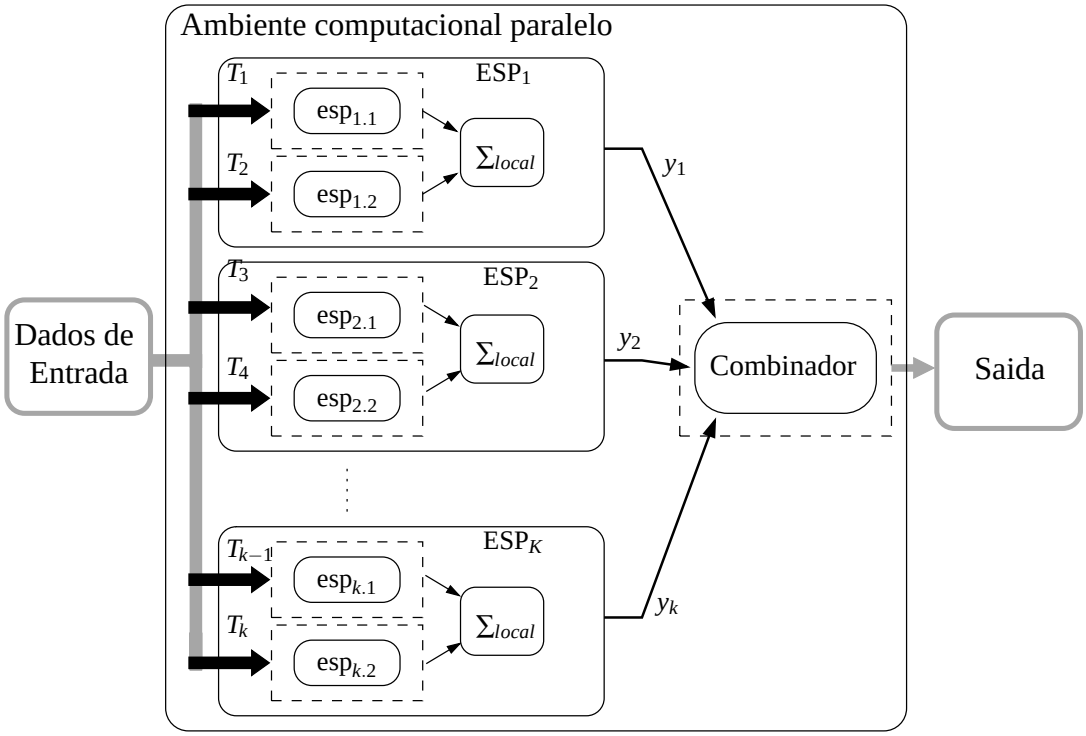

Figura 4.4: Diagrama em blocos da arquitetura paralela para máquinas de comitê estendida.

Treinamento

O treinamento da arquitetura de comitê de especialistas, mostrado na Figura 4.3, em paralelo, é realizado de forma semelhante ao do PMC usando retropropagação, adicionando-

se apenas a criação de tarefas e controle das mesmas. O programa é iniciado e são geradas *K* tarefas, cada uma contendo a estrutura de cada uma das redes especialists. Neste ponto cada especialista tem a autonomia de ter sua arquitetura desenvolvida de forma diferente, com os parâmetros de treinamento escolhidos livremente.

Cada rede é então treinada individualmente considerando-se uma quantidade de épocas pré-determinada ou até atingir um valor de erro desejado. Após o treinamento, as estruturas da rede podem ser armazenadas para execução posterior ou utilização em novo treinamento.

Da forma como o *software* foi implementado é possível estabelecer não apenas arquiteturas diferentes para cada especialistas, mas também valores específicos do algoritmo de aprendizagem, tais como taxas de aprendizagem, termos de momento e valores da regra delta-bar-delta. É possível, se desejado, a atribuição de conjuntos de treinamentos independentes para cada rede.

Algoritmo paralelo para aprendizagem da arquitetura de comitês de especialistas

1. *Inicialização*. Criar *K* tarefas, uma para cada rede especialista da arquitetura.
2. *Atribuição de variáveis*. Atribuir valores iniciais aos pesos das diferentes redes especialistas e suas variáveis de aprendizagem η, α, κ, β e ξ
3. *Treinamento*. Treinar cada rede com o conjunto de dados disponibilidados à mesma.
4. *Condição de parada*. Repetir o passo 3 até que cada uma das redes especialistas atinja um valor de erro mínimo especificado ou uma quantidade de máxima de épocas.
5. *Terminação*. Após o treinamento salvar os pesos para execução posterior.

Uma estratégia de treinamento pode incluir comunicação entre as épocas de treinamento, ou seja entre o passo 3 e 4. Isto pode ser feito incluindo uma transmissão de informação entre os especialistas após determinado número de épocas de forma semelhante ao treinamento competitivo. De forma semelhante, os parâmetros de treinamento podem ser alterados durante o treinamento, assim é possível utilizar conjuntos de treinamentos dinâmicos, ou taxas de aprendizagem também dinâmicas.

4.4.2 Rede modular com múltiplas camadas

Para o desenvolvimento de uma arquitetura modular foram analisados os seguintes aspectos: definição da arquitetura, papel de seus elementos constituintes, algoritmo de treinamento e as etapas de comunicação.

A arquitetura da rede modular encontra-se definida na Figura 2.7 e na Figura 4.1, apresentadas anteriormente. A estratégia de paralelização adotada é semelhante àquela utilizada na Seção 4.4.1, onde associa-se cada especialista a uma tarefa paralela. Variações dessa abordagem podem ser pensadas, tais como associar um especialista a mais de uma tarefa ou vários especialistas à mesma tarefa. A rede de passagem, que funciona como elemento avaliador nessa arquitetura, é constituída de estruturas semelhantes às das redes especialistas e também irá corresponder a uma tarefa específica.

Como o próprio nome indica, as redes modulares foram desenvolvidas e inspiradas na idéia de aprender a solucionar o problema de forma modularizada e dividida. O papel de seus elementos constituintes se divide em dois: os especialistas que aprendem sobre o problema e a fornecer respostas desejadas e a rede de passagem que aprende a escolher entre os especialistas. Computacionalmente o trabalho de agrupar as respostas das saídas individuais através de uma combinação é pequeno e pode ser agrupado junto a rede de passagem, por exemplo.

O algoritmo de treinamento para redes modulares denominado mistura hierárquica de especialistas, desenvolvido por Jacobs e Jordan [Jacobs & Jordan 1991], foi apresentado na seção 2.4.2 sendo aqui resumido na seguinte forma:

Algoritmo de mistura gaussiano associativo de especialistas

1 *Inicialização*. Criar $K+1$ tarefas, uma para cada uma das K redes especialistas e uma para a rede de passagem.
2 *Atribuição de variáveis*. Atribuir valores iniciais aos pesos das diferentes redes especialistas e da rede de passagem, além de suas variáveis de aprendizagem η, α, κ, β e ξ.
3 *Treinamento* Para cada exemplo de treinamento calcular:
(u_i) = saída da rede de passagem
(g_i) = valores das probabilidades *a priori* da rede de passagem
(y_i) = valores individuais de saída de cada rede especialista
(h_i) = valores das probabilidade *a posteriori* para cada rede especialista
(e_i) = valores de erro para cada neurônio de saída de cada especialista
(w_i) = fazer a atualização dos pesos sinápticos das redes especialistas com base nos valores dos pesos atuais, a taxa de aprendizado, a probabilidade *a posteriori*, o erro obtido e a saída fornecida.
(a_i) = fazer a atualização dos pesos sinápticos da rede de passagem com base nos valores atuais, taxa de aprendizado, saída da rede e a diferença entra as probabilidades *a posteriori* e *a priori*.
4 *Condição de parada*. Repetir o passo 3 até que cada uma das redes especialistas atinja um valor de erro mínimo especificado ou uma quantidade de máxima de épocas.
5 *Terminação*. Após o treinamento, salvar os pesos para execução posterior.

Antes do desenvolvimento do algoritmo paralelo, foram realizados experimentos computacionais com a arquitetura da rede modular proposta por Jacobs e Jordan e observou-se que o treinamento apresenta uma convergência bastante lenta, quando do treinamento por lote, ou seja, ainda mais lenta do que a atualização *online* (com atualização dos pesos a cada exemplo). Observou-se ainda uma instabilidade nos resultados obtidos durante a aprendizagem. Atribui-se esse fator ao número reduzido de parâmetros da rede (constituída de uma única camada de neurônios). A demanda por comunicação torna-se excessiva se o conjunto de dados de treinamento for pequeno, e seria atenuada apenas em problemas que possuam um número elevado de exemplos.

Com base nas possibilidades de paralelismo elencadas na Seção 4.3 algumas considerações podem ser estabelecidas na escolha do modelo de paralelismo a ser adotado

para a rede modular. A primeira possibilidade, que aborda a decomposição em domínio, ou seja, a partição do conjunto de dados de treinamento, apresenta-se de forma interessante na redução do tempo de processamento. A aplicação dessa abordagem exige o acúmulo das atualizações dos pesos sinápticos (w_i) e (a_i) sob todo o conjunto de treinamento utilizado, e isto acarreta em dois problemas: a atualização por lote e demanda por comunicação, uma vez que cada rede especialista precisa receber todas as partições do conjunto de treinamento e os valores das probabilidades *a posteriori*, calculados pela rede de passagem.

A segunda abordagem paralela para a rede modular poder ser explorada através da decomposição em arquitetura. A rede pode ser dividida entre as tarefas, sendo uma para cada rede especialista e uma para a rede de passagem. Assim, cada rede pode realizar sua computação individual tendo em sua memória uma cópia do conjunto de treinamento. Entretanto, essa abordagem possui o mesmo inconveniente da anterior, no que diz respeito à necessidade de cada tarefa dispor dos valores das probabilidades *a posteriori*, calculados pela rede de passagem.

Uma solução para contornar este problema seria replicar a rede de passagem em cada um dos especialistas, porém para realização do cálculo das probabilidades *a posteriori* (h_i) também é necessário o conhecimento dos erros de saída de todas as redes especialistas. O algoritmo baseado nessa abordagem certamente possui um tempo comunicação entre as tarefas elevado, degradando seu desempenho.

A terceira proposta de aplicação do paralelismo baseia-se na aplicação da competitividade. Isto pode ser feito replicando-se toda a estrutura da rede modular em todas as tarefas e utilizando-se de comunicações e atualizações como as propostas por Alves na Seção 3.2.4. A Figura 4.5 apresenta a distribuição das tarefas, ou seja as redes modulares, entre os processadores p_k do ambiente paralelo.

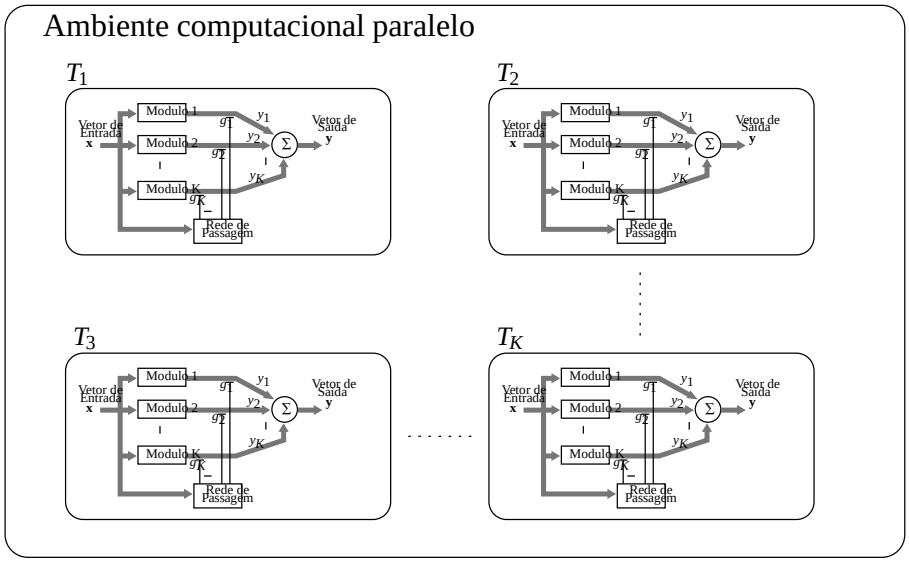

Figura 4.5: Diagrama em blocos da distribuição da arquitetura paralela para redes modulares.

Modificação da arquitetura

Durante a análise e experimentação do algoritmo de mistura de especialistas como foi apresentado por Jacobs e Jordan [Jacobs & Jordan 1991] e sumariado por Haykin [Haykin 2001], observou-se que a estrutura era eficiente para alguns problemas bastante simples. A aplicação do algoritmo para problemas mais complexos não apresentou boas soluções, provavelmente devido a simplicidade dos especialistas: primeiro, por apresentar apenas uma camada, e segundo por não incluir na saídas dos neurônios a presença de funções de ativação não-lineares. Também, a falta do *bias* força a arquitetura a fornecer apenas saída nulas para entradas nulas .

Houve então a necessidade de desenvolvimento de uma arquitetura ampliada. Para tanto optou-se pela adição de camadas ocultas, assim como nas redes perceptron de múltiplas camadas, e neurônios com função de ativação não lineares, tanto nas redes especialistas como na rede de passagem. Além disso em cada camada houve a adição de elementos de bias. Na Figura 4.6 mostra-se um diagrama da arquitetura modular estendida.

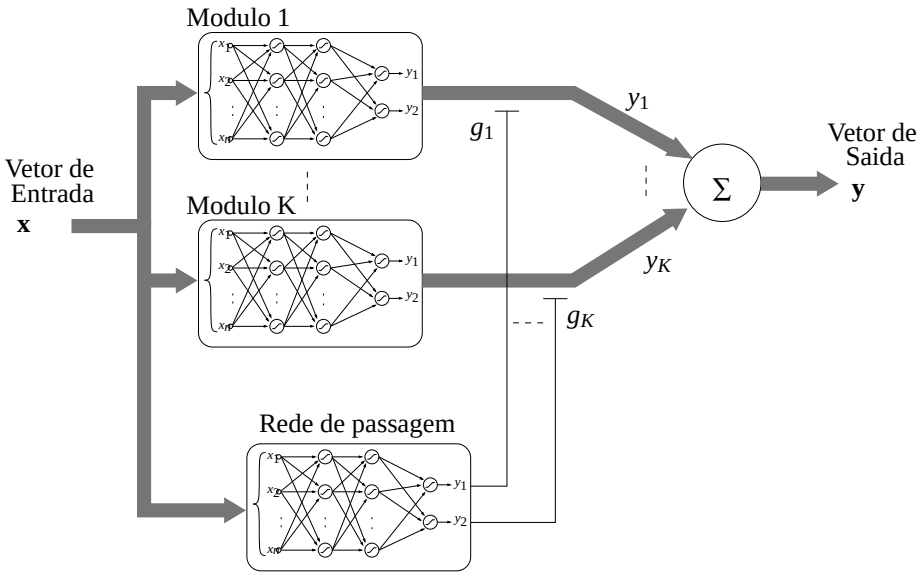

Figura 4.6: Diagrama em blocos da rede modular estendida.

Treinamento

Para treinar a rede modular modificada foram necessários ajustes no algoritmo de mistura gaussiana de especialistas. A literatura pesquisada acerca da utilização de especialistas com múltiplas camadas em redes modulares não apresenta nenhum estudo sobre como isso deve ser feito. Neste trabalho, propõe-se uma forma original de se adaptar os algoritmos de mistura de especialistas e o algoritmo da retropropagação, em um novo algoritmo de treinamento.

Para fins de comparação, os três algoritmos são mostrados a seguir. Antes de discutí-los, porém, algumas convenções devem ser estabelecidas:

- A rede modular proposta em Jacobs e Jordan [Jacobs & Jordan 1991] é composta de K especialistas (indexados pela variável i, $i = 1,...,K$). Cada especialista possui uma camada de entrada e uma camada de saída com $q^{(i)}$ neurônios (indexados pela variável m, $m = 1,...,q^{(i)}$). A rede de passagem também possui uma camada de entrada e uma camada de saída, com K neurônios. As funções de ativação de todos os neurônios da rede modular são do tipo linear;
- A rede PMC genérica considerada aqui é composta de L camadas, sendo uma camada de entrada e uma camada de saída. Cada camada possui $q^{(l)}$ neurônios (indexados pela variável i, $i = 1,...,q^{(l)}$). As funções de ativação de cada neurônio podem ser de qualquer tipo;
- A rede modular proposta neste trabalho é composta de K especialistas (indexados pela variável i, $i = 1,...,K$) do tipo PMC com L^{Esp_i} camadas e $q^{(l)_{Esp_i}}$ neurônios em cada uma delas (indexados pelas variáveis j, $j = 1,...,q^{(l)_{Esp_i}}$ e $l = 1,...,L^{(Esp_i)}$). A rede de passagem, também do tipo PMC, tem $L^{(Pas)}$ camadas e $q^{(l)_{Pas}}$ neurônios em cada uma delas (indexados pelas variáveis j, $j = 1,...,q^{(l)_{Pas}}$) e $l = 1,...,L^{Pas}$). As funções de ativação de cada neurônio podem ser de qualquer tipo.

O algoritmo de treinamento modificado, trata-se de uma adaptação do algorimo de Jacobs [Jacobs & Jordan 1991], com a introdução do cálculo do gradiente do erro. Para os módulos especialistas, além do erro de saída, é incluída também a probabilidade *a posteriori*, pois esta indica para cada exemplo fornecido, a contribuição daquele especialista na composição da saída da rede modular. Da mesma forma, na rede de passagem, o gradiente inclui a diferença entre as probabilidades *a posteriori* e *a priori*, uma vez que esta diferença indica o erro nas decisões sobre as contribuições dos especialistas.

Algoritmo de Treinamento da Rede Modular

$$
\begin{aligned}
u_i(n) &= \mathbf{x}^T(n)\mathbf{a}_i(n) \\
g_i(n) &= \frac{exp(u_i)}{\sum_{j=1}^{K} exp(u_j)} \\
y_i^{(m)}(n) &= \mathbf{x}^T(n)\mathbf{w}_i^{(m)}(n) \\
\mathbf{y}_i(n) &= [y_i^{(1)}(n), y_i^{(2)}(n), ..., y_i^{(q^{(i)})}(n)]^T \\
h_i(n) &= \frac{g_i(n)exp(-\frac{1}{2}\| \mathbf{d}(n) - \mathbf{y}_i(n) \|^2)}{\sum_{j=1}^{K} g_j(n)exp(-\frac{1}{2}\| \mathbf{d}(n) - \mathbf{y}_i(n) \|^2)} \\
\mathbf{e}_i(n) &= \mathbf{d}_i(n) - \mathbf{y}_i(n) \\
\mathbf{w}_i^{(m)}(n+1) &= \mathbf{w}_i^{(m)}(n) + \eta h_i(n) e_i^{(m)}(n)\mathbf{x}(n) \\
\mathbf{a}_i(n+1) &= \mathbf{a}_i(n) + \eta[h_i(n) - g_i(n)]\mathbf{x}(n)
\end{aligned}
$$

onde:

- $u_i(n)$ - saída do i-ésimo neurônio da rede de passagem, quando da aplicação do n-ésimo exemplo de treinamento;
- $\mathbf{x}(n)$ - vetor de entrada das redes, correspondente ao n-ésimo exemplo de treinamento;
- $\mathbf{a}_i(n)$ - pesos sinápticos entre a entrada $\mathbf{x}$ e o i-ésimo neurônio da rede de passagem;
- $g_i(n)$ - probabilidade *a priori* associada à saída do i-ésimo neurônio da rede de passagem, quando da aplicação do n-ésimo exemplo de treinamento;
- $y_i^{(m)}(n)$ - saída do m-ésimo neurônio do i-ésimo especialista, quando da aplicação do n-ésimo exemplo de treinamento;
- $h_i(n)$ - probabilidade *a posteriori* associada à saída do i-ésimo neurônio da rede de passagem, quando da aplicação do n-ésimo exemplo de treinamento;
- $\mathbf{d}(n)$ - saída desejada da rede modular, associada ao n-ésimo exemplo de treinamento;
- $\mathbf{e}_i(n)$ - erro entre a saída do i-ésimo especialista e a saída desejada da rede modular;
- $\mathbf{w}_i^{(m)}(n)$ - pesos sinápticos entre a entrada $\mathbf{x}$ e o m-ésimo neurônio do i-ésimo especialista.

Algoritmo de Treinamento da Rede PMC

$$
\begin{aligned}
v_i^{(l)}(n) &= \sum_{j=0}^{m} w_{ij}^{(l)}(n) y_j^{(l-1)}(n) \\
y_i^{(l)}(n) &= \varphi_i^{(l)}(v_i^{(l)}(n)) \\
e_i(n) &= d_i(n) - y_i^{(L)}(n)
\end{aligned}
$$

gradiente na camada de saída

$$
\delta_i^{(L)}(n) = e_i^{(L)}(n)\varphi'^{(L)}_i(v_i^{(L)}(n))
$$

gradiente nas camadas ocultas

$$
\delta_i^{(l)}(n) = \varphi'^{(l)}_i(v_i^{(l)}(n)) \sum_{k=1}^{q^{(l+1)}} \delta_k^{(l+1)}(n) w_{ki}^{(l+1)}(n)
$$

$$
\mathbf{w}_{ij}^{(l)}(n+1) = \mathbf{w}_{ij}^{(l)}(n) + \eta\delta_i^{(l)}(n) y_j^{(l-1)}(n)
$$

- $v_i^{(l)}(n)$ - potencial de ativação do i-ésimo neurônio da l-ésima camada, quando da aplicação do n-ésimo exemplo de treinamento;
- $w_{ij}^{(l)}(n)$ - peso sináptico associado à j-ésima entrada do i-ésimo neurônio da l-ésima camada, quando da aplicação do n-ésimo exemplo de treinamento;
- $y_j^{(l-1)}(n)$ - saída do j-ésimo neurônio da (l-1)-ésima camada, quando da aplicação do n-ésimo exemplo de treinamento;
- $e_i^{(n)}(n)$ - erro entre a saída do i-ésimo neurônio da camada de saída e a saída desejada da rede;
- $d_i(n)$ - saída desejada do i-ésimo neurônio da rede, associada ao n-ésimo exemplo de treinamento;
- $\varphi_i^{(l)}$ - função de ativação do i-ésimo neurônio da l-ésima camada;
- $\varphi'^{(l)}_i$ - derivada da função de ativação do i-ésimo neurônio da l-ésima camada;
- $\delta_i^{(l)}(n)$ - gradiente do erro associado ao i-ésimo neurônio da l-ésima camada, quando da aplicação do n-ésimo exemplo de treinamento.

Novo Algoritmo da Rede Modular com Múltiplas Camadas

$$
\begin{aligned}
v_{p_i}^{(l)}(n) &= \sum_{j=0}^{q^{(l)Pas}} a_{ij}^{(l)} u_i^{(l-1)}(n) \\
u_i^{(l)}(n) &= \varphi_{p_i}^{(l)}(v_{p_i}^{(l)}(n)) \\
g_i(n) &= \frac{exp(u_i^{(L)}(n))}{\sum_{j=1}^{K} exp(u_j^{(L)}(n))} \\
v_{e_i^{(k)}}^{(l)}(n) &= \sum_{j=0}^{q^{(l_{Esp_i})}} w_{e_i^{(k)}j}^{(l)} y_{e_i^{(k)}}^{(l-1)}(n) \\
y_{e_i^{(k)}}^{(l)}(n) &= \varphi_{e_i^{(k)}}^{(l)}(v_{e_i^{(k)}}^{(l)}(n)) \\
\mathbf{y}_{e^{(k)}}(n) &= [y_{e_i^{(k)}}^{(L)(1)}(n), y_{e_i^{(k)}}^{(L)(2)}(n), ..., y_{e_i^{(k)}}^{(L)(q)}(n)]^T \\
h_i(n) &= \frac{g_i(n) exp(-\frac{1}{2} \| \mathbf{d}(n) - \mathbf{y}_{e_i^{(k)}}(n) \|^2)}{\sum_{j=1}^{K} g_j(n) exp(-\frac{1}{2} \| \mathbf{d}(n) - \mathbf{y}_{e_i^{(k)}}(n) \|^2)} \\
\mathbf{e}_{e^{(k)}}(n) &= \mathbf{d}_i(n) - \mathbf{y}_{e^{(k)}}(n) \\
&\text{gradiente na camada de saída} \\
\delta_{e_i^{(k)}}^{(L)}(n) &= h_i(n) e_i(n)^{(L)} \varphi'^{(L)}_{e_i^{(k)}}(v_{e_i^{(k)}}^{(L)}(n)) \\
&\text{gradiente nas camadas ocultas} \\
\delta_{e_i^{(k)}}^{(l)}(n) &= \varphi'^{(l)}_{e_i^{(k)}}(v_{e_i^{(k)}}^{(l)}(n)) \sum_{m=1}^{q^{(l+1)Esp_i}} \delta_m^{(l+1)}(n) w_{me_i^{(k)}}^{(l+1)}(n) \\
w_{e_i^{(k)}j}^{(l)}(n+1) &= w_{e_i^{(k)}j}^{(l)}(n) + \eta \delta_{e_i^{(k)}}^{(l)}(n) y_j^{(l-1)}(n) \\
&\text{gradiente na camada de saída} \\
\delta_{p_i}^{(L)}(n) &= [h_i(n) - g_i(n)] \varphi'^{(L)}_{p_i}(v_{p_i}^{(L)}(n)) \\
&\text{gradiente nas camadas ocultas} \\
\delta_{p_i}^{(l)}(n) &= \varphi'^{(l)}_{p_i}(n)(v^{(l)} \sum_{m=1}^{q^{(l+1)Pas}} \delta_m^{(l+1)}(n) a_{mp_i}^{l+1}(n) \\
a_{p_i j}^{(l)}(n+1) &= a_{p_i j}^{(l)}(n) + \eta \delta_{p_i}^{(l)}(n) y_j^{(l-1)}(n)
\end{aligned}
$$

onde:

$v_{p_i}^{(l)}(n)$ - potencial de ativação do i-ésimo neurônio da l-ésima camada da rede de passagem, quando da aplicação do n-ésimo exemplo de treinamento;

$v_{e_i^{(k)}}^{(l)}(n)$ - potencial de ativação do i-ésimo neurônio da l-ésima camada do k-ésimo especialista, quando da aplicação do n-ésimo exemplo de treinamento;

$u_i^{(l)}(n)$ - saída do i-ésimo neurônio da l-ésima camada da rede de passagem, quando da aplicação do n-ésimo exemplo de treinamento;

$y_{e_i^{(k)}}^{(l)}(n)$ - saída do i-ésimo neurônio da l-ésima camada do k-ésimo especialista, quando da aplicação do n-ésimo exemplo de treinamento;

$g_i(n)$ - probabilidade *a priori* associada à saída do i-ésimo neurônio da camada de saída da rede de passagem, quando da aplicação do n-ésimo exemplo de treinamento;
$h_i(n)$ - probabilidade *a posteriori* associada à saída do i-ésimo neurônio da camada de saída da rede de passagem, quando da aplicação do n-ésimo exemplo de treinamento;
$\mathbf{e}_i(n)$ - erro entre a saída do k-ésimo especialista e a saída desejada da rede modular;
$\varphi_{p_i}^{(l)}$ - função de ativação do i-ésimo neurônio da l-ésima camada da rede de passagem;
$\varphi_{e_i^{(k)}}^{(l)}$ - função de ativação do i-ésimo neurônio da l-ésima camada do k-ésimo especialista;
$\varphi'^{(l)}_{p_i}$ - derivada da função de ativação do i-ésimo neurônio da l-ésima camada da rede de passagem;
$\varphi'^{(l)}_{e_i^{(k)}}$ - derivada da função de ativação do i-ésimo neurônio da l-ésima camada do k-ésimo especialista;
$\delta_{e_i^{(k)}}^{(l)}(n)$ - gradiente do erro associado ao i-ésimo neurônio da l-ésima camada do k-ésimo especialista, quando da aplicação do n-ésimo exemplo de treinamento.
$\delta_{p_i}^{(l)}(n)$ - gradiente do erro associado ao i-ésimo neurônio da l-ésima camada da rede de passagem, quando da aplicação do n-ésimo exemplo de treinamento.
$w_{e_i^{(k)}j}^{(l)}(n)$ - peso sináptico associado à j-ésima entrada do i-ésimo neurônio da l-ésima camada do k-ésimo especialista, quando da aplicação do n-ésimo exemplo de treinamento;
$a_{p_ij}^{(l)}(n)$ - peso sináptico associado à j-ésima entrada do i-ésimo neurônio da l-ésima camada da rede de passagem, quando da aplicação do n-ésimo exemplo de treinamento;

Atualização do pesos sinápticos

A taxa de aprendizagem η utilizada nas equações de atualização dos pesos sinápticos nos algoritmos apresentados na Seção 4.4.2, pode ser fonte de instabilidade na execução dos mesmos. Quando são utilizados valores muito elevados pode-se aumentar o tempo de convergência ou ainda manter-se preso em mínimos locais quando são utilizados taxas muito reduzidas. Para contornar este tipo de problema algumas variações são aplicadas, tais como, a introdução do termo do momento, uso de η adaptativo e aplicação da regra delta-bar-delta.

O termo do momento incrementa a velocidade de aprendizagem e previne o perigo de ocorrer instabilidades na redução do erro. Isto é feito através da inclusão de um termo na obtenção do Δ de atualização dos pesos de uma rede PMC

$$\Delta w_{ji}(n) = \alpha \Delta w_{ji}(n-1) + \eta \delta_i(n) y_i(n) \tag{4.2}$$

onde α é usualmente um valor positivo chamada de constante do *momentum*.

O segundo modo de alteração do valor da taxa de aprendizagem baseia-se na utilização de uma taxa dinâmica através do tempo (épocas de treinamento). A cada época de treinamento são avaliadas duas ou mais taxas de atualização, distantes de uma fração do valor de η igual $\Delta\eta$. A taxa que fornecer melhor resultado é utilizada na próxima época. Este procedimento exige K vezes o processamento normal, para K diferentes taxas de aprendizagem. Isto é interessante quando aplicado em ambiente paralelo pois cada tarefa pode ficar responsável por avaliar uma determinada taxa. O parâmetro $\Delta\eta$ pode ser alterado para criar taxas alternativas com distâncias não lineares.

O terceiro modo de otimização da taxa de aprendizagem se baseia na regra Delta-bar-Delta. Esta regra exige que seja criada uma taxa de aprendizagem η para cada peso da

arquitetura PMC, isto demanda instantâneamente um aumento da necessidade de memória e de processamento. A regra faz uso da observação do comportamento do erro da rede devido a uma determinada taxa de um peso específico da rede.

De forma resumida tem-se que

$$\Delta\eta_{ji}(n+1) = \begin{cases} \kappa & \text{se } S_{ji}(n-1)D_{ji}(n) > 0 \\ -\beta\eta_{ji}(n) & \text{se } S_{ji}(n-1)D_{ji}(n) < 0 \\ 0 & \text{caso contrário} \end{cases} \tag{4.3}$$

onde D_{ji} e S_{ji} são definidos, respectivamente, como

$$D_{ji} = \frac{\partial E(n)}{\partial w_{ji}(n)} \tag{4.4}$$

e

$$S_{ji}(n) = (1-\xi)D_{ji}(n-1) + \xi S_{ji}(n-1) \tag{4.5}$$

onde κ é uma constante positiva e ξ uma constante positiva contida no intervalo fechado $[0,1]$. A quantidade D_{ji} é o valor corrente da derivada parcial do valor do erro com respeito ao peso sináptico w_{ji}. O parâmetro $S_{ji}(n)$ é um somatório exponencial da derivada atual e passada do valor do erro com relação ao peso sináptico w_{ji}.

4.5 Software desenvolvido

O *software* é denominado de RNAP, um acrônimo de Redes Neurais Artificiais Paralelas. Objetiva-se que o trabalho de desenvolvimento da RNAP possa prover condições de ser utilizada como uma caixa de ferramenta para o desenvolvimento destes tipos de redes. A RNAP desenvolve-se de forma semelhante aos projetos desenvolvidos por Neumann [Neumann 2005] para a aprendizagem por reforço, e por Collobert [Collobert et al. 2002] para as redes neurais.

Foram desenvolvidas, como forma de protótipo, algumas das partes do *software* no software para simulação matemática Scilab[2]. Entre as linguagens de programação selecionadas como candidatas à implementação estiveram: C, C++, Fortran 77 e Fortran 90. As quatro linguagens são suportadas pelos principais sistemas que implementam o modelo de passagem de mensagens MPI. Além disso, deve ser uma linguagem que possibilite a implementação de sistemas de análise como o proposto por Moore et al [Moore et al. 2005], que permite, inclusive, uma análise escalonável do desempenho.

As linguagens C e Fortran 77 são orientadas a uma programação seqüencial e funcional. As linguagens C++ e Fortran 90 são linguagens mais recentes, e por isso, incorporam paradigmas de programação mais modernos. As facilidades das linguagens C++ e Fortran 90, incorporando características de programação orientada a objetos, facilitam ainda mais a reutilização de códigos além de permitirem melhores abstrações das estruturas necessá-

[2]Scilab é um software de simulação numérica de fácil utilização, desenvolvido pelo Scilab Consortium. O Scilab é um software com código aberto, que pode ser obtido junto a inúmeras bibliotecas sob o domínio <http://www.scilab.org>.

rias ao projeto a ser implementado. Com todo o diferencial apresentado pelas linguagens C++ e Fortran 90, fez-se descartada a escolha por uma das linguagens C ou Fortran 77.

Em um trabalho desenvolvido por Cary *et al* [Cary et al. 1996] é apresentado um estudo comparando as linguagens C++ e Fortran 90 no desenvolvimento de softwares científicos. A comparação inclui as facilidades oferecidas por cada uma delas, suas possibilidades de expansão e ainda gráficos de desempenhos. A linguagem C++ apresenta maiores recursos para o programador. A linguagem Fortran 90 apresenta-se com um desempenho ligeiramente superior na realização de tarefas com processamento matemático de grandes dimensões.

A escolha pela utilização da linguagem C++ ocorreu devido a familiaridade prévia do desenvolvedor, do material de suporte e referência disponível ser superior e por apresentar uma comunidade de desenvolvimento maior, facilitando dessa forma a resolução de dúvidas que pudessem aparecer durante o desenvolvimento do projeto.

A implementação da RNAP foi iniciada com a codificação da estrutura representacional das redes do tipo PMC (seção 2.2.1) e do seu algoritmo de treinamento (retropropagação). Para tanto, foi desenvolvida uma Classe denominada rna_mlp. A Classe rna_mlp é composta por atributos que representam a estrutura da arquitetura da rede, dos algoritmos utilizados pela mesma e também por métodos de acesso público para realização das tarefas. Entre os atributos da Classe encontram-se:

1. *Relacionados a arquitetura*. Pesos da rede, tipos de neurônios, tipos de função de ativação e quantidade de camadas.
2. *Relacionados ao algoritmo*. valor de bias, valor de η, α, quantidade de épocas, conjunto de dados de entrada e saída, erro máximo, épocas máximas e parâmetro de regra delta (κ, β, ξ).

Entre os métodos da Classe:

1. *Relacionados a arquitetura*. Construtor por cópia, construtor por tamanho de arquitetura (quantidade de camadas e neurônios em cada camada), construtor por arquivo (com formatação específica), alterar e exibir parametros de tamanho e tipos de funções, execução da rede para um determinada entrada, salvar e ler arquivo da estrutura, iniciar pesos da rede de forma aleatória ou zerados.
2. *Relacionados ao algoritmo*. Iniciar o algoritmo de treinamento, salvar e ler dados de treinamento, ler dados de validação, salvar validação, definir parâmetros de treinamento (α, η, épocas, erro, β, κ, ξ).

A implementação é realizada com a inclusão de sua documentação de forma concomitante no código. Os passos seguintes de desenvolvimento incluíram o desenvolvimento das classes rna_mod para a arquitetura de redes modulares e a classe rna_mlmod para a arquitetura modular desenvolvida com perceptron de múltiplas camadas.

Em cada uma das novas classes foram desenvolvidos igualmente métodos de acesso, construção, alteração e treinamento. Uma classe de dados foi criada para ajuste prévio dos dados de treinamento como, por exemplo, normalização, classificação e atribuição de valores. A construção das máquinas de comitê é realizada diretamente no programa a partir das classes desenvolvidas.

Ao todo foram criados no ambiente Scilab apenas para o protótipo cerca de 5000 (cinco mil linhas) de códigos para o treinamento e execução das arquiteturas. Na versão final do *software* foram criados mais de 10000 (dez mil linhas) de códigos em C++ para criação das arquiteturas PMC, modular e modular com múltiplas camadas.

4.6 Conclusão

Apresentou-se nesse capítulo a metodologia de desenvolvimento utilizada, as arquiteturas de redes implementadas junto a suas variações, bem como, os algoritmos utilizados para o treinamento das redes desenvolvidas. Assim, é estabelecido um ambiente para aplicações numéricas e verificações de desempenho. O capítulo seguinte trata da caracterização dos experimentos computacionais realizados no cluster computacional, e também, de uma discussão dos resultados obtidos com a execução dos programas.

Capítulo 5

Aplicações e Análise de Resultados

Este capítulo apresenta os experimentos realizados com a estrutura proposta. Isto foi feito para verificar a eficiência das arquiteturas propostas e a utilização do *cluster* com relação ao seu comportamento em execuções de comunicação e processamento.

A primeira parte do capítulo apresenta os experimentos realizados sobre um problema de classificação de padrões em máquinas de comitê do tipo média de ensemble. O segundo experimento trata da classificação de padrões utilizando as redes modulares com múltiplas camadas. O terceiro exemplo trata-se da aproximação de funções fazendo uso de redes PMC, modulares de múltiplas camadas e uma estrutura composta.

5.1 Classificação com média de ensemble

Realizou-se no cluster um experimento computacional com o intuito de avaliar a estrutura de médias de ensemble proposta. Foi utilizado a RNAP com o conjunto de bibliotecas de comunicação por passagem de mensagem MPI implementado pela LAM/MPI em sua versão 7.0.6.

5.1.1 Domínio do problema

O experimento consiste em classificar os padrões observados na Figura 5.1, separados por uma determinada distância. O conjunto de dados é composto por 1000 exemplos, onde cada exemplo é composto por 6 descritores, que representam respectivamente: coordenada cartesiana *x*, coordenada cartesiana *y* e um vetor binário de quatro posições que representam classe ao qual o ponto pertence.

Os exemplos se distribuem uniformemente através do intervalo em $x = [-10; 10]$ e $y = [-10; 10]$, e se classificam conforme sua distância em relação aos pontos: $x_{c_1} = (-2,5;\ -2,5)$, $x_{c_3} = (5;\ 0)$ e $x_{c_3} = (-5;\ 8)$. Assim, para que um ponto pertença a classe

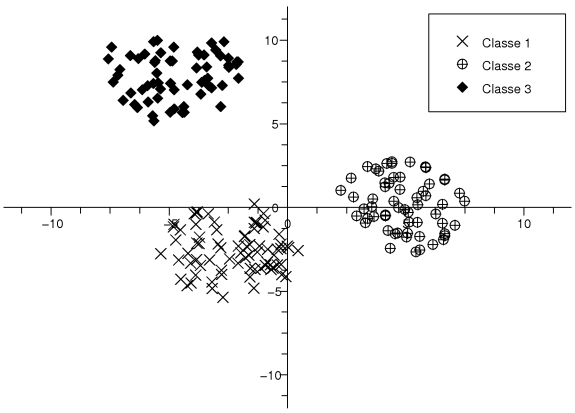

Figura 5.1: Exemplos dos pontos das classes do primeiro experimento.

um, dois, três, ou nenhuma, é necessário que se verifique a seguinte relação de pertinência

$$p(x,y) \in \begin{cases} \text{classe 1} & \text{se} \quad ||x - x_{c_1}||_2 < 3 \\ \text{classe 2} & \text{se} \quad ||x - x_{c_2}||_2 < 3 \\ \text{classe 3} & \text{se} \quad ||x - x_{c_3}||_2 < 3 \\ \text{nenhuma} & \text{caso contrário} \end{cases} \tag{5.1}$$

onde $||\cdot||_2$ representa a norma euclidiana. A Tabela 5.1 apresenta alguns exemplos de dados formatados segundo o arquivo de exemplos seguidos de suas respectivas classes.

Tabela 5.1: Exemplos de dados do conjunto de treinamento

x	y	c1	c2	c3	nc	$\in$
-3,3934	0,2028	1	0	0	0	classe 1
6,9949	1,5351	0	1	0	0	classe 2
-5,6707	7,8224	0	0	1	0	classe 3
3,3076	-4,3502	0	0	0	1	nenhuma

5.1.2 Arquiteturas

Foram criadas duas estruturas para a classificação proposta: a rede PMC padrão e a máquina de comitê do tipo média de ensemble. A rede de perceptrons de múltiplas camadas foi implementada de maneira usual com diferentes arquiteturas e diferentes taxas de aprendizado (η), incluindo casualmente o termo do *momentum* (α), porém, sem a utilização da regra delta-bar-delta. A exclusão da regra delta-bar-delta nesta aplicação

foi motivada pelo excesso do uso de memória e o aumento considerável de tempo de processamento.

A Tabela 5.2 apresenta as arquiteturas de PMC avaliadas pelos experimento. Onde o vetor que representa a arquitetura da rede é formado da seguinte forma: o primeiro valor define o tamanho da camada de entrada, os valores intermediários definem os número de neurônios das camadas ocultas e o último valor representa o número de neurônios da camada de saída. Em cada arquitetura foram executados testes variando o valor da taxa de aprendizagem e o valor de termo do *momemtum*.

Tabela 5.2: Arquiteturas PMC avaliadas

Arquitetura	η inicial	η final	α inicial	α final
[2 15 4]	3,86 x 10^{-1}	3,8 x 10^{-5}	0	10^{-5}
[2 25 4]	3,86 x 10^{-1}	3,8 x 10^{-5}	0	10^{-5}
[2 35 4]	3,86 x 10^{-1}	3,8 x 10^{-5}	0	10^{-5}
[2 10 10 4]	3,86 x 10^{-1}	3,8 x 10^{-5}	0	10^{-5}
[2 15 15 4]	3,86 x 10^{-1}	3,8 x 10^{-5}	0	10^{-5}

A segunda estrutura consiste de uma máquina de comitê do tipo média de ensemble (ME), compostas por diferentes quantidades de especialistas. Igualmente a arquitetura PMC não foram utilizados regras delta-bar-delta para o treinamento dos especialistas individualmente, porém foram variadas as taxas de aprendizagem (η) e o termo do momento (α). Foram desenvolvidas seis estruturas de médias de ensemble:

ME1 Dois especialistas cada um composto de arquitetura [2 15 4].
ME2 Quatro especialistas cada um composto de arquitetura [2 15 4].
ME3 Seis especialistas cada um composto de arquitetura [2 15 4].
ME4 Dois especialistas cada um composto de arquitetura [2 10 10 4].
ME5 Quatro especialistas cada um composto de arquitetura [2 10 10 4].
ME6 Seis especialistas cada um composto de arquitetura [2 10 10 4].

O treinamento das médias de ensemble foi baseado na estratégia de decomposição do domínio (seção 4.3.1), e foi realizado de forma que cada especialista utilize em seu processo de aprendizagem um conjunto de treinamento composto de uma quantidade de exemplos reduzidas, de forma semelhante a técnica *Bagging* (seção 2.4.1) com o diferencial de utilizarem conjuntos de treinamento disjuntos. Assim para o ensemble de dois especialistas, cada especialista aprende a partir de um conjunto de treinamento equivalente a metade do que foi utilizado para a rede PMC, logo, 500 exemplos. Já para o ensemble com 4 especialistas cada um treina com um conjunto de 250, e o último ensemble com seis especialistas e 166 exemplos. Apesar da redução do conjunto de treinamento para cada especialista, a quantidade de exemplos ainda mantém um proporcionalidade com relação a dimensão da rede, assim, atende à Regra de Baus-Haussler [Baum & Haussler 1990] possibilitando a aprendizagem da máquina.

A saída final da média de ensemble é gerada por um combinador que realiza a média das saídas individuais, desta forma cada especialista contribui igualmente com a saída, logo

$$\mathbf{Y} = \frac{1}{K}\sum_{i=1}^{K} Y_{esp_i} \tag{5.2}$$

onde K corresponde ao número total de especialistas e Y_{esp_i} a saída fornecida pelo i-ésimo especialista da média de ensemble.

5.1.3 Resultados obtidos

A Tabela 5.3 apresenta os melhores resultados obtidos com a execução do experimento pelas redes PMC. São apresentados a arquitetura, os parâmetros de aprendizagem, o tempo de execução e o menor valor de erro obtido.

Tabela 5.3: Resultados para arquiteturas PMC

Arquitetura	η	α	tempo	erro
[2 15 4]	3,8 x 10^{-3}	0,00	49,831s	0,04796
[2 15 4]	3,8 x 10^{-3}	10^{-2}	1m 6,767s	0,05980
[2 25 4]	3,8 x 10^{-4}	0,00	1m 46,241s	0,06511
[2 25 4]	3,8 x 10^{-3}	10^{-3}	1m 46,13s	0,04378
[2 35 4]	3,8 x 10^{-4}	0,00	2m 25,493s	0,04594
[2 35 4]	3,8 x 10^{-3}	10^{-3}	2m 25,274s	0,04052
[2 10 10 4]	3,8 x 10^{-3}	0,00	1m 10,091s	0,02099
[2 10 10 4]	3,8 x 10^{-3}	10^{-3}	1m 41,237s	0,01483
[2 15 15 4]	3,8 x 10^{-3}	0,00	2m 57,865s	0,02520
[2 15 15 4]	3,8 x 10^{-3}	10^{-3}	2m 58,400s	0,02181

Observa-se que em todos os casos em que as redes PMC apresentam apenas uma única camada oculta o erro mantém valores próximos entre o intervalo de [0,040 e 0,065]. A adição de uma camada oculta, caso das quatro últimas arquiteturas da tabela, acarreta na redução do erro e apresenta valores entre o intervalo de [0,0148 e 0,0252], uma diferença média de 0,014 mais baixo.

A Tabela 5.4 apresenta os resultados obtidos a partir da execução dos ensembles em paralelo, onde o resultado para um especialista (em um processador) trata-se da melhor execução com relação ao tempo da rede PMC com arquitetura semelhante, já apresentada na Tabela 5.3 anterior.

Verifica-se que o paralelismo apresentou efetividade na redução do tempo. Em ambos os casos os tempos de processamento foram reduzidos conforme a adição de especialistas e o erro manteve-se dentro das médias apresentadas pela execução serializada pelas redes PMC. Observa-se um crescimento do erro conforme a adição de especialistas para os ensembles ME1, ME2 e ME3, porém entre os ensembles ME5 e ME6 o erro apresenta uma ligeira redução quando ocorre o aumento de 4 para 6 especialistas.

Tabela 5.4: Resultados para médias de ensemble

Arquitetura	N. proc.	η	α	tempo	erro
[2 15 4]	1	3,8 x 10^{-3}	0,00	49,831s	0,04796
ME1	2	3,8 x 10^{-3}	0,00	25,130s	0,05438
ME2	4	3,8 x 10^{-3}	0,00	14,539s	0,05884
ME3	6	3,8 x 10^{-3}	0,00	10,864s	0,0600
[2 10 10 4]	1	3,8 x 10^{-3}	0,00	1m 10,091s	0,02099
ME4	2	3,8 x 10^{-3}	0,00	35,374s	0,02444
ME5	4	3,8 x 10^{-3}	0,00	19,741s	0,02959
ME6	6	3,8 x 10^{-3}	0,00	14,522s	0,02782

O ligeiro aumento do erro na adição de especialistas na média de ensemble pode ser atribuído ao tipo de treinamento utilizado, em que o conjunto de treinamento é reduzido conforme o acréscimo de especialistas. Assim para um conjunto diminuto em cada especialista acarreta numa redução da qualidade da resposta global. Isto pode ser solucionado alterando-se a estratégia de treinamento, utilizando por exemplo a técnica *Bagging* da maneira usual, quando o tempo de processamento não seja um fator condicionante.

A Tabela 5.5 demonstra a relação de ganhos e da eficiência dado a aplicação do paralelismo. Os valores são obtidos a partir da aplicação das equações Eq. 3.3 e Eq. 3.4, apresentadas no Capítulo 3. Na Figura 5.2 apresentam-se as curvas de ganhos paras os ensembles em função do número de especialistas (número de processadores).

Tabela 5.5: Tabelo de ganhos e eficiência obtidos com o paralelismo

Arquitetura	N. proc.	Tempo execução	Ganho	Eficiência
[2 15 4]	1	49,831s	1,00	1,00
ME1	2	25,130s	1,98	0,99
ME2	4	14,539s	3,42	0,85
ME3	6	10,864s	4,58	0,76
[2 10 10 4]	1	70,091s	1,00	1,00
ME4	2	35,374s	1,98	0,99
ME5	4	19,741s	3,55	0,88
ME6	6	14,522s	4,82	0,80

A partir da Tabela 5.5 observa-se que a utilização do paralelismo acarreta em redução do tempo de processamento para resolução do experimento. Observando-se a rede ME6 por exemplo, quando se aplica o paralelismo com 6 especialistas, o tempo necessário para treinamento é 20,71% do tempo necessário utilizando-se apenas um especialista.

As Figuras 5.3 e 5.4 apresentam gráficamente a atribuição de classes realizadas pela rede ME4 em um conjunto de pontos de teste. Na primeira Figura, a rede é avaliada após um treinamento de 250 épocas, nela apresenta-se a saída individual do especialista 1, do especialista 2 e a saída global da rede. Na segunda Figura a ME4 foi avaliada após um

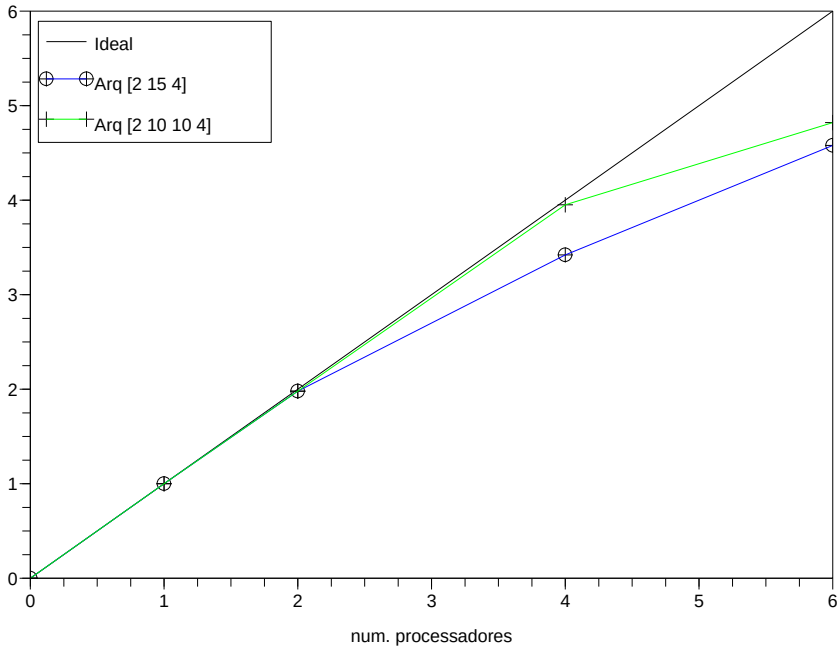

Figura 5.2: Curvas de ganho obtidas para as arquiteturas [2 15 4] (ME1, ME2e ME3) e [2 10 10 4] (ME4, ME5 e ME6).

treinamento de 1000 épocas.

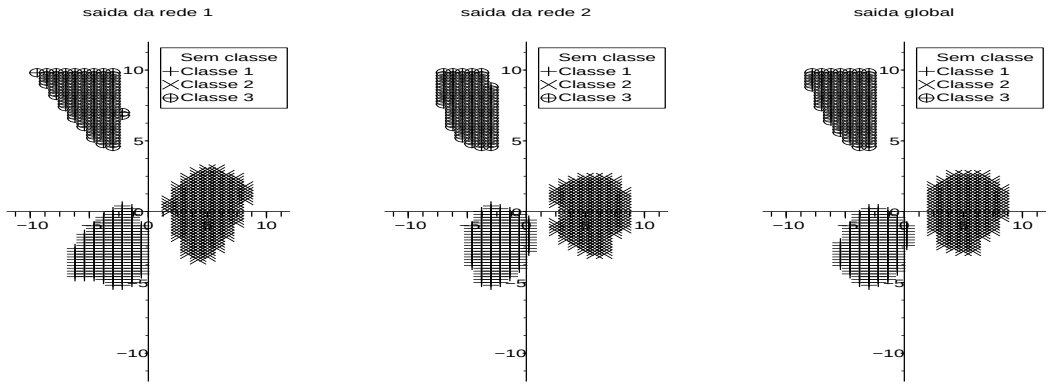

Figura 5.3: Saídas das redes especialistas e global para rede ME4 com dois especialistas após 250 épocas de treinamento.

Observa-se que os especialistas reduzem os valores de erro e aperfeiçoam suas seleções sobre os sub-conjuntos de exemplo que são treinados, o que não configura uma seleção adequada com relação ao conjunto completo de treinamento. Apesar disso, a combinação realizada pela média de ensemble é aperfeiçoada com o passar das épocas,

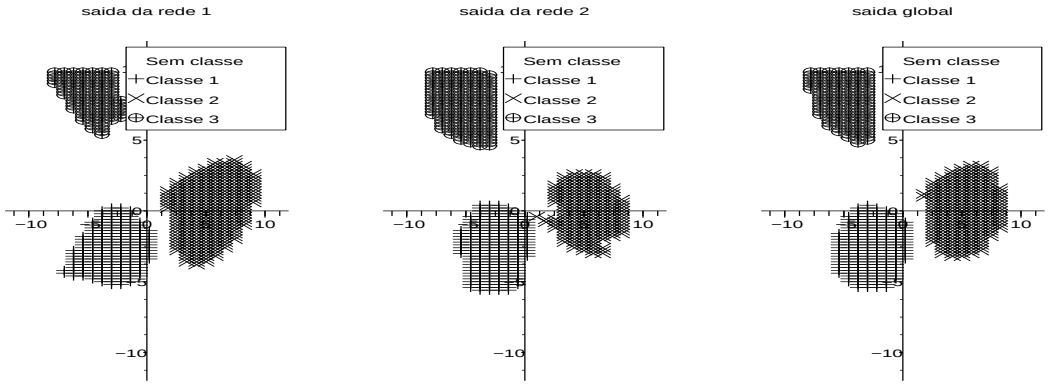

Figura 5.4: Saídas das redes especialistas e global para a rede ME4 com dois especialistas após 1000 épocas de treinamento.

conforme é possível observar por exemplo, na seleção das classes 1 e 2 que tornam-se mais circulares.

5.2 Classificação com rede modular de múltiplas camadas

Este segundo experimento trata da classificação de padrões onde cada classe tem diferentes dimensões sobre um plano cartesiano. Este problema foi desenvolvido para avaliar a estrutura da rede modular com múltiplas camadas e o novo algoritmo de treinamento desenvolvido, observando o seu comportamento durante o tempo (épocas de treinamento), aqui não foi realizado a aplicação do paralelismo.

5.2.1 Domínio do problema

A classificação a ser realizada neste problema trata-se da divisão das classes apresentadas na Figura 5.5, onde os pontos de treinamento dividem-se uniformemente entre o intervalo de $[-1;1]$ na coordenada x, e $[-1;1]$ na coordenada y. A primeira classe de pontos pertence a um circulo de raio unitário circunscrito em um quadrado com lado de dimensão dois e centro na origem do plano cartesiano. As demais classes distribuem-se sobre as regiões externas ao circulo em cada quadrante do plano individualmente.

A Tabela 5.6 apresenta alguns exemplos de dados formatados segundo seus descritores que incluem o valor da coordenada x, o valor da coordenada y e um vetor binário de cinco posições que representam a classe a qual o ponto pertence.

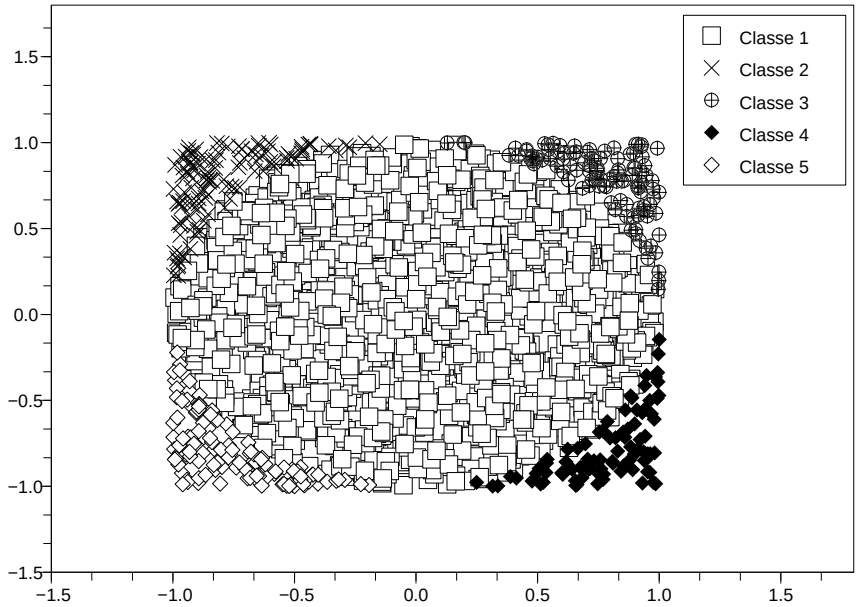

Figura 5.5: Exemplos dos pontos das classes do segundo experimento.

Tabela 5.6: Exemplos de dados do conjunto de treinamento do segundo experimento

x	y	c1	c2	c3	c4	c5	$\in$
-0.7865	0.3919	1	0	0	0	0	classe 1
-0.9343	0.7471	0	1	0	0	0	classe 2
0.6470	0.9637	0	0	1	0	0	classe 3
0.9541	-0.5873	0	0	0	1	0	classe 4
-0.3531	-0.9531	0	0	0	0	1	classe 5

5.2.2 Arquiteturas

Para verificação da rede modular com múltiplas camadas foram realizados treinamentos com redes arquiteturas e diferentes parâmetros de aprendizagem. A Tabela 5.7 apresenta algumas das arquiteturas de redes modulares (RM) avaliadas. O primeiro vetor apresenta a arquitetura de seus especialistas enquanto o segundo apresenta a arquitetura da rede de passagem, da mesma forma como apresentado no experimento anterior. Na representação das funções das redes especialistas e de passagem cada posição do vetor representa a função utilizada nos neurônios da camada de posição correspondente.

A função sigmóide utilizada no treinamento é a logística, que tem sua definição a partir da Eq. 5.3,

Tabela 5.7: Arquiteturas modulares com múltiplas camadas avaliadas

Rede	Arq. esp	Arq. passagem	Funções esp.	Funções pass.
RM1	[2 5 5]	[2 5 5]	[lin sig/th sig/th]	[lin sig/th sig]
RM2	[2 5 5]	[2 15 5]	[lin sig/th sig/th]	[lin sig/th sig]
RM3	[2 10 5]	[2 5 5]	[lin sig/th sig/th]	[lin sig/th sig]
RM4	[2 10 5]	[2 10 5]	[lin sig/th sig/th]	[lin sig/th sig]
RM5	[2 15 5]	[2 10 5]	[lin sig/th sig/th]	[lin sig/th sig]
RM6	[2 15 5]	[2 15 5]	[lin sig/th sig/th]	[lin sig/th sig]
RM7	[2 5 5 5]	[2 10 5]	[lin sig/th sig/th sig]	[lin sig/th sig]
RM8	[2 5 5 5]	[2 5 5 5]	[lin sig/th sig/th sig]	[lin sig/th sig/th sig]
RM9	[2 5 5 5]	[2 10 10 5]	[lin sig/th sig/th sig]	[lin sig/th sig/th sig]
RM10	[2 10 10 5]	[2 5 5 5]	[lin sig/th sig/th sig]	[lin sig/th sig/th sig]
RM11	[2 15 15 5]	[2 10 10 5]	[lin sig/th sig/th sig]	[lin sig/th sig/th sig]

Onde: lin = linear, sig = sigmóide, th = tangente hiperbólica

$$sig(x) = \frac{1}{1+exp(-x)} \tag{5.3}$$

já a função tangente hiperbólica utilizada no treinamento é definida na Eq. 5.4,

$$th(x) = a \times tanh(b \times x) \tag{5.4}$$

onde o parâmetro de amplitude *a* é igual a 1 e o parâmetro de inclinação *b* é igual a 2/3, assim a função tangente hiperbólica tem sua imagem definida dentro do intervalo de [-1 ; 1].

Em todos os casos foram avaliados diferentes parâmetros de treinamento, entre eles a taxa de aprendizagem η, o termo do *momentum* α, e os parâmetros da regra Delta-bar-delta, κ, β e ξ. A Tabela 5.8 apresenta os intervalos de utilização de cada parâmetro para as arquiteturas avaliadas.

Tabela 5.8: Variação dos parâmetros do algoritmo utilizados durante o treinamento

	Valor inicial	Valor final
taxa de aprendizagem η	3,8 x 10^{-1}	3,8x10^{-5}
termo do *momentum* α	0	10^{-5}
kappa κ	10^{-1}	10^{-6}
beta β	2,5 x 10^{-1}	2,5 x 10^{-6}
csi ξ	0,99	0,1

5.2.3 Resultados obtidos

A Tabela 5.9 apresenta os resultados obtidos após um treinamento com 500 épocas, incluindo seu tempo de execução e o erro obtido.

Tabela 5.9: Resultados de treinamentos para arquiteturas modulares com múltiplas camadas em 500 épocas

Rede	tempo execução	erro obtido		Rede	tempo execução	erro obtido
RM1	1m 45,883s	0,0662		RM7	2m 54,183s	0,0528
RM2	2m 7,865s	0,0633		RM8	3m 7,754s	0,0536
RM3	2m 52,512s	0,0655		RM9	3m 19,211s	0,0508
RM4	3m 4,093s	0,0659		RM10	6m 29,637s	0,0351
RM5	4m 12,393s	0,0538		RM11	15m 10,406s	0,0358
RM6	4m 24,078s	0,0609				

Para fazer uma análise do comportamento dos especialistas individualmente foi necessário a geração de gráficos das suas saídas individuais e da saída da rede de passagem. A Figura 5.6 mostra a classificação realizada pela rede RM1.

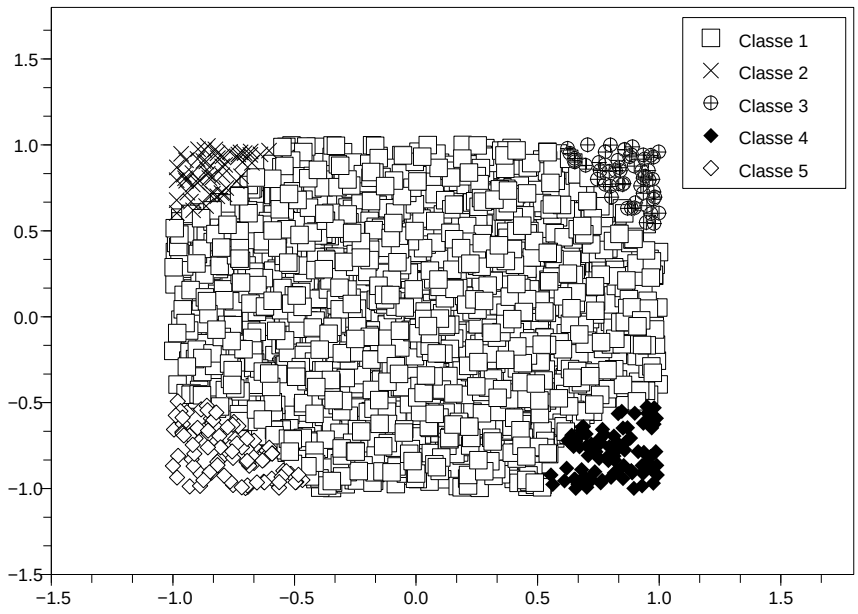

Figura 5.6: Classificação realizada pela rede RM1, após 500 épocas de treinamento.

A Figura 5.7 apresenta a saída global da rede RM1 para 55 exemplos de pontos que foram classificados como pertencentes a classe 2. O eixo x dos gráficos representam cada ponto individualmente, enquanto que o eixo y representa o valor da saída da rede.

Observa-se que há uma próximidade nos valores de saída na seleção da classe um e da classe dois (primeiro e segundo gráficos da primeira linha), isto se deve a fronteira de decisão entre as duas ser bastante próxima. Isto não ocorre com as outras classes (terceiro ao quinto gráfico), onde todas apresentam valores muito próximos de zero para os mesmos pontos.

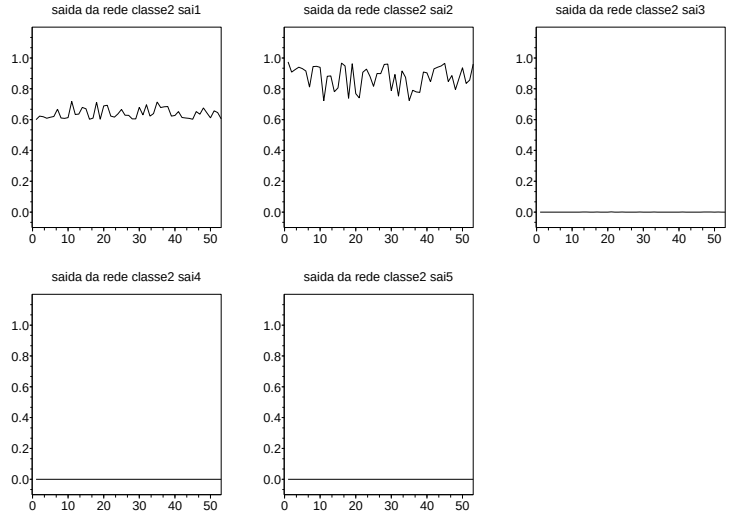

Figura 5.7: Saída global da rede modular com múltiplas camadas, RM1, na seleção dos pontos da classe 2, após 500 épocas de treinamento.

A Figura 5.8 apresenta a saída da rede de passagem da rede RM1 na classificação dos pontos pertencentes a classe 2. O eixo y apresenta o valor das probabilidades *a priori*, que representam o quanto cada especialista deve contribuir para o resultado final, dessa forma, observa-se que o treinamento atribuiu ao primeiro especialista um maior peso na decisão pelos elementos da classe 2, isto demonstra que está ocorrendo uma especialização localizada.

Já na Figura 5.9 apresenta-se a saída fornecida pelo especialista 1 para os pontos atribuídos a classe 2, onde a saída número dois, tem resultados significativamente superiores as demais.

Verificou-se todas as classes e saídas da rede de passagem e das redes especialistas em todos os pontos atribuídos as cinco classes diferentes, em todos os casos ocorreu uma especialização localizada, em alguns momentos distribuídas entre dois ou três especialistas. Verificou-se também, que em todos os casos a maior interferência foi provocada pela classe 1 dado o limiar divisório entre as mesmas.

Na tentativa de reduzir o impacto da interferência provocada pela classe 1, foram realizados experimentos onde os especialistas constituintes da rede modular possuíam diferentes arquiteturas. Esperava-se que um especialista de maior dimensão conseguisse se especializar melhor na primeira classe, dado que ela continha a maior quantidade de pontos, porém ocorreu o contrário, o especialista de maior dimensão se especializava sempre nas menores classes, enquanto os de menores dimensões continuavam com a primeira classe.

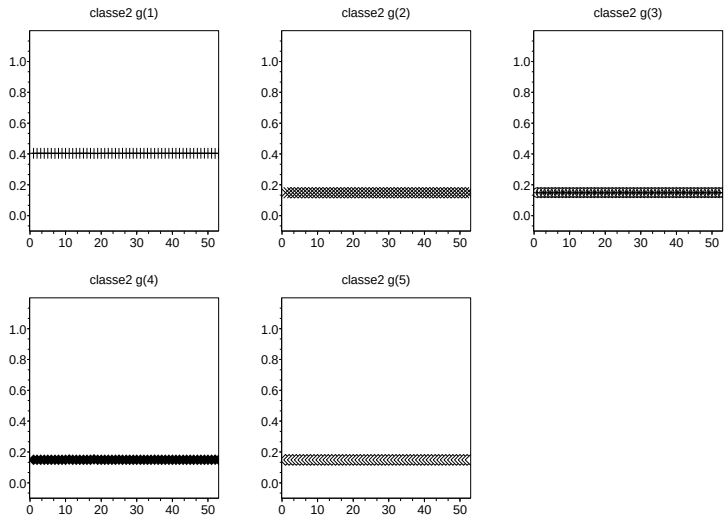

Figura 5.8: Saída da rede de passagem da rede modular com múltiplas camadas RM1, na seleção dos pontos da classe 2, após 500 épocas de treinamento.

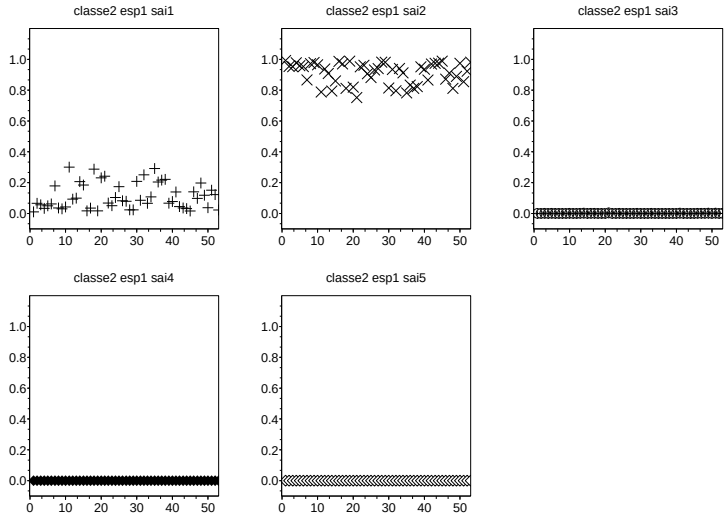

Figura 5.9: Saída fornecida pelo especialista 1 da rede RM1, na seleção dos pontos da classe 2.

Foram realizados experimentos com diferentes valores para a quantidade de épocas nos treinamentos da redes. Nas situações em que as arquiteturas tinham grandes dimensões, caso das redes RM10 e RM11, o acréscimo de épocas acarretou na redução do erro médio quadrado e apresentou melhores resultados. Nesses casos, ocorreu uma especialização maior e uma separação dos dados mais perceptível. Já nos casos em que as arquiteturas tinham pequenas dimensões, caso da rede RM1, o aumento da quantidade de épocas não provocou uma queda no valor do erro médio quadrado, e além disso ocorreu uma processo de generalização nos especialistas, ou seja, deixaram de ser especialistas localizados e passaram gradualmente a responder por toda a arquitetura da rede. Logo as

probabilidades *a priori* fornecidas pela rede de passagem tenderam a se estabilizar com valores próximos entre si.

5.3 Aproximação de funções com ensemble de redes modulares

O terceiro experimento trata da aplicação das redes PMC, redes modulares com múltiplas camadas e uma estrutura hierárquica em um *cluster* na resolução do problema de aproximação de funções. É feito uma avaliação do seu comportamento no ambiente paralelo bem como a comparação de resultados com estruturas PMC.

5.3.1 Domínio do problema

O problema de aproximação de funções é bastante difundido na literatura, motivo pelo qual foi incluído neste estudo de caso. A curva a ser aproximada neste problema é uma composição de várias outras funções distintas e em domínios diferentes. A Figura 5.10 apresenta a curva a ser aproximada em todo o seu domínio.

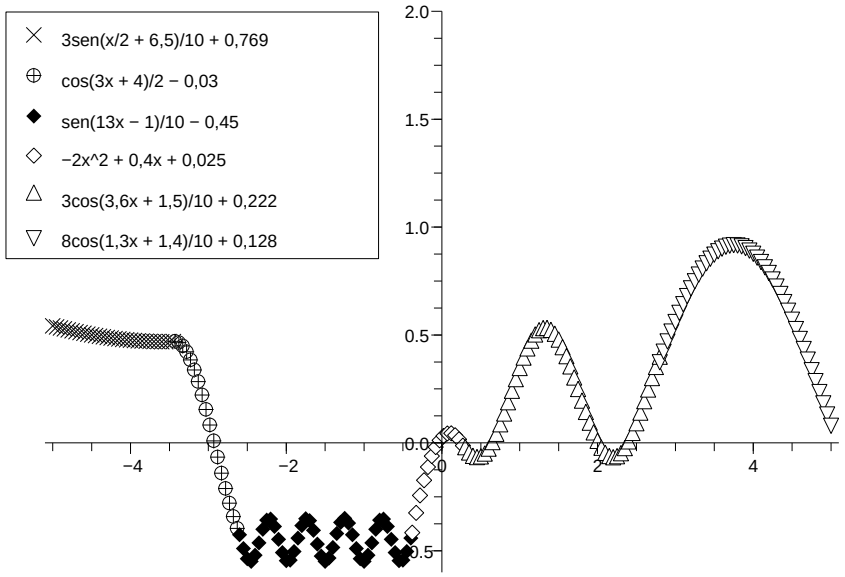

Figura 5.10: Curva a ser aproximada pelo experimento 3.

A função é definida entre o intervalo de $[-5;5]$ e trata-se de uma combinação de seis

funções, podendo ser obtida a partir da aplicação da Eq. 5.5 a seguir

$$f(x) = \begin{cases} 0,3sen(0,5x+6,5)+0,769 & \text{se} \quad -5 \leq x < -3,43 \\ 0,5cos(3x+4)-0,03 & \text{se} \quad -3,43 \leq x < -2,6 \\ 0,1sen(13x-1)-0,45 & \text{se} \quad -2,6 \leq x < -0,38 \\ -2x^2+0,4x+0,025 & \text{se} \quad -0,38 \leq x < 0,3 \\ 0,3cos(3,6x+1,5)+0,222 & \text{se} \quad 0,3 \leq x < 2,8 \\ 0,8cos(1,3x+1,4)+0,128 & \text{se} \quad 2,8 \leq x \leq 5 \end{cases} \tag{5.5}$$

5.3.2 Arquiteturas

Para realização deste experimento foram desenvolvidas três estruturas distintas: a primeira é uma rede perceptron de múltiplas camadas, a segunda trata-se de uma rede modular de múltiplas camadas e por último uma máquina de comitê do tipo média de ensemble de redes modulares de múltiplas camadas. Para a primeira estrutura foram realizados experimentos com diferentes redes PMC e diferentes parâmetros de ajuste do algoritmo da retropropagação. A Tabela 5.10 apresenta as arquiteturas das redes PMC avaliadas, com camadas de 15 a 100 neurônios, e os intervalos de ajustes para os parâmetros de treinamento: η, α, κ, β e ξ.

Tabela 5.10: Arquiteturas PMC avaliadas

Rede	Arquit.	η	α	κ	β	ξ
PMC1	[1 15 1]	3,3, a 10^{-8}	0 a 10^{-5}	0	0	0
PMC2	[1 25 1]	3,3 a 10^{-8}	0 a 10^{-5}	0	0	0
PMC3	[1 35 1]	3,3 a 10^{-8}	0 a 10^{-5}	0	0	0
PMC4	[1 50 1]	3,3 a 10^{-8}	0 a 10^{-5}	0	0	0
PMC5	[1 15 1]	3,86x10^{-3}	0	0,1 a 10^{-5}	0,2 a 2,5x10^{-5}	0,1 a 0,9
PMC6	[1 25 1]	3,86x10^{-3}	0	0,1 a 10^{-5}	0,2 a 2,5x10^{-5}	0,1 a 0,9
PMC7	[1 35 1]	3,86x10^{-3}	0	0,1 a 10^{-5}	0,2 a 2,5x10^{-5}	0,1 a 0,9
PMC8	[1 50 1]	3,86x10^{-3}	0	0,1 a 10^{-5}	0,2 a 2,5x10^{-5}	0,1 a 0,9
PMC9	[1 80 1]	3,86x10^{-3}	0	0,1 a 10^{-5}	0,2 a 2,5x10^{-5}	0,1 a 0,9
PMC10	[1 100 1]	3,86x10^{-3}	0	0,1 a 10^{-5}	0,2 a 2,5x10^{-5}	0,1 a 0,9

A redes modulares de múltiplas camadas avaliadas são constituídas de três especialistas, onde a Tabela 5.11 apresenta as arquiteturas dos especialistas e das redes de passagem avaliadas, e também suas respectivas funções de ativação para cada camada.

A terceira estrutura foi desenvolvida no intuito de verificar a capacidade do framework na criação de estruturas hierárquicas. Trata-se de um máquina de comitê do tipo média de ensemble em que seus constituintes especialistas são redes modulares de múltiplas camadas, representada pelo diagrama da Figura 5.11. Os ensembles foram construídos com duas, quatro e seis redes especialistas. Para este caso as redes modulares serão definidas como os especialistas das médias de ensemble, enquanto que suas próprias redes especia-

Tabela 5.11: Estruturas de redes modulares com múltiplas camadas avaliadas

Rede	Rede especialistas	Rede passagem	funções esp.	funções pass.
RM1	[1 5 1]	[1 5 3]	[lin sig/th th]	[lin sig/th sig]
RM2	[1 5 1]	[1 10 3]	[lin sig/th th]	[lin sig/th sig]
RM3	[1 10 1]	[1 5 3]	[lin sig/th th]	[lin sig/th sig]
RM4	[1 10 1]	[1 10 3]	[lin sig/th th]	[lin sig/th sig]
RM5	[1 5 5 1]	[1 5 3]	[lin sig/th th th]	[lin sig/th sig]
RM6	[1 5 5 1]	[1 10 3]	[lin sig/th sig/th th]	[lin sig/th sig]
RM7	[1 10 10 1]	[1 5 3]	[lin sig/th sig/th th]	[lin sig/th sig]

Onde: lin = linear, sig = sigmóide, th = tangente hiperbólica

listas são denominadas de sub-especialistas do ensemble. Cada rede modular especialista foi avaliada em três configurações, assim foram construídas ao total nove médias de ensemble, cujas arquiteturas estão apresentadas a seguir e sumarizadas na Tabela 5.12.

MERM1 Dois especialistas cada um constituído de três sub-especialistas de arquiteturas [1 5 1] e [1 5 3] para passagem.

MERM2 Quatro especialistas cada um constituído de três sub-especialistas de arquiteturas [1 5 1] e [1 5 3] para passagem.

MERM3 Seis especialistas cada um constituído de três sub-especialistas de arquiteturas [1 5 1] e [1 5 3] para passagem.

MERM4 Dois especialistas cada um constituído de três sub-especialistas de arquiteturas [1 10 1] e [1 10 3] para passagem.

MERM5 Quatro especialistas cada um constituído de três sub-especialistas de arquiteturas [1 10 1] e [1 10 3] para passagem.

MERM6 Seis especialistas cada um constituído de três sub-especialistas de arquiteturas [1 10 1] e [1 10 3] para passagem.

MERM7 Dois especialistas cada um constituído de três sub-especialistas de arquiteturas [1 5 5 1] e [1 10 3] para passagem.

MERM8 Quatro especialistas cada um constituído de três sub-especialistas de arquiteturas [1 5 5 1] e [1 10 3] para passagem.

MERM9 Seis especialistas cada um constituído de três sub-especialistas de arquiteturas [1 5 5 1] e [1 10 3] para passagem.

Ao contrário do primeiro experimento, este utiliza a técnica de modificação dos parâmetros de aprendizagem da regra delta-bar-delta, os diferentes parâmetros de treinamento testados incluem: $\eta, \alpha, \kappa, \beta, \xi$. a Tabela 5.13 apresenta as faixas variação dos parâmetros do algoritmo de treinamento.

O treinamento para esta estrutura hierárquica foi realizado de forma semelhante ao primeiro experimento, onde o conjunto de treinamento foi divido proporcionalmente ao número de especialistas. A saída global da rede é composta pela média aritmética das saídas individuais de cada rede especialistas, de tal forma que

Tabela 5.12: Arquiteturas das médias de ensembles compostas de redes modulares avaliadas.

Arquiteturas de Ensembles de Redes Modulares

N. Esp	Arq. esp.1			Arq. esp.2			Arq. esp.3		
	N. sub esp.	Arq. sub esp.	Arq. sub pass.	N. sub esp.	Arq. sub esp.	Arq. sub pass.	N. sub esp.	Arq. sub esp.	Arq. sub pass.
2	3	[1 5 1]	[1 5 3]	3	[1 10 1]	[1 10 3]	3	[1 5 5 1]	[1 10 3]
4	3	[1 5 1]	[1 5 3]	3	[1 10 1]	[1 10 3]	3	[1 5 5 1]	[1 10 3]
6	3	[1 5 1]	[1 5 3]	3	[1 10 1]	[1 10 3]	3	[1 5 5 1]	[1 10 3]

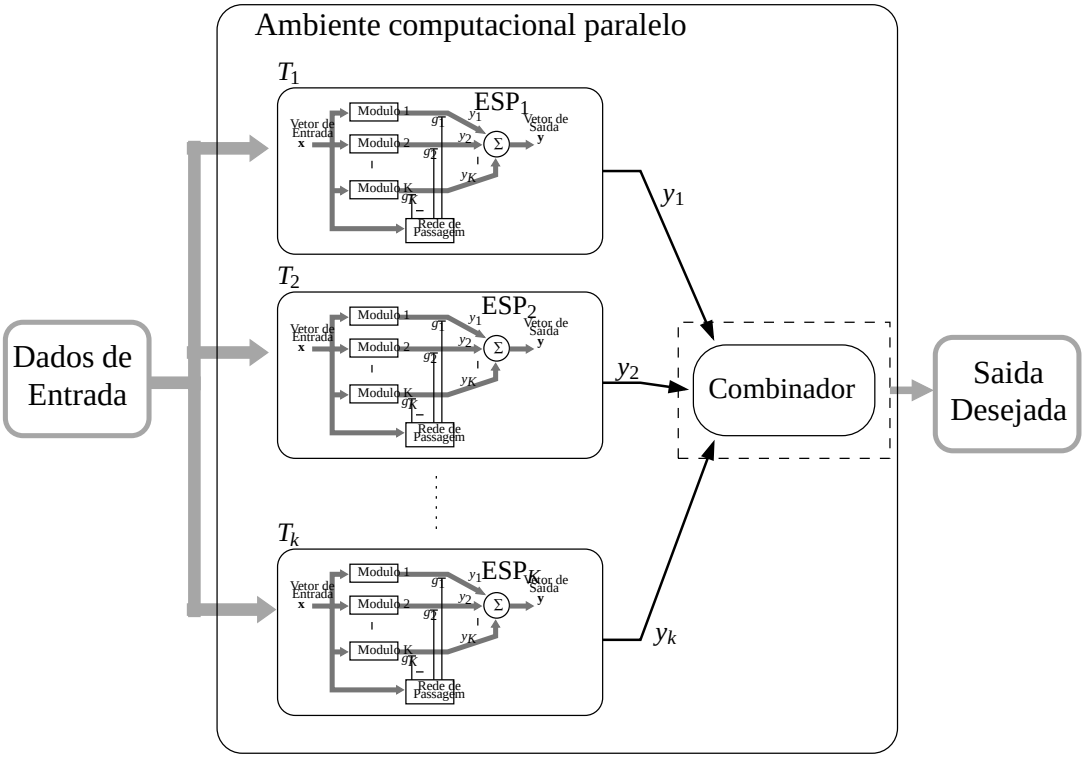

Figura 5.11: Diagrama em blocos da estrutura do tipo média de ensemble com especialistas de redes modulares com múltiplas camadas.

$$\mathbf{Y} = \frac{1}{K}\sum_{i=1}^{K}\mathbf{Y_{esp_i}} \tag{5.6}$$

onde K representa o número de especialistas, no caso de redes modulares, e Y_{esp_i} representa a saída fornecida pelo i-ésimo especialista, obtida a partir de

$$\mathbf{Y_{esp_i}} = \sum_{j=1}^{J} g_{ij}\mathbf{y_{ij}} \tag{5.7}$$

Tabela 5.13: Intervalo de ajuste dos parâmetros de treinamento do algoritmo

η	α	κ	β	ξ
3 a 3,8x10^{-7}	0 a 10^{-5}	10^{-1} a 10^{-5}	0,2 a 2,5x10^{-5}	0,1 a 0,9

onde J é igual ao número de sub-especialistas do especialista i, g_{ij} representa a j-ésima saída da rede de passagem do especialista i, e por fim y_{ij} representa a saída da j-ésima sub-rede do especialista i. A partir da Eq. 5.7 e 5.6 a saída global pode ser definida como

$$\mathbf{Y} = \frac{1}{K}\sum_{i=1}^{K}\sum_{j=1}^{J} g_{ij}\mathbf{y_{ij}} \tag{5.8}$$

5.3.3 Resultados obtidos

Durante o treinamento das redes PMC foram utilizados como parâmetro de parada o número máximo de épocas que foi estipulado em 10000 (dez mil) e o valor máximo para o erro, igual a 0,01%. A Tabela 5.14 apresenta os melhores resultados obtidos para as estruturas PMC, utilizando termo do *momentum*, com seus parâmetros de treinamento e tempos de execução. A Tabela 5.15 apresenta os melhores resultados obtidos com o uso da regra-delta-bar-delta.

Tabela 5.14: Resultados para estruturas PMC

Rede	Arquitetura	η	α	tempo	erro
PMC1	[1 15 1]	3,8 x 10^{-3}	0	6m 15,410s	0,00108040
PMC2	[1 25 1]	3,8 x 10^{-3}	10^{-5}	10m 19,206s	0,00109096
PMC3	[1 35 1]	3,8 x 10^{-3}	0	14m 20,100s	0,00098077
PMC4	[1 50 1]	3,8 x 10^{-3}	0	20m 31,381s	0,00114730

Tabela 5.15: Resultado para redes PMC com uso da regra delta-bar-delta

Rede	Arquit.	η	κ	β	ξ	tempo	erro
PMC5	[1 15 1]	3,8 x 10^{-3}	10^{-4}	0,025	0,7	6m 10,421s	0,00095282
PMC6	[1 25 1]	3,8 x 10^{-3}	10^{-4}	0,025	0,7	10m 10,814s	0,00106137
PMC7	[1 35 1]	3,8 x 10^{-4}	10^{-4}	0,025	0,7	14m 15,219s	0,00112938
PMC8	[1 50 1]	3,8 x 10^{-3}	10^{-4}	0,025	0,7	20m 6,218s	0,00116902
PMC9	[1 80 1]	3,8 x 10^{-3}	10^{-4}	0,025	0,6	32m 27,241s	0,00120582
PMC10	[1 100 1]	3,8 x 10^{-3}	10^{-4}	0,025	0,9	40m 55,681s	0,00154195

Observa-se que em todas as redes do tipo PMC o valor do erro atingido oscilou entre 0,0015 e 0,009. No caso do uso do termo do *momentum* o melhor resultado obtido foi através da rede PMC3 em que o erro obtido foi de 0,00098077 em um tempo de 10

minutos e 19 segundos. Já nas estruturas que fazem uso da regra delta-bar-delta o melhor resultado foi obtido através da rede PMC5, que atingiu o valor de erro de 0.00095282 em 6 minutos e 10 segundos.

Como a diferença total dos valores de erro não foi muito elevado, que foi de 0,006% para este problema, não justifica-se o aumento da arquitetura dado o excesso adicional no tempo de processamento. Para as arquiteturas de dimensão inferior, como a PMC5, o menor tempo de processamento foi de 6 minutos e 10 segundos, enquanto que para a arquitetura de maior dimensão, como a PMC10 com 100 neurônios, o tempo de processamento foi de 40 minutos e 55 segundos, um aumento de aproximadamente 683%.

A segunda etapa de treinamentos foi dividida em três momentos: no primeiro ocorreu a avaliação das redes modulares de forma serial sem a regra-delta-bar-delta, o segundo tratou das redes modulares com a regra delta e o terceiro a aplicação da estrutura hierárquica em paralelo com a comparação de resultados de ganho e eficiência inclusive com realação às redes PMC.

A Tabela 5.16 apresenta os resultados obtidos com a aplicação de redes modulares com três especialistas de múltiplas camadas sem a regra delta, que torna-se o equivalente a aplicação de um ensemble constituído de apenas um especialista modular por sua vez constituído de três sub-especialistas. São apresentados suas arquiteturas, parâmetros, tempo de processamento e erro obtido. A condição de parada foi atingir um erro inferior a 0,01% ou 10000 épocas.

Tabela 5.16: Resultados de treinamentos para arquiteturas modulares com três especialistas de múltiplas camadas em 10000 épocas

Rede	especialistas	passagem	η	α	tempo	erro obtido
RM1	[1 5 1]	[1 5 3]	1,8x10^{-2}	0	11m 50,749s	0,05595227
RM2	[1 5 1]	[1 10 3]	3,8x10^{-3}	10^{-6}	14m 51,145s	0,05527794
RM3	[1 10 1]	[1 5 3]	1,8x10^{-2}	0	18m 44,343s	0,05527794
RM5	[1 5 5 1]	[1 5 3]	1,8x10^{-2}	0	22m 46,542s	0,04670517
RM6	[1 5 5 1]	[1 10 3]	3,8x10^{-3}	0	25m 50,973s	0,04876834
RM7	[1 10 10 1]	[1 5 3]	1,8x10^{-2}	0	53m 44,765s	0,04552572

Na Tabela 5.17 apresentam-se os resultados obtidos com a aplicação de redes modulares com três especialistas de múltiplas camadas com o uso da regra delta-bar-delta. São expostas suas arquiteturas, parâmetros, tempo de processamento e erro obtido. A condição de parada foi atingir 2000 épocas de treinamento ou um erro menor que 0,01%.

Observa-se que a diferença entre os resultados obtidos com a utilização da regra delta-bar-delta nas redes modulares é elevada. Considerando a faixa de erro desejada com valor de 10^{-4}, existe uma diferença de $4x10^{-2}$ entre os erros obtidos pelas duas implementações. Considerando o tempo de processamento, na primeira implementação com arquiteturas reduzidas como a RM1, o menor tempo para realização das 10000 épocas foi de 11 minutos e 50 segundos alcançando um erro de 0,05595227, ao passo que na segunda implementação tem-se que para a maior RM10, o tempo necessário para convergir para um erro de 0,00094542 foi igual 7 minutos e 3 segundos em duas mil épocas. É um ganho

Tabela 5.17: Resultados de treinamentos para arquiteturas modulares com três especialistas de múltiplas camadas fazendo uso da regra delta-bar-delta em 2000 épocas

Rede	espec.	passag.	κ	β	ξ	época.	tempo	erro
RM1	[1 5 1]	[1 5 3]	10^{-3}	2,5x10^{-3}	0,7	2000	2m 18,6s	0,001936
RM2	[1 5 1]	[1 10 3]	10^{-3}	2,5x10^{-3}	0,7	2000	3m 24,0s	0,002755
RM4	[1 10 1]	[1 10 3]	10^{-3}	2,5x10^{-3}	0,7	1400	3m 22,1s	0,000932
RM8	[1 15 1]	[1 15 3]	10^{-3}	2,5x10^{-3}	0,7	1400	6m 10,7s	0,000955
RM9	[1 15 1]	[1 10 3]	10^{-3}	2,5x10^{-3}	0,7	2000	6m 12,2s	0,000939
RM10	[1 20 1]	[1 10 3]	10^{-3}	2,5x10^{-3}	0,7	2000	7m 3,6s	0,000945
RM11	[1 5 5 1]	[1 5 5 3]	10^{-3}	2,5x10^{-3}	0,7	1600	5m 2,0s	0,000922
RM6	[1 5 5 1]	[1 10 3]	10^{-3}	2,5x10^{-3}	0,7	1400	3m 57,1s	0,000894

em tempo de processamento de 15,7 vezes. Isso comprova que a eficiência da regra delta-bar-delta se estende as redes modulares conseguindo reduzir de maneira considerável o tempo necessário ao treinamento.

A Figura 5.12 apresenta algumas curvas do valor do erro através das épocas de treinamento para redes perceptrons de múltiplas camadas (PMC), redes modulares padrões e redes modulares com uso da regra delta. O desempenho da rede modular com múltiplas camadas apresenta-se superior, principalmente no início do treinamento, onde logo nas primeiras cem épocas ele já reduz o erro de maneira mais brusca que nas demais arquiteturas.

Foram realizados os treinamentos com a estrutura hierárquica no cluster, utilizando-se como condição de parada o número máximo de épocas igual a 2000. Os valores dos parâmetros de treinamento foram: η = 3,8x10^{-2}, α = 0, κ = 10^{-3}, β = 2,5x10^{-3} e ξ = 0,7. Estas escolhas foram baseadas nos melhores resultados obtidos com a implementação das redes modulares individuais. A Tabela 5.18 apresenta os resultados obtidos com as estruturas propostas, incluindo o tempo de processamento e o erro obtido.

Tabela 5.18: Resultados de treinamentos dos ensembles de redes modulares fazendo uso da regra delta-bar-delta em 2000 épocas

Rede	N. esp.(proc)	espec.	passag.	tempo	erro obtido
MERM1	2	[1 5 1]	[1 5 3]	1m 11,195	0,0025635
MERM2	4	[1 5 1]	[1 5 3]	36,069s	0,0029765
MERM3	6	[1 5 1]	[1 5 3]	23,693s	0,0035818
MERM4	2	[1 10 1]	[1 10 3]	2m 10,686s	0,0013087
MERM5	4	[1 10 1]	[1 10 3]	1m 5,367s	0,0015317
MERM6	6	[1 10 1]	[1 10 3]	42,730s	0,0021390
MERM7	2	[1 5 5 1]	[1 10 3]	2m 34,551	0,0010029
MERM8	4	[1 5 5 1]	[1 10 3]	1m 19,126	0,0015315
MERM9	6	[1 5 5 1]	[1 10 3]	51,510s	0,0020315

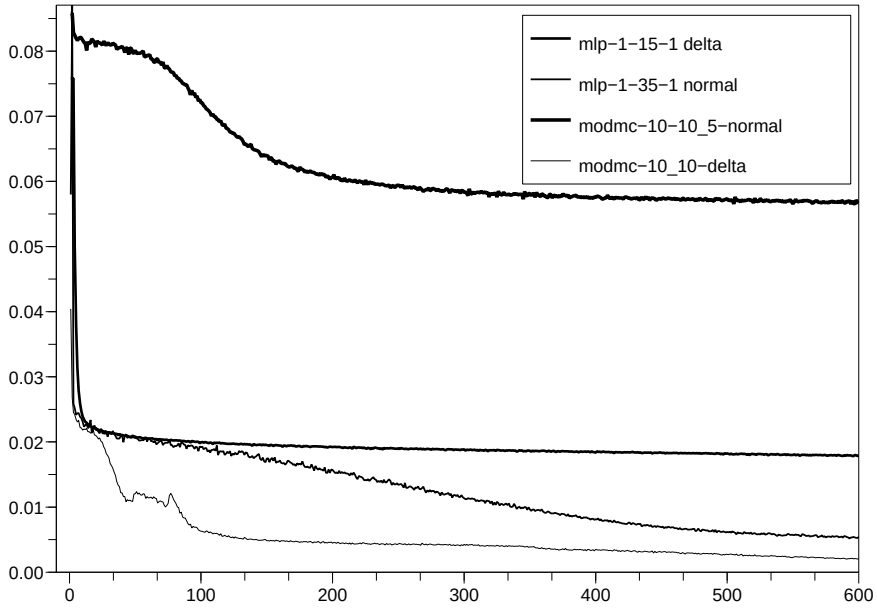

Figura 5.12: Curvas de treinamento para redes PMC e redes modulares com múltiplas camadas.

Assim como no experimento 1, a aplicação do paralelismo na estrutura da média de ensemble proporcionou um redução considerável no tempo de processamento. Também de forma semelhante ao primeiro experimento, ocorreu uma pequena elevação no valor do erro final obtido, atribuído a redução do conjunto de treinamento com o aumento do número de especialistas. A Tabela 5.19 apresenta os ganhos obtidos com a execução em paralelo, enquanto que a Figura 5.13 apresenta as curvas de ganho.

A partir das curvas da Figura 5.13 verifica-se que os ganhos obtidos com a arquitetura menor (RM6) apresentam um desempenho melhor quando aplicado o paralelismo, isto é atribuído à redução do tempo de comunicação. Para verificação da capacidade da estrutura de ensemble foi realizado um teste com a MERM7 onde o treinamento foi realizado com o conjunto de treinamentos com 1000 exemplos em cada especialistas e 10000 mil épocas de treinamento. O tempo total de treinamento foi de 28 minutos e 34 segundos, atingindo um erro de 0,0006223, correspondendo ao menor erro conseguido em todas as estruturas testadas, e um tempo inferior aos das redes PMC9, PMC10 e RM7 por exemplo.

A estrutura hierárquica apresentou resultados melhores que os atingidos pelas rede PMC e as rede modulares sem uso da regra delta-bar-delta. As redes modulares (RM1-RM12) apresentaram o melhor conjunto de resultados e um tempo de processamento reduzido se comparada as demais.

Tabela 5.19: Tabelo de ganhos e eficiência obtidos com o paralelismo da estrutura hierárquica

Arquitetura	N. espec.	Tempo execução	Ganho	Eficiência
RM1	1	2m 18,6s	1,00	1,00
MERM1	2	1m 11,1s	1,946	0,97
MERM2	4	36,0s	3,850	0,96
MERM3	6	23,6s	5,872	0,97
RM4	1	4m 48,714s	1,00	1,00
MERM4	2	2m 25,68s	1,98	0,99
MERM5	4	1m 15,36s	3,83	0,95
MERM6	6	52,73s 71	5,47	0,91
RM6	1	5m 38,7s	1,00	1,00
MERM7	2	2m52,0	1,96	0,98
MERM8	4	1m 29,1s 3,80	0,95	
MERM9	6	59,510s	5,69	0,94

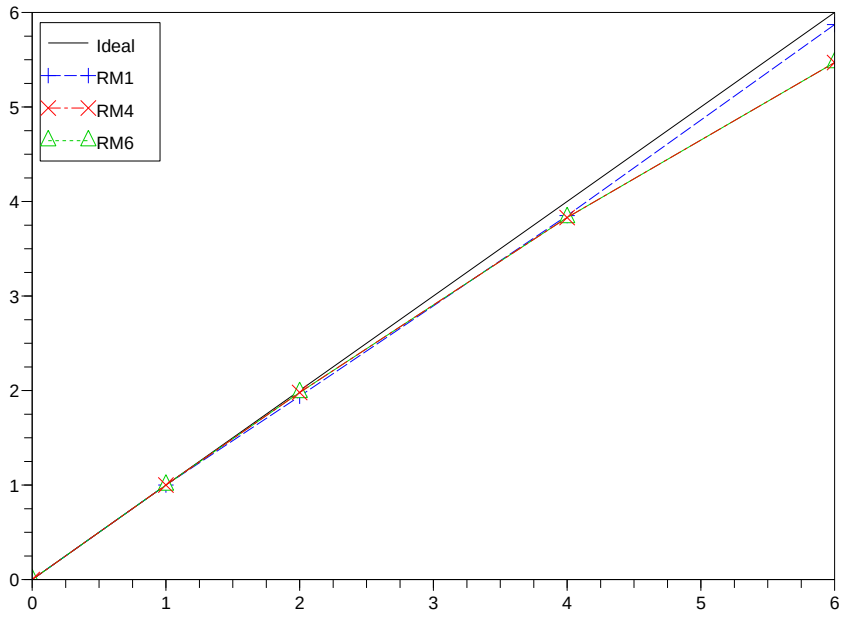

Figura 5.13: Curvas de ganho obtidas para as arquiteturas RM1 (MERM1, MERM2 e MERM3), RM4 (MERM4, MERM5 e MERM6) e RM6 (MERM7, MERM8 e MERM9).

Capítulo 6

Conclusão

Apresentou-se neste trabalho uma investigação da utilização do processamento paralelo no treinamento de redes neurais do tipo máquinas de comitê. Foram analisadas algumas das publicações existentes na literatura e suas características de abordagem utilizadas. Apresentou-se métodos e técnicas da aplicação do paralelismo às máquinas de comitê, incluindo as utilizadas no treinamento de redes de menor dimensão e de estruturas diferentes como as redes de perceptron de múltiplas camadas.

Realizou-se também, uma contribuição com a criação de um algoritmo de treinamento para as redes modulares de múltiplas camadas, isto foi feito a partir de uma combinação dos algoritmos de mistura gaussiano associativo para redes modulares e do algoritmo da retropropagação para redes perceptrons de múltiplas camadas.

Para verificação das contribuições foi criado um conjunto de *softwares* (*framework*), composto de estruturas e métodos para criação e treinamento de redes neurais (comitês, pmc, modulares e modulares estendidas), bem como um estudo sobre a aplicação deste na resolução de problemas de classificação de padrões e aproximação de funções. Os problemas foram avaliados quanto a aplicação em um *cluster*, o tempo de processamento, os resultados obtidos, comportamento das redes e a sua capacidade de criação de estruturas hierárquicas.

Em todas as aplicações que o paralelismo foi utilizado, ocorreu uma redução do tempo de processamento, obtendo desta forma ganho e eficiência no treinamento das redes avaliadas. A utilização do novo algoritmo de treinamento para as redes modulares de múltiplas camadas apresentou resultados superiores aos atingidos com as redes de perceptron de múltiplas camadas treinadas com o algoritmo da retropropagação. O *framework* apresentou uma capacidade de modularização suficiente ao possibilitar uma construção hierárquica de máquinas de comitês compostas de diferentes estruturas.

Torna-se possível, então, responder ao questionamento realizado no capítulo introdutório na seção 1.1. Mostrou-se viável a aplicação do paralelismo em diferentes níveis das redes do tipo máquinas de comitê, ou seja, aplica-se o paralelismo na estrutura global e também em suas redes especialistas. Foi apresentado, também, que a aplicação do paralelismo no algoritmo de treinamento de Jacobs e Jordan [Jacobs & Jordan 1991] é realizável, porém, não apresenta resultados satisfatórios. Entretanto, a aplicação do paralelismo competitivo na estrutura estendida da rede modular proposta, apresentou ganhos significativo no seu tempo de treinamento, validando desta forma a utilização do paralelismo.

O trabalho permite a abordagem de alguns aspectos não avaliados, como a aplicação em problemas de controle ou compactação, utilização de outras redes neurais como constituintes das médias de ensemble, tais como a RBF ou SVM. Há ainda, a possibilidade de abordagens diferentes na aplicação do paralelismo como a abordagem competitiva, ou experimentação de outras técnicas de agrupamentos do ensemble, como: combinadores inteligentes, *voting* ou *bagging* em sua forma padrão.

Referências Bibliográficas

Addahl, Gene M. (1967), The validity of the single processor approach to achieving large scale computing capabilities, *em* 'In AFIPS conference proceedings', Vol. 30, pp. 483–485.

Albuquerque, A. C. M. L., J. D. Melo & A. D. Doria Neto (2004), Algoritmos genéticos e processamento paralelo aplicado ao treinamento e definição de redes neurais perceptrons de múltiplas camadas, *em* 'XV Congresso Brasileiro de Automática', CBA, Gramado, pp. 01–06.

Alves, Robinson L. de S. (2002), Uma Contribuição ao Treinamento de Perceptrons de Múltiplas Camadas usando Retropropagação Competitiva e Processamento Paralelo, Dissertação de mestrado, Universidade Federal do Rio Grande do Norte.

Anderson, Charles W. & Zhaohui Hong (1994), Reinforcement Learning with Modular Neural Networks for Control, *em* 'Proceedings of NNACIP'94, the IEEE International Workshop on Neural Networks Applied to Control and Image Processing'.

Batista, Gustavo E. A. P. A. (2003), Pré-processamento de Dados em Aprendizado de Máquina Supervisionado, Tese de doutorado, Universidade de São Paulo, Instituto de Ciências Matemáticas e de Computação.

Baum, Eric B. & David Haussler (1990), 'What size net gives valid generalization?', *Neural Computation* **1**(1), 151–160.

Bittencourt, Valneide G. (2005), Aplicação de Técnicas de Aprendizado de Máquina no Reconhecimento de Classes Estruturais de Proteínas, Dissertação de mestrado, Universidade Federal do Rio Grande do Norte.

Cary, John R., Svetlana G. Shasharina, Julian C. Cummings, John V. W. Reynders & Paul J. Hinker (1996), Comparison of C++ and Fortran 90 for Object-Oriented Scientific Programming, Relatório Técnico LA-UR-96-4064, Los Alamos National Laboratory, Los Alamos.

Collobert, R., S. Bengio & J. Mariéthoz (2002), Torch: a modular machine learning software library, Relatório Técnico IDIAP-RR 02-46, IDIAP.

Cover, T. (1965), 'Geometrical and statistical properties of systems and linear inequalities with applications in pattern recognition', *IEEE Transactions on Electronic Computers* **19**, 326–334.

Dietterich, Thomas G. (1998), 'Machine-learning research: Four current directions', *The AI Magazine* **18**(4), 97–136.
URL: *citeseer.ist.psu.edu/dietterich97machine.html*

Dimitrakakis, Christos & Samy Bengio (2005), 'Online adaptive policies for ensemble classifiers', *Neurocomputing* **1**(64), 211–221.

Estévez, Pablo A., Hélène Paugam-Moisy, Didier Puzenat & Manual Ugarte (2002), 'A Scalable Parallel Algorithm for Training a Hierarchical Mixture of Neural Experts', *Parallel Computing* **1**(28), 861–891.

Foster, Ian (1995), *Designing and Building Parallel Programs*, Addison-Wesley.

Harmanani, Haidar (2002), 'A parallel neural networks algorithm for the clique partitioning problem', *The International Journal of Computers and Applications* **9**(2).

Haykin, Simon (2001), *Redes Neurais: Princípios e Práticas*, 2ª edição, Bookman, Porto Alegre.

Jacobs, Robert A. & Michael I. Jordan (1991), 'Adaptive Mixtures of Local Expert', *Neural Computation* **1**(3), 79–87.

Kecman, Vojislav (2001), *Learning and soft computing: support vector machines, neural networks, and fuzzy logic models*, MIT Press, Londres.

Konda, Vijay R. & John N. Tsitsiklis (2001), 'Actor-Critic Algorithms', *SIAM Journal on Control and Optimization* **42**(4), 1143–1166.

Kontár, Stanislav (2006), Parallel training of neural networks for speech recognition, *em* 'Proc. 12th International Conference on Soft Computing MENDEL'06'.

Kuncheva, Ludmila I. (2004), *Combining Pattern Classifiers: Methods and Algorithms*, John Wiley & Sons, New Jersey.

Li, Weigang & N. C. da Silva (1999), A study of parallel neural networks, *em* 'IJCNN'99. International Joint Conference on Neural Networks. Proceedings.', Vol. 2, IEEE Service Center, Piscataway, NJ, pp. 1113–16.

Mahapatra, S., R.N. Mahapatra & B.N. Chatterji (1999), 'Mapping of neural network models onto massively parallel hierarchical computer systems', *Journal of Systems Architecture* **45**(11), 919–929.

Merkey, Phil (2004), 'Beowulf History', Disponível em: <http://www.beowulf.org/ overview/ history.html >.
URL: *http://www.beowulf.org/overview/history.html*

Mitchell, Tom M. (1997), *Machine Learning*, McGraw-Hill.

Moore, Shirley, Felix Wolf, Jack Dongarra, Sameer Shende, Patricia Teller & Bernd Mohr (2005), A Scalable Approach to MPI Application Performance Analysis, *em* 'Proceedings of Recent Advances in Parallel Virtual Machine and Message Passing Interface: 12th European PVM/MPI Users Group Meeting', Vol. 3666, Springer-Verlag GmbH Lecture Notes in Computer Science, pp. 309–316.

Neumann, Gerhard (2005), The Reinforcement Learning Toolbox, Reinforcement Learning for Optimal Control Tasks, Dissertação de mestrado, Technischen Universität, Graz.

Optiz, David & Richard Maclin (1999), 'Popular Ensemble Methods: An Empirical Study', *Journal of Artificial Intelligence Research* **11**, 169–198.

Park, Jung-Wook & G. K. Venayagamoorthy (2003), Adaptive Critic Designs and their Implementations on Different Neural Network Architectures, *em* 'Proceedings of the International Joint Conference on Neural Networks', Vol. 3, pp. 1879–1884.

Petridis, V., P. Adamidis & K.G. Margaritis (1993), 'On the parallel simulation of artificial neural networks'.
URL: *http://www.citeseer.ist.psu.edu/60729.html*

Pitanga, Marcos (2004), *Construindo Supercomputadores com Linux*, 2ª edição, Brasport, Rio de Janeiro.

Prechelt, Lutz (1995), 'Data locality and load balancing for parallel neural network learning', *Computer and Software Engineering* .
URL: *citeseer.ist.psu.edu/prechelt95data.html*

Provost, Foster John & Daniel N. Hennessy (1996), Scaling up: Distributed machine learning with cooperation, *em* 'Proceedings of the Thirteenth National Conference on Artificial Intelligence', Portland, OR.
URL: *citeseer.ist.psu.edu/provost96scaling.html*

Rezende, Solange O. (2005), *Sistemas Inteligentes: Fundamentos e Aplicações*, Manole, Barueri.

Seiffer, Udo (2002), Artificial neural networks on massively parallel computer hardware, *em* 'In European Symposium on Artificial Neural Networks Proceedings', pp. 319–330.

Sloan, Joseph D. (2004), *High Performance Linux Clusters with OSCAR, Rocks, OpenMosix, and MPI*, O'Reilly, Sebastopol.

Sutton, Richard S. (1992), Reinforcement Learning Architectures, *em* 'Proc. Int. Symp. on Neural Information Processing'.

Sutton, Richard S. & Andrew G. Barto (1998), *Reinforcement Learning An Introduction*, Adaptive Computation and Machine Learning Series, MIT Press, Londres.
URL: *http://www.cs.ualberta.ca/ sutton/book/ebook/the-book.html*

Topping, B.H.V., J. Sziveri, , A. Bahreinejad, J.P.B. Leite & B. Cheng (1998), 'Parallel processing, neural networks and genetic algorithms', *Advances in Engineering Software* **29**(10), 763–786.

Torresen, J. (1996), Parallelization of Backpropagation Training for Feed-Forward Neural Networks, Tese de doutorado, The University of Trondhein.

Valentini, Giorgio & Francesco Masulli (2002), 'Ensembles of learning machines', in Neural Nets WIRN Vietri-02, Series Lecture Notes in Computer Sciences, Eds.: Springer-Verlag.

Wesley-Smith, Ian (2006), A parallel artificial neural network implementation, *em* 'Proceedings of The National Conference On Undergraduate Research (NCUR) 2006'.

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo