Universidade do Minho

Escola de Engenharia

Departamento de Informática

Resolução de Problemas em Ambientes Distribuídos

Uma Contribuição nas Áreas da Inteligência Artificial e da Saúde

Victor Manuel Rodrigues Alves

Tese de Doutoramento

2002

Universidade do Minho

Escola de Engenharia

Departamento de Informática

Resolução de Problemas em Ambientes Distribuídos

Uma Contribuição nas Áreas da Inteligência Artificial e da Saúde

Victor Manuel Rodrigues Alves

Tese submetida à Universidade do Minho
para obtenção do grau de Doutor em Informática, elaborada sob a orientação do
Professor Doutor José Carlos Ferreira Maia Neves

2002

Para a Margarida,

Para o João, o Daniel e o Pedro,

Para os meus Pais.

Agradecimentos

Em primeiro lugar gostaria de agradecer ao supervisor, Professor Doutor José Carlos Ferreira Maia Neves, que mostrou ser detentor de uma grande paciência, tendo-me dado muitos conselhos e orientação científica ao longo destes anos em que decorreram os meus estudos. Esteve sempre presente nos bons e maus momentos, dando apoio e encorajamento.

Foi um prazer ter trabalhado com o Abílio Ribeiro e o Filipe Santos, meus colegas de laboratório e que sempre acompanharam os meus trabalhos tendo contribuído com muitas e preciosas observações para este estudo. Um agradecimento especial também para o César Analide, Paulo Novais e Paulo Cortez, meus colegas de grupo.

Não posso deixar de agradecer em particular ao Luís Nelas e ao Dr. Moreira Maia toda a sua colaboração, ajuda e apoio, e ao Centro de Tomografia de Braga, em geral pela sua contribuição em termos técnicos e disponibilidade de equipamentos.

Agradeço a todos aqueles que tiveram o incómodo de colocarem os seus trabalhos e publicações na Internet, que tornaram a minha pesquisa bibliográfica mais fácil e agradável.

Por fim agradeço a todos aqueles que viveram comigo esta jornada, a minha família e os meus amigos, que sempre acreditaram que tudo tem um fim.

Este projecto foi suportado por uma bolsa de investigação com a duração de três anos por parte da JNICT[1] e uma bolsa complementar, visando os mesmos objectivos com a duração de um ano por parte do PRAXIS[2] XXI.

[1] Junta Nacional de Investigação Científica e Tecnológica, bolsa de doutoramento BD/1341/91-IA.

[2] Bolsa PRAXIS/BD/3387/94.

Resumo

O desenvolvimento de infra-estruturas computacionais nas unidades que prestam cuidados de saúde colocam sérios desafios à concepção e desenvolvimento de sistemas de *Software* (*SW*), e *Hardware* (*HW*), assim como às tecnologias de integração desses mesmos sistemas; i.e., à medida que a cardinalidade e complexidade dos sistemas computacionais aumenta, a integração de subsistemas, que se pretendem cooperantes e independentes, aproxima-se em dificuldade aos das aplicações de larga escala [Berner et al. 1996]. Itens da integração incluem suporte para uma grande variedade de *Hardware* computacional e acessórios, acesso a recursos computacionais aleatoriamente distribuídos, incorporação de sistemas de segurança, a possibilidade de parametrização e evolução de acordo com a progressão natural das necessidades dos utilizadores, o controlo do desenvolvimento de *Software* e dos custos de manutenção, só para referir alguns. Sistemas de Informação Médica (*SIM*) computorizados serão claramente cruciais para a viabilização de sistemas de saúde no futuro [Altman 1997]. *SIMs* serão necessários para interligar instituições de saúde de modo a garantir que os cuidados de saúde são providenciados de acordo com as melhores directrizes e protocolos, sendo esta uma área em que Medicina e as Ciências da Computação se poderão sobrepor. O processamento automático de informação pode providenciar eficiência e poupanças ao sistema de saúde, nomeadamente numa melhor utilização do tempo dos agentes de saúde, melhores diagnósticos e terapias, prevenção de erros, menores tempos de internamente, assim como maior bem estar para os pacientes [Barnett & Cimino 1987] [Neves et al. 1999a].

Informática Médica (*IM*) é sem dúvida um campo emergente e, naturalmente, novas tecnologias serão chamadas para sustentar os seus constantes avanços [Wolfram 1995]. Com a Internet abre-se uma oportunidade sem precedentes para a cooperação global, trabalho em rede, acesso à informação e sua partilha. Oferece aos médicos a oportunidade de fazer parte do esforço pioneiro para fornecer acesso multimédia aos recursos médicos globalmente ligados [Bodenreider et al. 1998] [Neves et al. 1999b]. O objectivo deste trabalho passa por desenvolver um novo paradigma para a resolução de problemas na área médica através de uma estreita ligação entre as Ciências Médicas e as Ciências da Computação, com aplicação à resolução de problemas em sistemas complexos e dinâmicos, como o são o das unidades prestadoras de cuidados de saúde.

Abstract

The construction of a highly integrated hospital-wide computing infrastructure poses serious challenges to today software design, development and integration technologies; i.e., as the size and complexity of the computing systems grows, integration of independent co-operating sub-systems is becoming a potential approach to large-scale applications [Berner et al. 1996]. Issues in integration include support for a variety of computing and censoring hardware, access to arbitrarily distributed computing resources, incorporation of legacy systems, the ability to customise and evolve with changing user needs, the control of software development and maintenance costs, just to name a few. Clearly, computerised Medical Information Systems (*MIS*) will be crucial for the enabling of more efficient health care systems in the future [Altman 1997]. *MIS* will be needed to link health care institutions together to ensure that health care is provided in accordance with the best care guidelines and protocols, an area in which Medicine and Computer Science may overlap. Clinical computing can provide efficiencies and savings to the health care system, namely better use of professional health care provider time, better patient diagnosis and therapy, error prevention, shortened length of stay, and better patient outcome. Medical Informatics (*MI*) is indeed an emerging field, and naturally, new techniques are and will be brought in to sustain new advances [Wolfram 1995]. With Internet, an unprecedented opportunity for global co-operation, networking, information access, and sharing is open. It gives physicians the opportunity to be part of pioneering efforts to provide multimedia access to globally linked medical resources; i.e., it will emphasise the effects of *MI* on patient care, patient education, medical education and clinical research

[Bodenreider et al. 1998]. One's goal in this work aims at a new paradigm for problem solving based on theorem proving with application to diagnosis in complex, dynamic environments, which is a typical situation in any health care unit.

Conteúdo

Lista de Figuras

Lista de Tabelas

Notação e Terminologia

Notação Geral

A notação ao longo do documento segue a convenção apresentada a seguir:

- *Texto em itálico* – para palavras em língua estrangeira (e.g., Inglês). Também utilizado para dar ênfase a um determinado termo ou expressão.
- **Texto em negrito** – utilizado para realçar um conceito ou palavra no meio de um parágrafo.
- *Parágrafo com espaçamento simples em itálico* – para extractos de código ou axiomas lógicos.

Acrónimos

AAAI	*American Association for Artificial Intelligence*. (Associação Americana para a Inteligência Artificial)
ACP	Análise de Componentes Principais.
ACR	*American College of Radiology*. (Colégio Americano de Radiologia)
ADI	Análise de Dados Inteligente.
AIC	*Akaike Information Criteria*. (Critério de Informação de Akaike)
ART	*Adaptive Resonance Theory*. (Teoria da Ressonância Adaptativa)
BC	Base de Conhecimento.

BD Base de Dados.

BIC ou SBC *Bayes Information Criteria*. (Critério de Informação de Bayes)

BP *Back-Propagation*.

CGI *Common Gateway Interface*.

DEC *Digital Equipment Company*.

DICOM *Digital Imaging and Communications in Medicine*.

DIMSE Objectos de Serviços de Mensagem.

HIS *Hospital Information System*. (Sistema de Informação Hospitalar)

HW *Hardware*.

IA Inteligência Artificial.

IAAI *Innovative Applications of Artificial Intelligence*. (Aplicações Inovadoras de Inteligência Artificial)

IAD Inteligência Artificial Distribuída.

IE Entidade de Informação.

IOD Definições de Objecto de Informação.

MAI Modulo de Aquisição de Informação.

MCR Modulo para a Configuração das *RNAs*.

MD Módulo de Diagnóstico.

MI ou IM *Medical Informatics*. (Informática Médica)

MIS *Medical Information Systems*. (Sistemas de Informação Médica)

MIT *Massachusetts Institute of Technology*.

MPN Módulo de Pré-Processamento e Normalização de Dados.

MTR Módulo de treino da *RNA*.

NEMA *National Electrical Manufactures Association*.

OA Operação de Alteração.

OSI *Open System Interconnection*. (Interconexão de Sistemas Abertos)

PACS *Picture Archiving and Communication Systems*. (Sistemas de Comunicação e Arquivo de Imagem)

PAP *Practical Applications of Prolog*. (Aplicações Praticas de Prolog)

PC *Personal Computer*. (Computador Pessoal)

PL Programação em Lógica.

PLC Programação em Lógica Contextual

PLE Programação em Lógica Estendida.

PMF Pressuposto do Mundo Fechado.

PPD Processamento Paralelo Distribuído.

QN Quadro Negro.

RAS *Remote Access Service*. (Serviço de Acesso Remoto)

RBF *Radial-Basis Functions*.

RDIS Rede Digital com Integração de Serviços

RFMC Redes Feedforward MultiCamada.

RFSC Redes Feedforward de uma Só Camada.

RIS *Radiology Information System*. (Sistema de Informação Radiológico)

RM Ressonância Magnética.

RMC Redes MultiCamada.

RNA Rede Neuronal Artificial.

ROI *Regions of Interest*. (regiões de Interesse)

RPAD Resolução de Problemas em Ambientes Distribuídos.

RR Redes Recorrentes.

RSNA Reunião anual da Sociedade de Radiologia da América do Norte.

RX Raio X

SADM Sistemas de Apoio ao Diagnóstico Médico.

SADMED Ambiente Computacional para Sistemas de Apoio ao Diagnóstico Médico.

SBQN Sistema Baseado em Quadros Negros.

SC Sistema Computacional.

SCP *Service Class Provider*. (Classe de Serviço Fornecedor)

SCU *Service Class User*. (Classe de Serviço Cliente)

SID Sistemas Inteligentes de Diagnóstico.

SIM Sistemas de Informação Médica.

SM Sistemas Multiagentes.

SOM *Self-Organizing Maps*.

SOP *Service/Object Pair*. (Par Serviço/Objecto)

SPD Sistema de Processamento Distribuído.

SPI *Standard Product Interconnect*.

SQL *Structured Query Language*.

SRCR Sistema de Representação de Conhecimento e Raciocínio.

SW *Software*.

TC Tomografia Computorizada.

TS *Tuple Space*. (Espaço de Tupletos)

UH Unidades Hounsfield.

WL *Window Level*. (Nível de Janela)

WW *Window Width*. (Largura de Janela)

Símbolos Gerais e Abreviaturas

$\boldsymbol{aga_k}$ Agentes de Apoio ao Diagnóstico do *SADMED*.

$\boldsymbol{agm}$ Agente Monitor do *SADMED*.

$\boldsymbol{ags_j}$ Agentes Servidores do *SADMED*.

$\boldsymbol{agc_i}$ Agentes de Conhecimento do *SADMED*.

$\boldsymbol{agr}$ Agente de Recursos do *SADMED*.

$\boldsymbol{qenv}$ O Ambiente do *SADMED*.

$\boldsymbol{teo}$ Teoria lógica.

$\boldsymbol{obj}$ Objectivo.

$\boldsymbol{prog}$ Programa lógico.

$\boldsymbol{\Xi}$ Sistema *SADMED*.

$\boldsymbol{C_{sadmed}}$ Contexto do sistema *SADMED*.

Δ_{sadmed} Conjunto das regras-ponte do sistema *SADMED*.

Capítulo 1
Introdução

Notas introdutórias, inteligência artificial em diagnóstico médico, evolução da inteligência artificial. Motivação, objectivos, e organização da tese.

Uma das linhas de investigação na área científica da Inteligência Artificial (*IA*) está directamente relacionada com a especificação e construção de entidades computacionais autónomas, que se revêem num certo universo de discurso. Cada uma destas entidades ou agentes é realizada através de uma função (um programa de computador) que transforma estímulos em acções e que, nos casos de insuficiência de informação, evidenciam capacidades de aprendizagem como forma de ultrapassarem tal senão, de se manterem autónomos e de equacionarem o seu próprio desempenho [Russel & Norvig 1995].

O desenvolvimento de infra-estruturas computacionais nas unidades de saúde coloca sérios desafios à concepção e desenvolvimento de *Software* e *Hardware*, e ás tecnologia de integração; i.e., à medida que a cardinalidade e complexidade dos sistemas computacionais aumenta, a integração de subsistemas cooperantes independentes revê-se numa aproximação potencial ás aplicações de larga escala [Berner et al. 1996]. Itens da integração incluem suporte para uma grande variedade de *Hardware* computacional e acessórios, acesso a recursos computacionais aleatoriamente distribuídos, incorporação de sistemas de segurança, a possibilidade

de parametrização e evolução de acordo com evolução natural das necessidades dos utilizadores, o controlo do desenvolvimento de Software e dos custos de manutenção, só para referir alguns desses indicadores. Sistemas de Informação Médica (*SIM*) computorizados serão claramente cruciais para a viabilização de sistemas de saúde mais eficientes, quer no presente, quer no futuro [Altman 1997]. *SIMs* serão necessários para interligar instituições de saúde de modo a garantir que os cuidados de saúde são providenciados de acordo com as melhores directrizes e protocolos, sendo esta uma área em que Medicina e as Ciências da Computação se poderão interligar. O processamento automático de informação clínica pode providenciar eficiência e poupanças ao sistema de saúde, nomeadamente numa melhor utilização do tempo dos agentes de saúde, melhores diagnósticos e terapias, prevenção de erros, menores tempos de internamento, e melhor qualidade de vida dos doentes [Barnett & Cimino 1987] [Neves et al. 1999a].

Informática Médica (*IM*) é sem dúvida um campo emergente, e naturalmente novas tecnologias são chamadas para sustentar os seus constantes avanços na melhoria da prestação dos cuidados de saúde [Wolfram 1995]. Com a Internet abre-se uma oportunidade sem precedentes para a cooperação global, trabalho em rede, acesso à informação e sua partilha. Oferece aos médicos a oportunidade de fazer parte do esforço pioneiro para fornecer acesso multimédia aos recursos médicos globalmente interligados [Bodenreider et al. 1998] [Neves et al. 1999b], assim como participar na sua elaboração. Em particular, em serem chamados a definir novos métodos de aprendizagem os quais, embora possam induzir algoritmos de diagnóstico fiáveis a partir duma descrição nem sempre completa do estado do paciente, tais ferramentas de diagnóstico não podem nem tão pouco é sua intenção substituir os médicos, mas poderão ser considerados ferramentas úteis na tarefa de melhorar o seu desempenho.

1.1 Enquadramento

Em termos de uma evolução a longo prazo, as futuras aplicações das tecnologias da *Inteligência Artificial* (*IA*), tendo embora em linha de conta que grande parte das previsões do passado relativamente a estas tecnologias nem sempre tenham levado aos resultados desejados, neste trabalho serão realçados alguns dos aspectos tecnológicos presentemente a serem utilizados na resolução de problemas em áreas científicas que são atravessadas horizontalmente pela área científica da *IA*.

Serão analisados alguns aspectos de algumas aplicações práticas, com referência a encontros que ocorrem à escala planetária, em que são de realçar a *Innovative Applications of Artificial Intelligence* (*IAAI*) e a *Practical Applications of Prolog* (*PAP*), com reflexão sobre o que se pôde e pode vir a aprender a partir de tais fóruns. De seguida será analisada a evolução dos equipamentos que servem de base e limitam as aplicações que têm vindo a ser estudadas. Poder-se-á, assim, chegar a novos e interessantes desenvolvimentos. Finalmente, ver-se-á como este tipo de desenvolvimento se poderá integrar com a prática académica, assim como com a noção de inteligência, particularmente a de raiz humana.

Os resultados a apresentar neste trabalho demonstram que é possível melhorar a exactidão do diagnóstico através da utilização de métodos de aprendizagem automática. Quando se considera a aplicação de sistemas de aprendizagem como forma de assistir o médico, há que considerar vários requisitos específicos que o sistema deverá contemplar. Serão discutidos vários temas relacionados com a utilização de métodos de aprendizagem no diagnóstico médico e problemas de previsão, ilustrando casos problemáticos subjacentes a várias aplicações desenvolvidas no passado. Estes métodos levam a que se classifiquem os algoritmos de aprendizagem em três grandes grupos [Michie et al. 1994]:

- Métodos estatísticos e de reconhecimento de padrões (e.g., k-vizinhos mais próximos, análise discriminante, classificadores Bayesianos);
- Aprendizagem indutiva de regras simbólicas (e.g., indução de árvores de decisão '*top-down*', regras de decisão e indução de programas lógicos); e
- Redes Neuronais Artificiais (*RNAs*) (e.g., *RNAs* multinível com aprendizagem por *backpropagation*, redes auto-organizativas de Kohonem, a memória associativa de Hopfield).

Aparentemente a tecnologia associada aos métodos de aprendizagem é adequada para o diagnóstico médico em alguns problemas específicos e de pequena dimensão. Frequentemente em hospitais especializados e seus departamentos existem dados acerca de diagnósticos sob a forma de registos médicos. Nestes casos basta introduzir os registos dos pacientes com os diagnósticos conhecidos à partida no sistema computacional e correr o algoritmo de aprendizagem. Trata-se de uma visão demasiado simplificada mas, em princípio, o conhecimento sobre o estado clínico de um dado paciente pode ser automaticamente derivado da descrição de casos resolvidos no passado. Estes sistemas poderão então ser usados não só para assistir o médico no seu dia-a-dia de modo a melhorar a velocidade, exactidão e/ou fiabilidade de diagnóstico, mas também na reciclagem/formação de estudantes ou médicos não especialistas.

Sistemas baseados em métodos de aprendizagem foram na realidade aplicados em muitos domínios da área médica (e.g., em oncologia [Bratko & Mulec 1980] [Zwitter et al. 1983] [Kononenko et al. 1984] [Bratko & Kononenko 1987] [Elomaa & Holskti 1989], em patologias ligadas com o fígado [Lesmo et al. 1982], no prognóstico de sobrevivência de pacientes com hepatite [Kononenko et al. 1984], em urologia [Kononenko et al. 1984] [Bratko & Kononenko, 1987] [Roskar et al. 1986], no diagnóstico de doenças da tiróide [Horn et al. 1985] [Hojker et al. 1988] [Quinlan et al. 1987], em reumatologia [Kononenko et al. 1988] [Karalic & Pirnat

1990] [Kern et al. 1990], no diagnóstico do síndroma craniostenosis [Baim 1988], nos diagnósticos dermatológicos [Chan & Wong 1989], em cardiologia [Bratko et al. 1989] [Clark & Boswell 1991] [Catlett 1991], em neuropsicologia [Muggleton 1990], em ginecologia [Nuñez 1990], e em perinatologia [Kern et al. 1990]).

Contudo, nem todos os sistemas de aprendizagem são igualmente apropriados. Quando se aplica um sistema de aprendizagem no diagnóstico médico, existem vários requisitos específicos que o sistema deve contemplar. Ir-se-á de seguida analisar vários assuntos relacionados com a utilização de métodos de aprendizagem no diagnóstico médico e em problemas de prognóstico, tais como em problemas de fiabilidade e transparência de decisões. Por exemplo, serão abordados assuntos tais como:

- Localização do tumor primário, prevendo a recorrência de cancro mamário;
- Diagnóstico de doenças da tiróide;
- Diagnóstico em reumatologia; e
- Previsão de complicações na recuperação de fracturas do pescoço.

Neste trabalho começa-se por:

- Descrever o processo típico de diagnóstico de um paciente e apresentam-se os resultados obtidos no passado por aplicação de algoritmos de aprendizagem em diversos domínios do foro médico; e
- Compara-se o desempenho dos médicos relativamente aos sistemas com aprendizagem, descrevem-se os requisitos específicos para que os sistemas com aprendizagem possam ser aplicados no diagnóstico médico e, finalmente são equacionados os quesitos porque estes sistemas são (não são) aceites na prática.

1.1.1 Diagnóstico Médico

Um processo de diagnóstico típico segue um padrão. Primeiro é efectuada uma entrevista ao paciente durante a qual são recolhidos os dados anamnésicos; de seguida são efectuados exames preliminares durante os quais são recolhidos dados relativos ao seu estado. Conforme os dados anamnésicos e de estado, o paciente efectua exames laboratoriais adicionais. O diagnóstico é depois realizado pelo médico que tem em atenção toda a informação disponível relativo ao estado de saúde do paciente. Dependente do diagnóstico é prescrito um tratamento, findo o qual todo o processo se poderá repetir. Em cada iteração o diagnóstico poderá ser confirmado, refinado ou rejeitado. A definição do diagnóstico final esta dependente do problema (médico) em equação. Em algumas situações o primeiro diagnóstico é também o final; noutros o diagnóstico final é concretizado após estarem disponíveis os resultados do tratamento; e em algumas situações não existe nenhum modo de se obter um diagnóstico final 100% fiável. Por exemplo, na situação de se pretender localizar o tumor primário, o diagnóstico final pode sempre ser obtido através de uma intervenção cirúrgica onde a localização do tumor primário é obtida, embora este acto médico seja geralmente evitado e substituído por outro tipo de exames. Na situação concreta da previsão de recorrência de um cancro de mama, após uma operação de remoção, o diagnóstico final é impossível até cinco anos após a intervenção cirúrgica. E em urologia, na situação de diagnóstico do tipo de incontinência, na prática o diagnóstico final nunca é obtido visto não existir nenhum método prático de verificação do próprio diagnóstico.

É sabido que o diagnóstico médico é subjectivo, dependendo não só dos dados disponíveis mas também da experiência do médico, da sua intuição e até da sua condição físico-psicológica no momento. Vários estudos mostraram que o diagnóstico de um paciente varia significativamente conforme o paciente seja

examinado por médicos diferentes ou até pelo mesmo médico em momentos diferentes (dia da semana ou hora do dia).

Os métodos de aprendizagem podem ser usados para derivar automaticamente regras de diagnóstico a partir de descrições de pacientes tratados no passado, para os quais o diagnóstico final foi verificado. Conhecimento sobre diagnóstico derivado automaticamente pode assistir os médicos de modo a tornar o processo de diagnóstico mais objectivo e fiável [Alves et al. 2001] [Holmes 1997].

1.1.2 Desempenho dos Médicos versus Sistemas de Aprendizagem

Em geral, o diagnóstico através de sistemas baseados em regras geradas automaticamente suplanta, embora ligeiramente, o de médicos especialistas numa situação em que estes dispõem da mesma informação que a máquina.
Na Tabela 1.1 apresenta-se uma comparação do desempenho de dois algoritmos que implementam métodos de aprendizagem (i.e., classificadores Naive Bayesianos e Assistant [Cestnik et al. 1987]), com o desempenho médio de quatro médicos especialistas em cada um dos quatro problemas de diagnóstico referidos no que se segue [Kononenko et al. 2000]:

- Localização do tumor primário – o tratamento médico de pacientes com metástases tem maior sucesso se a localização do tumor primário no corpo do paciente for conhecida. A tarefa diagnóstica passa por determinar uma das 22 possíveis localizações do tumor primário, dispondo como dados da idade, o sexo, tipo histológico do carcinoma, grau de diferenciação e 13 possíveis localizações das metástases descobertas;

- Previsão de recorrência de cancro da mama – entre os pacientes em que foram removidos cancros da mama a doença aparece até cinco anos após a operação em cerca de 20% dos casos. Para melhorar o tratamento torna-se necessário prever a possibilidade de recorrência tendo como dados a idade, tamanho e localização do tumor, e dados acerca de nodos linfáticos. O problema é particularmente difícil para os médicos especialistas devido ao longo período de observação (cinco anos). Pouca experiência com base no acto médico, ou seja na prática, pode ser obtida nestes casos;

- Doenças da tiróide – o problema está na determinação de um de quatro possíveis diagnósticos tendo como dados a idade, o sexo, dados histológicos e resultados de exames laboratoriais. Contudo, nas situações reais do dia a dia os médicos socorrem-se de elementos adicionais para o diagnóstico, os quais não estão, em princípio, disponíveis para processamento pelo computador; e

- Reumatologia – o problema está na selecção de um de seis grupos de diagnósticos possíveis a partir de dados anamnésicos e de estado. Para os médicos especialistas isto não será com certeza um problema. Contudo, os médicos de clínica geral têm de decidir entre doença reumatológica ou ortopédica para encaminhamento do paciente para um especialista. Estas decisões são geralmente pouco fiáveis e na opinião dos médicos especialistas em reumatologia, erradas em mais de 30% dos casos.

Tabela 1.1 – Comparação do desempenho de diferentes algoritmos de aprendizagem em quatro domínios médicos diferentes.

Classificador	Tumor primário	Cancro da mama	Tiróide	Reumatologia
Naive Baysiano	49%	78%	70%	67%
Assistant	44%	77%	73%	61%
Médicos	42%	64%	64%	56%

Ambos os algoritmos suplantam significativamente o desempenho dos médicos em termos de correcção da classificação. Contudo, estes resultados necessitam de ser avaliados. Deve-se salientar que nestas experiências, tanto os médicos como o computador dispuseram de exactamente da mesma informação. No caso do cancro da mama e da reumatologia, o diagnóstico de um paciente a partir de informação em papel é algo muito pouco natural; nas restantes situações isso até acontece na prática com alguma frequência. Este cenário é, de qualquer modo, pouco realista na prática médica. Durante o exame do paciente o médico geralmente observa o estado do paciente em termos de impressões intuitivas, as quais não podem ser, regra geral, formalmente descritas e, portanto, não podem ser alimentadas a um computador. A falta de tal informação pode em alguns casos ser de importância crucial para obtenção de diagnósticos mais fiáveis.

Os resultados apresentados na Tabela 1.1 devem portanto ser entendidos como uma estimativa para avaliação do desempenho dos algoritmos de aprendizagem e não necessariamente para avaliar um eventual mau desempenho dos clínicos; embora os algoritmos de aprendizagem potenciem programas de diagnóstico mais fiáveis a partir de descrições, porventura não isentas de ruído, dos pacientes. Estas ferramentas de diagnóstico não podem definitivamente, nem é sua intenção, substituir os médicos no seu papel, mas devem ser considerados mais uma ferramenta que poderá melhorar o seu desempenho (e.g., os resultados apresentados no Capítulo 5 demonstram que é possível ajudar no desempenho dos

Neuroradiologistas com ajuda de um algoritmo de aprendizagem baseado em *RNAs*).

1.1.3 Requerimentos específicos para sistemas de aprendizagem em diagnóstico médico

Para que os sistemas em que estão embebidos subsistemas de aprendizagem sejam úteis na resolução de tarefas relacionadas com o diagnóstico médico, devem apresentar, entre outras, valências do género:

- Bom desempenho – o algoritmo deve ter a capacidade de extrair informação significativa dos dados disponíveis. A correcção do diagnóstico tem de ser a maior possível (e.g., a maior parte dos algoritmos tem um desempenho pelo menos tão bom quanto o dos médicos, quando em presença de exactamente o mesmo modelo do paciente). Portanto, existindo a possibilidade de medir a correcção do diagnóstico médico, o seu desempenho pode ser utilizado para nivelar por baixo o desempenho que o(s) subsistema(s) de aprendizagem deverá(ão) apresentar numa dada situação;

- Lidar com informação incompleta – no processo que conduz ao diagnóstico médico é vulgar não se dispor de uma descrição completa do estado do paciente. Os algoritmos de aprendizagem têm de ter a capacidade de lidar apropriadamente com estas descrições, com um certo ruído, tornando um senão num auxiliar ao diagnóstico (e.g., criando diferentes cenários para o estado clínico do paciente);

- Transparência no conhecimento sobre o diagnóstico – o conhecimento gerado e a explicação de decisões devem ser transparentes para o médico,

que deve ser capaz de analisar e compreender o conhecimento gerado. Idealmente, o conhecimento gerado automaticamente deverá levar o clínico a equacionar a sua tomada de posição face a um dado problema (e.g., considerar novas inter-relações e (ir)regularidades nos dados não contemplados no diagnóstico inicial);

- Capacidade de explicação – o sistema deve ser capaz de explicar decisões aquando do diagnóstico de novos pacientes; e

- Redução do número de testes – na prática médica a recolha dos dados dos pacientes é geralmente dispendiosa, demorada e penosa para os pacientes. Daí que seja desejável dispor de um classificador com capacidade de diagnosticar, dispondo de uma quantidade mínima de dados acerca do paciente.

1.1.4 Aceitação na prática

Embora os resultados de aplicação dos diversos algoritmos de aprendizagem no diagnóstico médico aparentem ser excelentes, esta tecnologia não tem tido muita aceitação na área médica. As justificações apresentadas pelos próprios médicos são diversas, e tomam a forma:

- Inflexibilidade na representação de conhecimento. O conjunto de atributos que descrevem o modelo dos pacientes é em geral fixo. A informação usada na derivação do diagnóstico final esta limitado a parâmetros rigidamente definidos, enquanto que reacções subjectivas, informais e difusas (e.g., intuição, impressão) não são em geral consideradas;

- Os médicos argumentam frequentemente que se não têm a certeza acerca de um diagnóstico final, podem sempre solicitar mais exames (e.g., testes laboratoriais) para certificação do diagnóstico. Nas situações em que é fácil efectuar mais exames ao paciente, os médicos não sentem a necessidade de assistência no processo de diagnóstico. No prognóstico não existe a possibilidade de efectuar mais exames para a confirmação da previsão. Por esse motivo é que os problemas de prognóstico são ainda mais atractivos para aplicação da tecnologia da aprendizagem do que os de diagnóstico [Zwitter et al. 1983];

- Os médicos argumentam frequentemente que se encontram demasiado ocupados, não dispondo de tempo para utilizarem ferramentas de apoio à decisão. Na prática do dia a dia a digitalização dos dados no computador é demasiado desgastante em tempo e energia para utilizar o computador no processo de diagnóstico;

- Não responsabilizável, é outra das resistências à introdução de novas tecnologias de diagnóstico. É considerado que o problema de diagnóstico é porventura a tarefa mais sensível e crítica na sua actividade. Quando deixada ao cargo de máquinas deixarão os médicos sem poder de controlo e sem haver a quem imputar responsabilidades; e

- São encontradas também algumas razões algo irracionais na resistência ao diagnóstico computorizado. Argumentam que o diagnóstico é considerado a tarefa intelectual por excelência da sua profissão. Como tal, esta tarefa requer um conhecimento profundo, ideias inesperadas e uma particular intuição. Portanto o diagnóstico é um pouco como uma arte, impossível de explicar e formalizar. Como poderá então ser executada por computadores?

> E se os computadores o poderem executar isso será a destruição de toda a magia e orgulho de uma classe profissional.

Um dos objectivos do trabalho efectuado foi precisamente tornar a arquitectura e os sistemas desenvolvidos fáceis de utilizar por não especialistas em informática. Isto incluiu tornar as ferramentas o mais intuitivas possível e equipadas com interfaces visualmente e funcionalmente atractivas. Daí a opção pelas tecnologias da Internet em termos de interfaces médicas. Note-se também que nos sistemas desenvolvidos a necessidade de introdução de dados foi reduzida ao mínimo, daí a ligação directa aos equipamentos de imagem médica.

1.1.5 Evolução da Inteligência Artificial

Tendo como referência as aplicações de *IA* até ao momento desenvolvidas, colocam-se-nos algumas questões, deveras pertinentes, tais como:

- Qual é o aspecto com que se apresentam estas aplicações?
- Com que objectivo é que foram desenvolvidas?;
- Qual foi a sua evolução em termos de geração de mais valias, nomeadamente no que respeita à "Nova Economia"?; e
- Quais foram os atributos que se tornaram fundamentais para se definir o seu percurso?

Recuando para os primórdios da década de 80, na primeira conferência da *American Association for Artificial Intelligence* (*AAAI*), encontra-se uma publicação denominada *R1:An Expert in the Computer System Domain* em que se apresenta um sistema pericial baseado na trilogia *conhecimento*, *raciocínio* e *experiência*. Este sistema apresentava algumas características notáveis: era complexo, de grandes dimensões, tecnicamente sofisticado, abrangente no domínio de aplicação e

apresentava um bom desempenho. Outro dos aspectos únicos do sistema residia no facto de não ter sido patrocinado por nenhuma instituição pública mas sim por uma empresa privada, a DEC (*Digital Equipment Company*), na altura o segundo maior fabricante de computadores a nível mundial. Este patrocínio não se deveu a altruísmo nem teve a ver com potenciar o capital científico existente, mas pura e simplesmente porque a empresa tinha um problema real que pretendia resolver. Este sistema veio efectivamente a responder às expectativas em si depositadas, tornando-se crítico para a imagem da empresa. Este sistema não mais fazia do que tomar uma encomenda de um sistema computacional e apresentar uma solução para o cliente; i.e., configurava o sistema. Estava-se na altura dos grandes sistemas (minicomputadores), cuja configuração face a solicitações deveras diversas por parte dos utilizadores tinha-se tornado para as empresas num grande senão.

Esta tendência foi-se acentuando, em particular ao nível das médias e grandes organizações empresariais, sendo de mencionar, entre outras, entidades como *Manufacturers Hanover*, *IBM*, *Alcoa*, *GM*, *Artur D. Little*, *Pac Bell*, *Dupont*, *Ford*, *DEC*, *NASA*, *Boeig*, *Bellcore*, *US Navy*, *MCI*, *Met Life* e *American Express* [Luger & Stubblefield 1998].

É de salientar que muitas empresas e organizações não apresentam publicamente os sistemas que desenvolvem, argumentando que, se o sistema que desenvolveram teve sucesso, trata-se de uma mais-valia, pelo que a sua divulgação prejudicaria a empresa e, se o sistema foi um fracasso, então a divulgação deste facto seria embaraçosa para a empresa, prejudicando a sua imagem. Esta posição do tecido empresarial faz com que os sistemas apresentados sejam sempre em muito menor numero do que os que foram efectivamente desenvolvidos.

No historial das conferências em *IA*, e em particular da *Innovative Applications of Artificial Intelligence* (*IAAI*), verifica-se uma amálgama praticamente constante de

um certo tipo de aplicações (e.g., manufactura e *design*, negócios, médicas, finança, diagnóstico, telecomunicações, direito, auditoria, computadores e engenharia de *software*, militares, pesquisa de informação e classificação, espaço, atendimento de clientes). Ora, embora estas aplicações cubram amplos domínios de utilização do conhecimento, cingem-se, contudo, à tomada de decisão, no processo de funcionamento normal das organizações, tendo como objectivo genérico o aumento de produtividade. Actualmente sopram uns certos ventos de mudança, são apresentadas soluções para problemas baseadas em agentes que funcionam em tempo real, como sejam as aplicações de planeamento e de assistência pessoal, o que prenuncia o aparecimento de uma nova forma de estar no sector.

Na primeira conferência de *IAAI*, realizada em 1989, dizia-se que a concretização de certos objectivos científicos, preestabelecidos, iria permitir atacar novos problemas. Embora ainda não se tivesse atingido a meta de cognição automática (o que continua a ser verdade mais do que uma década depois), a tecnologia estava no entanto a provocar um enorme impacto no funcionamento diário das grandes organizações. Esta conferência foi realizada tendo como mote permitir a troca de informação acerca de aplicações que realmente funcionavam e quais eram os problemas concretos. O objectivo era o de traçar caminhos no sentido de uma melhor tecnologia, encontrar e remediar as deficiências actuais dessa tecnologia e, solucionar problemas reais.

Numa segunda conferência, realizada em 1990, tinha-se como mote a demonstração da utilidade destas aplicações. Torna-se interessante comparar estas duas aproximações para a solução do mesmo problema: na primeira afirmava-se que as concretizações científicas motivaram e fixaram a plataforma para as aplicações. Na segunda já se tentava marcar uma posição política: pretendia-se mostrar a utilidade das aplicações e mostrar que eram plenamente integradas nos ambientes computacionais e corporativos existentes.

Segue-se uma terceira conferência, em que se afirma que se está a trabalhar para a causa do utilizador empresarial, que se está a tentar demonstrar o que se pode conseguir com a tecnologia corrente na solução de problemas reais. Dizia-se ainda que as aplicações ditavam o caminho para a investigação e que, dado o sucesso das aplicações desenvolvidas, se pretendia sugerir alguns nichos do conhecimento para a investigação fundamental.

O objecto do discurso das conferências *IAAI* seguintes, não trouxe grandes novidades. Procurou-se demonstrar a maturidade da *IA* como uma tecnologia comercialmente viável. Em particular, no papel de *fermento* das tecnologias da *IA* no desenvolvimento de aplicações em geral. Em qualquer aplicação bem sucedida que tenha empregue várias tecnologias para o tratamento da informação, o componente de *IA* poderá ser dos mais insignificantes em termos de custos de desenvolvimento, mas é aquele que fará a diferença no produto final.

A razão porque foi feita esta resenha histórica está em que o que foi dito perspectiva uma evolução dos pressupostos e das questões que têm vindo a influenciar a evolução do que é entendido por *IA*. Nos primeiros anos, os objectivos a atingir passavam por demonstrar que *IA* era uma tecnologia viável, em contraponto ao sentimento de que tempos difíceis estavam para vir nesta área. Nos anos seguintes defendeu-se que embora fosse realmente possível por a componente *IA* a funcionar, com um efectivo valor acrescentado (i.e., mais valia) para as aplicações finais, o problema mais difícil de resolver estava em pô-la a funcionar em contexto. Os grandes problemas a resolver reduziam-se a problemas de integração (e.g., tratava-se de equacionar o resultado das mudanças na base computacional, das mudanças nas linguagens de programação dominantes). Estas questões, quando vistas segundo uma perspectiva de resolução de problemas a longo prazo, revelaram-se totalmente inconsequentes. A linguagem de programação dominante afirma-se com um ciclo de

vida de cinco anos, logo uma solução para o problema passa por levar a um certo acomodamento a esse facto. Uma vez que estas questões aparentavam terem sido tratadas adequadamente, a atenção foi mudada para a demonstração de que a aplicação das tecnologias de *IA* se tinha tornado na prática uma rotina e, como tal, devia ser tratada como muitas das outras áreas das tecnologias da informação.

Parece, por conseguinte, chegada a altura de declarar que a *IA* atingiu a maturidade; i.e., as aplicações envolvendo a *IA* estão bem estabelecidas e constituem uma tecnologia com utilização na prática, encontrando-se exemplos disso nos mais variados sistemas em funcionamento nos dias de hoje. Um dos motivos apontados para justificar o pessimismo inicial, passava pelo mau desempenho das empresas que vendiam ferramentas de suporte para estas tecnologias. Ora, o que veio a acontecer com as tecnologias e ferramentas de *IA*, foi que estas migraram para dentro das grandes organizações, onde grupos de especialistas as utilizam para desenvolver sistemas periciais com interesse para a empresa. Estes grupos não são constituídos por especialistas das Ciências da Computação, mas sim por especialistas nas áreas de aplicação.

O paradigma computacional que serve de motor a estas aplicações é dado sob o ponto de vista pictórico na Figura 1.1; no centro posiciona-se o cérebro electrónico, um programa (porventura de grandes dimensões) usando *IA* que trata da tomada de decisões. Um problema com origem no normal funcionamento da organização é alimentado ao programa na forma de uma questão de que se pretende uma resposta. Estes programas dão corpo a sistemas de grandes dimensões sendo baseados em conhecimento; isto em comparação com os sistemas actuais em que, embora a dimensão da base de conhecimento seja modesta, apresenta um grau de complexidade muito superior ao que se fazia nos primórdios da *IA*.

Figura 1.1 – O Paradigma Tradicional

Estes sistemas operam como demonstradores de teoremas; i.e., uma questão é formulada e enviada ao sistema tendo como base um teorema a demonstrar. A questão poderá ser formulada através de um interface gráfico, ou vir directamente de outros sistemas da organização. Uma vez produzida uma solução, a mesma é enviada para a fonte que colocou a questão, sob uma forma de uma estrutura de dados, instância de um arquétipo(s) que corporiza(m) as possíveis soluções para os problemas da organização.

Sob o ponto de vista computacional, tem-se um modelo de funcionamento em tudo semelhante ao apresentado na Figura 1.2. A organização apresenta em termos operacionais, um fluxo de problemas cuja solução envolve, geralmente, o dispêndio de avultadas quantias em dinheiro, e em que os ganhos em eficiência trazidos pela *IA* embora sejam pequenos, traduzem-se num retorno monetário não negligenciável quando em comparação com os investimentos necessários. Constata-se que este investimento se paga rapidamente, embora não seja esta sempre e só a razão para recurso a esta tecnologia. Não raras vezes reside na possibilidade de se conseguir efectuar uma tarefa a razão para o uso deste tipo de soluções.

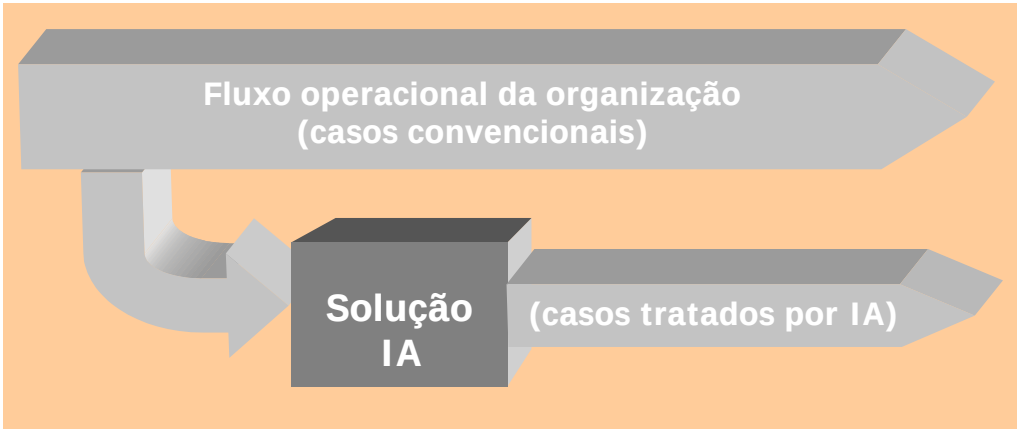


Figura 1.2 – O Modelo de funcionamento para aplicações tradicionais de *IA*

Este modelo assenta em três pilares ou atributos fundamentais (Figura 1.3): ***abrangência***, porque não existe retorno (e.g., financeiro) se não se conseguir cobrir um número suficiente de casos; ***correcção***, porque se torna demasiado dispendioso forçar a solução de uma grande quantidade dos casos; e ***explicável***, porque se torna necessário explicar o que correu mal quando uma questão é incorrectamente processada. Em muitos casos é também útil informar sobre o processo de raciocínio utilizado, mesmo quando se concluiu a resolução do problema com sucesso. Não raras vezes tem-se de justificar uma decisão tomada e guardá-la como referência para o futuro; ora isto é particularmente verdade para sistemas que se apresentam já contextualizados.

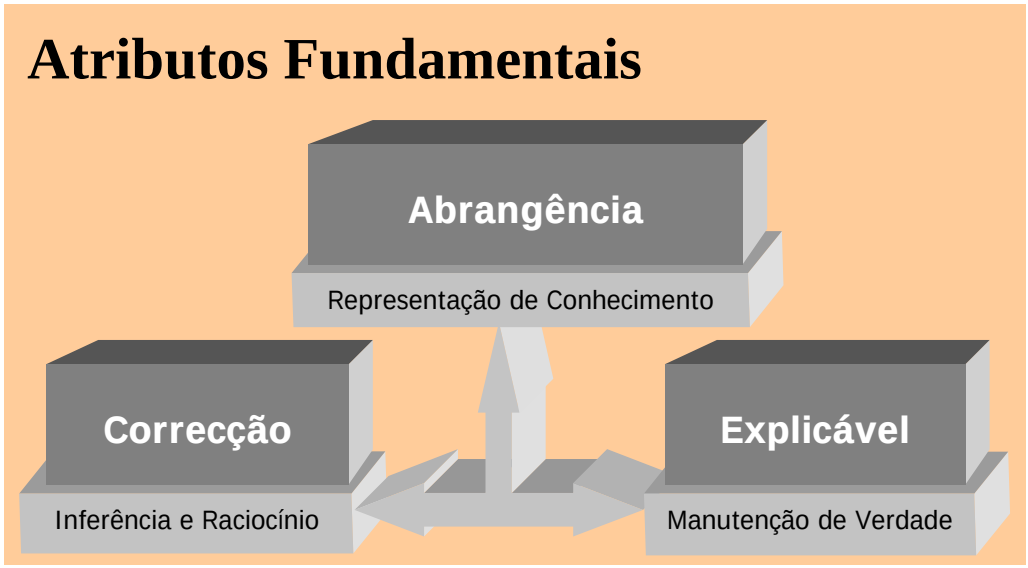


Figura 1.3 – Atributos fundamentais em aplicações tradicionais de Inteligência Artificial

Resumindo, tem-se que qualquer aplicação com componente de *IA*, e em termos de funcionamento, deverá ter como axiomas:

- Reclamar a solução de um número suficiente de casos para tornar a aplicação valiosa;
- Forçar uma solução para a menor quantidade possível de casos reclamados (i.e., sem solução numa primeira iteração); e
- Explicar as soluções dos casos forçados para facilitar o trabalho de re-arranjo da lista de casos a submeter ao sistema com vista a obter uma solução.

Estes atributos marcaram a agenda tradicional da investigação em Inteligência Artificial; o interesse foi focalizado em formas de representação de conhecimento para a resolução de problemas de *abrangência*, em inferência e raciocínio para o

tratamento de problemas de *correcção*; e em gestão de dependências e manutenção de verdade para tornar o sistema *explicável*.

É neste ponto em que se encontra o conhecimento actual e, o futuro mostra-se promissor. Mesmo que os tempos difíceis tenham existido, o que é refutado por muitos, pode-se dizer que estão a caminho do fim.

O que é que é que o futuro reserva? Comece-se por analisar a evolução dos processadores (componentes fundamentais da base computacional) que a indústria do sector tem fornecido. O quadro da Tabela 1.2 dá uma indicação do número de microprocessadores vendidos em sistemas de secretaria e em servidores; i.e., nas situações mais usuais de utilização de microprocessadores. Os dados referem-se a 1997. O nível de venda destes microprocessadores era na altura de cerca de 100 milhões de unidades por ano. No mesmo quadro também se apresenta o número de microcontroladores e outros processadores que são embebidos num sem número de aplicações. Houve uma venda de 1.3 biliões de microcontroladores de 4 bits comparados com os 103 milhões de microprocessadores e 2.4 biliões de microcontroladores de 8 bits.

A diferença entre as vendas de microprocessadores de 16 bits (cuja principal aplicação na altura seriam os computadores pessoais) e os microcontroladores de 4 e 8 bits é enorme. Mesmo nos 16 bits tem-se 331 milhões vendidos como microcontroladores contra os 103 milhões vendidos para aplicação em computadores pessoais. Até na categoria dos 32 bits venderam-se 13 milhões de microcontroladores em 1997.

Tabela 1.2 – Evolução do mercado dos processadores (*UMP* e *UMC* denotam, respectivamente, o número de unidades de microprocessadores e o número de unidades de microcontroladores).

Fornecimento de processadores pela industria			
	1997	**2000**	**Crescimento**
Microprocessadores	103,405,000	150,000,000	1.45
Microcontroladores - 4 bits	1,310,000,000	1,680,000,000	1.28
Microcontroladores - 8 bits	2,450,000,000	4,770,000,000	1.95
Microcontroladores - 16 bits	331,000,000	764,000,000	2.31
Microcontroladores - 32 bits	13,000,000	43,000,000	3.31
Total Microcontroladores	4,104,000,000	7,257,000,000	1.77
Razão UMP/UMC	39.69	48.38	1.22
Microprocessadores Embebidos - 8 bits	39,171,000	20,200,000	0.52
Microprocessadores Embebidos - 16 bits	56,589,000	108,000,000	1.91
Microprocessadores Embebidos - 32 bits	103,405,000	153,100,000	1.48
Total Microprocessadores Embebidos	199,165,000	281,300,000	1.41

No seu conjunto, o mercado dos microcontroladores é descomunal, sendo 40 vezes superior ao dos microprocessadores em 1997 e 49 em 2000. O gráfico da Figura 1.4 torna tudo mais claro. As barras cilíndricas denotam os microcontroladores enquanto que as barras rectangulares denotam os microprocessadores utilizados no fabrico de computadores. Efectivamente, hoje em dia, um veículo de gama baixa chega a ter integrados, para seu normal funcionamento, 16 microcontroladores e, se for de gama alta, pode atingir 36 ou mais. Se se questionar alguém sobre quantos processadores possui, a resposta não se deve basear nos computadores de que é proprietário, pois estes constituirão uma amostra praticamente irrelevante relativamente aos que estão nos seus electrodomésticos, carro e outras aparelhagens.

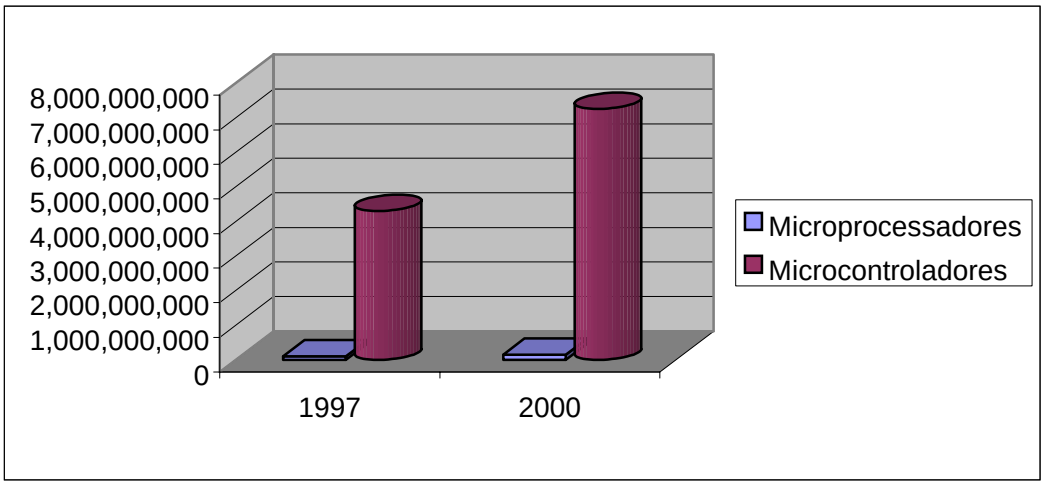


Figura 1.4 – Gráfico da evolução no fornecimento de processadores.

Constata-se que as vendas de processadores para serem utilizados em computadores subiram entre 1997 e 2000 de 103 milhões para os 150 milhões, o que representa um crescimento de 45%. Em contrapartida, o fornecimento dos microcontroladores foi de cerca de 4,104 milhões em 1997 e 7,257 milhões em 2000, o que se traduz num crescimento de 77%, valor bastante superior ao dos microprocessadores. Ou seja, regista-se um maior crescimento, no sector dos microcontroladores, no mesmo período, sendo o crescimento tanto maior quanto maior for o desempenho do processador: 28% para 4 bits, 95% para 8 bits, 131% para 16 bits e 231% para 32 bits.

A data em que ocorrerá uma viragem; i.e., o momento em que o fornecimento de microcontroladores de gama alta, vai ultrapassar o dos microprocessadores para computadores está certamente muito próxima (veja-se o exemplo da *PlayStation* 2, a consola de jogos da *SONY*).

O que isto significa é que a maior parte do poder computacional estará embebido em sistemas físicos que interactuam com o ambiente que os rodeia. Um novo paradigma emerge: torna-te físico, dá olhos, ouvidos e mãos ao cérebro electrónico de modo a que este se possa aperceber do que se passa no ambiente que o rodeia, de modo a que possa inter-actuar e alterar esse mesmo ambiente (Figura 1.5).

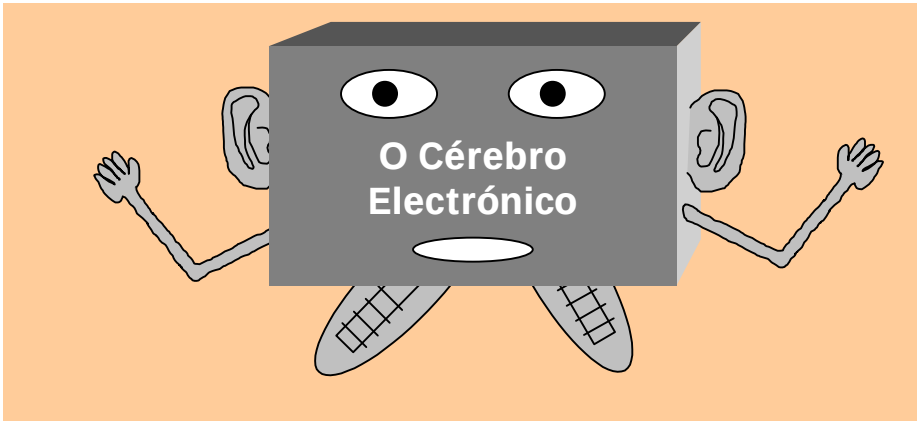


Figura 1.5 – O Paradigma Emergente: Sistemas Inteligentes Embebidos

Já existem vários sistemas em funcionamento que apresentam estas características; e.g., o *Image-Guided Neurosurgery System* [Grimson et al. 1996] que é um sistema presentemente em utilização num hospital em Brigham. Trata-se de um sistema de apoio a intervenções cirúrgicas cerebrais, que utiliza câmaras de vídeo e imagens geradas por um aparelho de Tomografia Computorizada (*TC*) com sensores. Com as imagens do *TC* é construído um modelo tridimensional do cérebro do paciente a ser operado. Durante a cirurgia, a imagem da câmara de vídeo é usada para acertar com precisão a orientação e posição da cabeça do paciente, sendo a imagem tridimensional do cérebro sobreposta à imagem vídeo de modo a ajudar o cirurgião a guiar os instrumentos, conforme vão entrando no cérebro para retirar o tumor. Este sistema permite que o cirurgião faça intervenções que anteriormente não eram feitas devido ao exagerado risco que acarretariam. Embora se trate de um sistema que

funciona a partir da percepção do ambiente e, esteja em pleno e regular funcionamento, ainda não se pode caracteriza-lo como universal.

Outra aplicação representativa deste novo paradigma dá pelo nome de projecto *Quarto Inteligente* do *Massachusetts Institute of Technology* (*MIT*). Trata-se de um quarto que apresenta certas propriedades especiais, tais como o reconhecimento da voz, visão e reconhecimento de gestos, e um sistema bem diferente dos convencionais para a gestão e consulta de informação; i.e., um sistema com o qual se interactua falando, gesticulando, desenhando, vindo a resposta sob a forma de imagens e voz. O teclado só é utilizado como último recurso, para as situações em que não há alternativas para interactuar com o sistema. Neste sistema o quarto tem autonomia para levar a bom termo tarefas como acender e apagar a luz, abrir e fechar as persianas, seleccionar e pôr música, etc. Tudo sobre o controlo do computador, que não mais faz do que dar corpo às mãos do quarto (inteligente). Estes são dois exemplos do que é possível num sistema que disponha de dispositivos de percepção e actuadores.

A implementação generalizada destes sistemas, no futuro, não vai de modo algum ser uma tarefa fácil. Existem problemas de acompanhamento de diálogos que não são triviais. Existem, de igual modo questões relacionadas com a compreensão da linguagem, para já não falar daquelas que estão relacionadas com problemas que têm a ver com a representação do conhecimento, formas de raciocínio, complexidade e estruturas computacionais, e de Direito.

Um outro aspecto do mesmo problema que está em rápida mudança e que tem a ver com a base computacional a utilizar é o das comunicações. De dia para dia aumenta a largura de banda disponível e as ofertas em termos de tecnologias, em contraponto com os preços que vão baixando. A primeira grande surpresa foi a de constatar-se que se está a caminhar para uma época de abundância em computação embebida.

Realmente poucos se aperceberão, de que os processadores que geralmente se interpretam como computadores são uma amostra estatisticamente insignificante na população dos processadores instalados e, que os processadores embebidos estão a ficar surpreendentemente poderosos. Mas esta constatação não é nada em comparação com o facto do custo das comunicações tender para zero. De facto, a largura de banda esta a crescer a um ritmo tão acelerado que a torna provavelmente no factor tecnológico de liderança. Existem vários factores que contribuem para esta evolução, alguns são tecnológicos, como a multiplexagem de divisão de onda em fibras ópticas e, outros são relativamente mundanos. Por exemplo, para a colocação de cabos de fibra óptica abrem-se valas no solo; ora, o custo de abertura de valas é muito maior que o custo dos cabos que lá se pretendem colocar. Então, já que se abriu a vala, o que se faz é enterrar o máximo de cabos possível. Veja-se o que fez a *BRISA*; juntamente com a construção das auto-estradas instalou uma infra-estrutura de comunicações para mais tarde a rentabilizar. O mesmo fez empresas como a *CP – Caminhos de Ferro* e a *EDP – Electricidade de Portugal* que aproveitaram obras em curso para as instalar. Ou seja, sempre que alguém se propõe aumentar a largura de banda, esta aumenta muitíssimo mais do que o necessário. O mesmo arquétipo aplica-se também aos satélites, já que se vai ter o incómodo de colocar um satélite em órbita, tem-se o cuidado de garantir que este possa tratar o maior número de canais possíveis, tendo em conta que fica algo caro mandar equipamentos para o espaço.

O importante não é saber se a procura apanhará ou não a oferta, mas ter em linha de conta que existe uma imensa oferta de soluções à espera de novos clientes. Entrou-se numa era de abundância no que respeita à disponibilidade de comunicações e computação embebida. Isto irá certamente permitir que projectos como os de Telemedicina e Realidade Virtual Reforçada possam proliferar. Os dispositivos e a largura de banda estarão lá para isso. O que realmente os tem retido na gaveta, mais

do que qualquer outro motivo que se possa invocar, reside no facto de que as aplicações que têm embebido a videoconferência não são amigas do utilizador.

O que aqui está subentendido é que o modo mais comum de computação passará muito brevemente por um conjunto altamente distribuído de processadores, embebidos em dispositivos de comunicação, com sensores e actuadores (Figura 1.5). O último degrau na rede de comunicações passa certamente pelos dispositivos de rede sem fios para permitir mobilidade; ora estes dispositivos já aí estão, com uma largura de banda razoável (11Mb/s) a custos muito acessíveis.

Se esta é na verdade a realidade computacional emergente, está-se perante um mundo muito diferente daquele a que estamos habituados. Sensores e actuadores serão colocados onde for possível, e não forçosamente, só nos locais teoricamente mais aconselháveis. Mas mesmo que fossem, não raras vezes falham. A falha é uma fatalidade deste mundo. Quanto mais dispositivos se tiverem instalado, maior a possibilidade de alguns avariarem. Isto é particularmente verdade quando as comunicações fazem parte de todo o processo. As comunicações avariam muito mais rapidamente do que os processadores. Existem imensos tipos de sensores, não raras vezes imperfeitos! Não só, imperfeitos no sentido de que avariam, mas também no sentido de que são sensíveis ao ruído. O factor ruído aparece no nosso caminho sempre que se implementam soluções que envolvam sensores. O desafio está em contornar o problema, a partir de percepções correctas do ambiente; i.e., usando um conjunto distribuído de componentes falíveis; em recuperar das falhas dos componentes; em compensar o ruído, quer a sua origem esteja nos componentes quer esteja no próprio ambiente e, em utilizar toda esta infra-estrutura que está a crescer com as maiores vantagens possíveis.

Anteriormente, foi feita referência aos atributos ***abrangência***, ***correcção*** e ***explicável*** e, como estes foram fundamentais na agenda tradicional da investigação

em Inteligência Artificial. Estes atributos e a agenda de investigação que inspiraram não irá desaparecer, mas há uma nova tríade de atributos a considerar no mundo emergente, em que há abundância em computação embebida. Dão pelo nome de ***autonomia***, uma vez que os sistemas estão espalhados pelo ambiente e têm de planear por si próprios; ***robustez***, no sentido de serem capazes de recuperar de falhas e, ***coerência***, no sentido de serem capazes de formar uma imagem coerente a partir de montes de informação parcial e com ruído originária dos sensores (Figura 1.6). Estes atributos levam a que se estabeleça uma agenda de investigação deveras aliciante. Embora se note uma certa sobreposição com a agenda anterior, existe uma nítida alteração de ênfase. O objectivo de ***autonomia*** orienta o focos no sentido do planeamento, escalonamento e alocação de recursos. A tomada de decisão ocorre tendo como base universos de discurso em que se torna necessário o raciocínio com informação incompleta e incertezas, dado que nenhum dispositivo está livre de ruído e por outro lado, nem toda a informação estará necessariamente disponível. O objectivo de ***robustez*** aponta para que haja um enorme focos na monitorização, quaisquer que sejam os planos previamente efectuados, diagnosticando o que funcionou indevidamente seguido de processos de recuperação e replanejamento. O objectivo da ***coerência*** leva a que se foque a atenção em processos de aprendizagem; levantamentos estatísticos; raciocínio por evidência; fusão de informação e gestão de dispositivos em que se decide aqueles que devem ser utilizados, local onde se instalam, etc.

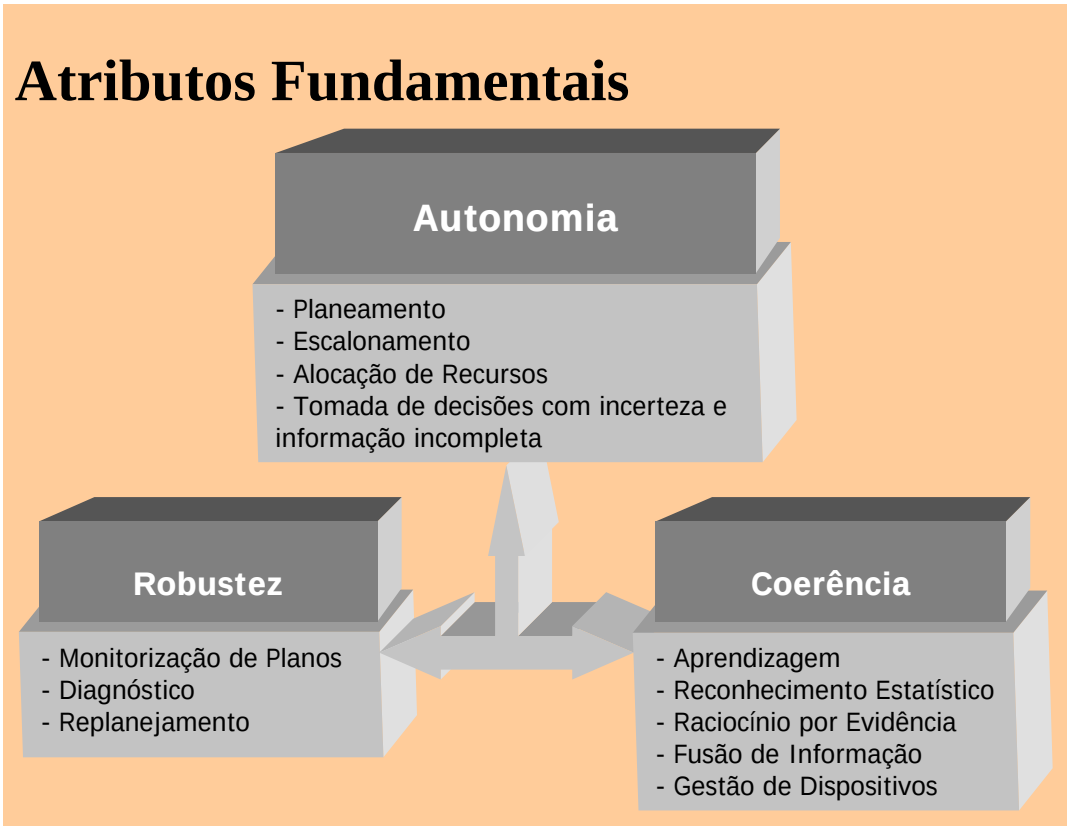


Figura 1.6 – Atributos fundamentais de um paradigma emergente para a Resolução de Problemas

No que se segue ver-se-á como estes assuntos poderão ser abordados e quais os resultados já disponíveis. O objectivo da ***autonomia*** denota que estes sistemas terão de ser capazes de planear e programar as suas próprias acções. Terão de fazer tudo isto em tempo real, o que por sua vez levanta a questão: até que ponto serão capazes de desenvolver um plano suficientemente bom, dado que terá de ser desenvolvido num intervalo de tempo que o torne efectivamente útil. Um plano perfeito só disponível após ter sido necessário não é lá muito útil. Em aplicações militares, este pressuposto é crucial (e.g., um plano perfeito para colocar um helicóptero num ponto de observação 10 minutos após o fim dos combates, não é um plano porventura muito útil). Por conseguinte, o planeamento tem de ser situado num

contexto temporal e satisfazer as restrições computacionais dadas, deixando a optimização de ser o ponto-chave. O objectivo é obter uma resposta suficientemente boa dentro das restrições impostas; sabe-se, contudo, que não há soluções perfeitas, mas há técnicas suficientemente bem elaboradas para construir alguns destes sistemas, sistemas estes que terão necessidade de alocar recursos de uma forma dinâmica, não necessariamente conhecida à partida. Os sistemas terão de funcionar em situações críticas, com diversas tarefas a serem executadas em paralelo, disputando, por conseguinte, a utilização de recursos comuns.

Existem casos particularmente conhecidos de sistemas autónomos embebidos, tais como o sistema *Remote Agent* no projecto *Deep Space 1* [Nayak & Rajan 1999]. Alguns são algo peculiares, porque quando nos referimos a tempo real fazemo-lo de acordo com os nossos padrões; porém, quando nos referimos a missões espaciais, o tempo real não é medido em segundos mas em horas, dias e até semanas. A experiência de autonomia do *Deep Space 1* só foi efectuada no ano passado e já foi declarada um sucesso. O agente *Remote Agent* conseguiu operar, conforme o que era esperado, a sonda durante 1 dia [Kanna et al. 2000].

Passe-se de seguida, à questão da ***robustez***. *Robustez* é normalmente interpretada de forma diferente por pessoas diferentes. O termo, utilizado neste contexto, tem a ver com a faculdade do sistema em recuperar de situações de falha do *software* e/ou *hardware*. De facto, o trabalho da *NASA* com o *Deep Space 1* teve muito a ver com a recuperação de falhas de *hardware*.

Contudo, estes sistemas que sentem o ambiente e dependem da captação de percepções, enfrentam um outro problema. O ponto fraco destes sistemas não está somente nos sensores, está também na teoria de percepção; i.e., a operação correcta a ser executada de modo a obter o tipo de resultado pretendido não é necessariamente conhecida, pelo que é necessário construir sistemas que estejam

preparados para recuperar das suas próprias falhas. Como é que se poderá conseguir isso? Pelo menos uma das abordagens para a solução deste problema poderá passar por integrar no sistema representações da sua própria topologia. O sistema deverá conhecer como é que os seus componentes interactuam no sentido de conseguir atingir o seu objectivo final. O facto do sistema conhecer a sua topologia permite-lhe monitorizar a execução de programas, fazer auto-diagnósticos e entrar em fase de reparação, ou recuperar de falhas [Kanna et al. 2000].

Nos sistemas onde a ***robustez*** for um atributo fundamental, é desejável que o ambiente de programação permita incluir no sistema algo mais do que o código. Seria, por exemplo, desejável representar o modo em que o programa se encontra estruturado, por que camadas de serviços e em que domínios específicos [Rich et al. 1988] [Rich et al. 1990], quais os meta-procedimentos já definidos (i.e., os serviços ao maior nível de abstracção possível). A maior parte dos programas são efectivamente dados por combinações de termos lógicos, em notação clausal, em forma de padrões, sendo estes padrões (comuns) que acabam por ser reutilizados. Cada um destes meta-procedimentos ou padrão pode ser implementado de diversos modos (e.g., um meta-procedimento em processamento de imagem pode envolver uma operação de filtragem). Ora, existem imensas formas de concretizar um filtro em particular. Todos nos levam a bons resultados, mas fazem-no, contudo, de modos diferentes e alguns condicionam o seu desempenho ás características próprias do problema. A importância em se focar a nossa atenção em padrões ou meta-procedimentos está em que, associado a cada meta-procedimento pode existir um plano que explica porque este funciona. Pode-se levar esta informação para o ambiente em que se executam os programas através de um processo de monitorização estabelecido à volta dos componentes do sistema, de modo a garantir que os invariantes do plano são realmente executados; i.e., verificam em tempo real que os componentes do sistema funcionam de acordo com o que era pretendido. Se não for este o caso, então o processo de monitorização deverá criar as condições

para que os procedimentos de diagnóstico sejam invocados. Estes poderão usar diferentes técnicas para a resolução de problemas como forma de descobrir como e onde é que o programa falhou no seu objectivo de atingir uma solução. O insucesso poderá ser devido a uma deficiente caracterização do domínio de discurso; ou poderá ser devido à falha de um dos seus componentes; ou, em certos contextos, poderá até ser devido a sabotagem. Todos estes são motivos perfeitamente válidos para justificar a construção de sistemas deste tipo.

Tudo o que foi dito até ao momento marca o caminho a seguir no sentido de se ter uma visão do ambiente de desenvolvimento de programas como sendo um sistema rico em informação e conhecimento. O ambiente de desenvolvimento não captura só o código do programa em si, mas também a fundamentação lógica para o código, para o plano, e assim sucessivamente. Presta um grande conjunto de serviços; estende o sistema com código que actualmente tem de ser manuscrito; i.e., insere código que ajuda a detectar condições excepcionais e a determinar pontos a partir dos quais as operações de recuperação podem ser iniciadas, o que encaminha a agenda para o desenvolvimento de funcionalidades que actualmente ainda não existem em nenhum ambiente de análise e desenvolvimento de *software*. Afinal, trata-se daquele código que se sente que se deve criar mas que raramente se cria. O código para tratamento de erros costuma ser a última das preocupações. Isto tem a ver com o facto de que até se ter em funcionamento todas as funcionalidades pretendidas, não vai haver grande preocupação com as excepções. Existem alguns novos serviços ao nível da execução de programas que farão parte da infra-estrutura desta nova geração de *software*; estes incluem os serviços ao nível da execução que se encarregarão de efectuar os diagnósticos, reparações e recuperações. Trata-se de um modelo para um novo ambiente de análise e desenvolvimento de *software,* que é rico em conhecimento tanto ao nível da componente de análise e desenvolvimento, como ao nível da componente de execução de programas.

O tema final a ser considerado é o da ***coerência***. Como já foi referido, no ambiente têm-se imensos sensores com uma certa tendência para a falha; algumas vezes irão induzir ruído, e noutras até irão avariar. O ambiente real em que se vive é um lugar algo disforme. É a esse lugar que os sensores terão de se habituar. Um modo de abordar o tema da ***coerência*** passa por se considerar que o ambiente é uma boa representação de si mesmo. Geralmente pode-se recorrer a eventos do ambiente para a calibração mútua de diferentes sensores e actuadores que estão embebidos no ambiente. Em boa verdade, se observarem o mesmo evento e partilharem as suas próprias percepções desse evento, poderão aprender muito acerca um do outro e acerca do ambiente. Pode-se, por exemplo, imaginar uma situação em que uma câmara numa videoconferência se dirige ao sistema de som, informando-o que a pessoa que está a falar se encontra na posição de coordenadas cartesianas (x_1,y_1). Esta por sua vez responde informando que tinha como adquirido que o som estava a vir da posição de coordenadas(x_2,y_2), pelo que irá tomar providências no sentido de ajustar o seu sistema de coordenadas (Figura 1.7). O ajustamento seguido, desenvolvido de forma interactiva, usando a informação passada pelo ambiente, poderá levar a técnicas chave para construir uma forma de avaliar percepções que se apresentam como coerentes.

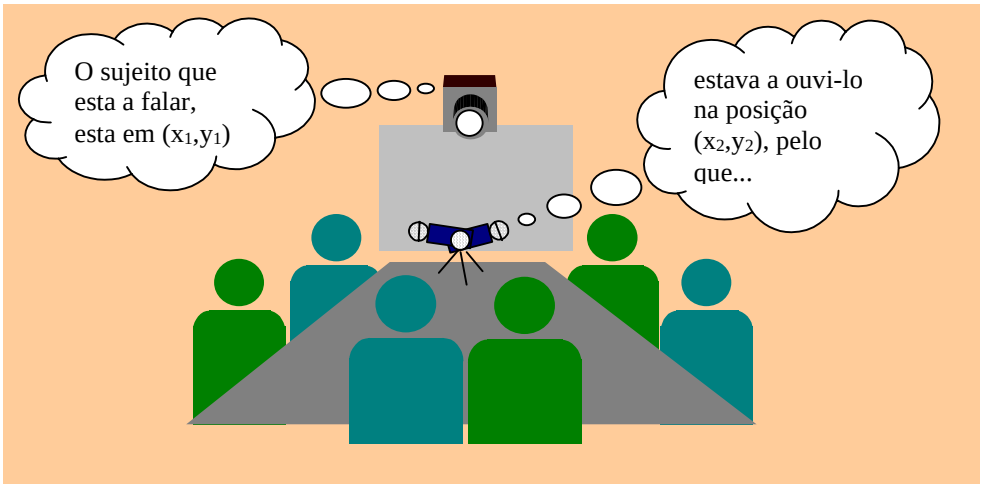


Figura 1.7 – A coerência através da calibração mutua.

Considere-se agora o exemplo (Figura 1.8). Neste caso podem-se ver quatro fotografias, que foram compostas por sobreposição, para dar uma visão deveras abrangente de um edifício (instalações da Universidade do Minho em Gualtar). Por muito cuidado que se tenha nessa operação é muito difícil, senão mesmo impossível, acertar os alinhamentos, cor, brilho, etc. Contudo, é possível digitalizar as imagens e calcular como é que os pixels das imagens se deveriam sobrepor, alterar e ajustar de modo a que se forme uma imagem perfeita. É pura e simplesmente uma questão de computação (Figura 1.9). Pode-se, portanto, passar de uma imagem incoerente para uma coerente, através de sensorização e de computação. Isto torna-se deveras interessante, pois com uma simples máquina digital e um sistema de tratamento de imagem, passa-se a dispor de imagens a que só se teria acesso, e nem sempre esse seria o caso, com equipamentos, dispendiosos, os quais nem sempre nos dariam a solução desejada (o sistema não é escalável).

Figura 1.8 – Imagem composta por sobreposição manual.

Figura 1.9 – Imagem composta por programa de tratamento de imagem

Encerra-se este capítulo com algumas observações acerca da interacção entre dois tipos de trabalho neste campo. Refiro-me ao desenvolvimento e aplicação prática da nova tecnologia e, à agenda científica da comunidade de *IA*. Estas actividades estão de certo modo relacionadas: ao objectivo científico tendo em vista a compreensão da *IA*, corresponde uma agenda prática de engenharia aplicada à construção de programas inteligentes. Para muitos, a definição do espaço em que se move a *IA* tem a ver com a compreensão da inteligência humana. Isto porque o tipo de *IA* a corporizar deve sê-lo feito à nossa imagem. Se é este o objectivo, então o seu lado

experimental poderá passar pela modelação do raciocínio humano, utilizando programas em computador.

No que respeita ao trabalho aqui descrito e que é motivado por objectivos de ciência fundamental, tende-se a colocar questões do género: "*como é que as pessoas apreendem tarefas de modo a torná-las computacionalmente tratáveis*", "*que conhecimento, método de resolução de problemas e técnicas de percepção fazem com que tudo funcione*". No domínio aplicacional põem-se questões similares: "*como é que apreendemos uma qualquer tarefa específica de modo a torná-la computacionalmente tratável para o tipo de máquinas computacionais de que dispomos*", "*que conhecimento e método de resolução de problemas são apropriados*". Ora existe uma área que é comum a este tipo de análise: a dos princípios em que assenta a própria definição de inteligência. Uns têm como objectivo entender a inteligência, os outros têm como objectivo a construção de sistemas úteis. Não estão de modo algum em conflito, são complementares.

Investigadores de *IA* que se preocupam com o problema da compreensão da inteligência humana, tentam obter representações do mundo que do ponto de vista da Psicologia e da Filosofia sejam realistas. Será que isso porventura desvendará o processo que a mente e em particular a mente humana poderá utilizar para realizar uma dada tarefa?

Trata-se de uma questão que é bastante polémica (e.g., se se tenta planear como agarrar um copo com a mão, o sistema de coordenadas a utilizar (o mais óbvio) é o sistema de coordenadas da mão (ou do cotovelo)). Afinal de contas é a mão que se vai mover e o que é necessário é descobrir a configuração e posição da dobra do cotovelo. Contudo, segundo trabalhos recentemente publicados na revista *Science* (que mostram que as coordenadas usadas por uma parte significativa do nosso sistema motor são coordenadas centradas na visão), o que acontece é que a mão é

movida segundo as directrizes do nosso sistema visual. Nenhum dos nossos movimentos é independente dos outros sentidos. Nos primatas a visão é a modalidade de percepção principal. Daí que, como as várias componentes do cérebro apreendem a trabalhar em conjunto, as coordenadas visuais são enviadas às outras partes do cérebro. Acontece que isto também é verdade relativamente ao nosso sistema auditivo; existem partes significativas deste sistema que utiliza coordenadas centradas no sistema visual.

Os neurocientistas podem, por conseguinte, ser uma das fontes de resultados interessantes do ponto de vista de investigação em *IA*; isto é inovador. Repare-se que, o que os neurocientistas reportaram não foi que a taxa de activação do neurónio *x* na região *y* do cérebro é 23 blips por milisegundo, como seria de esperar. Hoje os neurocientistas fornecem descrições funcionais das representações usadas nas diferentes partes do cérebro; isto é um tipo de informação qualitativamente diferente que configura uma mais valia enorme. Houve um progresso substancial na neurociência e muito mais está na forja. Pode mesmo vir a acontecer que uma hipótese colocada na prática de *IA* (e.g., na construção de um ambiente inteligente) venha a ser suportada por evidência pela neurociência. Isto seria deveras interessante e motivador de novas experiências. Admita-se que se construiu um sistema que usa processos de aprendizagem para suportar a calibração mútua das suas partes motoras e visuais. Será que tal sistema acaba por ficar com a componente motora a funcionar com coordenadas visuais? Em nosso entender, é para este tipo de sistemas que se caminha.

Sumariando: a evolução tecnológica está aí e vai oferecer muitas oportunidades para que se construam sistemas complexos embebidos no seu próprio ambiente. Está-se a caminhar para uma era de abundância no domínio das comunicações, pelo que se deve começar a equacionar a computação em novos termos. A construção destes sistemas também chamará a atenção para a necessidade de utilização de novos

atributos e novas agendas técnicas, que serão cruciais na compreensão desses mesmos atributos. Por outro lado, a construção destes sistemas (inteligentes e embebidos) corporizará uma das forças unificadoras desta disciplina, unificando aplicações e ciência fundamental, *IA* e áreas científicas afins.

1.2 Objectivos da Tese

Desenvolver um novo paradigma para a resolução de problemas na área Médica que passa por uma ligação estreita entre as Ciências Médicas e as Ciências da Computação, com aplicação à resolução de problemas em sistemas complexos e dinâmicos, como o são o das unidades prestadoras de cuidados de saúde.

1.3 Organização da Tese

A presente tese, para além deste capítulo introdutório, apresenta um conjunto de cinco capítulos, organizados da seguinte forma:

Capítulo 2
Sistemas Multiagente e Representação de Conhecimento.
Fornece-se uma descrição dos Sistemas Multiagente (*SM*) e retrata-se o estado da arte. Como forma de se compreender o significado de *SM*, apresenta-se a sua constituição e modo de funcionamento, indicando-se para cada unidade constituinte a sua caracterização. Esta abordagem não só permite clarificar as propriedades de um *SM*, mas também facilita a sua análise, desenvolvimento e implementação como um programa de computador. São também apresentados sistemas de representação de conhecimento, e esquemas de raciocínio, em Programação em Lógica Estendida

(*PLE*), assim como em Lógica Auto-Epistémica, Circunscrição e Lógica por Omissão.

Capítulo 3

Sistema de Aprendizagem Baseado em Redes Neuronais Artificiais.

Descreve o funcionamento e características dos sistemas para a Resolução de Problemas usando Redes Neuronais Artificias (*RNAs*).

Capítulo 4

***SADMED* – Ambiente Computacional para Sistemas de Apoio ao Diagnóstico Médico.**

Descreve formalmente o sistema *SADMED*. Apresenta a sua arquitectura, caracteriza não só cada constituinte como descreve o seu *modus operandi*.

Capítulo 5

Diagnóstico Médico Baseado em Tomografia Computorizada.

É exemplificada e avaliada a utilização do sistema *SADMED* no diagnóstico de imagens geradas por equipamentos de Tomografia Computorizada.

Capítulo 6

Conclusões e Trabalho Futuro.

Faz-se o fecho do trabalho desenvolvido, sumariando e lançando-se as pontes para trabalho futuro.

Capítulo 2
Sistemas Multiagente e Representação de Conhecimento

Fornece-se uma abordagem aos sistemas multiagente, e a conceitos relacionados com a representação de conhecimento.

Embora a Epistemologia, o estudo do conhecimento, tenha uma longa e respeitável tradição em Filosofia, começando com os gregos, as tentativas de representação rigorosa do conhecimento, assim como a análise formal de sistemas de raciocínio baseados em conhecimento, são de alguma forma mais recentes, com início nos anos cinquenta.

2.1 Introdução

Nos anos sessenta assistiu-se a um desusado interesse nesta área por parte da comunidade de Filosofia. Axiomas para a representação de conhecimento foram sugeridos, atacados e defendidos. Modelos para as diferentes axiomatizações foram propostos, em particular em termos de uma semântica de mundos possíveis, e então, de novo atacados e defendidos.

Mais recentemente, a representação de conhecimento e o raciocínio acerca de conhecimento encontraram aplicação em áreas tão diversas como a Economia, o Direito, a Linguística, a Inteligência Artificial e as Ciências de Computação. Embora a investigação nestas áreas tenha vindo a ser conduzida por indivíduos que mostram preferência em olhar para a Filosofia como a sua fonte de inspiração, constata-se que as suas preocupações mais pragmáticas, normalmente centradas em questões de natureza computacional, tal como a dificuldade no processamento de conhecimento, não foram tratadas nos textos de Filosofia.

Em Ciências da Computação, o paradigma do conhecimento tem sido aplicado com sucesso na análise de problemas em sistemas distribuídos, bases de dados e criptografia. Em Inteligência Artificial, o conhecimento apresenta-se como o conceito unificador para as teorias do raciocínio através do senso comum, da aprendizagem e da conversação. Em Economia, a compreensão profunda das regras de interacção entre os agentes económicos é conseguida através da utilização de conhecimento geral ou ordinário. Em Filosofia, tem havido progresso em aspectos de ciência fundamental como o da compreensão da semântica de atitudes relativas a proposições e proposições circulares [Shinghal 1992] [Searle 1990].

2.1.1 A lógica de predicados

A lógica de predicados é uma lógica clássica, monótona, que proporciona bases lógico-matemáticas para a representação do conhecimento e formas de raciocínio. De certo modo, qualquer forma de raciocínio não monótono vai além da lógica clássica, suportando mais conclusões, por conseguinte, é imperativo começar-se com um conhecimento básico.

2.1.2 Representação de conhecimento e raciocínio hipotético

Após a dilucidação do conceito genérico de Sistema de Representação de Conhecimento e Raciocínio, (*SRCR*) apresenta-se a forma como o conhecimento disjuntivo é tratado neste contexto. O caso fundamental desta construção reside na extensão de bases de factos para bases de factos disjuntivos, o que está em relação directa com as denominadas especificações epistémicas de Gelfond [Gelfond & Lifschitz 1990]. Este processo de construção da base de conhecimento baseia-se num processo operacional de prova de teoremas e não na usual teoria de modelos de Kripke [Kripke 1971].

O conceito de sistema de representação de conhecimento e raciocínio apoia-se, em princípio, em duas componentes centrais: uma operação de inferência e outra de alteração de bases de conhecimento concebidas como objectos abstractos. Por outro lado, ao definirem-se operacionalmente as tarefas de inferência e alteração, estes sistemas podem servir de base para uma definição operacional de lógicas. Quer as bases de dados relacionais quer as dedutivas podem ser consideradas como paradigmas computacionais para este tipo de sistemas. Estas implementam uma forma de raciocínio não monótono, devido ao uso da falha na prova para expressar por omissão informação negativa. Por outro lado, as bases de dados relacionais e as dedutivas, assim como os programas lógicos em geral, não são capazes de representar e processar informação negativa dada explicitamente. Esta limitação levou a que se considerassem extensões à programação em lógica, através da adição de uma segunda negação ou negação forte [Neves 1984] [Gelfond & Lifschitz 1990]. Neste tipo de sistemas um significado especial é dado ao *Pressuposto do Mundo Fechado*; i.e., este faz a ligação entre a negação por falha e a negação forte, em combinação com a representação parcial ou total de predicados. Se o *Pressuposto do Mundo Fechado* se verificar para um predicado, a negação por falha

determina a negação forte, ou seja, uma base atómica formada a partir de tal predicado é falsa se já o for devido a omissão.

Outras formas de raciocínio não monótono foram objecto de estudo, nomeadamente as referidas a seguir. Não foram contudo consideradas como alternativa para extensões à programação em lógica, formalismos lógico-matemáticos utilizados neste trabalho, por considerações que têm a ver com o seu desempenho computacional e tratamento de informação disjuntiva e negativa.

2.1.2.1 Lógica auto-epistémica

Desenvolvida por Moore nos princípios dos anos oitenta, a lógica auto-epistémica afirma-se como um formalismo para o raciocínio não monótono, passando a ideia base pela descrição rigorosa de um agente a raciocinar acerca do seu próprio conhecimento ou crenças. Considere-se o diálogo:

> *Será que o Rui Veloso irá dar um concerto no Coliseu do Porto na próxima semana?*

A partir deste exemplo torna-se claro que não se dispõe de conhecimento preciso para concluir que o Rui Veloso não irá dar um concerto no Coliseu do Porto na semana seguinte.

Neste sentido o nosso conhecimento é incompleto e, dando uma resposta pela negativa, estamos a presumir. Esta presunção baseia-se numa reflexão sobre o conhecimento de que dispomos (se algo importante está para acontecer na nossa cidade, então estaremos informados sobre isso).

Suponha-se que no dia a seguir à conversa mencionada em epígrafe, a primeira notícia do jornal *O Público* apresenta como título:

> *O concerto do milénio. Rui Veloso no Coliseu do Porto na próxima semana.*

A situação alterou-se. Neste momento sabe-se que o Rui Veloso irá dar um concerto, pelo que a resposta à questão colocada anteriormente é *sim*. Isto significa que a conclusão a que anteriormente se tinha chegado por introspecção, não é doravante válida e deve ser reformulada. Por conseguinte, este raciocínio é não monótono; i.e., novos dados invalidaram uma conclusão anterior. A única diferença consiste em que agora se sabe que o concerto se irá realizar, não se podendo concluir o contrário por introspecção.

A lógica auto-epistémica dá corpo a esta forma de raciocínio, passando pela introdução do operador modal L_φ, que é aplicado a *formula* de primeira ordem. L_φ tem o significado:

> Nós conhecemos φ (ou nós acreditamos em φ)

O exemplo anterior é agora representado na forma:

$concerto \rightarrow L_{concerto}$	(se um concerto se realiza então nós estamos informados acerca disso)
$\neg L_{concerto}$	(não estamos informados de que um concerto se irá realizar).

A semântica da lógica auto-epistémica é produzida em termos de *expansões*; i.e., fragmentos de conhecimento que definem vistas do mundo compatíveis com e

baseadas no conhecimento disponível. Ora uma das características principais das expansões é a sua estabilidade; i.e., um conjunto fechado de *formula* é considerado estável caso se verifiquem as seguintes condições:

$$se\ \varphi \in E\ então\ L_{\varphi} \in E$$
$$se\ \varphi \notin E\ então\ \neg L_{\varphi} \in E$$

Este conceito de estabilidade espelha de uma forma clara a problemática da introspecção, ou seja, o raciocínio auto-epistémico, assim como as suas limitações.

2.1.2.2 Circunscrição

Desenvolvida por McCarthy [McCarthy 1980] a *circunscrição* é mais um formalismo para o raciocínio não monótono, permitindo-nos extrair conclusões a partir de informação incompleta, em casos em que a lógica clássica é insuficiente. Na sua forma mais simples, a circunscrição apega-se à lógica de predicados, fazendo uso de uma ideação muito simples: dada uma teoria de primeira ordem *T*, completa-a um conjunto adicional de *formula*, que designaremos pelo conjunto *circunscrição (T)*. Por conseguinte, é utilizado raciocínio nos termos em que é entendido em lógica clássica. Considere-se então a teoria de primeira ordem *T* dada pelo conjunto de *formula* lógicas:

$$\forall X(ave(X) \wedge \neg\, excepção(X) \rightarrow voa(X))$$
$$ave(piu\text{-}piu)$$

que se pode ler na forma *todas as aves que não são dadas como excepções voam*, o que é nem mais nem menos do que uma forma de representar a regra por omissão: as aves *normalmente voam*. Pretendendo-se inferir que *voa(piu-piu)*, a partir de *T*, segue-se que *voa(piu-piu)* não é uma consequência lógica de *T* na lógica de

predicados, uma vez que é impossível provar que *piu-p*iu não seja uma excepção; *piu-piu* pode ser mesmo uma excepção, apesar de tudo. O princípio por detrás de *circunscrição* é o seguinte: restrinja-se o conjunto de objectos para os quais o predicado *excepção( )* é verdadeiro aos objectos para os quais a informação existente leva a que *excepção( )* seja verdadeiro. No caso presente não há nenhuma evidência que nos leve a concluir que haja aves que possam ser consideradas como excepções. Por conseguinte, a *formulae*:

$$\forall X \neg \, excepção(X)$$

é inserida na teoria *T*, como elemento do conjunto *circunscrição(T).* Agora é fácil verificar que a conclusão *voa(piu-piu)* é uma consequência lógica de $T \cup$ *circunscrição(T).*

O lado semântico do problema passou pela eliminação de todos os modelos de *T* nos quais a interpretação de *excepção( )* é um conjunto não vazio de *formula* lógicas. Os modelos que não são eliminados são mínimos na interpretação de *excepção( )*. De um modo geral, a problemática da circunscrição passa pela minimização das interpretações de predicados específicos, logo eliminando muitos dos modelos de *T* e dando corpo a um conjunto maior de conclusões. Não desenvolveremos este tema com maior detalhe, por duas razões principais:

- Sob o ponto de vista teórico, a circunscrição alcança a plenitude quando lógicas de segunda ordem são chamadas a terreiro; e
- Sob o ponto de vista prático, parece-nos difícil conceber-se a forma ou modo em que a circunscrição poderá ser implementada eficientemente e usada em aplicações (especialmente se a lógica de segunda ordem for usada).

A razão pela qual consideramos a *circunscrição* como uma técnica útil, reside no facto de dar a oportunidade de compreender por que motivo se encontra o conceito de modelos mínimos por detrás dos sistemas de raciocínio não monótono.

2.1.2.3 Lógica por omissão

Introduzida por Reiter [Reiter 1980] afirmou-se como uma das vias mais promissoras para o raciocínio não monótono. A lógica por omissão faz a distinção entre dois tipos de conhecimento, nomeadamente *formula* em lógica de predicados (chamadas axiomas ou factos) e regras de referência (denominadas ausências ou omissões). Por conseguinte, uma lógica de omissão é composta por um conjunto de factos que representam certa informação, embora normalmente incompleta, acerca do universo de discurso; i.e., um conjunto de regras de referência que sancionam conclusões plausíveis, mas não necessariamente verdadeiras.

Ora isto significa que há conclusões que não são revistas quando mais informação acerca do universo de discurso fica disponível. Um exemplo simples de uma ausência ou omissão é dado por:

$$\frac{ave(X) : voa(X)}{voa(X)}$$

que se por ler na forma *se X é uma ave e se é congruente assumir que X voa, então concluir que X voa*. Na ausência de conhecimento que indicie o contrário é razoável assumir-se que qualquer ave em particular pode voar, ou mais simplesmente que usualmente as aves voam. Dada a informação de que o *piu-piu* é uma ave pode-se concluir que o *piu-piu* voa, Porém, se mais tarde se vier a saber que o *piu-piu* não pode voar (e.g., porque o *piu-piu* é um pinguim), a regra de referência deixa de ser aplicável; i.e., não nos é permitido continuar a assumir que o *piu-piu* voa. Assim, o

nosso comportamento é não monótono. A semântica operacional da *lógica por omissão* é definida em termos de *extensões*, ou seja, por um conjunto de crenças que se subscrevem em relação ao mundo descrito pela teoria em mãos.

Uma teoria (lógica) de omissão *T* é um par (*F*, *O*), em que *F* denota um conjunto de *formula* em lógica de predicados (denominadas factos ou axiomas de *T*) e *O* um conjunto enumerável de regras de referência ou omissões. Uma regra de referência α tem a forma:

$$\frac{\varphi : \gamma_1,...,\gamma_n}{\beta}$$

em que φ, γ_1, ..., γ_n, β são *formula* em lógica de predicados, e $n > 0$. A *formulae* φ é designada de pré-requisito, γ_1, ..., γ_n, as justificações, e β a consequência lógica de α. O exposto caracteriza a sintaxe de uma lógica por omissão.

Sendo $\frac{\varphi : \gamma_1,...,\gamma_n}{\beta}$ uma regra de referência ou omissão, o seu significado é o seguinte:

> Se φ é conhecido, e se é congruente assumir γ_1, ..., γ_n, então concluir β.

Esta interpretação pode ser, por sua vez, objecto de uma caracterização rigorosa ou formal, desde que se explicite onde é que φ deve ser incluído e com quem é que γ_1, ..., γ_n, devem ser congruentes. Uma interpretação algo mais precisa é então dada na forma:

> Se φ é conhecido, e se todos os $\gamma_{i(i=1,...,n)}$, são congruentes com o estado corrente da base de conhecimento, então concluir β. O estado corrente

da base de conhecimento é constituído por todas as consequências lógicas derivadas por aplicações prévias das regras de referência ou omissão.

A definição formal: $\frac{\varphi : \gamma_1,\ldots,\gamma_n}{\beta}$ é aplicável a um conjunto de *formula F* se, e somente se, $\varphi \in F$ e $\neg\gamma_1 \notin F,\ldots, \neg\gamma_n \notin F$.

Um dos seus senãos está no facto, já referido em epígrafe, de que poder-se-á dar o caso de haver conclusões que não são revistas quando mais informação acerca do universo de discurso fica disponível.

2.1.3 Conhecimento sobre o Conhecimento ou Metaconhecimento

Um metaprograma é qualquer programa que trate quaisquer outros programas como dados. A linguagem em que estes programas são desenvolvidos designa-se por metalinguagem. A linguagem em que é escrito o programa, quais dados do metaprograma, é a linguagem objecto. Nos casos em que a linguagem objecto e a metalinguagem são idênticas, como metaprogramas tem-se ainda o que se designa por interpretadores metacirculares, ou seja, que foram escritos na linguagem que está a ser interpretada.

Inferência ao metanível tem vindo a ser largamente aplicada na Programação em Lógica e Inteligência Artificial. Dada uma linguagem *L*, uma afirmação ao metanível *S* com respeito a *L* é uma afirmação acerca dos objectos sintácticos de *L*. Por outro lado, é geralmente aceite que o conhecimento é constituído por crenças verdadeiras. Por conseguinte, são necessários mecanismos para raciocinar não só acerca de conhecimento mas também acerca de crenças. Por exemplo, pode-se

acreditar (embora incorrectamente) que Braga é a capital da Galiza. Mais tarde, porém, alguém nos prova que a capital da Galiza é a Corunha, pelo que há que rever as nossas crenças. Situações como esta levam-nos a colocar questões do tipo:

- Como podemos raciocinar acerca de conhecimento e crenças?;
- Poder-se-á desenvolver uma linguagem de programação para raciocinar acerca de conhecimento e crenças?;
- Poder-se-á raciocinar com base no conhecimento e crenças de uma dada entidade em particular ou agente?; e
- Como poderão dois ou mais agentes conciliar as suas crenças e conhecimento?

2.1.4 As raízes da Inteligência Artificial Distribuída

Em meados dos anos 70, a investigação em Inteligência Artificial (*IA*) tinha feito progressos significativos em diferentes áreas do saber, em particular através do emprego de métodos de trabalho que faziam recurso a conhecimento de uma forma intensiva. Por outro lado, o sucesso conseguido com diferentes tipos de sistemas, tais como os sistemas de produção, em que o conhecimento é codificado e tratado a partir de estruturas de dados de reduzida dimensão, altamente manipuláveis, conduziu a tentativas de construção de sistemas modelares que exploraram a metáfora dos especialistas cooperantes.

Sistemas baseados no paradigma dos Quadros Negros [Nii 1986] [Nii 1989] capturaram então o cerne das ideias. Lançando as pontes entre este tipo de sistemas e o trabalho então em curso sobre as Redes Neuronais Artificias, criou-se o enquadramento necessário para o influxo de tecnologia de redes, o que levou os sistemas distribuídos baseados em conhecimento a afirmarem-se como um ramo autónomo de pesquisa em *IA*, e enquadráveis com o trabalho aqui desenvolvido.

Embora construindo um novo paradigma para a resolução de problemas, por força das raízes históricas em que era então desenvolvido o trabalho em *IA*, os primeiros investigadores em Inteligência Artificial Distribuída (*IAD*) adoptaram uma postura em tudo análoga à da *IA*; i.e., dado um problema, como poderiam eles construir sistemas, sistemas distribuídos, para a sua resolução? A ênfase de todo este esforço era colocada no problema e no modo de administrar múltiplos agentes por forma a que trabalhassem em equipa.

Procurava-se, assim, utilizar formas de organização social previamente conhecidas e de planeamento de tarefas em tempo real, troca de objectivos e de soluções parciais, de modo a potenciar e aumentar a coerência da actividade em grupo, sem que isso levasse a uma excessiva sobrecarga de trabalho.

2.1.5 Incorporando Novas Metáforas: Sistemas Multiagente

As raízes históricas dos Sistemas de Processamento Distribuído (*SPD*) residem no modo como a resolução de problemas se efectua a partir de computadores. Por conseguinte, é natural assumir que as entidades constituintes de um *SPD*, sendo computadores programáveis, possam estar na dependência das acções a desenvolver, num dado instante ou momento. Mas esta presunção de que o indivíduo faz o que lhe é dito para fazer não é aplicável à modelação de sistemas sociais, na qual se fundamenta muita da investigação desenvolvida em *SPD*. Por exemplo, em Economia, Teoria dos Jogos, Psicologia, estudam-se entidades que não são facilmente programáveis, em que o trabalho de investigação em curso visa determinar os tipos de comportamento que se desenvolvem entre os actores em jogo, conhecidas certas condicionantes dos sistemas, ou o modo de estabelecer condições no sistema que levem a comportamentos pré-estabelecidos. Enquanto os *SPD* tomam como axioma que os agentes deverão ser capazes de dialogar e

entender-se, partilhar tarefas, comunicar com verdade, etc., experiências em Ciências Sociais tornam claro que conseguir tais atributos em grupos de indivíduos não é uma tarefa fácil. Isto leva-nos a sustentar que, se diferentes indivíduos programarem diferentes agentes, estes últimos competirão, estarão em desacordo e actuarão de modo geral tendo em conta os interesses dos seus progenitores e não o melhor interesse do grupo, considerado como um todo [Franklin & Graesser 1996].

Se os agentes não podem ser levados a cooperar, partilhar tarefas, revelar-se honestos, nesse caso sobre que princípios são construídos? O trabalho que tem sido desenvolvido em Sistemas Multiagente (*SM*), leva-nos a presumir, tendo em atenção as disciplinas da Teoria dos Jogos e das Ciências Sociais, que um agente deve ser racional; i.e., seja o que for que estiver a fazer, deve esforçar-se por maximizar a sua própria componente de benefício/custo. Mas então que dizer acerca de agrupamentos de agentes racionais, considerados como um todo?

Há, porém, trabalho em *SM* orientado no sentido de identificar as condições (e.g., qual deve ser o grau de conhecimento mútuo entre agentes) que possam levar os agentes racionais a ter determinado comportamento, de forma a possibilitar o desenvolvimento de sinergias ao nível da sociedade de agentes. Por conseguinte, em *SM* tem sido colocada ênfase no agente e no modo de levá-lo a interagir de uma forma construtiva com outros agentes. Há investigadores, por exemplo, que fazem incidir o seu trabalho sobre agentes com convicções e interesses muito próprios, mas que podem, contudo, vir a cooperar, se as tarefas que lhes forem adstritas lhes trouxerem algum benefício. Por outro lado, outros investigadores dedicam-se ao estudo de protocolos que poderão levar os agentes a falar verdade, ao mesmo tempo que alcançam consenso sobre planos de acção, ou a exibir propriedades de grupo tais como a eficiência, a simplicidade, a estabilidade, etc.

A presente exposição tem como objectivo apresentar o modo de relacionar *SM* e *SPD*, os quais, não sendo mutuamente exclusivos, se apresentam, de facto, como tendo sido construídos um sobre o outro, numa certa extensão. A escolha desta metodologia baseada em *SM* para a resolução de problemas tem a ver com o objectivo de longo prazo que passa pelo desenvolvimento de agentes dirigidos para tarefas específicas, susceptíveis de agir, planear e adquirir conhecimento em situações de informação incompleta. Estes agentes deverão ter a capacidade de intervir em processos que emanam do universo do discurso, clarificar situações e definir estratégias para a resolução de problemas, através de uma interacção contínua com o meio [Haddadi 1996] [Witting 1992].

Segundo a visão de que um *SPD* é um subconjunto de *SM*; i.e., um *SM* é um *SPD* quando determinado número de presunções se materializa.

Visão 1

- A presunção de benevolência – qualquer sistema prefigura um *SPD* se os agentes no sistema se assumirem como benevolentes;
- A presunção de objectivos comuns – uma motivação para que os agentes sejam benevolentes e tenham objectivos comuns; e
- A presunção de um intento centralizador na concepção do sistema – esta perspectiva de entender o problema inclui as anteriores, uma vez que, se os objectivos por que se rege o construtor do sistema forem inoculados nos agentes, aquele, estando preocupado em fazer com que as partes funcionem como um todo, tornará necessariamente benevolente cada um dos agentes do sistema.

Visão 2 – Os *SM* disponibilizam a base para *SPD*

Enquanto os *SPD* assumem o facto de que, quaisquer que sejam as propriedades desejadas para os agentes, estas podem ser induzidas, os *SM* preocupam-se, em geral, com o modo de induzir nos agentes do sistema estas propriedades em primeiro lugar. Isto é, os *SM* apenas fazem presunções acerca das propriedades dos agentes e especulam sobre que propriedades poderão vir a adquirir, tendo em conta certo número de incentivos e características do ambiente.

Visão 3 – Os *SM* e os *SPD* prefiguram agendas complementares em termos de investigação e desenvolvimento

Embora isso esteja implícito no que caracteriza a **Visão 2**, as questões e/ou problemas colocados por investigadores em *SM* são de algum modo diferentes daqueles que se colocam em trabalhos que têm por base *SPD*. Isto leva-nos a concluir que *SM* e *SPD* não são etiquetas para certos tipos de sistemas, mas antes agendas para investigação e desenvolvimento.

2.2 Representação de Informação Incompleta

Qualquer sistema computacional necessita de armazenar e manipular informação. Os sistemas de *Bases de Dados* (*BDs*) e os sistemas de representação de conhecimento lidam com aspectos concretos do mundo real e podem-se comparar nos termos em que dividem a utilização da informação. Ambos os sistemas distinguem as funções de *representação* (descrição do esquema conceptual e dos dados) e de *computação* (construção de respostas a questões e manipulação dos dados).

Assim, os requisitos das *BDs* e dos sistemas de representação de conhecimento para o tratamento da informação são diferentes, como caracterizou Ullrich Hustadt

[Hustadt 1994], questionando a necessidade da adopção do *Pressuposto do Mundo Fechado* (*PMF*) em sistemas de representação de conhecimento.

As linguagens de manipulação de informação num sistema de *BDs* alicerçam-se, basicamente, nos seguintes pressupostos:

- *Pressuposto do Mundo Fechado* – toda a informação que não existe mencionada na base de dados é considerada falsa;
- *Pressuposto dos Nomes Únicos* – duas constantes diferentes (que definam valores atómicos ou objectos) designam, necessariamente, duas entidades diferentes do universo de discurso; e
- *Pressuposto do Domínio Fechado* – **não** existem mais objectos no universo de discurso para além daqueles designados por constantes na base de dados.

Um dos modelos mais importantes, no que concerne à manipulação de informação de *BDs*, é o *modelo relacional*, que descreve a informação em termos de valores atómicos e relações entre conjuntos desses valores. Contudo, a tipificação da informação de um sistema de representação de conhecimento não se pode fazer nos mesmos termos em que se caracteriza uma *BD* convencional. Nem sempre se pretende assumir que a informação representada é a única que é válida e, muito menos, que as entidades representadas sejam as únicas existentes no mundo exterior. Os pressupostos em que se pretende basear um sistema de representação de conhecimento, passam por:

- *Pressuposto do Mundo Aberto* – podem existir outros factos ou conclusões verdadeiros para além daqueles representados na base de conhecimento;

- *Pressuposto dos Nomes Únicos* – duas constantes diferentes (que definam valores atómicos ou objectos) designam, necessariamente, duas entidades diferentes do universo de discurso; e
- *Pressuposto do Domínio Aberto* – podem existir mais objectos do universo de discurso para além daqueles designados pelas constantes da base de conhecimento.

Todo este trabalho vai no sentido de dotar um sistema de representação de conhecimento de características capazes de o levar a operar segundo estes três pressupostos. A implementação de esquemas de raciocínio não-monótono, abordado em Analide [Analide 1996], proporcionam uma maior flexibilidade no mecanismo de inferência e no processo de obtenção de conclusões, mas fazem-no recorrendo à negação por falha, adoptando o *PMF*. Um sistema capaz de completar raciocínios não-monótonos não significa, por si só, que possa tratar informação incompleta.

As linguagens tradicionais de *Programação em Lógica* (*PL*) proporcionam uma ferramenta poderosa para a representação de conhecimento, uma vez que a característica não-monótona da negação por falha torna possível expressar várias situações de senso comum que não são disponibilizadas pela lógica clássica. Contudo, a *PL* tradicional não permite representar directamente informação incompleta. Um sistema de raciocínio sobre uma base de conhecimento com informação incompleta, contempla questões de representação e de inferência diferentes das que se podem identificar num sistema não-monótono, função do tipo de respostas que se pretendem obter em cada um dos sistemas.

Apesar de, numa teoria lógica clássica, o conhecimento ser equacionado em termos do que se prova ser ***verdadeiro***, o que se prova ser ***falso*** e o que representa informação sobre a qual não se pode ser conclusivo, num programa em lógica, as respostas às questões colocadas são, apenas, de dois tipos: ***verdadeiro*** ou ***falso***. Isto

deve-se ao facto de um programa em lógica apresentar várias limitações em termos da representação de conhecimento (não permite representar explicitamente informação negativa ou disjuntiva), para além do facto de que, em termos de uma semântica operacional se aplicar, automaticamente, o *PMF* a todos os predicados.

A generalidade dos programas escritos em lógica representa implicitamente a informação negativa, assumindo a aplicação do raciocínio segundo o *PMF*. Uma extensão de um programa em lógica pode incluir informação negativa explicitamente, bem como explicitar directamente o *PMF* para alguns predicados. Consequentemente, torna-se possível distinguir três tipos de conclusões para uma questão: esta pode ser ***verdadeira***, ***falsa*** ou, quando não existe informação que permita inferir uma ou outra das conclusões anteriores, a resposta à questão será ***desconhecida***.

2.2.1 Extensão à Programação em Lógica

De acordo com o que foi explicitado em epígrafe, um programa em lógica determina respostas em termos da veracidade ou falsidade das questões, não sendo possível, assim, implementar um mecanismo de raciocínio que possibilite abordar a representação de informação incompleta. De uma forma bastante simples, pode-se descrever através de um programa em lógica, a asserção segundo a qual *as aves voam*, representando-a na forma:

$$voa(X) \leftarrow ave(X)$$

De forma idêntica, é possível corporizar a afirmação de que *as avestruzes não voam*, pela introdução de um segundo predicado, *não-voa(X)*, cujo significado é a antítese do predicado *voa(X)*, e declarar:

$$não\text{-}voa(X) \leftarrow avestruz(X)$$

ou seja, o elemento *X* ***não voa*** se for uma avestruz. Contudo, esta representação não capta a relação que deve existir entre a informação representada por cada um dos

predicados. Isto porque, computacionalmente, se tratam de dois predicados diferentes, sem qualquer tipo de relação definida entre si. Intuitivamente, é óbvio que essa relação deveria existir.

A abordagem de problemas em *PL* seria muito mais natural se fosse possível declarar explicitamente informação falsa (negativa), já que este tipo de informação tem um papel influente no raciocínio por senso comum. O objectivo de estender a *PL* é o de que passe a permitir representar informação negativa explicitamente. A extensão de um programa em lógica passa a contar com dois tipos de negação: a ***negação por falha***, característica dos programas em lógica tradicionais, representada pelo termo *não*, e a ***negação forte*** ou ***clássica***, como forma de identificar informação negativa, ou falsa, representada pela conectiva '¬'.

Deste modo, a forma de dar corpo à proposição *as avestruzes não voam* será descrita pela negação, explícita, da extensão do próprio predicado *voa(X)*, o que é o mesmo que dizer, pela negação, explícita, do próprio predicado:

$$\neg voa(X) \leftarrow avestruz(X)$$

O significado a atribuir a esta última expressão é o de que é ***falso*** que *X* voe, se for uma avestruz. Para já, ganha-se na relação entre as afirmações, o que estava omisso quando representadas num programa em lógica tradicional. Em [McCarthy 1980], *John McCarthy* apresenta um exemplo bastante feliz de concretização deste tipo de sistemas. Considere-se a situação em que se pretende permitir que um autocarro escolar atravesse a linha do caminho-de-ferro, na condição de que não se aproxime nenhum comboio. Esta afirmação pode ser representada por:

$$atravessar \leftarrow não\ comboio$$

se a inexistência do átomo *comboio* na base de conhecimento for interpretada como a ausência do comboio em aproximação. Mas esta convenção de notação é inaceitável se a informação acerca da presença ou ausência do comboio não estiver disponível. Se, por exemplo, o átomo *comboio* não existir devido ao facto de o

condutor do autocarro ter a visão bloqueada, continua a não se pretender que o autocarro atravesse a linha.

No entanto, esta diferença entre a intuição que se tem do caso e o significado da regra representada pode desaparecer, considerando a utilização da negação clássica para descrever a situação:

$$atravessar \leftarrow \neg\, comboio$$

O significado desta expressão é, agora, de que a conclusão *atravessar* só é possível se for ***falsa*** a existência do termo *comboio*, ou seja, se for falsa a existência do comboio em aproximação. Desta forma, a conclusão *atravessar* não fará parte das soluções para a questão a menos que o facto $\neg$ *comboio* seja incluído na base de conhecimento.

A diferença entre estes dois casos está em que, na primeira situação, o autocarro atravessará a linha se *não existir uma prova* de que o comboio esteja a aproximar-se e, na segunda, o autocarro atravessará se *existir uma prova* de que o comboio *não está* a aproximar-se. A distinção entre o significado de *não P* e $\neg P$ é essencial quando não se quer (ou não se pretende) assumir que a informação existente sobre *P* é completa; i.e., quando não se aplica o *PMF* a *P*, não podendo assumir que a ausência de informação denote, inequivocamente, a sua falsidade.

Para além do mais, a introdução da negação explícita, juntamente com a negação por falha, permite uma maior expressividade da linguagem. Se o problema for o de o condutor do autocarro *não ter a certeza absoluta* de que o comboio *não está a aproximar-se*, então *não deverá atravessar* a linha. Esta afirmação pode ser descrita de uma forma muito natural, nomeadamente:

$$\neg\, atravessar \leftarrow não\ \neg\, comboio$$

Assim, declarações do tipo ***não*** *é possível mostrar que* ***não*** *possa acontecer P*, podem ser descritas por formulações da forma *não* $\neg$ *P*. Exemplos de situações

como esta são comuns em problemas do dia-a-dia, que envolvem o raciocínio por senso comum.

Em sistemas construídos com base nesta forma de extensão à *PL*, passa-se a ter uma terceira hipótese de responder às questões que porventura lhe sejam colocadas: para além de serem verdadeiras as conclusões obtidas a partir da informação positiva e falsas as obtidas em função da informação negativa, torna-se possível concluir que a resposta é ***desconhecida***, se nenhuma das conclusões anteriores for possível.

Formalmente, pode-se definir um programa em lógica estendida como um conjunto de regras da forma:

$$q \leftarrow p_1 , \ldots , p_m , \textit{não } p_{m+1} , \ldots, \textit{não } p_{m+n}$$

em que tanto o q como os p_i são literais, isto é, fórmulas da forma a ou $\neg a$, sendo a um átomo, e $m, n \geq 0$.

A semântica de um programa em lógica estendido atribui, ao programa, um ***conjunto de respostas***, $\mathcal{R}$, como sendo o conjunto das fórmulas que correspondem às conclusões que podem ser obtidas por um sistema de inferência capaz de implementar o esquema de raciocínio adequado. Diremos que $\neg q$ é ***verdadeiro***, para um determinado conjunto de respostas, $\mathcal{R}$, se $\neg q \in \mathcal{R}$, enquanto que *não q* é verdadeiro, em $\mathcal{R}$, se $q \notin \mathcal{R}$.

A interpretação de uma questão q colocada a um sistema capaz de raciocinar em termos da Programação em Lógica Estendida (*PLE*), Π, pode ser definida, informalmente, da seguinte forma:

- ***verdadeira*** se q é ***verdadeiro*** em todos os conjuntos de respostas do programa Π;

- ***falsa*** se $\neg q$ é ***verdadeiro*** em todos os conjuntos de respostas de Π; e
- ***desconhecida*** em qualquer outro caso.

ou seja, em termos práticos, considerando uma questão abstracta *q(X)*, em que *X* pode tomar a forma $X_1, X_2, ..., X_n$, definem-se as seguintes três respostas possíveis para essa questão:

- ***verdadeiro*** *se* $\exists X : q(X)$;
- ***falso*** *se* $\exists X : \neg q(X)$; e
- ***desconhecido*** *se* $\neg \exists X : q(X) \vee \neg q(X)$

em que '*:*' é a notação para *tal que*.

Outra importante motivação para a introdução da negação explícita em programas em lógica está relacionada com o paralelismo existente entre informação positiva e negativa. Podem ocorrer situações em que seja mais fácil ou mais natural expressar directamente a informação negativa e, posteriormente, tirar conclusões em função dessa representação.

2.2.2 O *PMF* na Programação em Lógica Estendida

A diferença substancial entre um programa em lógica e um programa em lógica estendido está em que, no primeiro, a única forma de negação utilizada é a negação por falha na prova e, é aplicado o *PMF* a todos os predicados do programa, enquanto que num programa em lógica estendido, para além da negação por falha, faz-se uso, ainda, da negação explícita para a representação de informação, situação que é *simulada* pela negação por falha num programa em lógica.

À partida, num programa em lógica estendido, não é aplicado o *PMF* a nenhum dos predicados. Sintacticamente, um programa em lógica é, em certa medida, um caso

especial de um programa em lógica estendido. Contudo, semanticamente, existem diferenças na interpretação dos conjuntos de regras num programa em lógica e aquele a dar às que se encontram no contexto de um programa em lógica estendido.

Atente-se no programa dado na Figura 2.1 que apresenta uma extensão do predicado `par`, que é utilizado na geração de números pares.

$$par(0)$$
$$par(s(s(X))) \leftarrow par(X)$$

Figura 2.1 – Extensão do predicado `par`

Em termos da Programação em Lógica Estendida (*PLE*), o conjunto de soluções que se obtém a partir da extensão do predicado `par` é dado por:

$$\{par(0), par(s(s(0))), par(s(s(s(s(0))))), \ldots\}$$

o que denota, uma vez que não contém a declaração *par(s(0))* nem $\neg par(s(0))$, que a resposta à questão *par(s(0))* será ***desconhecido***, o que não é propriamente o que se pretende expressar neste caso. Nesta situação, o que se pretende afirmar pela ausência de uma solução em particular, não é que esta seja desconhecida mas, antes, de que essa solução, de facto, é falsa (porque não existe); ou seja, pretender-se-ia ver-se aqui aplicado o *PMF*, para que a ausência de informação fosse considerada como a negação da sua existência.

Este significado pode ser contemplado no predicado descrito pelo programa em lógica estendido anterior, se lhe for adicionada uma regra que formalize o conceito subjacente ao *PMF*, expressando que é falso tudo aquilo que não for possível provar como sendo verdadeiro. Neste caso, é falso que um número seja par se não existir uma prova de que o seja; i.e.,

$$\neg par(X) \leftarrow não\ par(X)$$

Incluindo esta regra no programa dado pela Figura 2.1, o predicado `par(X)` toma a forma dada pela Figura 2.2:

par(0)
par(s(s(X))) ← par(X)
¬ par(X) ← não par(X)

Figura 2.2 - Formalização do *PMF* para o predicado *par*

Pelo que o conjunto de soluções dado pelo programa da Figura 2.2, para os números pares, toma a forma:

{*par(0), ¬ par(s(0)), par(s(s(0))), ...*}

Como já havia sido referido, a ausência de um determinado átomo *A*, num programa em lógica, significa que *A* é falso e que a resposta a questões que envolvam *A* deverá ser ***falsa***, enquanto que a falta desse mesmo átomo *A* no conjunto de regras que corporizam um programa em lógica estendido (e está-se a falar do mesmo conjunto de regras), indica que a resposta à questão colocada deva ser ***desconhecido***.

Este exemplo sugere que um programa em lógica é semanticamente equivalente a um programa em lógica estendido ao qual se adicionam regras com a formalização do *PMF* para cada um dos predicados em presença. Genericamente, pode-se definir o *PMF* para um predicado *p*, na linguagem de um programa em lógica estendido, pela regra:

$$\neg p(X) \leftarrow \text{não } p(X) \qquad (2.1)$$

em que *X* pode tomar a forma $X_1, X_2, ..., X_n$, sendo *n* o número de argumentos do predicado *p*.

Um programa em lógica, ao qual se adicionam regras como (2.1) para cada um dos predicados existentes no programa, transforma-se num programa em lógica estendido, semanticamente equivalente a um programa em lógica. Refira-se que um sistema capaz de raciocinar em termos de informação incompleta tal como o que foi apresentado nesta secção não é, necessariamente, um sistema não-monótono [Analide 1996].

2.2.3 Valores Nulos

Nas secções anteriores foi abordada a problemática do raciocínio em face de falta de informação ou onde se admite a representação parcial de informação. Nesta secção pretende-se fazer um estudo do tipo de valores que podem surgir numa situação de informação incompleta. Esta identificação de certo tipo de valores, designados por ***valores nulos***, surge como uma estratégia para a enumeração de casos, para os quais urge que se distinga entre situações em que as respostas a uma dada questão(ões) deverão ser do tipo ***conhecidas*** (verdadeiras ou falsas) ou ***desconhecidas*** [Traylor & Gelfond 1993].

A representação de valores nulos será abordada no âmbito da *PLE* que tem vindo a ser apresentada neste capítulo. Serão três os valores nulos aqui tratados: o primeiro permitirá representar valores desconhecidos e não necessariamente de um conjunto determinado de valores; o segundo representará valores desconhecidos, mas de um conjunto finito e determinado de valores; por fim, o terceiro género de valores nulos será de um tipo ligeiramente diferente e, representará valores não permitidos, considerados, nomeadamente, na assimilação de informação nas bases de conhecimento. Nas subsecções seguintes faz-se uma apresentação pormenorizada destes tipos de valores nulos.

2.2.3.1 Valor Nulo do Tipo Desconhecido

Considere-se o caso em que se pretende identificar o tipo de canto de algumas aves. A Tabela 2.1 representa a totalidade da informação conhecida, ou seja, transcreve ***completamente*** o conhecimento que se tem, neste momento, sobre o canto das aves.

Tabela 2.1 – Tipos de canto das aves

AVE	*SOM*
periquito	*chilro*
canário	*assobio*

Esta descrição informal do conhecimento pode ser equacionada através da *PLE*, pela introdução do predicado `canto` com dois argumentos, o primeiro representando a *AVE* e o segundo identificando o *SOM* que esta emite. A informação contida na Tabela 2.1 pode ser dada pelo programa em lógica constante da Figura 2.3.

canto(periquito, chilro)
canto(canário, assobio)
¬ canto(Ave, Som) ← não canto(Ave, Som)

Figura 2.3 – Representação formal da informação da Tabela 2.1

A terceira cláusula do Programa da Figura 2.3 materializa o *PMF* para o predicado `canto`, o que se justifica, neste caso, por se ter assumido que a Tabela 2.1 é o repositório de toda a informação para o universo de discurso objecto de consideração, e permite concluir, correctamente que, por exemplo, *¬canto(canário,chilro)*, ou seja, que é falso que o canto do canário seja o chilro.

No entanto, a situação modifica-se se a Tabela 2.1 for substituída pela Tabela 2.2.

Tabela 2.2 – Expansão à informação da Tabela 2.1

AVE	*SOM*
periquito	*chilro*
canário	*assobio*
ave rara	*fado*

Nesta tabela, a entidade *ave rara* representa um valor específico do tipo *AVE*. Este é um ***valor nulo*** do tipo ***desconhecido e não necessariamente de um conjunto determinado de valores*** que denota uma ave ***desconhecida***, talvez diferente de *periquito* ou *canário*, que são as duas únicas aves conhecidas.

Neste contexto, a resposta à questão *canto(X,assobio) – qual é a ave cujo canto é o assobio* – deverá ser *canário*, enquanto que a resposta à questão *canto(X,fado) – qual é a ave cujo canto é o fado* – deverá ser ***desconhecida***, no sentido em que a questão é verdadeira, porque tem solução, mas a concretização dessa solução é que é desconhecida, já que não se sabe de nenhuma ave em concreto que satisfaça as condições enunciadas; i.e., apenas se sabe que existe uma ave rara que canta o fado.

Em continuação, a resposta à questão *canto(canário,chilro)* continuará a ser ***falsa***, enquanto que a questão *canto(canário,fado)* não terá uma resposta conclusiva por ser ***desconhecida*** a ave que canta o fado. Após esta análise do problema proposto e, considerando a informação da Tabela 2.2, parece óbvio que se represente essa informação acrescentando ao programa da Figura 2.3 a cláusula:

canto(ave rara, fado)

Contudo, esta alteração ainda não permite obter as conclusões desejadas, já que a questão *canto(canário,fado)* é ***falsa*** por não existir uma prova de que seja verdadeira (através da cláusula de formalização do *PMF*) quando, em função do exposto, se pretenderia que fosse ***desconhecida***.

O que está a acontecer é que a conclusão obtida a partir da cláusula que formaliza o *PMF* para o predicado *canto* não está a permitir a flexibilização das respostas por não admitir excepções. E a questão é mesmo esta: o valor nulo que identifica a ave rara que canta o fado deverá ser considerado como uma excepção à regra de formalização do *PMF*.

Para representar correctamente esta informação torna-se necessária uma formalização adequada do *PMF* numa linguagem de *PL* que permita tratar valores nulos. O que está em causa neste momento é a mudança de estado da base de conhecimento entre o instante em que se definiu a informação contida na Tabela 2.1 e o instante que deu origem à informação constante da Tabela 2.2, ou seja, o problema está relacionado com a flexibilidade do sistema proposto no que respeita à evolução natural do conhecimento.

Estas questões foram abordadas em [Analide 1996], onde se faz uma referência à representação de excepções como situações anómalas em relação à extensão de um determinado predicado.

Considere-se, então, que a identificação de valores nulos do tipo desconhecido será feita através da introdução de uma situação de excepção, correspondendo a uma condição anómala, na cláusula de formalização do *PMF*. Assim, a terceira cláusula do programa dado na Figura 2.3 deverá ser substituída por:

$$\neg\, canto(Ave, Som) \leftarrow$$
$$\quad não\ canto(Ave, Som),$$
$$\quad não\ excepção_{canto}(Ave, Som)$$

e deverá ser identificada uma excepção relativamente ao canto das aves na forma:

$$excepção_{canto}(Ave, Som) \leftarrow$$
$$\quad canto(ave\ rara, Som)$$

Com esta formulação, passa-se a considerar não só que é ***falso*** que o *canto* da *Ave* seja porventura um dado *Som* a não ser que exista uma prova de que assim seja, mas

também que não existe uma prova de que haja uma situação anómala definida entre essa *Ave* e esse *Som* para o *canto* daquela.

O programa resultante é o constante da Figura 2.4 que expressa correctamente a informação representada na Tabela 2.2. Generalizando, tem-se que valores nulos do tipo desconhecido se representam, no contexto de um programa em lógica estendido, como situações anómalas a serem identificadas na definição dos casos que são considerados excepções à afirmação da informação negativa.

canto(periquito, chilro)
canto(canário, assobio)
canto(ave rara, fado)
¬canto(Ave, Som) ←
não canto(Ave, Som),
não excepção$_{canto}$(Ave, Som)
excepção$_{canto}$(Ave, Som) ← canto(ave rara, Som)

Figura 2.4 – Representação da informação da Tabela 2.2

Para um predicado *p*, representa-se um valor nulo v_i do tipo desconhecido, como uma excepção a $\neg p$, tem-se que:

$$\neg p(X) \leftarrow \text{não excepção}_p(X)$$

e, de igual modo, as excepções que se pretendem definir como valores nulos:

$$\text{excepção}_p(X) \leftarrow p(v_1)$$
$$\text{excepção}_p(X) \leftarrow p(v_2)$$
$$\ldots$$

Nestas expressões, considerando *X* como uma sequência de variáveis $X_1, X_2, \ldots, X_n$, em que *n* é o número de argumentos de *p*, então v_i representará a mesma sequência, mas na qual se substituirá a variável X_j pelo valor nulo dado por *v* (e.g., $X_1, \ldots, v, \ldots, X_n$).

No caso da aplicação que tem sido utilizada, tal foi efectuado incluindo a excepção no corpo da cláusula de formalização do *PMF* que já fazia parte do programa em lógica, correspondendo à forma genérica:

$$\neg p(X) \leftarrow \text{não } p(X) \text{ , não } \text{excepção}_p(X)$$

a qual se aplicam as mesmas considerações enunciadas em epígrafe para as situações concretas que foram objecto de estudo.

2.2.3.2 Valor Nulo do Tipo Desconhecido, de um Conjunto Dado de Valores

A diferença entre o valor nulo do tipo desconhecido e não necessariamente de um conjunto determinado de valores, visto na subsecção anterior, e o valor nulo que se pretende apresentar, ***desconhecido mas de um conjunto finito e determinado de valores***, está precisamente neste facto: este último valor nulo representará um (ou mais) valores de um conjunto finito de valores bem determinados; só não é conhecido, especificamente, qual dos valores validará a(s) questão(ões) porventura colocada(s) ao sistema.

Considere-se uma extensão à Tabela 2.2 através da asserção de que o *SOM* do canto da *AVE pinguim* atende a uma de duas hipóteses: *ópera* ou música *clássica*. Represente-se esta informação através do conteúdo da Tabela 2.3.

Tabela 2.3 – Informação sobre os pinguins com novos dados

AVE	*SOM*
periquito	*chilro*
canário	*assobio*
ave rara	*fado*

pinguim	*{ópera, clássica}*

A expressão *{ópera, clássica}* denota o conjunto finito de possíveis valores que corporizam o tipo de informação existente sobre o *SOM* do *pinguim*. Numa primeira aproximação para a resolução do problema, podem-se representar estes novos itens de informação através de uma disjunção de termos:

$$canto(pinguim, \underline{ópera}) \vee canto(pinguim, \underline{clássica})$$

Figura 2.5 – Representação de parte da informação da Tabela 2.3

e incluir esta declaração no programa dado pela Figura 2.4. Formalmente, a base de conhecimento resultante denota correctamente a informação da Tabela 2.3. Contudo, a introdução da disjunção de termos não é, normalmente, implementada em sistemas baseados na *PLE*, pelo menos directamente. Quando isso é feito, paga-se um custo computacional elevado pela utilização de mecanismos de inferência sobre bases de conhecimento disjuntivas.

Sendo assim, torna-se necessário reformular esta aproximação de forma a tornear este senão, o que será conseguido através de uma abordagem à resolução semelhante à realizada para o caso de valores nulos apresentado anteriormente.

A informação representada pela expressão da Figura 2.5 não pode ser utilizada, considerando, isoladamente, cada um dos termos da disjunção, uma vez que não se sabe, concretamente, qual das partes é verdadeira. No entanto, pode ser considerada como uma excepção em relação àquilo que é negado ser verdadeiro.

Relativamente à identificação das excepções à cláusula que formaliza o *PMF* para o predicado *canto*, esta situação envolve um caso muito particular de senãos. Ser

possível ao pinguim cantar ópera é uma excepção à negação de que o faça, tal como o é o facto de poder cantar música clássica. Face ao que foi dito, e atendendo às situações já tratadas em epígrafe, para a resolução do problema ir-se-ão elaborar duas excepções, que correspondem a outras tantas possibilidades de resposta para a questão que foi formulada, e dadas na forma:

$excepção_{canto}(pinguim, ópera)$
$excepção_{canto}(pinguim, clássica)$

Este tipo de excepções não têm a ver com o facto de o pinguim cantar ou não cantar o que quer que seja. São, simplesmente, excepções a tomar em consideração quando se possa vir a adoptar pela conclusão de que não o faça (i.e., cante). Acrescentando estas asserções ao programa dado pela Figura 2.4, obtemos como resultado o programa constante da Figura 2.6, que é capaz de lidar com informação do tipo disjuntiva, tal como foi declarado através da expressão da Figura 2.5.

canto(periquito, chilro)
canto(canário, assobio)
canto(ave rara, fado)
¬ canto(Ave, Som) ←
    não canto(Ave, Som) ,
    não $excepção_{canto}(Ave, Som)$

$excepção_{canto}(Ave, Som) \leftarrow$
    canto(ave rara, Som)

$excepção_{canto}(pinguim, ópera)$
$excepção_{canto}(pinguim, clássica)$

Figura 2.6 - Representação de informação disjuntiva

Em virtude da interpretação dada a um programa em lógica estendido, apresentada na secção *Extensão à Programação em Lógica* e, uma vez que não existe informação positiva sobre o assunto, não se pode concluir que o pinguim cante

ópera (ou música clássica) mas, como existem excepções que impedem que a conclusão seja falsa, a resposta a qualquer uma das questões *canto(pinguim,ópera)* ou *canto(pinguim,clássica)* deverá ser desconhecida. Sem embargo, a resposta à questão *canto(pinguim,assobio)*, por exemplo, continuará a ser falsa, uma vez que o pinguim não canta qualquer coisa, só canta ópera ou música clássica, de acordo com o que está declarado no programa da Figura 2.6.

Concluindo, pode-se dizer que valores nulos do tipo analisado nesta subsecção representam-se como um conjunto de excepções à informação negativa representada explicitamente na base de conhecimento. Generalizando, tem-se que, dado um predicado p, define-se um conjunto de valores nulos v_i do tipo desconhecido, de um conjunto determinado de valores, como uma sequência de excepções a $\neg p$, dadas por:

$$\neg p(X) \leftarrow \text{não } excepção_p(X)$$

ou, em termos da aplicação que tem servido de base ao presente trabalho, na forma:

$$\neg p(X) \leftarrow \text{não } p(X) \wedge \text{não } excepção_p(X)$$

e concretizadas como excepções, para cada um dos valores do tipo nulo que denotam possíveis soluções para o problema, na forma:

$excepção_p(v_1)$
$excepção_p(v_2)$
...

em que os v_i denotam valores do tipo nulo tomados do conjunto de valores possíveis.

2.2.3.3 Valor Nulo do Tipo Não Permitido

O terceiro tipo de valores nulos a ser abordado, difere ligeiramente dos que foram considerados anteriormente, uma vez que, além de identificarem, também, valores desconhecidos, caracterizam, um tipo de dados que não se pretende que venham a fazer parte da base de conhecimento.

Para se representar este valor nulo e, na linha do que tem vindo a ser descrito, considere-se a informação constante da Tabela 2.4. O valor dado por ω, para além de identificar um valor ***desconhecido***, denota um valor do tipo ***não é permitido*** (i.e., não é passível de ser conhecido). Por exemplo, no caso de se reportar a aves que cantam o hino, qualquer tentativa para inserir informação acerca das aves que cantam o hino, que venha a ser feita posteriormente à declaração deste facto à base de conhecimento do sistema deve ser rejeitada, por ser fonte de inconsistência. Isto significa que a última expansão à base de conhecimento, de que resultou a Tabela 2.4, introduziu a informação de que existe uma ave, cuja identificação não é passível de ser conhecida, e que canta o hino.

Tabela 2.4 – Introdução de valores do tipo não permitido

AVE	*SOM*
periquito	*chilro*
canário	*assobio*
ave rara	*fado*
pinguim	*{ópera, clássica}*
ω	*hino*

Se se começar por considerar este valor nulo como sendo do mesmo tipo daquele que primeiro foi referido em epígrafe, isto é, um nulo do tipo desconhecido e não

necessariamente de um conjunto determinado de valores, pode-se representá-lo na base de conhecimento, através de uma produção, na forma:

canto(ω,hino)

e considerá-lo como uma excepção à informação negativa constante da extensão do predicado `canto`, o que pode ser expresso na forma:

$excepção_{canto}(Ave, Som) \leftarrow$
$\quad canto(\omega, Som)$

Deste modo, a resposta para a questão *canto(periquito,chilro)* continua a ser dada pela constante lógica ***verdadeiro*** (outra coisa não seria de esperar), a questão *canto(canário,chilro)* continuará a ter como resposta a constante lógica ***falso***, enquanto que a resposta à questão *canto(canário,hino)* será dada pela constante lógica ***desconhecido***, uma vez que não se conhece (nem se permite conhecer) o valor concreto da ave que canta o hino.

O programa resultante, que corresponde à última expansão da informação constante da Tabela 2.4, é dado pela Figura 2.7.

canto(periquito, chilro)
canto(canário, assobio)
canto(ave rara, fado)
canto(ω,hino)
$\neg$ *canto(Ave, Som)* $\leftarrow$
$\quad$ *não canto(Ave, Som)* ,
$\quad$ *não* $excepção_{canto}$*(Ave, Som)*

$excepção_{canto}(Ave, Som) \leftarrow$
$\quad canto(ave\ rara, Som)$

$excepção_{canto}(pinguim, ópera)$
$excepção_{canto}(pinguim, clássica)$
$excepção_{canto}(Ave, Som) \leftarrow$
$\quad canto(\omega, Som)$

Figura 2.7 – Representação de valores nulos do tipo não permitido

O programa constante da Figura 2.7 representa correctamente a informação constante da Tabela 2.4. No entanto, esta representação não é suficiente para capturar o carácter dinâmico a associar aos valores nulos do tipo não permitido, cujas posteriores alterações deverão ser impedidas. Como exemplo, considere-se que é *forçada* a inserção na base de conhecimento do termo:

canto(papagaio, hino) (2.2)

Torna-se evidente que não é provocado nenhum problema de inconsistência com a informação que já existia na base de conhecimento. Porém, o significado que se atribuiu ao tipo de valor nulo em observação, um valor nulo do tipo não permitido, não está a ser observado, porque a partir desse momento seria possível identificar uma ave – *papagaio* – que cantaria o hino.

Se é a nossa intuição que não está a ser devidamente descrita pela informação constante da base de conhecimento, é porque ainda não se estabeleceu nenhum formalismo capaz de a representar. Mais uma vez o problema ocorre durante o processo de alteração da informação existente na base de conhecimento. No caso em discussão, o que se pretende é que não exista qualquer evolução do conhecimento constante da base se esta se vier a efectuar através da inclusão de informação que viole a condição imposta pelo valor nulo do tipo não permitido.

A questão está em identificar as restrições implicitamente presentes num valor nulo do tipo não permitido. Qualquer expressão em lógica (regra) que procure implementar um teste à consistência da informação da base de conhecimento, terá necessariamente que exprimir que uma *AVE*, cujo *CANTO* é um determinado *SOM*, no caso presente o *hino*, só deverá estar declarada como tal na base de conhecimento se essa *AVE* não for indiciada em quaisquer valores nulos do tipo não permitido.

Comece-se por considerar a representação da informação que denota um valor nulo do tipo não permitido, introduzindo o predicado `nulo`, com apenas um argumento, que corporizará esse valor. O facto de ω ser um valor com estas características dá origem à produção:

nulo(ω)

a ser inserida na base de conhecimento. Um método para a verificação da consistência da informação da base de conhecimento com o significado pretendido poderá ser dado pela expressão (termo):

$$canto(Ave, hino) \wedge não\ nulo(Ave) \tag{2.3}$$

Através da invocação deste termo é possível constatar que se encontra uma inconsistência, se existir uma ave que cante o hino e que não esteja identificada como sendo um valor nulo do tipo não permitido. Formalmente, em *PL*, a fórmula (2.3) deve ser representada por uma regra cuja conclusão seja sempre falsa, isto é, correspondendo a uma cláusula com a cabeça vazia, da forma:

$$\neg (canto(Ave, hino), não\ nulo(Ave)) \tag{2.4}$$

O programa em lógica que descreve correctamente a informação constante na Tabela 2.4, deverá contemplar, então, uma descrição do valor nulo ω como tal, bem como incluir a regra (2.4), que denota o invariante que captura o significado dinâmico associado ao valor nulo, para além da formulação que já havia sido proposta no programa dado pela Figura 2.7. O programa em lógica resultante é o constante da Figura 2.8.

```
canto(periquito, chilro)
canto(canário, assobio)
```

canto(ave rara, fado)
canto(ω,hino)
¬ canto(Ave, Som) ←
não canto(Ave, Som) ,
não excepção$_{canto}$(Ave, Som)

excepção$_{canto}$(Ave, Som) ←

canto(ave rara, Som)

excepção$_{canto}$(pinguim, ópera)
excepção$_{canto}$(pinguim, clássica)
excepção$_{canto}$(Ave, Som) ←
canto(ω, Som)
nulo(ω)
¬ (canto(Ave, hino) , não nulo(Ave))

Figura 2.8 – Representação completa de valores nulos do tipo não permitido

Nesta altura se se *forçar* a inclusão na base de conhecimento do termo referido em (2.2), este será claramente rejeitado uma vez que viola a restrição imposta (2.4). Generalizando, dado um predicado p, e um valor nulo v_i do tipo não permitido, é possível representá-lo como uma excepção a $\neg p$, na forma:

$$\neg p(X) \leftarrow \text{não excepção}_p(X)$$

ou então dado por:

$$\neg p(X) \leftarrow \text{não } p(X) \text{ , não excepção}_p(X)$$

e concretizados como excepções para cada um dos valores nulos do tipo não permitido que denotam possíveis soluções para o problema, na forma:

$$\text{excepção}_p(X) \leftarrow p(v_1)$$
$$\text{excepção}_p(X) \leftarrow p(v_2)$$
...

em que X denota uma sequência de variáveis $X_1, X_2, ..., X_n$, e em que cada v_i se refere a um valor nulo do tipo não permitido tomado de um conjunto de valores possíveis (e.g., é então possível escrever na forma $X_1, ..., v, ..., X_n$., em que a variável X_j foi substituída pelo nulo do tipo não permitido v).

Por outro lado, cada um dos valores nulos denotados por v_i deve ainda ser dado como extensão do predicado *nulo*, na forma:

nulo(v_1)
nulo(v_2)
...

assim como deve ser incluída uma restrição (ou restrições) que impeça a posterior inclusão na base de conhecimento de valores que violem a condição imposta pelo valor nulo do tipo não permitido, o que é atingido através da consideração de expressões na forma:

$\leftarrow p(X)$, *não nulo*(X)

De notar que, para cada predicado *p*, pode existir mais do que um valor nulo do tipo não permitido, com significados diferentes (e.g., pode existir um ω_1 que cante o hino, um ω_2 que cante mal, um ω_3 que cante outra coisa qualquer, e assim por diante).

2.2.3.4 Interpretação de Valores Nulos

Neste momento torna-se crucial apresentar um sistema que permita implementar mecanismos de inferência sobre estes três tipos de valores nulos, que concretizam situações de informação incompleta. Tal sistema alicerçar-se-á no paradigma da *PL*, suportando as três respostas possíveis para quaisquer questões colocadas neste contexto: respostas dos tipos ***verdadeira***, ***falsa*** e ***desconhecida***.

A implementação prática deste sistema não será mais do que um interpretador (i.e., um demonstrador de teoremas) para questões colocadas em termos de um programa em lógica estendido, interpretação essa que será efectuada nos termos apresentados na secção *Extensão à Programação em Lógica*. Esse interpretador será traduzido

pela extensão de um meta-predicado que determinará o valor de verdade para a questão (ou teorema) colocada ao sistema, em termos da sua especificidade.

Em primeiro lugar há que definir a estrutura do predicado que se designará por *demo* (de *demo*nstração). Deste predicado, e como argumentos, ter-se-ão o Teorema a demonstrar e o respectivo resultado, que estará entre um de três valores possíveis: a veracidade, a falsidade ou o desconhecimento de resposta para a questão interpretada. O predicado pode, então, ser definido nos termos:

demo: Questão, Resposta $\rightarrow$ *{verdadeiro, falso}*

isto é, o predicado *demo* é constituído por dois argumentos, sendo o primeiro respeitante à *Questão* a colocar ao sistema e o segundo a *Resposta* para a questão dada. Uma vez que a extensão deste predicado faz parte de um programa em lógica, tem como contradomínio o conjunto de valores *{verdadeiro*, *falso}*, em que "*{*" e "*}*" é a notação utilizada para denotar conjuntos.

Tendo-se em linha de conta o que já foi mencionado sobre a demonstração de teoremas num programa em lógica estendido, obtém-se como extensão do predicado *demo* o programa em lógica dado na Figura 2.9.

demo(Q, verdadeiro) $\leftarrow$
    Q
demo(Q, falso) $\leftarrow$
    $\neg$ *Q*
demo(Q, desconhecido) $\leftarrow$
    não $\neg$ *Q ,*
    não Q

Figura 2.9 – Interpretador de questões

A primeira cláusula do programa indica que a resposta a uma questão *Q* é dada pelo valor de verdade *verdadeiro* se for possível desenvolver uma prova de *Q* a partir da

informação positiva existente na base de conhecimento. A segunda cláusula frisa que a resposta a uma questão *Q* é dada pelo valor de verdade *falso* se for possível provar $\neg Q$. A resposta a *Q* será do tipo *desconhecido* se *não* existir quer uma prova de $\neg Q$ quer uma prova de *Q* [Neves 1984].

Nestas condições, pode-se utilizar a extensão do predicado *demo* dado pela Figura 2.9 para dar resposta a algumas das questões que se podem colocar a um sistema baseado em *PLE*, como é o caso dos exemplos que foram sendo apresentados ao longo deste capítulo.

Considere-se o programa constante da Figura 2.3, que descreve a informação dada na Tabela 2.1. Uma vez que para o único predicado do programa, o canto, se aplicou o conceito do *PMF*, qualquer questão para a qual não seja possível encontrar um valor de verdade em termos das duas primeiras cláusulas (o *periquito* cujo *canto* é o *chilro* e o *canário* cujo *canto* é o *assobio*), deverá ser dada como não provada, e ser-lhe associado o valor de verdade *falso*.

Admita-se que agora se coloca a questão:

$$\neg demo(canto(canário, chilro), Resposta) \tag{2.5}$$

É fácil chegar à conclusão de que a *Resposta* à questão é dada pelo valor de verdade *falso*. Primeiro, porque a partir da primeira cláusula do predicado *demo* dado pela Figura 2.9, não há qualquer possibilidade de fazer a prova do teorema associado à questão dada através da informação positiva constante do programa a que se refere a Figura 2.3. Segundo, porque a partir da segunda cláusula do predicado *demo* é possível atribuir o valor de verdade *falso* ao termo *canto(canário,chilro)* (por não ser possível associar-lhe o valor de verdade `verdadeiro`, em virtude do *PMF* para o predicado *canto* do programa dado pela Figura 2.3).

Já no que diz respeito à informação da Tabela 2.2 e ao programa da Figura 2.4, onde se trata de valores nulos do tipo desconhecido a colocação da questão (2.5) levará aos mesmos resultados pelas razões já explicitadas, enquanto que a questão:

$$\neg demo(canto(canário, fado), Resposta) \quad (2.6)$$

analisada através de um procedimento em tudo análogo ao utilizado no caso anterior, se pode concluir que a resposta para *Resposta* é o valor de verdade *desconhecido*. Não é possível desenvolver uma prova para o cálculo de valor de verdade do termo *canto(canário,fado),* fazendo uso da primeira cláusula do programa dado na Figura 2.9, por inexistência de informação credível ao nível da base de conhecimento. Através da segunda cláusula do mesmo programa é cancelada a prova a partir do momento em que se identifica uma excepção que corresponde ao facto de o *fado* ser cantado por uma *ave rara* que não se conhece. A utilização da terceira cláusula resulta na obtenção do valor de verdade *desconhecido* para *Resposta*, por não ser possível desenvolver uma prova em que o valor de verdade associado a *Resposta* seja *verdadeiro* ou *falso*. Uma situação idêntica ocorre com a questão:

$$\neg demo(canto(X, assobio), Resposta)$$

em que o valor de verdade associado a *Resposta* é a constante lógica *verdadeiro*, a que corresponde a instanciação da variável *X* ao termo *canário*, e o valor de verdade *falso*, nas restantes situações, uma vez que não existe qualquer excepção associada a quem tem como canto o *assobio.*

Avançando para o caso de valores nulos do tipo desconhecido, mas de um conjunto determinado de valores, concretizados pelo programa da Figura 2.6, tem-se que tanto para a questão:

$$\neg demo(canto(pinguim, \underline{ópera}), Resposta)$$

como para a questão:

$\neg demo(canto(pinguim, clássica), Resposta)$

o valor de verdade associado a *Resposta* é dado pela constante lógica *desconhecido*; i.e., os teoremas associados às questões que estão a ser objecto de estudo não se enquadram na prova através da primeira cláusula do programa dado pela Figura 2.9 por não existir informação positiva que a concretize; não se aplica a segunda cláusula do mesmo programa por existirem excepções que o impedem; resta a aplicação da terceira cláusula, que determina que o valor de verdade a atribuir a *Resposta* seja dado pela constante lógica *desconhecido*.

Já para a questão:

$\neg demo(canto(pinguim, assobio), Resposta)$

o valor de verdade a atribuir a *Resposta* será dado pela constante lógica *falso*, já que as excepções não tratam este caso, pelo que se torna possível obter uma prova a partir da segunda cláusula do predicado *demo*. Uma situação semelhante se passa com a questão:

$\neg demo(canto(periquito, ópera), Resposta)$

o valor de verdade a atribuir a *Resposta* é agora dado pela constante lógica *falso*, por motivos idênticos aos enunciados para o caso anterior.

No que concerne aos valores nulos do tipo não permitido tem-se que, em termos de uma visão estática do conhecimento constante da base de conhecimento, segue-se um procedimento em tudo análogo ao aplicado aos valores do tipo desconhecido. Considere-se, por exemplo, a questão:

$\neg demo(canto(canário, hino), Resposta)$

questão esta que tem um tipo de tratamento igual ao dispensado para a questão referida em (2.6) uma vez que o *hino* é cantado por uma ave desconhecida, de tal modo que a *Resposta* é atribuído o valor de verdade *desconhecido*.

Apenas em face de problemas de assimilação de conhecimento é que se justifica a utilização deste tipo de valor nulo, já que a não permissão de se aceder à informação se refere, acima de tudo, a eventuais alterações à base de conhecimento.

A temática da assimilação de conhecimento é tratada com bastante detalhe em [Kowalski 1990] e [Kowalski 1995].

2.3 Conclusões

Neste capítulo é feita uma abordagem à problemática dos Sistemas Multiagente bem como a sistemas de representação de conhecimento e raciocínio, que contextualizam situações em que não existe informação completa acerca do problema em equação [Neves et al.1997].

Faz-se uma abordagem, em particular, à *PL* como ferramenta para a representação de conhecimento, tornando-se necessário considerar a introdução de uma extensão a esta forma de representação, de modo a ultrapassar a problemática do tratamento de informação incompleta.

A extensão à *PL* faz-se pela introdução, na linguagem, de dois tipos de negação: a negação por falha na prova, denotada pelo termo *não*, e a negação forte ou clássica, denotada por '$\neg$'. Neste contexto, passa a ser possível distinguir situações que são falsas de situações para as quais não existe prova de que sejam verdadeiras.

É tratado, ainda, o problema do *PMF* na *PLE*, abordando-se possíveis senãos que a formalização do conceito do *PMF* possa trazer à extensão de predicados de um programa em lógica estendido.

De seguida faz-se uma abordagem a diferentes tipos de valores que podem estar presentes em sistemas onde se verifiquem situações de informação incompleta, designados por valores nulos. São considerados três tipos de valores nulos: do tipo desconhecido, do tipo desconhecido mas de um conjunto determinado de valores e, do tipo não permitido.

Para os valores nulos apresentados, desenvolve-se um programa em lógica estendido dado pela Figura 2.8 que permite o tratamento desses valores nos termos já enunciados.

Por fim, apresenta-se o desenvolvimento de um meta-predicado, dado pela Figura 2.9 para a manipulação de programas como os mencionados ao longo do capítulo, tendo-se concretizado a sua utilização através do tratamento de situações introduzidas durante a apresentação dos diversos tipos de valores nulos.

Capítulo 3
Sistemas de Aprendizagem Baseados em Redes Neuronais Artificiais

Descrevem-se as Redes Neuronais Artificiais, suas características e potencialidades.

As *Redes Neuronais Artificiais (RNAs),* também designadas por *sistemas conexionistas,* comportam-se como modelos simplificados do sistema nervoso central do ser humano. Trata-se de uma estrutura extremamente interligada de *unidades computacionais*, frequentemente designadas por *neurónios ou nodos,* com capacidade de aprendizagem.

3.1 Redes Neuronais Artificiais

É dada a seguir uma definição de uma *RNA* vista como uma máquina adaptativa [Gallant 1993]:

> Uma *RNA* corporiza uma máquina computacional eminentemente paralela, composta por simples unidades de processamento, com uma propensão natural

para armazenar conhecimento empírico e torná-lo acessível ao utilizador. O seu comportamento assemelha-se ao do cérebro humano em dois aspectos:

- O conhecimento é adquirido a partir do ambiente que materializa o universo de discurso, através de um processo de aprendizagem; e
- O conhecimento é armazenado nas *conexões*, também designadas por *ligações* ou *sinapses,* entre nodos.

Durante o processo de aprendizagem, dado por um *algoritmo de aprendizagem* ou *de treino*, a força (ou *peso*) das conexões é ajustada de forma a se atingir um dado estado entálpico ou estado de conhecimento da rede. Embora seja esta a forma tradicional de construir *RNAs,* também é possível modificar a sua própria estrutura interna (ou *topologia),* à imagem e semelhança do que se passa no cérebro humano, onde neurónios podem morrer e novas sinapses (e mesmo neurónios) se podem desenvolver.

3.1.1 Inspiração Biológica: O Cérebro Humano

A maior parte da investigação em *RNAs* foi inspirada e influenciada pelos sistemas nervosos dos seres vivos, em particular do ser humano. É possível que as *RNAs* ofereçam uma das abordagens mais promissoras para a construção de verdadeiros sistemas inteligentes, tendo capacidade para ultrapassar a explosão combinatória associada à computação simbólica baseada em arquitecturas de *von Neumann*[3] [Neves & Cortez 1998].

[3] John von Neumann (1903-1957), matemático húngaro-americano que teve uma grande contribuição na definição da arquitectura de máquinas sequenciais, onde um programa é armazenado na mesma memória de dados que o programa utiliza. Hoje em dia, quase todos computadores são do tipo von Neumann.

O sistema nervoso central fornece uma forte base de sustentação a esta tese. O *cérebro* é uma estrutura altamente complexa, não linear e paralela. Possui uma capacidade de organizar os seus blocos constituintes, conhecidos por *neurónios,* de modo a executarem certas tarefas complexas (e.g., reconhecimento de padrões ou voz), de uma forma inatingível pelo computador mais sofisticado até hoje concebido.

Apesar dos grandes avanços científicos, o conhecimento acerca do modo como o cérebro humano opera está longe de estar completo. No entanto, alguns factos importantes são já conhecidos. Quando alguém nasce, o seu cérebro apresenta-se já com uma estrutura fortemente conexionista, com capacidade de aprender através da *experiência.* Este conhecimento evolui em função do tempo, embora o desenvolvimento seja mais acentuado nos primeiros dois anos de vida. Em termos de velocidade de processamento, um neurónio é cerca de 5 a 6 vezes mais lento do que uma porta lógica de silício. Todavia, o cérebro ultrapassa essa lentidão no processamento utilizando uma estrutura maciçamente paralela. Estima-se que o *córtex* humano possui cerca de 10 biliões de neurónios e 60 triliões de sinapses [Gallant 1993].

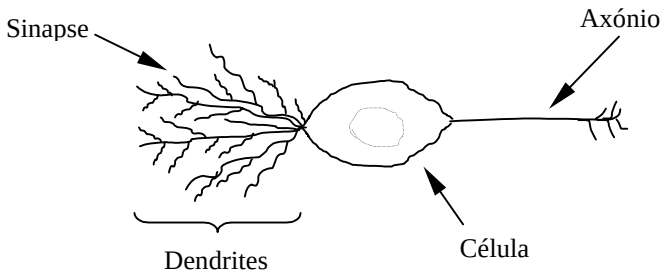


Figura 3.1 – Estrutura de um neurónio natural

Um neurónio é uma célula complexa que responde a sinais electroquímicos, sendo composto por um *núcleo,* por um *corpo celular,* por um numeroso conjunto de *dendrites* (i.e., *entidades* que recebem sinais de outros neurónios via sinapses) e, por um *axónio,* que transmite um estímulo a outros neurónios, através das já referidas sinapses (Figura 3.1) [Bose & Liang 1996]. Um único neurónio pode estar ligado a centenas ou mesmo dezenas de milhares de neurónios. Num cérebro existem estruturas anatómicas de pequena, média e alta complexidade, com diferentes funções, sendo possíveis parcerias. Os neurónios tendem a agrupar-se em camadas, sendo três os principais tipos de conexões (Figura 3.2), e definidos nos seguintes termos:

- *Divergentes*, onde as conexões se desenvolvem a partir de um neurónio que pode estar ligado a outros neurónios via uma arborização do axónio;
- *Convergentes*, onde as conexões têm em linha de conta vários neurónios que podem estar ligados a um único neurónio; e
- *Encadeadas ou Cíclicas*, onde as conexões podem vir a envolver vários neurónios e a formarem ciclos.

Por sua vez, as conexões são constituídas por dois tipos de sinapses: *inibitórias* e *excitatórias.*

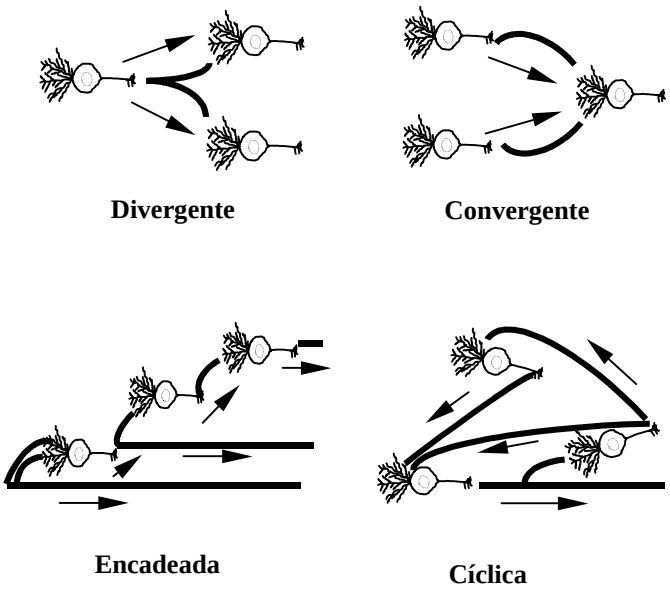


Figura 3.2 – Os diferentes tipos de conexões

3.1.2 Benefícios das *RNAs*

O poder computacional de uma *RNA* alicerça-se em dois aspectos fundamentais: numa topologia que premeia o paralelismo e, por outro lado, na sua capacidade de aprendizagem e generalização; i.e., conseguir responder adequadamente a novas situações com base em experiências passadas. São estas duas características que tornam possível a resolução de problemas, que de outra forma seriam inatingíveis. Isto não quer dizer que as *RNAs* sejam caixas mágicas que consigam por si dar resposta a qualquer problema. Em vez disso precisam, não raras vezes, de serem integradas com outros sistemas ou paradigmas computacionais para a resolução de problemas. Convém, ainda, reconhecer que ainda se está muito longe de atingir uma arquitectura que mimetize o cérebro humano [Gallant 1993]. As *RNAs* apresentam

características únicas que não se encontram em outros mecanismos ou técnicas computacionais associadas à resolução de problemas [Neves & Cortez 1998] [Azoff 1994] [Gallant 1993], algumas das quais são mencionadas no texto que se segue:

- *Aprendizagem e generalização*; i.e., conseguindo descrever o todo a partir de algumas partes, constituindo-se como formas eficientes de aprendizagem e armazenamento de conhecimento;
- *Processamento maciçamente paralelo*; i.e., permitindo que tarefas complexas sejam realizadas num curto espaço de tempo;
- *Transparência*; i.e., podendo ser vistas como uma caixa negra que transforma vectores de entrada em vectores de saída, via uma função de transformação em princípio desconhecida;
- *Não linearidade*; i.e., atendendo a que muitos dos problemas reais a equacionar e resolver são de natureza não linear;
- *Adaptabilidade*; i.e., podendo adaptar a sua topologia de acordo com mudanças do ambiente de trabalho;
- *Resposta evidencial*; i.e., onde uma saída da rede traduz não só um processo de decisão mas também o grau de confiança a conferir a esta;
- *Robustez e degradação suave*; i.e., permitindo processar o ruído ou informação incompleta de forma eficiente, assim como sendo capazes de manter o seu desempenho quando há desactivação de algumas das suas conexões e/ou nodos; e
- *Flexibilidade*; i.e., actuando com uma grande margem de aplicabilidade.

Finalmente, convém referir que apesar do facto das *RNAs* partilharem muitas das características do cérebro humano, são carentes relativamente a outras, como seja o *esquecimento.*

3.1.3 Neurónio Artificial ou Nodo

Um *nodo,* termo usado para distinguir entre um neurónio natural e um artificial, é a unidade de processamento chave para a operação de uma *RNA.* Embora existam diversos tipos de nodos, em princípio, cada nodo comporta-se como um comparador que produz uma saída quando o efeito cumulativo das entradas excede um dado valor limite. Um nodo pode ser dissecado de acordo com o exposto na Figura 3.3 e definido em termos de:

- Um *conjunto de conexões* (w_{ij})*,* cada uma etiquetada por um *peso;* i.e., um número real ou binário, que corporiza um efeito excitatório para valores positivos e inibitório para valores negativos da etiqueta. Assim, o *sinal* ou *estímulo* (x_j) como entrada da conexão é multiplicado pelo correspondente peso w_{ij}, onde *i* denota o nodo objecto de estudo e *j* o nodo de onde partiu o sinal. Pode ainda existir uma conexão extra, denominada de *bias,* cuja entrada toma o valor *+1*, que estabelece uma certa tendência ou inclinação no processo computacional; i.e., adiciona uma constante (w_{i0}) para que se estabeleçam as correctas condições operacionais para o nodo;

- Um *integrador* (*g*)*,* que reduz os *n* argumentos de entrada (estímulos) a um único valor. Frequentemente é utilizada a função *adição* (Σ), pesando todas as entradas e combinando-as linearmente; e

- Uma *função de activação* (*f*)*,* que condiciona o sinal de saída, introduzindo uma componente de não linearidade no processo computacional.

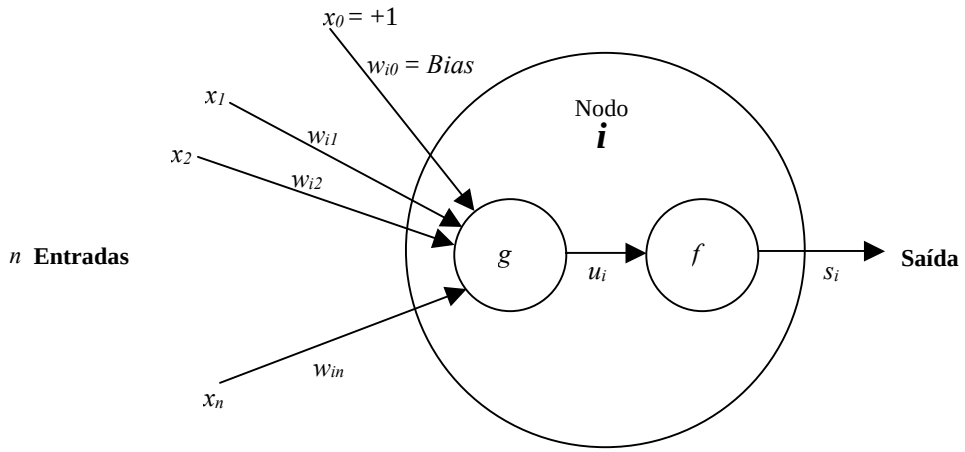


Figura 3.3 – Estrutura geral de um nodo de uma *RNA*

Dado um nodo *i*, o seu comportamento é descrito pelas equações (3.1) e (3.2), que constam do texto que se segue:

$$u_i = g(1 \times w_{io}, x_1 \times w_{i1}, x_2 \times w_{i2}, ..., x_n \times w_{in}) \quad (3.1)$$

$$s_i = f(u_i) \quad (3.2)$$

para um nodo *i* com *n* entradas e uma saída, onde u_i representa o ganho do nodo *i* e s_i a *saída* do nodo.

Tabela 3.1 – Funções de activação

Nome	Função *f*	Contradomínio
limiar	$\begin{cases} 1 , u_i \geq 0 \\ 0 , u_i < 0 \end{cases}$	{ 0, 1 }
linear	u_i	] -∞, +∞ [

Nome	Função f	Contradomínio
por troços	$\begin{cases} 1\ ,\ u_i \geq 0{,}5 \\ ku_i\ ,\ -0{,}5 < u_i < 0{,}5 \\ 0\ ,\ u_i \leq -0{,}5 \end{cases}$	[0, 1]
logística ou sigmoid	$\frac{1}{1+\exp(-ku_i)}$	[0, 1]
tangente hiperbólica	$\tanh(ku_i)$	[-1, 1]
Sen	$\text{sen}(u_i \ mod\ 2\Pi)$	[-1, 1]
Cos	$\cos(u_i \ mod\ 2\Pi)$	[-1, 1]
gaussiana	$\exp\left(\frac{-u_i^2}{2k^2}\right)$	[-1, 1]
Quadrada	$-sign(u_i)u_i^2$	] -∞, +∞ [

A Tabela 3.1 apresenta algumas das funções de activação mais utilizadas com *RNAs*, onde *k* denota a *inclinação da* função, *mod o* resto de uma divisão inteira e *sign(x) = x/|x|* [Azoff 1994] (Figura 3.4).

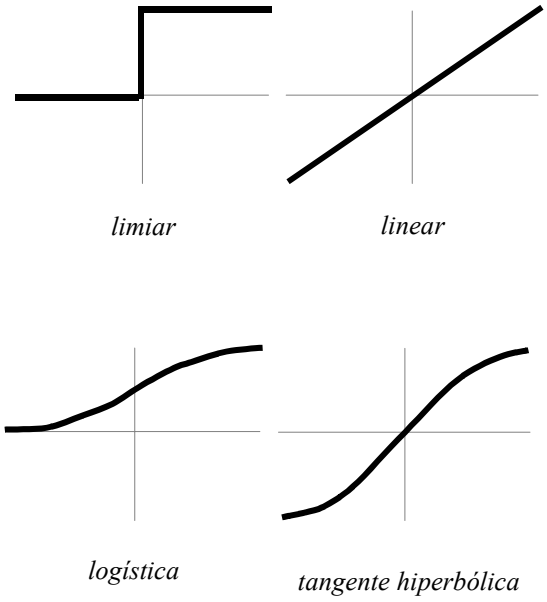


Figura 3.4 – Funções de activação usualmente utilizadas em *RNAs*.

A primeira função, também designada por função de *heaviside*, é normalmente utilizada em nodos do tipo *McCullochPitts* [LeCun 1993], em que a saída toma o valor *+1* apenas se o ganho for não-negativo, de acordo com uma filosofia do *tudo-ou-nada. Em* seguida, aparecem duas outras funções lineares, com a última a ter como contradomínio o intervalo [*0,1*]. A não linearidade é definida em termos das restantes funções, onde uma especial atenção deve ser dada à função logística, também conhecida por função *sigmoid*. Esta função, cuja forma é modelada a um *S*, é de longe a função mais utilizada no uso de *RNAs.* É uma função crescente que exibe um balanceamento gracioso entre um comportamento linear e não linear (Figura 3.5) [Hopfield 1982]. Ao se variar a *inclinação* (*k*) obtêm-se funções com diferentes declives; no limite, em que *k* se aproxima do infinito, a função tende para a função *limiar*.

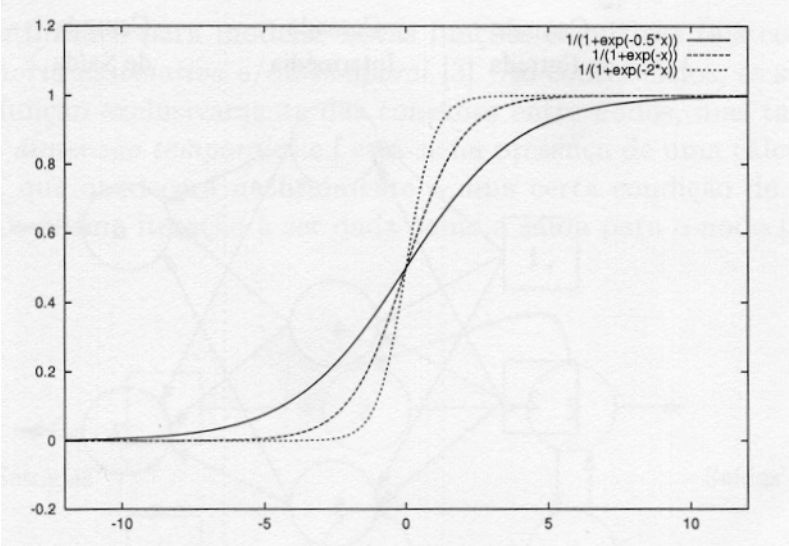


Figura 3.5 – A função *logística* (*sigmoid*) para inclinações de $k = 0.5$, $k = 1.0$ e $k = 2.0$

3.1.4 Arquitecturas de Rede

A forma como os nodos se interligam numa estrutura de rede é denominada por *arquitectura* ou *topologia.* Existem inúmeros tipos de arquitecturas de *RNAs,* cada uma com as suas próprias potencialidades. Em geral, caem dentro de três categorias [Gallant 1993], que são descritas no texto que se segue:

- *Redes Feedforward de uma Só Camada (RFSC)* – Uma *RNA feedforward* pode ser organizada por camadas, pois não existem ciclos, dado que as conexões são sempre unidireccionais (*convergentes* ou *divergentes*). Na sua forma mais simples, uma rede é composta por uma *camada de entrada,* cujos valores de saída são fixados externamente, e por uma *camada de saída* (Figura 3.6). De referir que a camada de entrada não é contabilizada como camada numa *RNA* dado o facto de nesta não se efectuarem quaisquer formas de cálculo;

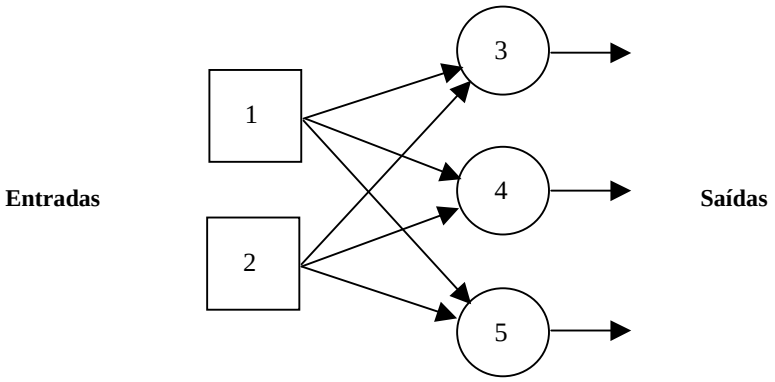


Figura 3.6 – Arquitectura de uma *RFSC*

- *Redes Feedforward MultiCamada (RFMC)* – A segunda classe de redes *feedforward* distingue-se pelo facto de possuir uma ou mais *camadas intermédias,* cujos nodos são designados por *nodos intermédios* (Figura 3.7), tendo como função intervir de forma útil entre a entrada e a saída da rede. Ao se acrescentarem camadas intermédias está-se a aumentar a capacidade da rede em modelar funções de maior complexidade, uma particularidade bastante útil quando o número de nodos na camada de entrada é elevado. Por outro lado, este aumento também transporta um senão, uma vez que o tempo de aprendizagem aumenta de forma exponencial; e

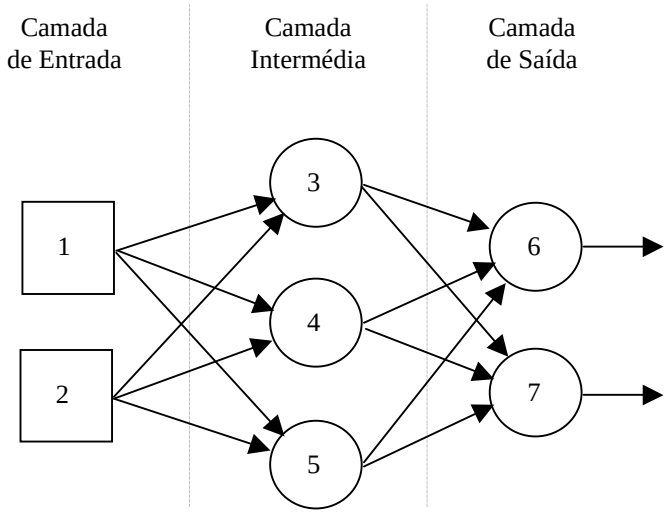


Figura 3.7 – Arquitectura de uma *RFMC*

- *Redes Recorrentes* (*RR*) – A recorrência existe em sistemas dinâmicos quando uma saída de um elemento influência de algum modo a entrada para esse mesmo elemento, criando-se assim um ou mais circuitos fechados tal como é apresentado na Figura 3.8. Assim que uma ou mais conexões cíclicas são incluídas numa rede, esta passa a ter um comportamento não linear, de natureza espacial e/ou temporal, que podem ser utilizados para modelar novas funções cognitivas, tais como as de memória associativa e/ou temporal [Bose & Liang 1996]. Ao conter ciclos, as saídas não são função exclusivamente das ligações entre nodos, mas também de uma dimensão temporal; i.e., está-se na presença de uma cálculo recursivo, que obedecerá naturalmente a uma certa condição de paragem, com a última iteração a ser dada como a saída para o nodo [Riedmiller & Braun 1993].

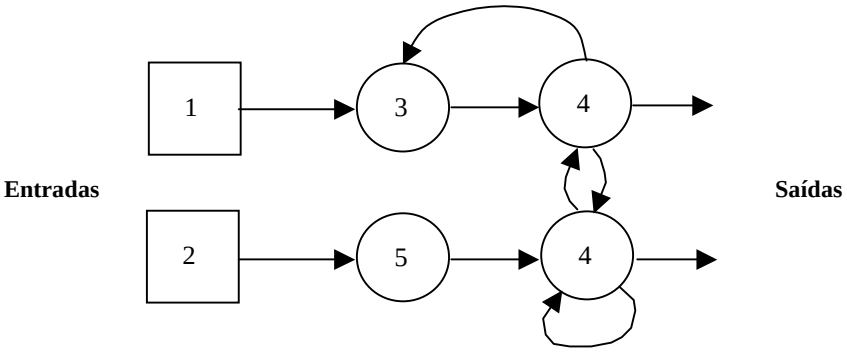


Figura 3.8 – Arquitectura de uma *RR*

3.1.5 Aprendizagem

Como já foi referido, uma propriedade importante das *RNAs* é a sua capacidade para aprender a partir da interacção com o seu ambiente ou universo de discurso. A aprendizagem de uma *RNA* envolve a sequência de eventos dada a seguir [Gallant 1993]:

- A *RNA* é estimulada por informação que emana de um dado ambiente ou universo de discurso em que se desenvolve;
- Certos parâmetros livres, normalmente os pesos das conexões, são alterados em resultado desse estímulo; e
- A *RNA* responde de uma nova forma a estímulos que emanem do ambiente em virtude das alterações sofridas pela sua estrutura interna.

Esta aprendizagem é executada a partir de um conjunto bem conhecido de *regras,* chamado de *algoritmo de aprendizagem* ou *treino.* Outro facto a ter em conta é a

forma como uma *RNA* se relaciona com o seu ambiente. Neste contexto, está-se a falar de um *paradigma;* i.e., o modelo do ambiente em que a rede opera. Assim, os diversos algoritmos de treino distinguem-se pelo paradigma e regras de aprendizagem que utilizam, cada um oferecendo as suas próprias vantagens e, naturalmente, alguns senões.

3.1.5.1 Aprendizagem em *RNAs*

Existem três paradigmas fundamentais a considerar em termos de aprendizagem em *RNAs* [Neves & Cortez 1998], que serão apresentadas no texto que se segue:

- *Supervisionada* – Trata-se de um método bastante popular que passa pela presença de um especialista, ou seja, são fornecidas respostas correctas à rede. A rede aprende a partir de um conjunto de casos (*P*), onde cada *caso* (*p*), também chamado de *exemplo* ou *caso de treino*, é composto por um vector de entrada (x^p) e por um vector de resposta ou saída (s^p). Durante o processo de aprendizagem, é efectuada uma comparação entre o valor desejado (t^p) com o valor de saída da rede, originando um erro ($e^p = t^p \ \theta \ y^p$, sendo θ a função de erro). O erro resultante é utilizado para de alguma forma ajustar os *pesos* da rede. Uma invocação do procedimento de treino é composta por ajustamentos *iterativos* ou em *lote* para todos os casos de treino. A aprendizagem é então conseguida quando, após várias iterações, o erro é reduzido para valores aceitáveis;

- *De Reforço* – Neste tipo de aprendizagem conta-se com a presença de especialistas acerca do universo de discurso, embora a resposta correcta não seja apresentada à rede; i.e., apenas se fornece uma indicação sobre se a resposta da rede está correcta ou errada, tendo a rede de usar essa informação para melhorar o seu desempenho. Em princípio, um prémio é

dado em termos do reforço dos pesos das conexões que contribuem para uma resposta correcta e, uma penalidade para a situação em contrário; e

- *Não Supervisionada* – Trata-se de uma abordagem diferente das anteriores no que concerne ao treino da *RNA*, onde não é fornecida ao sistema qualquer indicação, partindo do exterior, acerca da resposta correcta. Por conseguinte, a aprendizagem da *RNA* é feita pela descoberta de similaridades nos dados de entrada, que são agrupados segundo regularidades estatísticas ou segundo padrões de comportamento, de acordo com os casos de treino. Como caso típico deste tipo de aprendizagem têm-se as redes de *Kohonen* [Kohavi 1995].

3.1.5.2 Regras de Aprendizagem para *RNAs*

As regras de aprendizagem podem-se dividir em cinco tipos básicos [Gallant 1993], de acordo com o que é postulado no texto que se segue.

- *Hebbian* – Trata-se do mais antigo e porventura do mais conhecido processo de aprendizagem, proposto por Hebb em 1949 e exposto nos termos: "Quando um axónio da célula *A* está demasiado próximo para excitar uma célula *B* e influencia esta de forma repetitiva ou persistente, então algum processo metabólico ocorre numa ou ambas as células de forma a aumentar a eficiência da célula *A* em influenciar *B*". Este axioma, constituído num contexto neurobiológico, foi expandido, traduzindo-se em dois procedimentos (ou regras) de aprendizagem, com o seguinte teor:

  – Se dois nodos em cada lado de uma ligação são activados simultaneamente, então a *força* dessa ligação é progressivamente aumentada; e

– Se dois nodos em cada lado de uma ligação são activados de forma assíncrona, então a ligação é progressivamente enfraquecida ou eliminada.

Esta ligação, designada por *sinapse de Hebbian,* é função do tempo e da sua localização espacial. Estas regras são particularmente utilizadas na aprendizagem não supervisionada, (e.g., segundo *Hopfield* – redes *autoassociativas* [Haykin 1999] ou em *Memória Associativa Bidireccional – redes heteroassociativas* [Kohonen 1982]);

- *Competitiva* – Como o nome indica, as saídas dos nodos que respeitam a uma mesma camada competem entre si para pontuarem, com apenas um nodo a ser activado num dado instante, segundo a estratégia do *vencedor-toma-tudo.* No início do processo, os nodos de uma camada possuem ligações com pesos pequenos mas desiguais. Quando um padrão é fornecido à rede, um dos elementos da camada responderá melhor do que os outros, sendo por isso *premiado* com um reforço dos pesos das suas ligações. Em alguns casos, os pesos dos nodos vizinhos também podem ser reforçados. Esta regra é bastante apropriada para a *extracção de características,* sendo utilizada em problemas de *classificação*. As redes *SOM* (do inglês *Self-Organizing Maps)* como as *Kohonen* [Kohavi 1995] e *ART* [Carpinter & Grossberg 1988] utilizam este tipo de aprendizagem;

- *Estocástica* – Neste tipo de aprendizagem, os pesos são ajustados de um modo casuístico. Como exemplos temos o *Simulated Annealing* aplicado a máquinas de *Cauchy* e *Boltzmann* onde os estados dos nodos são função de uma distribuição de probabilidade [Riedmiller & Braun 1993];

- *Baseada na Memória* – Esta forma de aprendizagem passa pela consideração de todas (ou quase todas) as experiências passadas, que são explicitamente armazenadas na forma de pares entrada-saída. Quando surge um novo vector, x^p, nunca antes alimentado à rede, o sistema responde pela procura de vectores na região vizinha de x^p. Normalmente, este procedimento é dado na forma:

  - Um critério de definição da vizinhança local do vector x^p; e
  - Uma regra de aprendizagem aplicada aos casos de treino da vizinhança local de x^p.

  Como exemplo deste tipo de aprendizagem têm-se as redes *Radial-Basis Functions (RBF)* [Broomhead & Lowe 1988]; e

- *Gradiente Descendente* – Como foi anteriormente referido, na aprendizagem supervisionada pretende-se reduzir o erro entre o valor pretendido e o valor de saída da rede. O *erro* actua assim como um mecanismo de controlo, onde os sucessivos ajustamentos deste vão originar melhores respostas. O objectivo é minimizar uma função de custo, ξ, definida em termos do sinal de erro e^p. A regra de aprendizagem, conhecida como a regra *delta* ou de *Widrow-Hoff,* baseia-se na resolução da seguinte equação [Riedmiller 1994b]:

$$\Delta w = \eta \nabla \xi \tag{3.3}$$

  onde η denota uma constante positiva, chamada de *taxa de aprendizagem*, e $\nabla \xi$ o *gradiente* da função de custo. Trata-se de uma aprendizagem deveras utilizada e conhecida, associada a *RMC* e ao popular algoritmo de *Backpropagation* [Rojas 1996].

3.1.6 Tarefas de Aprendizagem em *RNAs*

A escolha da arquitectura e do método de aprendizagem é influenciada pela tarefa de aprendizagem a ser desempenhada pela *RNA,* sendo possível identificar diversas vias alternativas [Neves & Cortez 1998] [Gallant 1993]:

- *Memória Associativa* – Trata-se de *RNAs* que implementam uma forma de memória distribuída, que aprende por *associação*. Esta pode tomar duas formas: *autoassociação* ou *heteroassociação*. No primeiro caso, a rede armazena um conjunto de padrões de comportamento respeitantes ao funcionamento de um qualquer sistema. Mais tarde, é apresentado à rede uma versão distorcida do(s) padrão(ões) original(ais), pelo que a rede deve ser capaz de devolver o padrão ou padrões original(ais). Na segunda situação, a diferença reside no facto dos padrões de comportamento do sistema que está a ser objecto de estudo estarem organizados por pares de entradas e saídas. Aqui, está-se na presença de uma aprendizagem supervisionada, ao contrário da autoassociação que usa uma aprendizagem não supervisionada;

- *Diagnóstico* – Esta é uma tarefa deveras comum em áreas distintas como, por exemplo, a *Medicina* ou a *Engenharia*. Trata-se essencialmente de uma tarefa de *classificação*; i.e., exige uma correcta associação entre entradas, que representam dados indicadores de um estado do sistema que está a ser objecto de estudo (e.g., sintomas a que se podem associar diversas patologias ou comportamento anormal de uma máquina ou conjuntos de máquinas), com o correspondente diagnóstico (e.g., doença ou falha do equipamento). De uma forma geral, as *RNAs* são integradas em sistemas periciais complexos, envolvendo outras formas de aprendizagem (e.g., *aprendizagem* a partir de regras);

- *Reconhecimento de Padrões* – Formalmente, esta tarefa define-se como o processo pelo qual um sinal/padrão recebido pela *RNA* é atribuído a uma de diversas categorias possíveis. Em primeiro lugar, é necessário treinar uma rede, onde os sinais/padrões, associados à respectiva categoria, são alimentados à rede de forma repetitiva. Mais tarde, um novo padrão é alimentado por sua vez à rede, que terá de ser capaz de identificar a categoria em que esta se inclui, de acordo com a informação constante da *RNA*. De uma forma geral, o reconhecimento de padrões por *RNAs* pode tomar duas formas. Na forma mais simples, utiliza-se uma única *RFMC* com recurso a um algoritmo de aprendizagem supervisionado, tendo os nodos intermédios da rede a função de *extracção de características* porventura implícitas, nos dados constantes do vector de entrada para a *RNA*; i.e., uma transformação da entrada (x) num ponto intermédio (y), com uma dimensão inferior. Esta redução facilita a tarefa de *classificação*, descrita como uma transformação do ponto intermédio (y) em uma das classes possíveis num espaço de decisão r-dimensional, para r classes distintas. Na segunda forma, a máquina de aprendizagem é dada por duas partes constituintes, a primeira para a *extracção* de *características*, via uma rede não supervisionada, sendo a última para a *classificação*, via uma rede supervisionada. Importa referir ainda que as *RNAs* são bastante eficazes na aprendizagem de tarefas preceptivas como o *processamento de imagem*, o *reconhecimento de voz* ou o *reconhecimento da escrita*, ultrapassando em desempenho os métodos estatísticos convencionais (e.g., *classificadores Bayesianos*);

- *Regressão/Previsão* – O objectivo é conceber uma *RNA* capaz de modelar uma função desconhecida $f(\cdot)$ que se aproxime da função $F(\cdot)$ dada por um conjunto de vectores etiquetados; i.e., compostos por um par entrada-saída

($x \rightarrow y$), de forma que a distância *euclidiana* entre os valores que tomam as funções seja tão pequena quanto possível; i.e.,

$$\forall x, \|F(x) - f(x)\| < \rho \tag{3.4}$$

onde ρ denota um valor infinitamente pequeno. A regressão é uma tarefa perfeita para a aprendizagem supervisionada. A *previsão*, caso particular da regressão, é uma tarefa comum em diversas áreas do conhecimento como sejam a *Economia*, a *Medicina* ou a *Meteorologia.* Aqui pretende-se perspectivar os valores a serem dados por uma certa função com base em valores conhecidos do passado. As *RNAs* têm-se mostrado eficazes como ferramentas de previsão, tanto no que respeita à ocorrência (ou não) de eventos, como ao momento da ocorrência e respectiva intensidade;

- *Controlo* – Esta tarefa envolve um processo ou uma parte crítica de um sistema que tem de ser mantido sob controlo. O principal objectivo do controlador é fornecer os sinais apropriados ao sistema de forma que a saída (y) deste acompanhe uma entrada de referência (x). Como exemplos têm-se o controlo de veículos autónomos, robôs ou centrais de energia nuclear;

- *Optimização* – Existe um variado conjunto de problemas que envolvem a procura de soluções óptimas ou razoavelmente óptimas, tratando simultaneamente vários parâmetros, envolvendo um espaço de procura demasiado extenso, onde a exaustão de todas as alternativas a considerar para a resolução do problema se torna incomportável em termos computacionais (e.g., é o caso dos *problemas do caixeiro viajante* ou o do *escalonamento de tarefas*). A resolução deste tipo de problemas passa pelo uso de heurísticas e/ou máquinas de aprendizagem (incluindo as *RNAs*) que

reduzem o espaço de procura de forma a que se obtenham soluções para o problema passíveis de serem aceites; e

- *Filtragem/Compressão de Dados* – O termo *filtro* usualmente designa um dispositivo ou algoritmo para a extracção de informação de interesse a partir de dados com ruído. Por sua vez, a *compressão* envolve uma redução de um espaço *n*-dimensional para um espaço *m*-dimensional, sendo $n > m$. Ambas as tarefas se tornam particularmente importantes quando existem quantidades enormes de dados a serem processados (e.g., processamento de imagens médicas ou de satélite).

3.1.7 A Inteligência Artificial e as *RNAs*

Os objectivos a atingir com a *Inteligência Artificial* (*IA*) passam pelo desenvolvimento de novos paradigmas e procedimentos para a realização de tarefas de natureza cognitiva, associadas à resolução de problemas, as quais são actualmente melhor executadas pelo homem. Um sistema de *IA* deve subscrever, entre outros, os requisitos [Gallant 1993] (Figura 3.9):

- Armazenar conhecimento; i.e., deve atender a diferentes formas de *representação* do conhecimento;
- Utilizar o conhecimento adquirido para a resolução de problemas; i.e., deve considerar diferentes formas de *raciocínio*; e
- Adquirir novo conhecimento através da experiência; i.e., deve contemplar diferentes formas de *aprendizagem*.

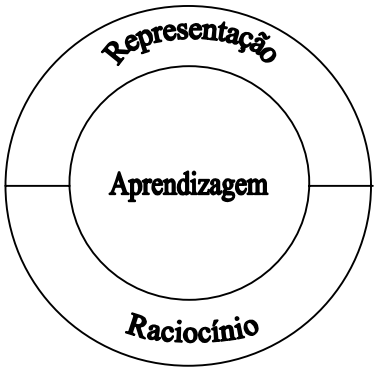


Figura 3.9 – Os três componentes fundamentais de um sistema de *IA*

O primeiro requisito envolve a consideração de diferentes formas de *representação do conhecimento,* de modo a tratar a informação disponível da forma o mais convenientemente possível, ou seja:

- O actual estado de conhecimento do sistema em termos dos objectos e relações entre objectos que são detectados no universo de discurso; e
- Novas observações ou alterações ao universo de discurso (ambiente), sem esquecer os problemas de ruído e de informação incompleta.

Na *IA*, este conhecimento tem vindo a ser objecto de estudo mediante o uso de linguagens *simbólicas,* através da utilização de uma colecção de factos e regras que são utilizadas quer em termos de representação, quer em termos da sua manipulação do conhecimento para a resolução de problemas.

Por outro lado, em termos dos processos abertos para o tratamento da problemática da *aprendizagem* automática, duas diferentes formas de processamento de informação serão consideradas: a *indutiva* e a *dedutiva.* Na primeira forma, padrões de comportamento e regras são obtidas a partir da análise de dados e experiências

passadas (e.g., *aprendizagem baseada em casos*), enquanto que na última, as regras são utilizadas para criar novos factos (e.g., através da *demonstração de teoremas*).

As máquinas de *IA* podem ser comparadas às *RNAs* a três níveis [McCulloch & Pits 1943], nos termos que são referidos no texto que se segue:

- *Explicação* – Nos primórdios da *IA* a ênfase foi colocada na construção de representações simbólicas, como forma de corporizar o estado de conhecimento da mente. Por outro lado, em termos das *RNAs* a ênfase foi colocada no desenvolvimento de modelos de *Processamento Paralelo Distribuído* (*PPD*), em que o processamento é feito num elevado número de unidades computacionais, em que uma explicação neurobiológica para os fenómenos cognitivos é objecto de consideração;

- *Estilo de Processamento* – Na *IA* convencional, a execução de tarefas é sequencial e, mesmo quando não existe uma ordem pré-definida, estas são executadas *passo a passo*. Em contraste, o processamento paralelo está no cerne das *RNAs*, sendo a sua fonte de flexibilidade, robustez e imunidade ao ruído; e

- *Estrutura de Representação* – A *IA* clássica utiliza linguagens do tipo simbólico, onde se compõem expressões complexas a partir da combinação de símbolos elementares, sendo estas expressões que corporizam o estado de conhecimento do sistema. Ao contrário, nas *RNAs* o conhecimento é armazenado na sua estrutura interna, no valor dos pesos das suas ligações, em termos de valores numéricos.

O movimento conexionista surgiu dentro da *IA* como uma reacção aos sistemas de representação simbólicos, considerados demasiado rígidos e especializados; i.e.,

esperava-se que este novo paradigma potenciasse a criação de sistemas de aprendizagem mais flexíveis e universais. Contudo, neste momento, caminha-se para uma integração das potencialidades oferecidas por ambos os paradigmas, em sistemas híbridos, adicionando as múltiplas capacidades das *RNAs*, como a adaptabilidade e robustez, à capacidade de representação, inferência e universalidade da *IA* simbólica [Memmi 1989].

3.1.8 O Movimento Conexionista numa Perspectiva Histórica

A reflexão filosófica sobre o que é a consciência e qual o órgão que a contém, dura desde épocas anteriores a Cristo, com os filósofos gregos a serem dos primeiros a especular sobre a localização física da alma. O conhecimento existente hoje sobre o funcionamento do cérebro é o resultado da investigação feita nos últimos 100 anos. Ramón y Cajal em *1894* [Ramon & Cajal 1990] foi o primeiro a propor uma teoria para o funcionamento do cérebro em termos de unidades constituintes a que denominou de neurónios [Thimm & Fiesler 1997]. Desde então, numerosos foram os progressos obtidos na compreensão do funcionamento do cérebro humano: descreveram-se os axónios, as dendrites, as sinapses e as activações electroquímicas [Riedmiller & Braun 1993].

A era moderna das *RNAs* surgiu com o trabalho pioneiro de McCulloch e Pitts [McCulloch & Pitts 1943], a que se segue o trabalho de Neves e Garrido [Neves & Garrido 1991], descrevendo o cálculo lógico utilizando *RNAs*, procurando unificar os estudos neurobiológicos com a Lógica Matemática. Desde então, um enorme progresso tem sido feito na interligação colaboração de ambas as áreas, com um particular destaque para o ocorrido na década de 80, onde se deu o ressurgimento do interesse em *RNAs* como mecanismos a utilizar na resolução de problemas com contribuições em diversas frentes:

- Grossberg estabeleceu um novo tipo de *SOM* conhecido como *ART* (do inglês *Adaptive Resonance Theory*) [Carpinter & Grossberg 1988];
- Em 1982, Hopfield desenvolve um novo tipo de *RNAs* baseado em *RR* com ligações simétricas. No mesmo ano surgem as redes *SOM* do tipo *Kohonen* [Kohavi 1995];
- O famoso algoritmo de *Back-Propagation* surgiu em 1986 pelo trabalho de Rumelhart, Hinton e Williams [Rumelhart et al. 1986], tornando-se no algoritmo de treino mais popular para *RMC*;
- Em 1988, Broomhead e Lowe [Broomhead & Lowe 1988] descrevem as redes *RBF*, fornecendo assim uma alternativa às *RMC*; e
- No início dos anos 90, Vapnik e seus colaboradores apresentam uma poderosa classe de redes supervisionadas, designadas de *Support Vector Machines*, para a regressão e o reconhecimento de padrões [Boser et al. 1992].

Hoje em dia procuram-se não só redes mais eficientes como também melhores algoritmos de treino [Sarle 1999]. Por outro lado, espera-se que a aplicação de *RNAs* a outras áreas do conhecimento se generalize, seja à Medicina, à Economia, ao Processamento de Sinal, à Visão por Computador, à Robótica, à Automação ou aos Sistemas Periciais, para além da Estatística [Jordan 1995] [Alves et al. 2001a] [Alves et al. 2001b]. Neste último domínio tem-se a estimação não paramétrica, a aprendizagem por agentes económicos e a modelação de séries temporais [Neves & Cortez 1998] [Sarle 1999] [Minsky 1990].

3.2 Redes MultiCamada com Aprendizagem

3.2.1 A Arquitectura *Feedforward* MultiCamada

As *Redes Feedforward MultiCamada* (*RFMC*), também conhecidas por *Redes Perceptrão MultiCamada,* constituem uma das mais importantes e populares classes de *RNAs,* com um vasto leque de aplicabilidade, utilizáveis em problemas de *memória associativa, classificação, reconhecimento de padrões, optimização* e *regressão* [Neves & Cortez 1998].

De um modo geral, as *RFMC* são definidas pelo tuplo (*E, I, S, C, F*) [Riedmiller & Braun 1993] [Faraday & Chatfield 1998] (Figura 3.10):

- Um conjunto de *nodos de entrada* (*E*), catalisadores dos estímulos à rede provenientes do ambiente;
- Um conjunto de *nodos intermédios* (*I*), unidades internas de processamento que aumentam a capacidade de aprendizagem de tarefas complexas por parte da rede, pela extracção progressiva de mais características do ambiente ou universo de discurso presentes nos dados alimentados à rede;
- Um conjunto de *nodos de saída* (*S*), que devolvem a resposta da rede;
- Um conjunto de *conexões unidireccionais* (*C*), definidos pelo tuplo (*i, j, w*) ou abreviadamente w_{ij}, em que $i \in I \cup S, j \in E \cup I, j < i$ e $w \in \Re$; e
- Um conjunto de *funções de activação* (*F*), normalmente do tipo não linear e diferenciáveis, sendo a função logística (ou sigmoid) uma das mais utilizadas (Tabela 3.1).

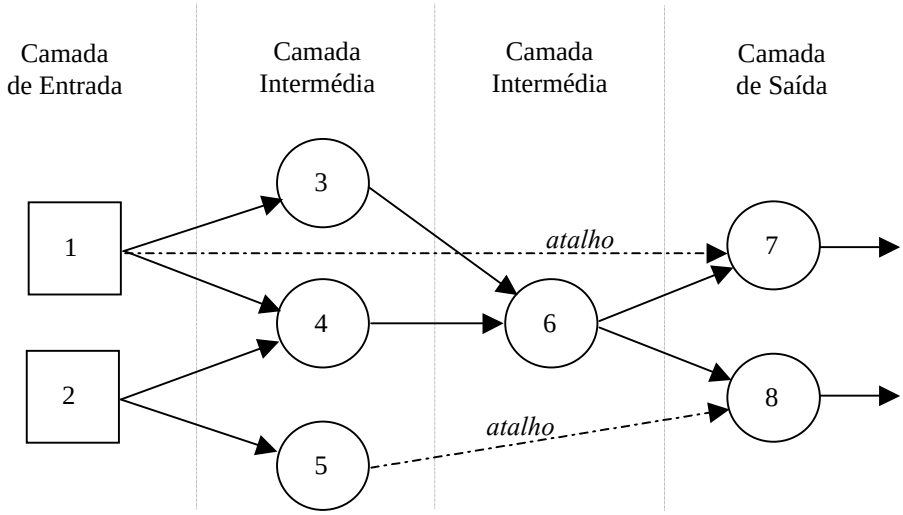


Figura 3.10 – Estrutura de uma *RFMC* com a topologia *2-3-1-2*

O sinal de entrada propaga-se para os nodos a jusante, através da rede, camada a camada, não existindo ciclos. Nas redes multicamada tem-se que as unidades intermédias (*I*) se dividem em *c* subconjuntos, ou camadas, $(I_1,I_2,...,I_c)$. É comum representar as camadas na forma E - I_1 - . . . - S.

Uma *RNA* designa-se *por completamente interligada* quando são levadas à prática todas as ligações possíveis entre os nodos de duas camadas adjacentes (Figura 3.7). Caso contrário, a rede designa-se por *parcialmente interligada.* Por vezes existem ligações directas entre as entradas e os neurónios de saída ou que "saltam" camadas (Figura 3.10). Estas conexões designam-se por *atalhos* [Patterson 1996].

A não linearidade, a existência de nodos intermédios e o seu alto grau de conectividade tornam a arquitectura *RFMC* deveras poderosa para uma máquina de aprendizagem. Por outro lado, são estas mesmas características que dificultam uma análise formalmente sustentada ao processo de aprendizagem [Gallant 1993].

3.2.2 O Algoritmo de *Backpropagation*

Dentro dos algoritmos supervisionados, o mais popular e mais usado em aprendizagem em *RNAs* é o algoritmo de *Backpropagation* (*BP*) ou então os seus derivados [Faraday & Chatfield 1998]. Este algoritmo prevalece como um marco para a comunidade das *RNAs,* já que constitui um método eficiente de computação para o treino de *RFMCs,* procurando o mínimo da função de erro no espaço de procura dos pesos das ligações entre nodos, baseando-se em métodos de gradiente descendente. A combinação de pesos que minimiza a função de erro é considerada a solução para o problema de aprendizagem. Dado que este método exige o cálculo do gradiente, torna-se necessário que a função de erro (ξ) seja contínua e diferenciável, o que acontece quando se usam funções de activação diferenciáveis [Riedmiller & Braun 1993]. O algoritmo de *BP* corre a dois passos [Bose & Liang 1996] [Riedmiller 1994b]:

- *Em frente,* onde o vector de entrada (x^p) é fornecido aos nodos de entrada, propagando-se para jusante, camada a camada, de acordo com as equações 3.1 e 3.2 em que o integrador (g) utiliza a função adição ($\sum$) ;i.e.,

$$u_i = \sum_j s_j w_{ij} \qquad (3.5)$$

em que,

$s_o = 1$ (valor de entrada da conexão de *bias*),

$$s_1 = x_1^p, \ldots, \; s_E = x_E^p,$$

para E nodos de entrada. Em seguida, calcula-se o erro, com base na função de custo, normalmente dada por:

$$\xi = \frac{1}{2}\sum_{k \in S}(t_k^p - s_k^p)^2 \tag{3.6}$$

- *Retropropagação*, onde o erro é propagado para atrás, desde a saída até os nodos de entrada. De seguida, os pesos são ajustados segundo a regra de Widrow-Hoff (equação 3.3). Para um único peso tem-se que:

$$\Delta w_{ij}(t) = -\eta * \frac{\delta \xi}{\delta w_{ij}}(t) \tag{3.7}$$

em que *t* representa a ordem da iteração e η a taxa de aprendizagem [Riedmiller 1994b]. As derivadas parciais são calculadas segundo o clausulado:

$$\frac{\delta E}{\delta w_{ij}} = \frac{\delta \xi}{\delta s_i}\frac{\delta si}{w_{ij}} \tag{3.8}$$

onde

$$\frac{\delta s_i}{w_{ij}} = f^{'}(u_i)s_j \tag{3.9}$$

Para se obter $\frac{\delta \xi}{\delta s_i}$, ou seja a influência da saída s_i do nodo *i* no erro global ξ, é necessário atender ao tipo de nodo; i.e.,

$$\frac{\delta \xi}{\delta s_i} = \begin{cases} -(t_i - s_i) & , i \in S \\ \sum_{j \in succ(i)} \frac{\delta \xi}{\delta s_i} f^{'}(u_j) w_{ji} & , i \notin S \end{cases}$$

onde *succ(i)* representa o conjunto dos nodos *j* da camada com que o nodo *i* se relaciona (ou estabelece ligações).

Antes de se iniciar o treino de uma rede, há que proceder à escolha dos valores iniciais dos pesos associados às ligações entre nodos, que em geral pertencem ao intervalo [0,1] ou [-1,1] e são gerados de forma aleatória. Pode-se então partir para o treino da rede, começando-se por seleccionar um caso de treino, caso se opte por um processo de aprendizagem com base em iterações, ou então em subscrever todos os casos, na forma *em lote.* Normalmente, convém que esta selecção seja aleatória. Em seguida, calcula-se o gradiente e ajustam-se os pesos. Uma *iteração* termina quando todos os casos disponíveis tiverem sido considerados. O processo computacional é condicionado por critérios de paragem (e.g., quando as mudanças nos pesos e na função de erro forem irrelevantes). De notar que o algoritmo pode convergir para um mínimo local (Figura 3.11); na prática, porém, constata-se que quando se parte de um número elevado de casos de treino, esta questão não se coloca, ou então não se assume como um problema sério.

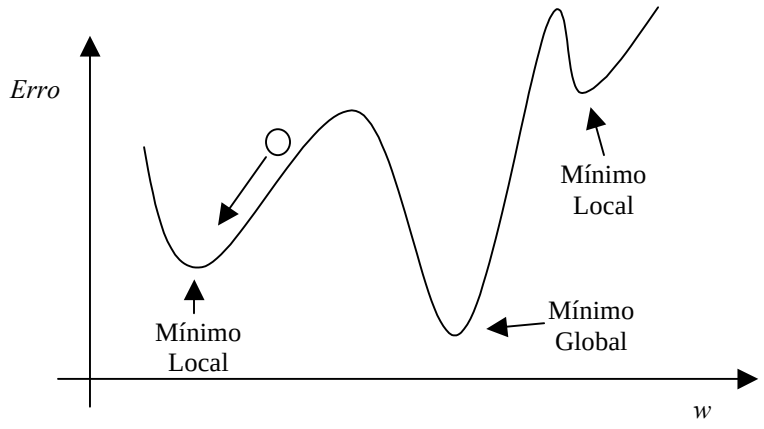


Figura 3.11 – Mínimos locais e globais

3.2.3 Alterações ao Algoritmo de *BP*

A primeira apresentação do algoritmo alterou o panorama da investigação em *RFMCs* e desde então têm surgido uma miríade de novos algoritmos de treino. Esta explosão deve-se a duas razões fundamentais. Por um lado, o algoritmo de *BP* é pesado, de convergência lenta. Por outro lado, baseia-se no gradiente descendente, pelo que, todas as técnicas de optimização não linear do gradiente podem ser aplicadas [Bose & Liang 1996]. No pior dos casos, a aprendizagem de *RNAs* é *NP-completa*; i.e., o esforço computacional envolvido cresce de um modo exponencial com o aumento de parâmetros do processo (i.e., pesos). Assim, existe espaço para a proposta de alternativas que possam acelerar o processo de aprendizagem, embora seja sempre possível enganar o "melhor" método com uma dada tarefa de aprendizagem, desde que se conheçam os seus pontos fortes e fracos [Riedmiller & Braun 1993]. Por conseguinte, diversas variantes rápidas do algoritmo de *BP,* que se baseiam no uso de uma topologia fixa, foram propostas nos últimos anos. Todavia, talvez as melhorias realmente significativas advenham do

uso de topologias *adaptativas*; i.e., algoritmos que adaptam não só os pesos mas também a topologia interna da rede a uma dada tarefa.

3.2.3.1 Melhorias ao algoritmo de *BP*

Existem formas simples de acelerar o algoritmo de *BP* [Gallant 1993] [Riedmiller & Braun 1993], as quais se podem sintetizar nos termos:

- *Taxa de Aprendizagem* – Quanto mais pequena for η menores vão ser as mudanças nos pesos das conexões da *RNA*, de modo que a procura do mínimo global será favorecida pelo uso de saltos mais suaves. O problema está em que desta forma converge-se para uma aprendizagem mais lenta. Por outro lado, se se aumentar em demasia o valor de η, então os saltos desmedidos nas mudanças dos pesos poderão provocar instabilidade no treino (e.g., movimento oscilatório). Uma forma de aumentar a taxa de aprendizagem sem provocar instabilidade é alterar a regra de aprendizagem de modo que esta inclua um termo de *momentum*:

$$\Delta w_{ij}(t) = -\eta * \frac{\delta \xi}{\delta w_{ij}}(t) + \mu \Delta w_{ij}(t-1) \qquad (3.10)$$

  onde μ representa a constante de *momentum,* usualmente dada por um valor positivo escolhido do intervalo [*0,1*]. Este termo de *momentum* funciona como uma memória que acelera descidas do gradiente, tendo um efeito estabilizador em situações oscilatórias. Outra forma distinta para lidar com este problema reside no uso de diferentes taxas de aprendizagem, uma por cada nodo. LeCun [Kwok & Yeung 1999] sugere que se utilize $\eta = \frac{1}{\sqrt{z}}$, para um nodo com *z* conexões;

- *Modo de* Treino – Durante a aprendizagem, uma *iteração* corresponde a uma completa apresentação de todos os exemplos de treino à rede. Convém que a apresentação seja aleatória, para que não exista uma tendência para valorizar certos casos de treino. Esta apresentação pode ser efectuada de duas formas:

    - *Modo sequencial,* onde se calculam os ajustamentos aos pesos logo após a escolha de um exemplo de treino; e
    - *Modo em lote* ou *por iteração*, onde os ajustamentos aos pesos só são efectuados após a apresentação de todos os casos de treino.

  O primeiro modo exige uma menor memória local de armazenamento, para além da apresentação aleatória tornar difícil que o algoritmo fique preso em mínimos locais, sendo deveras útil quando o conjunto de casos de treino é elevado e existe bastante informação redundante. Em contraste, o modo *em lote* melhora a estimativa do vector de gradiente, sendo também um processo mais fácil de se paralelizar;

- *Funções de Activação* – Embora seja habitual o uso de funções de activação com saídas dentro do intervalo [*0,1*], como no caso da função *logística padrão* existem pressupostos teóricos e empíricos que favorecem saídas dentro do domínio [*-1,1*]. A função *tanh* (Tabela 3.1) é um exemplo deste tipo de funções. A função *logística* também pode ser adaptada a este intervalo, tomando então a forma:

$$s_i = \frac{1-\exp(-u_i)}{1+\exp(-u_i)} \tag{3.11}$$

Outro dos aspectos computacionais do algoritmo *BP* está relacionado com o número de operações de vírgula flutuante cujo número poderá ser reduzido pelo uso de funções de activação mais simples (e.g., para a função *logística* existe uma versão que não necessita do cálculo de expoentes [Elliot 1993], e que é dada na forma: $s_i = \frac{1}{(1+|u_i|)^2}$);

- *Maximizando o Conteúdo de Informação* – Existem heurísticas simples para melhorar a informação fornecida à rede [Gallant 1993], e que pode ser expressa na forma:

    - Escolha do caso do conjunto de treino que origina o maior erro no treino da *RNA*; e
    - Escolha do caso do conjunto de treino que se afigure como o mais distinto em relação aos casos já alimentados à *RNA*.

- *Pré-Processamento de Dados* – Em muitas situações torna-se útil pré-processar os dados de um problema antes de estes serem alimentados à rede. A ideia é ajustar os dados de alguma forma tal que a rede os possa tratar de forma eficiente. Existem diversas formas de pré-processamento que serão descritas a seguir;

- *Escolha dos Pesos Iniciais* – Uma boa escolha dos valores iniciais dos pesos das ligações entre nodos da rede pode acelerar o processo de aprendizagem. O problema está em definir o que é uma boa escolha*?* Valores elevados podem levar a um processo de uma saturação da rede, atrasando o processo de aprendizagem. No entanto, valores demasiado pequenos podem levar a que o algoritmo de *BP* opere numa área demasiado plana que circunda a superfície de erro. Assim quer valores

demasiado elevados ou demasiado pequenos devem ser evitados. Uma boa estratégia é gerar valores aleatórios com média igual a *zero*, e que dependem do número de conexões de um nodo. Por exemplo, Gallant [Faraday & Chatfield 1998] sugere o intervalo $\left[\frac{-2}{z};\frac{2}{z}\right]$, para um nodo com *z* entradas.

3.2.3.2 Algoritmos de Adaptação Local

As variantes do algoritmo de *BP* podem ser classificadas em duas categorias: de adaptação global ou local [Riedmiller 1994b]. Os primeiros utilizam um conhecimento global do estado completo da rede, como a direcção de propagação de todo o vector de actualização dos pesos. Em contraste, os últimos são baseados na informação específica de um peso, como o comportamento temporal da sua derivada parcial. Este tipo de estratégia é mais próxima ao conceito em que se baseia todo o desenvolvimento operado nas *RNAs,* sendo mais facilmente paralelizável. Para além disso, estas técnicas tendem a ser mais eficazes e robustas, apesar de usarem menos informação. Neste contexto, estão dois algoritmos importantes de adaptação local: o *QuickProp* e o *RPROP* [Fahlman 1989] [Bose & Liang 1996] [Riedmiller 1994a] [Riedmiller 1994b] [Prechelt 1994].

3.2.4 Capacidades e Limitações das *RFMCs*

Uma rede *RFMC* treinada por *Backpropagation* pode ser vista como uma forma prática para efectuar uma qualquer correspondência não linear com uma dada função. A questão pode ser posta nos termos: dada uma função *g* desconhecida que faz uma correspondência entre padrões de um espaço *n*-dimensional para um espaço *m*-dimensional, $g : \Re^n \rightarrow \Re^m$, será possível encontrar uma *RFMC* que realize a correspondência exigida ou pelo menos se aproxime dela com alguma eficácia? A

resposta vem do *teorema universal de aproximação de funções* que estipula que uma camada intermédia é suficiente para uma *RFMC* computar uma aproximação de uma qualquer função contínua, embora o teorema não afirme que esta estrutura seja óptima em termos de tempo de aprendizagem e generalização [Gallant 1993]. Também está provado que com duas camadas intermédias até funções descontínuas podem ser representadas [Rumelhart et al. 1986].

Outro aspecto importante diz respeito ao tempo de aprendizagem. A aprendizagem implica a procura dos elementos desconhecidos de uma *RNA,* normalmente pelo ajuste dos pesos. Ora, a aprendizagem numa rede com 100 pesos é bastante mais pesada em termos computacionais do que a de uma rede com 10 pesos, sendo uma relação bem maior daquela que o factor 1:10 poderia sugerir. Seria muito útil que o tempo de aprendizagem fosse limitado por uma função polinomial sobre o número de variáveis. No entanto, tal não sucede. Está provado que o problema geral de aprendizagem em *RNAs* é intratável; i.e., não pode ser resolvido de forma eficiente para todas as instâncias da rede. Não é conhecido um algoritmo que consiga realizar a aprendizagem num tempo polinomial, sendo improvável que tal algoritmo venha a existir. Assim, diz-se que em geral o problema de aprendizagem em *RNAs* é NP-completo [Riedmiller & Braun 1993]. Todavia, um problema que seja de difícil aprendizagem para uma dada estrutura de rede poderá ser de fácil aprendizagem para outra. Assim, uma das possibilidades para ultrapassar a aprendizagem NP-completa das *RNAs* reside no uso de arquitecturas adaptativas [Kosko 1988].

3.2.5 Pré-Processamento dos Dados

O pré-processamento dos dados deve ser seriamente considerado antes de treinar uma rede. Embora não seja essencial em certos casos, torna-se vital noutros. São várias as operações em que se pode desdobrar o pré-processamento [Azoff 1994] [Sarle 1995]:

- *Verificação da Integridade dos Dados* – Trata-se de um procedimento de validação dos dados, verificando-se se existem erros de alguma espécie ou procedência;

- *Representação dos Dados* – Esta fase envolve uma conversão de dados. Dado que as *RNAs* apenas lidam com números, os dados terão de ser codificados numa representação binária, discreta ou real. Por outro lado, uma variável pode ser representada por um ou mais nodos. Por exemplo, em problemas de classificação, com *M* classes, é comum representar uma saída por um vector binário M-dimensional. A classe *k* poderá então ser representada por um vector em que o *k* elemento toma o valor *1* (verdadeiro) e os restantes elementos o valor *0* (falso);

- *Escalonamento dos Dados* – O objectivo deste procedimento é realizar uma transformação nos dados de modo a acelerar e melhorar o processo de aprendizagem. Este escalonamento depende do tipo de dados:
    - *Entradas.* O escalonamento das variáveis de entrada tem diversos efeitos conforme os algoritmos de aprendizagem. De um modo particular, algoritmos de gradiente descendente, como o conhecido *BP*, são bastante sensíveis ao escalonamento. Assim, cada variável de entrada deve ser escalonada para que a sua média sobre o conjunto de casos de treino esteja à volta do valor zero, pelo que o escalonamento para o intervalo [*-1,1*] funcionará melhor do que para o intervalo [*0,1*]. Existem duas formas eficientes de realizar este escalonamento:

        - *Média Zero e Desvio Padrão 1*

$$y = \frac{x - \overline{\chi}}{std(x)} \tag{3.12}$$

onde $\overline{x}$ denota a média de *x* e *std*(*x*) o desvio padrão de *x*; i.e.,

$$std(x) = \sqrt{\frac{\sum_i (x_i - \overline{x})^2}{n-1}}$$

para *n* casos de treino.

- *Valor Mínimo -1 e Valor Máximo 1*

$$y = \frac{x - (max + min)/2}{(max - min)/2} \tag{3.13}$$

em que *min* e *max* denotam os valores mínimo e máximo a tomar pela variável *x*.

- *Saídas.* Existem duas fortes razões para escalonar as saídas. Em primeiro lugar, quando se usa mais de uma saída e se a função de erro é sensível à escala, como acontece no caso da aprendizagem do gradiente descendente, então a diferença de escalas entre as saídas pode afectar a forma de aprendizagem da rede. Por exemplo, se uma saída tem valores entre *0* e *1* enquanto outra tem valores entre *0* e *1000000*, então o algoritmo de treino irá utilizar a maior parte do seu esforço na aprendizagem da segunda saída. Assim, as saídas que têm a mesma importância devem ser escalonadas para a mesma escala. Neste caso, deve-se proceder a um escalonamento com a mesma escala ou desvio padrão

(usando por exemplo a equação 3.12). Em segundo lugar, pode-se querer ajustar as variáveis de saída aos valores do contradomínio da função de activação (e.g., a função *logística* produz valores de saída dentro do domínio [*0,1*]). Neste caso, é importante conhecer os limites máximos e mínimos da variável em vez dos valores máximos e mínimos dos casos de treino. Neste caso poder-se-á utilizar a seguinte fórmula:

$$y = \frac{(x - L_{min})(B - A)}{L_{max} - L_{min}} \tag{3.14}$$

para um escalonamento no domínio [*A,B*], sendo L_{max} e L_{min} os limites máximo e mínimo da variável. Se a variável de saída tem limites desconhecidos então é preferível utilizar a função linear (ou outra não limitada) para os nodos de saída.

- *Redução da Dimensão ou Cardinalidade da Rede* – O número de elementos num vector de entrada, a sua dimensão ou cardinalidade, pode atingir valores elevados, principalmente quando não existe um grande conhecimento sobre a sua natureza e relevância. Em geral, redes de maior cardinalidade têm uma aprendizagem mais difícil. Uma das formas de aliviar este problema consiste na redução da dimensão dos vectores de entrada, extraindo-se a informação relevante. Existem várias técnicas para atingir este objectivo, nomeadamente as referidas a seguir:
  - *RFMC para Compressão*. Uma forma simples consiste no uso de uma *RFMC* com uma camada intermédia, que contém menos nodos do que os nodos de entrada, em que apenas conexões entre camadas adjacentes são permitidas. Esta rede contém um igual número de nodos de entrada e de saída, que estão de acordo com

a dimensão original do problema. Os vectores de saída também são iguais aos vectores de entrada, pelo que a rede é treinada para reproduzir a sua própria entrada. Devido ao desenho em funil, a camada intermédia aprende a comprimir os dados. No final do treino, existirá um erro residual, que depende do grau de compressão. Caso este não possa ser reduzido a um valor aceitável, é sinal de que a compressão não se adequa ao problema. Por fim, a camada de saída é removida e os pesos entre a camada de entrada e intermédia são fixos. A estrutura resultante pode então ser acrescentada a uma rede de dimensão reduzida, que irá aprender a tarefa original.

- *Análise de Componentes Principais (ACP).* O critério crucial na selecção das variáveis está na sua contribuição para se conseguir o objectivo pretendido. Ora, certas variáveis podem influenciar outras, existindo por isso uma duplicação de informação ou *correlação.* A separação da informação relevante de cada variável facilita a aprendizagem. Este processo, designado de *descorrelação,* pode ser visto como atingível através de uma rotação dos eixos cartesianos das entradas.
- *Transformada de Onda.* Esta fornece uma nova representação dos dados, com vectores formados por funções de onda, cujos componentes correspondem aos diversos graus de detalhe dos dados originais. A compressão de dados é então obtida pelo uso de um subconjunto destes componentes.

- *Filtragem de Ruído* – Por vezes torna-se útil o uso de técnicas de filtragem para a eliminação de ruído, suavizando a função de aprendizagem, facilitando o treino. Como exemplo temos a *análise espectral*, uma técnica muito usada na manipulação de imagens, onde um

sinal de entrada é decomposto em ondas sinusoidais, pela *Transformada de Fourier*.

3.2.6 Generalização

Diz-se que uma *RNA* possui uma boa *generalização* quando a correspondência entre entradas e saídas é correcta (ou próxima disso) para *dados de teste,* retirados da mesma população, nunca antes utilizados na criação ou treino da rede. O processo de aprendizagem pode ser visto como um problema de ajustamento de curvas ou de aproximação de funções, onde a rede tenta efectuar uma boa interpolação não linear dos dados [Riedmiller & Braun 1993]. A Figura 3.12 mostra como podem ocorrer duas generalizações distintas para o mesmo conjunto de dados de treino. Aqui, uma boa generalização ocorre com a curva *A*, com um erro mínimo para os dados de teste. O mesmo já não sucede com a curva *B*, que origina um erro maior para os casos de teste, isto apesar de apresentar um menor erro para os dados de treino. Tal fenómeno, designado de *overfitting,* ocorre quando uma *RNA* memoriza em demasia os exemplos de treino, tratando-se de um dos problemas mais sérios relacionados com o uso de *RNAs* [Russell & Norvig 1995]. Durante o processo de aprendizagem, a rede pode captar certas características, como o ruído, que estão presentes nos dados de treino mas não na função implícita a ser aprendida. Este exemplo ilustra os dois objectivos contraditórios da aproximação funcional. Por um lado tem-se a minimização do erro de treino, pelo outro tem-se a minimização do erro para as entradas desconhecidas. Assim, uma *RNA* que seja treinada em demasia perde capacidade para generalizar.

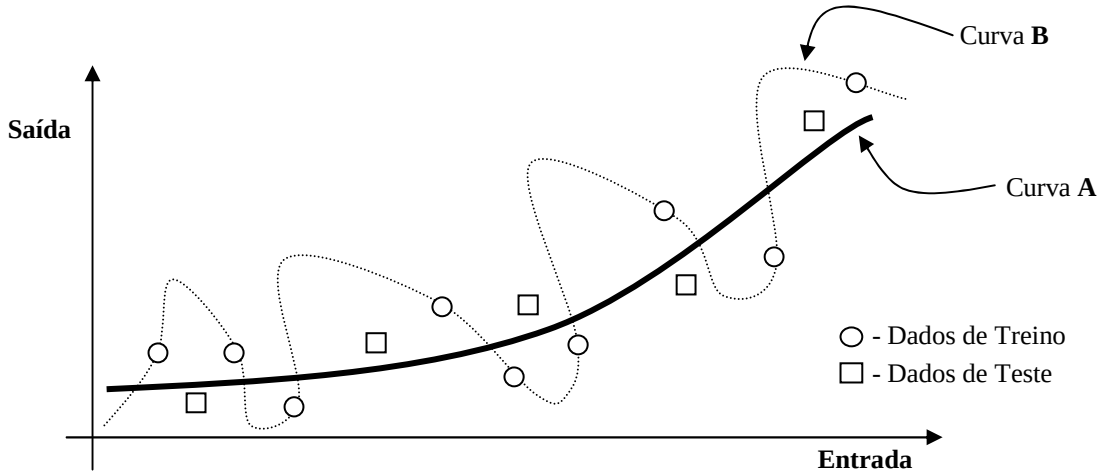


Figura 3.12 – Generalização e *overfitting*

A generalização é influenciada por três factores [Gallant 1993] [Sarle 1995], que podem ser expressas na forma:

- *A complexidade do problema a ser aprendido* – Trata-se de um factor de difícil controlo. As entradas devem conter informação suficiente para permitir a obtenção das saídas desejadas; i.e., tem de existir uma função matemática com algum grau de precisão que relacione as entradas com as saídas. Por outro lado, convém que esta função seja *suave*; i.e., pequenas alterações nas entradas devem provocar pequenas alterações nas saídas, para a maior parte dos casos. Por vezes, uma transformação não linear nas entradas pode melhorar a sua suavidade (e.g., transformação logarítmica);

- *Os casos de treino* – A sua cardinalidade deve ser suficientemente representativa, com exemplos (ou *amostras* na terminologia estatística) que caracterizem o ambiente (ou população). A generalização é sempre

efectuada a partir de dois tipos de duas situações: *interpolação e extrapolação*. No primeiro caso, um valor é calculado a partir da informação dos valores constantes de casos na vizinhança. A segunda situação engloba tudo o resto, ou seja, casos fora do domínio dos dados de treino. Enquanto que a interpolação pode ser efectuada com relativa acuidade, o mesmo já não se passa com a extrapolação, notoriamente menos fiável;

- *A arquitectura da RNA*; i.e., o número de parâmetros livres que denotam os pesos das ligações entre nodos e a sua capacidade de aprendizagem bem como a sua complexidade. Uma rede não propriamente complexa irá falhar na aproximação à função a aprender. Por outro lado, uma rede demasiado complexa irá fixar o ruído existente nos dados, provocando overfitting. A Figura 3.13 mostra uma variação típica do erro de uma *RNA* com uma camada intermédia, para os casos de treino e de teste, com o incremento do número de nodos intermédios. À medida que estes aumentam o erro de treino diminui. A dada altura, a curva de erro para os casos de teste inflecte, perdendo-se em generalização.

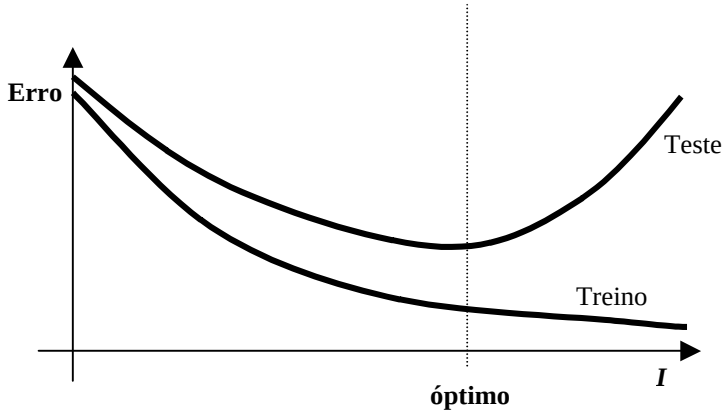


Figura 3.13 – Erro típico que ocorre com o aumento do número de nodos intermédios

A melhor forma de evitar o *overfitting é* utilizar uma quantidade elevada de casos de treino. Quando este número for pelo menos 30 vezes superior ao número de conexões, então é muito improvável que ocorra *overfitting*. O problema está em que nem sempre existem muitos casos de treino disponíveis e não se deve reduzir o número de conexões de um modo arbitrário, devido a problemas de insuficiência de complexidade da rede. Dado um número fixo de casos de treino existem duas grandes alternativas eficientes para controlar a complexidade da rede [Russell & Norvig 1995] [Sarle 1995]: *regularização* e *selecção de modelos.*

3.2.7 Regularização

A regularização baseia-se num controlo dos valores dos pesos das conexões da rede para se obter uma boa generalização. Entre os diversos métodos de regularização têm-se os definidos por [Russell & Norvig 1995] [Sarle 1995], e a seguir referenciados.

3.2.7.1 Decaimento dos Pesos

A estratégia passa por acrescentar uma penalidade à função de erro, de modo a reduzir os pesos das conexões, em particular as mais expressivas, visto que estas prejudicam o processo de generalização, dando origem a funções irregulares, por vezes na vizinhança de descontinuidades. Por outras palavras, pesos cujo $|\bullet|>>0$ causam uma excessiva variância nas saídas ($|\bullet|$ denota a função módulo) [Barlett 1997]. Normalmente esta penalidade é dada pela expressão:

$$d \times \sum w_{ij}^2 \tag{3.15}$$

onde *d* representa a *constante de decaimento,* cuja escolha é crucial para uma boa generalização.

3.2.7.2 Adição de Ruído

O objectivo é acrescentar deliberadamente ruído artificial às entradas durante o treino. Esta estratégia funciona porque a maior parte das funções a serem aprendidas pela rede são suaves. Assim, em cada iteração do algoritmo de treino, novos casos de treino são criados, pelo acrescento de ruído. Este nem deve ser demasiado pequeno, produzindo pouco efeito, nem demasiado grande, pois obviamente desvirtuará a função implícita a ser aprendida. Este ruído é produzido por um gerador de números aleatórios, usualmente seguindo uma distribuição normal com média *0* e desvio padrão *s*, cujo valor deverá ser estimado de algum modo (e.g., de modo a que seja menor do que o erro de generalização, medido por um estimador).

3.2.7.3 Paragem Antecipada

Trata-se de um dos mais populares métodos de regularização, onde os dados de treino são divididos em dois tipos de casos: *de treino* e *de validação.* Os primeiros são utilizados na aprendizagem da rede, enquanto que os últimos são utilizados para aferir da qualidade da aprendizagem; i.e., para estimar o erro de generalização. De notar que podem ser utilizados novos *casos de teste* para medir o desempenho da rede após o treino. Durante a fase de treino, calcula-se o erro de validação de forma periódica, parando-se quando este começa a aumentar. Todavia, esquemas de paragem mais elaborados têm de ser adoptados, dado que a função de erro pode

apresentar diversos mínimos locais. Por exemplo, Prechelt [Patterson 1996] defende o uso de três critérios de paragem:

- *Falha de Progresso do Treino* – A estratégia passa por tentar avaliar o progresso do treino; i.e., qual a diminuição do erro sobre os casos de treino, ξ_{tr}, durante uma dada *fatia do treino*, com k iterações. A função de progresso, avaliada em cada k iterações, toma a forma:

$$P_k(t) = 1000 \times \left(\frac{\sum_{t' \in t-k+1 \ldots t} \xi_{tr}(t')}{k \times min_{t' \in t-k+1 \ldots t} \xi_{tr}(t')} - 1 \right) \tag{3.16}$$

  O progresso no treino é elevado nas suas fases de maior instabilidade, onde o erro para os casos de treino sobe em vez de diminuir. No entanto, tende para *zero* a longo prazo, a não ser que o treino se torne oscilante. O treino é parado se $P_k(t) < \beta$, em que β é uma medida de erro em estado estacionário;

- *Perda de Generalização* – Esta ocorre sempre que há uma inversão de sinal nos valores da derivada da função de erro para os casos de validação, ξ_{va}, passando estes de negativos a positivos. A função de avaliação, também medida de k em k iterações, toma a forma:

$$G_k(t) = 100 \times \left(\frac{\xi_{va}(t)}{min_{t' \le t} \xi_{va}(t')} - 1 \right) \tag{3.17}$$

  Uma grande perda de generalização é uma boa razão para se parar o treino, pelo que o treino termina se $G_k(t) > \alpha$, onde α denota a perda de poder de generalização aconselhável para a rede; e

- *Número Máximo de Iterações* – Este critério é aplicado quando os anteriores critérios de paragem falham, de modo a garantir que o treino termine.

A paragem antecipada é bastante utilizada porque é simples e rápida, podendo ser aplicada a *RNAs* com um grande número de conexões. Todavia, possui algumas desvantagens. Em primeiro lugar, é bastante sensível à forma como é feita a divisão entre casos de treino e de validação; i.e., quais e quantos casos devo usar em cada categoria. Por outro lado, não aproveita toda a informação disponível para a aprendizagem. Daí que certos autores defendam o uso de estimadores mais sofisticados para o erro de generalização [Kernsley & Martinez 1992].

3.2.8 Selecção de Modelos

A regularização diminui o efeito de *overfitting* pelo estímulo dado à aprendizagem de funções suaves. No entanto, utiliza uma estrutura fixa, que deve ser especificada em avanço pelo utilizador. Embora se possa utilizar uma grande estrutura, com um grande número de nodos intermédios, na prática, a optimização dos pesos torna-se de difícil ajuste, exigindo um grande esforço computacional. Mais ainda, em geral, são métodos que exigem um delicado balanço, controlado por um (ou mais) parâmetro(s) de regularização. Mais recentemente, métodos *Bayesianos* têm sido incorporados na regularização, para eliminar alguns destes problemas. Trata-se de uma abordagem promissora embora ainda pouco desenvolvida. Para além disso, estes métodos assumem certos tipos de distribuição entre casos de treino e teste que podem falhar quando o número de conexões da rede é grande, quando comparado com a cardinalidade dos casos de treino [Kosko 1988]. Uma alternativa distinta baseia-se na procura de uma topologia para uma *RNA,* em termos do número de conexões, número de nodos e camadas intermédias. Os defensores desta estratégia argumentam que é mais fácil adaptar a complexidade da rede ao problema a ser

resolvido. Assim, um problema que seja de difícil aprendizagem para uma rede poderá ser facilmente aprendido por outra rede.

A abordagem estatística à resolução deste problema passa pela estimativa do erro de generalização para cada um dos modelos, ou topologias de rede, escolhendo-se o modelo que minimiza essa estimativa.

3.2.8.1 Métodos de Estimação do Erro de Generalização

Existem diversos métodos para estimar a capacidade de generalização de uma *RNA* [Efron & Tibshirani 1993] [Kernsley & Martinez 1992], alguns dos quais são enunciados a seguir:

- *Estatísticas Simples* – Diversas métricas foram desenvolvidas tendo em conta modelos lineares, baseando-se em suposições sobre as amostras de dados. Entre estas, podem-se referenciar:
    - *Critério de Informação de Akaike,* conhecido por *AIC,* que tende a admitir o *overfitting* em *RNAs*:

$$AIC = n \ln(SSE / n) + 2p \qquad (3.18)$$

      onde *SSE* representa o somatório do quadrado dos erros para todos os casos de treino, *n* representa o número de casos de treino e *p* o número de parâmetros livres da rede, ou seja, o número de pesos das ligações entre os nodos da rede (#*C*); e
    - *Critério de Informação de Bayes,* designado por *BIC* ou *SBC*, que normalmente funciona bem com *RNAs:*

$$BIC = n \ln(SSE / n) + p \ln(n) + p \qquad (3.19)$$

- *Validação com Divisão da Amostra* – O método mais popular para a estimação do erro de generalização de uma *RNA*, geralmente associado à paragem antecipada do treino da rede, baseia-se numa divisão dos dados do problema em casos de treino, para a rede aprender, e casos de validação, para estimar o erro de validação. Como ponto forte deste processo tem-se a sua simplicidade e rapidez, embora produza uma redução efectiva dos casos disponíveis para treino.

- *Validação Cruzada* – Trata-se de um melhoramento introduzido no método anterior, que permite utilizar todos os casos disponíveis. Na validação cruzada *k-desdobrável*, os dados (P) são divididos em k subconjuntos mutualmente exclusivos ($P_1, P_2, \ldots, P_k$) de cardinalidade aproximadamente igual. A rede é então treinada e testada k vezes. Em cada tempo $t \in \{1, 2, ..., k\}$, a rede aprende a partir do conjunto com $P \setminus P_t$, estimando-se o erro de validação a partir dos casos restantes, P_t. O erro final de generalização é então dado pela média dos erros de validação obtidos durante k vezes. Os valores de k podem variar entre 2 e n, embora um valor de referência seja k=10. Um valor pequeno para k origina uma má generalização enquanto que um valor elevado aumenta consideravelmente o esforço de computação, visto que a rede é treinada k vezes para cada invocação do algoritmo de treino. A validação cruzada é notavelmente superior à validação com divisão da amostra para pequenos conjuntos de casos de treino. Todavia, tal realização é conseguida à custa de um maior esforço computacional; e

- *Bootstrapping* – Dado um conjunto de dados com n elementos, uma amostra de *bootstrap* é criada por uma selecção de n instâncias dessa amostra a partir dos dados. A probabilidade de uma dada instância não ser escolhida após n

amostragens é dada por $(1 - 1/n)^n \approx 0{,}368$. Assim, o número esperado de instâncias distintas da amostra é dada por $0{,}623n$. O método consiste em utilizar a instância da amostra de *bootstrap* para treinar a rede e as instâncias restantes para validação da mesma. A estimativa de erro é então dada por:

$$\frac{1}{b}\sum_{i=1}^{b}(0{,}632\xi_{va_i} + 0{,}368\xi_{tr}) \tag{3.20}$$

onde ξ_{va_i} e ξ_{tr_i} denotam respectivamente a estimativa do erro de validação e de treino para cada amostra i. Os valores típicos para b variam entre 20 e 200. O *bootstrapping* parece funcionar melhor que a validação cruzada em muitas situações, embora também à custa de ainda mais esforço computacional.

3.2.8.2 Escolha da Topologia de Rede

A topologia ideal (em termos de camadas, nodos e conexões) de uma *RFMC* depende fundamentalmente do problema a ser resolvido. Normalmente tenta-se reduzir o número de opções (ou espaço de procura) adoptando certos pressupostos, como o uso de redes completamente interligadas, evitando-se neste caso a preocupação sobre que conexões devem existir. Também é comum adoptar-se apenas uma camada intermédia, ficando a escolha dos nodos intermédios a ser decidida por regras empíricas. Estratégias mais elaboradas passam por procedimentos de *tentativa-e-erro,* utilizando uma procura aleatória, embora seja altamente desejável a utilização de métodos eficientes e automáticos, com base em heurísticas (e.g., *escalar-montanhas ou algoritmos genéticos*) [Rumelhart et al. 1986]. Neste caso existem duas importantes estratégias: *algoritmos de corte* e *algoritmos construtivos* [Neves & Cortez 1998].

Os algoritmos de corte envolvem a remoção de nodos e conexões desnecessárias, segundo o princípio de *Ockham*[4]; *i*.e., tentando-se obter a rede mais simples que dê respostas a um dado problema. Estes algoritmos começam com uma *RNA* que seja suficientemente complexa para garantir um treino com sucesso. A seguir, algumas conexões e/ou nodos são removidos e a rede volta a ser treinada. Este processo continua até que uma solução aceitável seja encontrada, devolvendo-se no final a rede mais simples que garante uma boa aprendizagem. Independentemente do método de corte, é difícil avaliar a qualidade da rede. Tal critério deve depender do número de camadas intermédias, nodos, conexões e restrições da aplicação (e.g., requisitos de obtenção de respostas em tempo real podem implicar um limite para a quantidade de camadas intermédias). Outro critério importante a ter em conta é a capacidade de generalização da rede. Em geral, a avaliação da rede deve depender do tempo de treino, da complexidade da rede e a da sua capacidade de generalização.

As principais decisões de um algoritmo de corte envolvem a escolha da(s) unidade(s) a ser cortada(s), quando efectuar o corte e quando parar o treino. Existem vários algoritmos de corte que diferem fundamentalmente na primeira decisão, a parte mais crítica do processo. Esta escolha, sobre o que cortar, é feita com base em heurísticas (e.g., a heurística simples da remoção das conexões com os pesos menos significativos) [Sharda & Rampal 1996].

Algoritmos construtivos. Trata-se de uma estratégia oposta, seguindo-se outra direcção de procura: começa-se com uma rede simples e depois nodos intermédios e conexões adicionais são acrescentadas a esta, até que uma solução satisfatória seja encontrada. São várias as vantagens desta abordagem sobre a anterior. Em primeiro lugar, é fácil especificar a topologia inicial da rede, enquanto que nos algoritmos de

[4] William of Ockham - 1349. Filósofo e frade franciscano inglês, conhecido pela defesa da aplicação da regra económica ontológica à teologia: “entidades não são multiplicadas para além da necessidade”, tão frequentemente utilizada que se tornou famosa como a *navalha de Ockham.*

corte não se sabe à partida qual deve ser a complexidade da rede. Redes menos complexas são testadas no início, sendo assim mais económicas em termos de custos de computação. Mais ainda, dado que diversas redes de diferente complexidade podem ser capazes de constituir soluções aceitáveis, os algoritmos construtivos têm uma maior probabilidade de encontrar redes menos complexas do que as obtidas por processos de corte. Outra motivação para este tipo de algoritmo está relacionada com o tempo de aprendizagem. A flexibilidade destes torna estas redes em máquinas universais de aprendizagem; i.e., capazes de resolver em tempo polinomial qualquer classe de problemas que possa ser aprendida em tempo polinomial por outros métodos. Como dificuldades mais importantes desta estratégia tem-se a altura em que o processo de construção da rede deve terminar, as heurísticas de procura (que podem ser sub-óptimas) e os cuidados a ter com a generalização da rede.

Os diversos algoritmos construtivos diferenciam-se, fundamentalmente, pela sua *estratégia de construção* ou *de transição de estado,* existindo duas importantes categorias: *de um só valor* ou *multi-valor.* A primeira estratégia caracteriza-se pelo uso de apenas um único estado possível de transição, ao contrário da procura multi-valor, onde existem vários estados possíveis, designados por *candidatos,* sendo cada um estruturalmente distinto dos outros. Os algoritmos construtivos também se podem distinguir pelo tipo de treino aplicado à nova estrutura, que pode ser *total,* treinando-se todos os pesos da rede, ou *parcial,* congelando-se os valores dos pesos antigos e treinando-se apenas as conexões dos novos nodos, [Kosko 1988].

3.3 Conclusões

As *RFMCs* têm sido empregues com sucesso em diversas domínios, incluindo Engenharia, Direito, Ciências da Computação, Processos de Controlo, Robótica e

Automação, Estatística, Medicina, Produção, Economia, entre outros. Esta grande popularidade das *RFMCs* advém das suas capacidades de correspondência não linear entre padrões. Normalmente, para um bom conjunto de dados de treino, uma rede com apenas uma (ou quanto muito duas) camada(s) intermédia(s) serve para aprender uma dada tarefa com mestria. Como exemplos de aplicações de *RFMCs* tem-se [Neves & Cortez 1998] os casos de:

- *Classificação e Diagnóstico*
    - Classificação de células para um diagnóstico de cancro; e
    - Identificação de falhas na comutação de circuitos telefónicos.

- *Controlo e Optimização*
    - Condução de veículos autónomos; e
    - Controladores inteligentes para a produção de ferro.

- *Regressão/Previsão*
    - Atribuição de montantes de crédito;
    - Previsão de séries temporais;
    - Previsão da evolução de acções na bolsa; e
    - Previsão de resultados desportivos.

- *Reconhecimento de Padrões*
    - Reconhecimento de escrita manual; e
    - Identificação automática de indivíduos.

Neste trabalho, as *RNAs* são objecto de uma amálgama com sistemas de representação de conhecimento e raciocínio baseados numa extensão à programação em lógica (i.e., à *PLE*) , e usadas na classificação de imagens do foro médico.

Capítulo 4
SADMED – Ambiente Computacional para Sistemas de Apoio ao Diagnóstico Médico

Descreve formalmente o sistema SADMED. Apresenta a sua arquitectura, caracterizando não só cada entidade constituinte do sistema, como descrevendo o seu modus operandi.

O trabalho aqui apresentado tem como objectivo a concretização de uma arquitectura distribuída para *Sistemas de Apoio ao Diagnóstico Médico* (*SADM*), em que a distribuição é utilizada como forma de melhorar não só o seu desempenho mas também a sua esfera de aplicabilidade. O sistema, *Um Ambiente Computacional para Sistemas de Apoio ao Diagnóstico Médico* (*SADMED*), foi elaborado com recurso a entidades denominadas de agentes, com diferentes características, que partilham entre si informação de controlo e conhecimento, usando como plataforma de comunicação um *Quadro Negro* (*QN*), como forma de potenciar a resolução de problemas através da demonstração de teoremas [Engelmore et al. 1988] [Jagannathan et al. 1989] [Neves et al. 1994]. Por outro lado, e apesar dos muitos adjectivos que abonam a favor da *Programação em Lógica* (*PL*), esta não disponibiliza as abstracções e ferramentas necessárias para o

desenvolvimento de *software* complexo e distribuído no contexto da área da *IAD*. Estas limitações estão bem identificadas e têm motivado o aparecimento de várias extensões à *PL*, muitas das quais, no sentido de permitirem a especificação de *SPD* e *SM*. Uma das extensões criadas, que é aplicada na especificação do sistema *SADMED*, tem os seus alicerces na Programação em Lógica Contextual (*PLC*) [Cavedon & Tilhar 1995] [Denti et al. 1995] [Santos & Neves 1998] [Giunchiglia et al. 1993] [Cimatti & Serafini 1995]. Para ilustrar o conceito de base que está na origem da *PLC*, suponha-se que *teo* é uma teoria lógica; i.e., um conjunto identificado de cláusulas universalmente quantificadas. Colocar uma questão na forma de uma implicação $teo \rightarrow obj$ em relação a um programa *prog*, significa que a prova de *obj* seja realizada a partir do programa *prog* a que se associam os axiomas ou hipóteses contidas na teoria *teo*. A semântica operacional pode então ser formalizada através da regra de inferência:

$$\frac{prog \cup teo \vdash obj}{prog \vdash teo \rightarrow obj}$$

onde o símbolo $\vdash$ denota a dedução lógica (derivabilidade) e $\rightarrow$ a implicação lógica. Esta regra é facilmente implementada se se estender o programa *prog* com as cláusulas de *teo* antes de tentar provar *obj* e, a seguir, quer *obj* seja objecto de prova, ou não, desprezar essas cláusulas. Esta é a ideia chave subjacente à *PLC* e faz com que que cada teoria lógica seja concebida numa perspectiva de alguma incompletude relativamente ao conhecimento e numa óptica de mundo aberto (por oposição ao prossuposto do mundo fechado). Os programas passam a ser construídos pela composição dinâmica de teorias, como forma de criar bases de conhecimento completas, chamadas *contextos*.

O *contexto* pode ser interpretado (num determinado nível de abstracção) como um ambiente, uma hierarquia de componentes de *software* com herança, uma instância

de um objecto, uma linha de raciocínio, um subsistema, um agente ou uma comunidade de agentes.

Para especificação do sistema *SADMED* foi utilizado um formalismo desenvolvido para a especificação, modelação e raciocínio sobre um *SM* que utiliza a noção de *contexto* e incorpora algumas facilidades da programação orientada ao objecto, nomeadamente a abstracção, o encapsulamento, a modularidade e a hierarquia [Cavedon & Tilhar 1995] [Neves & Machado 1997]. Através deste formalismo é possível de uma forma flexível e modular, especificar:

- Os componentes do sistema ou agentes;
- O processo de socialização ou formas possíveis de agregação e cooperação;
- O procedimento de coordenação dos agentes; e
- O comportamento global do sistema.

Este formalismo permite a especificação de cada uma das entidades que compõem o sistema, uma vez que cada entidade apenas reporta ao seu próprio universo, assim como as interacções entre si. Nesta óptica, cada agente é modelado num *contexto* em separado, com a sua própria lógica, o que apresenta algumas vantagens competitivas, sendo desejáveis nestas entidades, entre outras, algumas das características a seguir enunciadas:

- Cada agente é uma entidade tão simples quanto possível, de tal modo que um agente precisa apenas de ser relacionado com as proposições no mundo ou universo de discurso com o qual está directamente relacionado;
- Agentes diferentes podem ser modelados como tendo interesses distintos, ou pelo menos perspectivas diferentes do mundo dentro do seu universo de discurso; e

- O desempenho de um agente pode ser maximizado, desde que lhe seja limitado o acesso a conhecimento, na medida em que se está em presença de conhecimento que se aplica à resolução de um certo tipo de problemas.

Um *SM* ou um *SPD* é especificado como um multi-sistema composto por agentes. Formalmente é definido através de:

- Um *contexto* que denota uma teoria em particular;
- Um conjunto de regras-ponte que funcionam como o interface entre os agentes e entre sistemas de agentes (comunidades); e
- Um conjunto não vazio de subsistemas, correspondendo cada um deles, por sua vez, a um multi-sistema com o seu próprio *contexto*.

As regras-ponte podem assumir a forma de:

- Regras subordinantes na forma $\frac{ocorre(e,i)}{C_k : ocorre(e',i')}$, denotando que se no sistema ocorre o evento *e* no instante *i*, então no contexto C_k ocorrerá o evento *e'* no instante *i'*;
- Regras subordinadas na forma $\frac{C_k : ocorre(e,i)}{ocorre(e',i')}$, denotando que se num determinado contexto C_k occorre o evento *e* no instante *i*, então no sistema ocorrerá o evento *e'* no instante *i'*; e
- Regras ordinárias na forma $\frac{C_k : ocorre(e,i)}{C_l : ocorre(e',i')}$, denotando que se num determinado contexto C_k occorre o evento *e* no instante *i*, então no contexto C_l ocorrerá o evento *e'* no instante *i'*.

As regras-ponte dão corpo, de uma forma explícita, à interacção e coordenação entre dois subsistemas. As restrições impostas por estas regras reflectem a esperança de que qualquer procedimento seja efectuado (com privacidade) dentro dos componentes do sistema (racionalidade emergente). Os componentes de um sistema não são visíveis fora desse sistema.

Neste trabalho é também proposto um formalismo para equacionar variantes de raciocínio acerca dos eventos de forma a capturar o comportamento interno dos agentes (subsistemas). Aos agentes está associado um conjunto de *fluentes*, que designam as propriedades do agente e que são dinâmicos ao longo do tempo; um conjunto dos valores que cada fluente pode tomar num determinado instante; um conjunto de eventos que o agente é capaz de executar; e um conjunto de instantes no tempo. A cada tipo de evento estão ainda associados dois conjuntos de pares fluente-valor, as pré-condições e as pós-condições de cada evento.

É altura, por conseguinte, de se fazer a apresentação da arquitectura computacional sobre a qual o sistema *SADMED* corre.

4.1 Uma Arquitectura Universal para a Resolução de Problemas em Ambiente Distribuído

A arquitectura aqui apresentada foi concebida para alto desempenho com regras de distribuição de cargas computacionais geradas por um processo de aprendizagem. Implementa um multicomputador constituído por um conjunto de nodos ligados por rede, tendo sido implementada em *PROLOG* utilizando o modelo *LINDA* para comunicação entre processos [Alves et al. 1993].

4.1.1 O modelo *LINDA*

O modelo *LINDA* desenvolvido por David Gelernter na Universidade de Yale [Ahuja et al. 1986] [Carriero & Gelernter 1989] [Freeman 1996], é uma ferramenta de alto nível para apoio à programação paralela, não suportada nem dependente de uma plataforma de hardware específica e assenta, tipicamente, num modelo de memória partilhada por um conjunto de processos distribuídos por várias plataformas computacionais.

A zona de memória do *LINDA*, designada por "*Tuple Space*" ou Quadro Negro (*QN*), além de suportar a comunicação entre processos permite-lhes também a partilha de conhecimento. O tipo de conhecimento que flúi neste espaço de memória partilhada apresenta-se sob a forma de tuplos. De forma a se poder manusear a informação residente no *QN*, são várias as operações que sobre este se concretizam, podendo-se enumerá-las (Tabela 4.1). Este conjunto único de operações foi aumentado no âmbito do trabalho presente com as operações da Tabela 4.2. Estas operações podem ser realizadas com ou sem suspensão do processo computacional em curso, consoante se pretenda uma execução síncrona ou assíncrona dos processos de acesso ao *QN*.

Tabela 4.1 – Operações primitivas *LINDA*.

Operação	Descrição
out(T)	- Permite a colocação do tuplo (ou termo) ***T*** no *QN*.
in(T)	- Testa se o tuplo ***T*** está ou não no *QN* e, se tal é o caso, remove-o.
rd(T)	- Verifica se o tuplo ***T*** está ou não no *QN* e, se tal for o caso, instância as variáveis em ***T***.
eval(T)	- Cria um novo processo em que ***T*** será avaliado, criando assim a possibilidade de processamento distribuído.

Tabela 4.2 – Operações extra baseadas em *LINDA*.

Operação	Descrição
In_noblock(T)	- Semelhante a ***in(T)*** com a diferença de neste caso o processo abortar se ***T*** não estiver disponível.
rd_noblock(T)	- Semelhante a ***rd(T)*** com a diferença de que neste caso o processo aborta se ***T*** não estiver disponível.
bagof_in_noblock(Template,T,Bag)	- ***Bag*** é a colecção de todas as instâncias de ***Template*** que ***T*** unifica em *TS*. Todos os ***T'***s são removidos.
bagof_rd_noblock(Template,T,Bag)	- Semelhante ao ***bagof_in_noblock(Template,T,Bag)*** com a diferença de que neste caso todos os ***T****s* mantêm-se no *TS*.

Um tuplo existe independentemente do processo que o criou. Na realidade, podem existir diversos tuplos independentes dos processos que os criaram os quais, em conjunto, podem mesmo constituir uma estrutura de dados no *QN*.

O modelo *LINDA* permite a distinção entre servidor e clientes quando há envio de mensagens, tanto no espaço como no tempo. As mensagens são separadas no tempo na medida em que não só a operação *out* é assíncrona mas também porque as mensagens circulam através de um espaço comum, o *QN*. Esta característica é deveras importante já que permite a comunicação em simultâneo de mensagens entre programas diferentes. As mensagens são separadas em espaço já que são identificadas pelo seu próprio conteúdo; i.e., nem o servidor nem o cliente têm conhecimento acerca da sua localização ou identidade, o que possibilita que os processos de comunicação se façam de um modo dinâmico. As acções de emissão ou de recepção de mensagens podem ser alocadas a qualquer processador do sistema ou mesmo transferidas de forma dinâmica entre processadores, sem que afectem os programas que aí se executam.

4.1.2 Controlo de execução e mecanismos de protecção

A necessidade de evitar a falha do sistema global, mesmo na situação de alguns dos seus componentes abortarem, levou à introdução de um mecanismo de meta-execução como unidade básica de controlo de execução e protecção do mesmo.

As vantagens e utilidade da meta-interpretação como técnica utilizada na programação em lógica é amplamente reconhecida tanto na teoria como na prática. Meta-interpretadores têm sido utilizados com evidentes vantagens, entre outras, no desenvolvimento de ferramentas algorítmicas de correcção de erros, controlo de expressões em programas lógicos ou para conseguir flexibilidade em sistemas baseados em conhecimento. Os meta-interpretadores em lógica mais divulgados estão ao nível da redução de cláusulas. O meta-interpretador que se lista de seguida é a forma mais comum de um tal procedimento:

```
exec(verdade).
exec((P,Q)) :-
        exec(P),
        exec(Q).
exec(P) :-
        clause((P:-Corpo)),
        exec(Corpo).
exec(P) :-
        sys(P),
        P.
```

Implementa Prolog puro acrescido de predicados do sistema, permitindo a execução ao nível meta de *exec(Objectivo)*. Esta meta-chamada executa simplesmente o objectivo *Objectivo* dentro de *exec*, sendo o resultado computacional equivalente ao obtido pela execução directa de *Objectivo*.

Um meta-interpretador à prova de falha com conjunção de objectivos como lista de atributos, pode agora ser equacionado através das operações básicas de modulação, incremento e mutação:

```
exec([G|TG],R) :-
        exec(G,R1),
        exec(TG,R2),
        and_result(R1,R2,R).
exec([],R) :-
        R=sucesso.
exec(verdade,R) :-
        R=sucesso.
exec(falso,R) :-
        R=falha.
exec(P,R) :-
        not_sys(P),
        clause((P:-Corpo)),
        exec(Corpo,R).
exec(P,R) :-
        sys(P),
        sys_exe(P,R).
```

onde a meta-chamada *exec(LG,R)* avalia a lista dada de objectivos *LG* e retorna sucesso no caso de ter sucesso, e falha se falhar. Assim previne-se a possibilidade de falha do sistema mesmo quando um objectivo *G* na lista *LG* falha.

As melhorias de desempenho são conseguidas por contagem de redução [Foster 1987]. Contagem de redução corresponde ao tempo computacional em sistemas convencionais. No meta-interpretador que se segue o argumento *CR* denota a contagem de redução corrente; i.e., o numero de vezes que regras de transformação são aplicadas a um determinado objectivo:

```
exec(LG,R,CR) :-
        exec(LG,R,0,CR).
exec([G|TG],R,CR,NCR) :-
```

```
        exec(G,R1,CR,CR1),
        exec(TG,R2,CR1,NCR),
        and_result(R1,R2,R).
exec([],R,CR,CR) :-
        R=sucesso(CR).
exec(verdade,R,CR,CR) :-
        R=sucesso(CR).
exec(falso,R,CR,CR) :-
        R=falha(CR).
exec(P,R,CR,NCR) :-
        not_sys(P),
        clause((P:-Corpo),CR,CR1),
        exec(Corpo,R,CR1,NCR).
exec(P,R,CR,NCR) :-
        sys(P),
        sys_exe(P,R,CR,NCR).
```

Os predicados *clause* e *sys_exe* incrementam a contagem de redução cláusula a cláusula, quando o cálculo lógico tem realmente lugar.

4.1.3 Estrutura global do sistema

As comunicações entre as unidades computacionais do sistema é fornecida pelo *LINDA*. A estrutura do sistema em camadas é mostrada na Figura 4.1, cuja implementação foi efectuada em computadores pessoais com sistemas operativos *LINUX* e *Microsoft Windows* sobre uma rede *Ethernet* com *TCP/IP*. A perspectiva da passagem de mensagens juntamente com o modelo *LINDA* foi utilizada para se conseguir a distribuição do sistema.

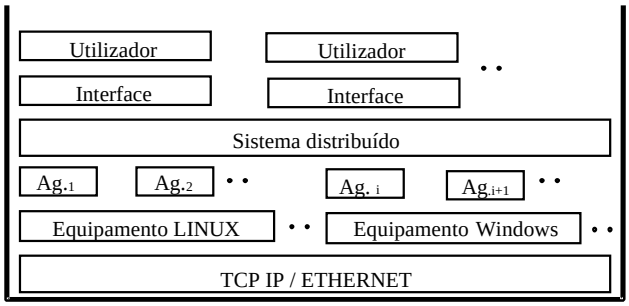


Figura 4.1 – Estrutura global por camadas do sistema

4.1.3.1 Processos, intercomunicação e nodos do sistema

Os nodos, processos e fluxo de mensagens são mostrados na Figura 4.2. As setas denotam o fluxo de mensagens entre processos. Cada nodo do sistema está identificado por uma referência e é considerado autónomo com controlo local.

O controlo local é efectuado pelo processo *servidor* que gere a comunicação no espaço de tupletos; i.e., no *QN*, pelo processo *aprendizagem* que gere as regras de distribuição, pelo processo *monitor* que gere as mensagens de controlo e pelo processo *comunicação* que é responsável pela transferência de mensagens entre nodos. Existem ainda outros dois tipos de processos, os processos *shell* que tratam da interface com os utilizadores e os processos *processador* que constituem as unidades de processamento do sistema.

Os processos *servidor*, *aprendizagem*, *monitor* e *comunicação* são processos que existem enquanto o sistema corre. *Shell* são processos cuja existência depende dos utilizadores e *processador* são processos que dependem do problema a resolver.

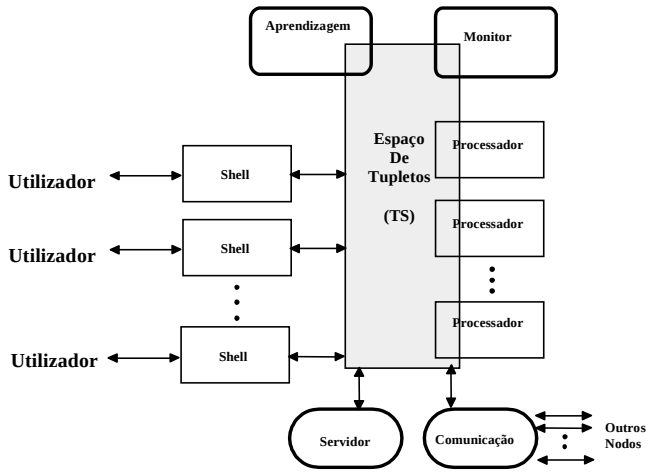


Figura 4.2 – Nodo do sistema, seus processos e fluxo de mensagens

4.1.3.2 O processo *Servidor*

O *servidor* é o processo que gere a comunicação no *QN*. Trata-se de um processo que pode correr em qualquer dos equipamentos do sistema.

A descrição de alto nível do processo *server* pode ser expressa do seguinte modo:

```
server(S0) :-
        waits_for_arrival(S0,RS,S1),
        server2(RS,S1,S2),
        server(S2).
server2([],S,S).
server2([IS|SSs],SS0,SS) :-
        server_one_stream(IS,SS0,SS1),
        server2(SSs,SS1,SS).
```

O predicado de meta-nível *server* tem um só argumento, *S0*, que denota a lista de todas as referências de ligações (i.e., pares de endereços de máquinas – porta usada na ligação) dos equipamentos com o processo servidor. Trata-se, basicamente, de um predicado recursivo não terminal; i.e., recursividade infinita, que gere todas as

ligações com os outros processos no sistema. O predicado *wait_for_arrival* espera pela chegada de pedidos de ligação sendo nesse caso acrescentadas à lista *S0*, dando *S1*. Verifica também todas as ligações existentes em *S0* à procura de dados, dando a lista de ligações *RS*.

O predicado *server2* responde a todos os pedidos de operações *LINDA*, estando escrito em termos do predicado *server_one_stream*, o qual interpreta as primitivas *LINDA*.

4.1.3.3 O processo *Shell*

A *shell* é o processo que trata da interacção com o mundo exterior (i.e., com os utilizadores). Recebe os dados dos utilizadores, efectua a sua tradução para a mensagem apropriada e coloca-a no *QN*. A utilização de mensagens permite que os diversos processos *shell* corram dum modo assíncrono em relação aos restantes processos do sistema e entre eles próprios. Trata-se de um processo que pode correr em qualquer dos equipamentos do sistema.

A descrição alto nível do processo *shell* pode ser expressa do seguinte modo:

```
shell(NodeID,Address) :-
        linda_client(NodeID,Address),
        shell,
        close_client.
```

O predicado de meta-nível *shell(NodeID,Address)* tem dois argumentos, *Address* que denota o endereço *TCP/IP (Hostname:Portnumber)* do servidor do *QN* e *NodeID* que denota a identificação do nodo a que se pretende ligar. Efectua uma ligação ao servidor *LINDA* através do predicado *linda_client(NodeID,Address)* e efectua todo o serviço de interface até que o utilizador requeira a sua desactivação, encerrando a ligação *(close_client)*.

4.1.3.4 O processo *Monitor*

A principal função do processo *monitor* é a gestão e execução das mensagens de controlo. Estas funções são monitorizadas pelo processo *aprendizagem*, em especial para a geração de novos processos e nodos da rede computacional. Este processo desempenha um papel central no sistema e pode correr em qualquer dos equipamentos do sistema.

A descrição alto nível do processo *monitor* pode ser expressa do seguinte modo:

```
monitor(NodeID,Address) :-
        linda_client(NodeID,Address),
        message_monitoring,
        close_client.
```

O predicado de meta-nível monitor*(NodeID,Address)* tem dois argumentos, *Address* que denota o endereço *TCP/IP (Hostname:Portnumber)* do servidor do *QN* e *NodeID* que denota a identificação do nodo a que se pretende ligar. Efectua uma ligação ao servidor *LINDA* através do predicado *linda_client(NodeID,Address)* e efectua todo o serviço de controlo e monitorização até que o utilizador requeira a sua desactivação, encerrando a ligação (i.e., através da invocação de *close_client*).

O predicado *message_monitoring* é definido do seguinte modo:

```
message_monitoring :-
        in(monitor(Message)),
        do(Message).
do(end).
do(write(X)) :-
        write(X),
        message_monitoring.
do(eval(Goal,Domain,Modules)) :-
        convert(Goal,Domain,Modules,Problem),
        out(learning(suggest(Problem))),
```

```
        in(monitor(suggest(Problem,Danswer))),
        new_processor(Goal,Domain,Modules,Danswer),
        message_monitoring.
...
```

O predicado *message_monitoring* vai interpretando as mensagens de controlo que surjam no *QN* ficando num estado de suspensão, quando no *QN* não se encontrar nenhuma mensagem. A mensagem *eval* requer a execução do objectivo *Goal* utilizando os módulos enumerados na lista *Modules* no domínio do *QN*. Isto é, convertendo *Goal*, *Domain* e *Modules* para um formato adequado e enviando uma mensagem com este problema *out(learning(suggest(Problem)))* ao processo *aprendizagem*, suspendendo a sua execução até receber uma resposta *in(monitor(suggest(Problem,Danswer)))*, em termos das regras a aplicar na distribuição dos módulos *(Danswer)*. Isto é efectuado pelo predicado *new_processor* que utiliza ou gera novos processos *processor* de acordo com as regras em *Danswer*. Mensagens com o objectivo, nome dos módulos, e domínio axiomático, são então enviados aos processos *processor*.

4.1.3.5 O processo *Aprendizagem*

A principal função do processo a*prendizagem* tem a ver com a gestão das regras de distribuição de cargas computacionais por máquina. Implementa um processo de aprendizagem baseado num sistema de classificação; i.e., um sistema de aprendizagem que aprende regras em texto sintacticamente simples, através de produções denominadas classificadores, para controlo do seu desempenho [Goldberg 1989] [Santos 1999]. Este processo desempenha um papel central no sistema e pode correr em qualquer dos equipamentos da rede de computadores.

A descrição de alto nível do processo a*prendizagem* pode ser expressa na forma:

```
learning(NodeID, Address) :-
```

```
        linda_client(NodeID, Address),
        message_learning,
        close_client.
```

O predicado de meta-nível *learning(NodeID,Address)* possui dois argumentos, *Address* que denota o endereço *TCP/IP (Hostname:Portnumber)* do servidor do *QN* e *NodeID* que denota a identificação do nodo a que se pretende ligar. Efectua uma ligação ao servidor *LINDA* através do predicado *linda_client(NodeID,Address)* e trata as mensagens que lhe são dirigidas até que o utilizador requeira a sua desactivação, encerrando a ligação *(close_client)*.

A extensão do predicado *message_learning* é definido em termos das produções:

```
message_learning :-
        in(learning(Message)),
        do(Message).
do(end).
do(suggest(Problem)) :-
        detector(Problem, MessageL),
        match(MessageL, AclassifierL),
        auction(AclassifierL, WclassifierL),
        clearinghouse(WclassifierL),
        taxcollect(AclassifierL),
        postmessage(WclassifierL),
        effector(WclassifierL, Answer),
        out(monitor(suggest(Problem,Danswer))),
        reward,
        ga,
        message_learning .
```

O predicado *message_learning* interpreta as mensagens que lhe são dirigidas e que surjam no *QN*, auto-suspendendo-se, quando no *QN* não há mais mensagens a processar. A mensagem *suggest* é enviada pelo processo *monitor* e solicita regras para a distribuição das cargas computacionais com interesse para a resolução do problema *Problem* [Santos & Neves 1993] [Santos 1999].

4.1.3.6 O processo *Processador*

A principal função do processo processador passa pelo processamento de informação. Os processos *processador* representam as unidades básicas de processamento do sistema, podendo correr em qualquer dos equipamentos do sistema.

Uma descrição de alto nível do processo *processador* (*processor*) pode ser expressa pelas produções:

```
processor(NodeID,Address,Ref) :-
        linda_client(NodeID,Address),
        message_processing(Ref),
        close_client.
```

O predicado de meta-nível *processor(NodeID,Address,Ref)* apresenta-se com três argumentos, *Address* que denota o endereço *TCP/IP (Hostname:Portnumber)* do servidor do *QN*, *NodeID* que denota a identificação do nodo a que se pretende ligar e *Ref* que denota a referência unívoca deste processo. Efectua uma ligação ao servidor *LINDA* através do predicado *linda_client(NodeID,Address)* e processa todas as mensagens que lhe são dirigidas até que o utilizador requeira a sua desactivação, encerrando a ligação através da invocação de *close_client*.

A extensão do predicado *message_processing* é definida em termos das produções:

```
message_processing(Ref) :-
        in(processor(Ref,Message)),
        do(Ref, Message).
do(_, end).
do(Ref, write(X)) :-
        write(X),
        message_processing(Ref).
do(Ref, eval(system, Domain, M)) :-
```

```
        system(Domain,M),
        message_processing(Ref).
...
```

O predicado *message_processing* vai interpretando as mensagens que lhe são dirigidas que surjam no *QN*, auto-suspendendo-se quando no *QN* não existirem quaisquer mensagens a processar.

4.1.3.7 O processo *Comunicação*

A principal função do processo *comunicação* é a de garantir a comunicação entre nodos. O processo *comunicação* é responsável pela transferência de mensagens entre os diferentes nodos da rede. Trata-se de um processo que pode correr em qualquer dos equipamentos do sistema.

Uma descrição de alto nível do processo *comunicação* (i.e., "communication") pode ser expressa em termos das produções:

```
communication(NodeID,Address) :-
        linda_client(NodeID,Address),
        message_passing(NodeID),
        close_client.
```

O predicado de meta-nível *communication(NodeID,Address)* surge com dois argumentos, *Address* que denota o endereço *TCP/IP (Hostname:Portnumber)* do servidor do *QN* e *NodeID* que denota a identificação do nodo a que se pretende ligar. Efectua uma ligação ao servidor *LINDA* através do predicado *linda_client(NodeID,Address)* e trata do envio para outros nodos das mensagens que lhes são dirigidas até que o utilizador requeira a sua desactivação, encerrando a ligação com a invocação de *close_client*.

A extensão do predicado *message_passing* é dada pelas produções:

```
message_passing(NodeID) :-
        in(communication(Message)),
        do(NodeID, Message).
do(_, end).
do(NodeID, new_connection(To_NodeID,Address)) :-
        linda_client(To_NodeID,Address), message_passing(NodeID).
do(NodeID,node_message(To_NodeID,Message)) :-
        out(To_NodeID,node_message(NodeID,Message)),
        message_passing(NodeID).
```

O predicado *message_passing* interpreta as mensagens que lhe são dirigidas e que surgem no *QN*, auto-suspendendo-se, quando no *QN* não há mais mensagens a processar. A mensagem *new_connection* é interpretada como denotando o pedido para uma ligação ao nodo *To_NodeID*; i.e., este processo torna-se um cliente *LINDA* de um outro servidor *LINDA*. A mensagem *node_message* trata da transferência de mensagens entre nodos (i.e., do nodo *NodeID* para o nodo *To_NodeID*).

4.1.3.8 O Arranque do Sistema

O arranque do sistema passa por se activarem todos os processos críticos do sistema, o que é expresso na forma:

```
├ ∀R ∀RC
        exec([server([])], R, RC).
├ ∀NodeID ∀Address ∀R ∀RC
        exec([monitor(NodeID, Address)],R,RC).
├ ∀NodeID ∀Address ∀R ∀RC
        exec([learning(NodeID,Address)],R,RC).
├ ∀NodeID ∀Address ∀R ∀RC
        exec([communication(NodeID,Address)],R,RC).
```

Em que "├" denota a relação de derivabilidade. A distribuição dos diversos nodos do sistema por uma rede de computadores é conseguida através de uma hierarquia de nodos, todos ligados a um processo de aprendizagem central (Figura 4.3). O

processo de aprendizagem central gere as regras de distribuição de conjuntos de nodos e está activo enquanto o sistema esta activo. O seu comportamento é semelhante ao do processo *aprendizagem* que se aplica nodo a nodo, com a diferença de que as regras que gera serem relativas à distribuição global de conjuntos de nodos [Santos 1999].

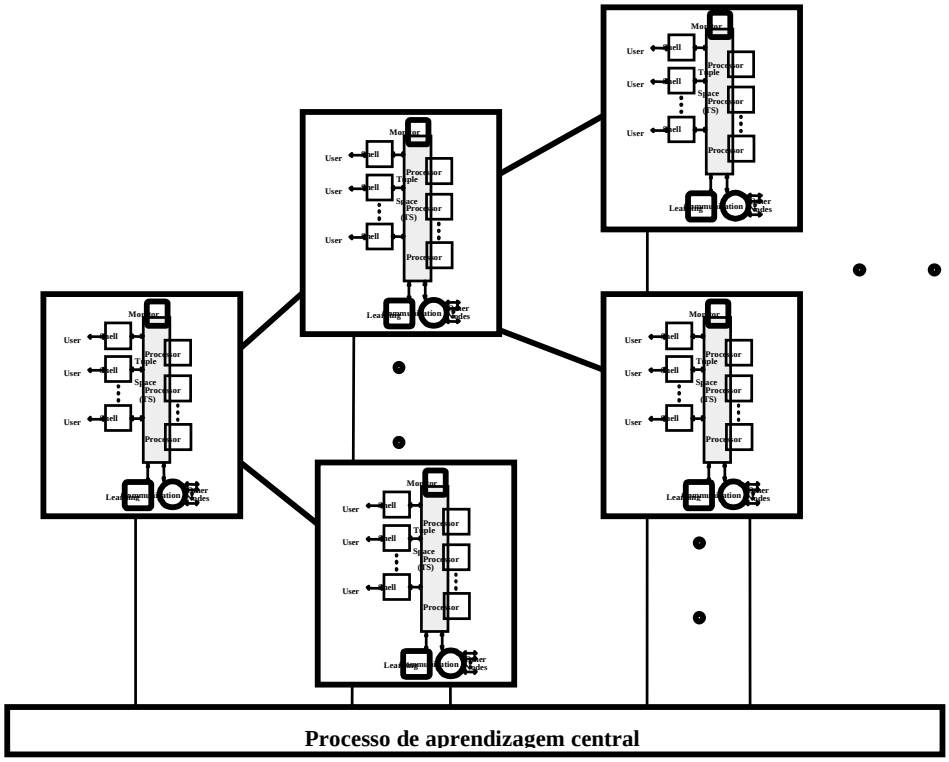


Figura 4.3 – Uma arquitectura universal para a associação de conjuntos de nodos

4.2 A Arquitectura do Sistema *SADMED*

A arquitectura do sistema *SADMED* é dada na Figura 4.4, com base nas suas entidades constituintes:

O Quadro Negro – É uma zona de memória partilhada pelos agentes do sistema, que a utilizam como plataforma de comunicação entre si e para partilha de estruturas de dados e informação de controlo e coordenação, de uma forma assíncrona e estruturada. Denota o ambiente em que operam os agentes do sistema;

O Agente Monitor – Permite a observação e o registo do funcionamento do sistema. Faz a interface com o administrador e pode, em algumas circunstâncias, tomar parte activa no controlo do sistema;

Os Agentes de Apoio Médico – Encarregados da interface entre os médicos e o sistema;

Os Agentes de Conhecimento – Repositórios de conhecimento (e.g., baseados em sistemas simbólicos, *RNAs* ou híbridos, que contribuem activamente para a elaboração do diagnóstico). Incorporam, embora em pequena escala, procedimentos de coordenação e controlo;

Os Agentes Servidor – Encarregados do registo e manutenção da informação para o funcionamento do sistema. Têm parte activa no seu controlo e na interacção com o mundo exterior; i.e., com o ambiente exterior (ao nível da recolha de informação dos equipamentos médicos e interacção com outros sistemas de informação porventura já instalados); e

O Agente Recursos – Encarregado de gerir os recursos dos equipamentos, em especial os relacionados com a ocupação da memória secundária e cópias de segurança.

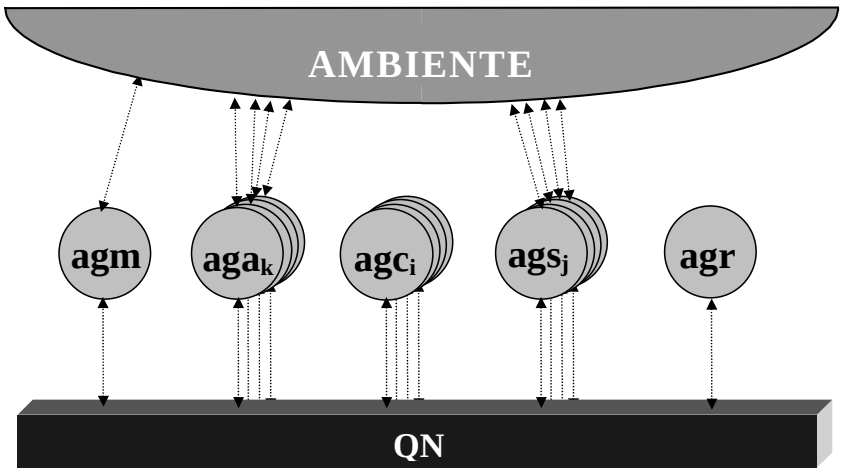


Figura 4.4 – Arquitectura do sistema *SADMED*.

Para descrever o sistema *SADMED* seguiu-se a via sugerida por Cavedon e Tilhar com recurso à *PLC* [Cavedon & Tilhar 1995], e com recurso à *PLE* [Neves et al. 1997] [Santos & Neves 1998] e a considerações de natureza geométrica ou social intrínsecas ao universo de discurso [Denti et al. 1995].

Definição 4.1 (O Sistema SADMED)

Formalmente, o sistema *SADMED* dá forma a um multi-sistema $\Xi = \langle Csadmed, \Delta sadmed, agm, aga_1,..., aga_k, agc_1,..., agc_i, ags_1,..., ags_j, agr, qenv \rangle$, com $i,j,k \geq 1$, em que:

Csadmed – denota o contexto em que o sistema se insere e opera, definido em termos de uma teoria ou programa em lógica dada pelo tripleto $\langle Lg, Ax, \Delta \rangle$, onde *Lg* é a linguagem de programação em lógica, neste caso uma extensão à programação em lógica [Traylor & Gelfond 1993] [Neves 1984], *Ax* o conjunto de axiomas de *Lg* e Δ o conjunto de regras de inferência;

Δsadmed – denota o conjunto das regras-ponte, que definem o processo de interacção entre os componentes do sistema. Definem a geometria do sistema, dando uma medida do significado e da interacção possível entre as suas partes constituintes. Estabelecem o modo como os eventos que ocorrem ao nível dos subsistemas ou agentes são coordenados, e expressam o grau de paralelismo presente no sistema;

agm – denota o agente monitor;

$\boldsymbol{aga_k}$ – denota um agente de apoio ao diagnóstico médico ($k=1...o$), em que o é um parâmetro do sistema;

$\boldsymbol{agc_i}$ – denota um agente de conhecimento ($i=1...m$), em que m é um parâmetro do sistema;

$\boldsymbol{ags_j}$ – denota um agente servidor ($j=1...n$), em que n é um parâmetro do sistema;

agr – denota o agente recursos; e

qenv – denota o ambiente ou universo de discurso.

4.3 Agentes do Sistema *SADMED*

O sistema *SADMED* é um Sistema de Processamento Distribuído (*SPD*); i.e., um sistema computacional no qual um conjunto de entidades semi-autónomas e heterogéneas interagem, através de uma insuspeita e terceira entidade, o *QN*, que num processo de consolidação, a um nível meta, criam um Sistema Computacional

(*SC*). Como se pode observar, o *QN* denota um espaço de memória partilhado, um meio de comunicação entre os demais componentes do sistema. Todas as interacções entre os agentes do sistema efectuam-se através do *QN*, colocando ou retirando informação com significado no contexto *Csadmed*.

Os agentes de conhecimento, agente servidor e agente monitor que compõem o sistema *SADMED* são similares ao agente reactivo com estado interno apresentado em [Russel & Norvig 1995] e [Weiss 1999], possuindo uma arquitectura do tipo não deliberativo [Coelho 1995]. O comportamento destes agentes pode ser equacionado através de um processo de transição de estado. Deste ponto de vista assume-se que o estado do ambiente de um agente pode ser caracterizado como um conjunto (possivelmente infinito) $EAA=\{eaa_1, eaa_2, ...\}$ de estados *ecológicos*. Num determinado instante, o ambiente encontra-se num destes estados. A capacidade do agente actuar no ambiente pode, por outro lado, ser representada pelo conjunto $Ev=\{ev_1, ev_2, ...\}$ de *eventos* (acções). Considere-se *Per* como o conjunto dos estímulos que o agente pode receber do ambiente, e *Int* o conjunto de todos os seus estados internos. Num determinado instante o agente é caracterizado pelo estado interno em que se encontra. Com base nestes postulados é possível então descrever de forma abstracta o comportamento do agente através das funções que são apresentados a seguir:

observa – $EAA \rightarrow Per$, que faz o mapeamento de um estado ecológico numa percepção do agente;

pensa – $Int \times Per \rightarrow Int$, que possibilita o mapeamento de um estado interno do agente e de um estímulo recebido do ambiente num novo estado interno do agente ($\times$ denota produto cartesiano); e

actua – Int → Ev, que se encarrega de mapear um estado interno do agente numa acção a executar sobre o ambiente.

Algoritmo 4.1 (Ciclo de Vida dos Agentes).

O comportamento de um agente (Figura 4.5) pode ser, em abstracto, dado pelo algoritmo ou procedimento:

proc *Agente* ≡
    seja *Ev={ev_1, ev_2, ...} o conjunto de acções que representam a aptidão de o agente actuar sobre o ambiente*
    seja *Per o conjunto (não vazio) dos estímulos que o agente pode receber do ambiente*
    seja *int_0 o estado de conhecimento inicial do agente*
    Estado ← int_0
    inicia(Estado)
    repetir
        Estímulo ← observa(Per)
        Estado ← pensa(Estado, Estímulo)
        Atitude ← actua(Estado, Ev)
    até *Atitude=fim_de_sessão*
    fim

O *modus vivendi* de um agente é descrito por uma sequência de transições de estado, na forma de *observa*, *pensa* e *actua*. O agente começa com um determinado estado inicial int_0. Então observa que o seu ambiente se encontra num estado *eaa*, e gera um estímulo *observa(eaa)*. O estado interno do agente é actualizado e passa a igualar o resultado da função *pensa(int_0, observa(eaa))*. Esta acção é então executada sobre o ambiente e o agente entra noutro ciclo, perscrutando o mundo via a função *observa*, actualizando o seu estado interno via a função *pensa* e seleccionando e perpetrando uma acção a executar via a função *actua*. O critério de paragem do ciclo de vida do agente passa por se desligar do sistema (i.e., o processo computacional que está associado ao agente é cancelado ao nível do sistema operativo da(s) máquina(s) em que o agente opera).

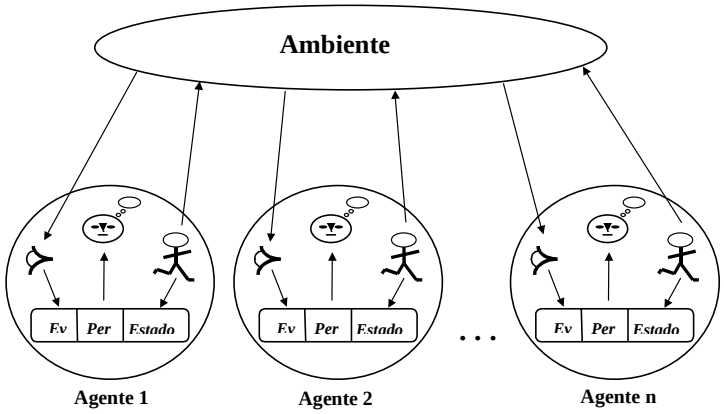


Figura 4.5 – Comportamento dos agentes no sistema *SADMED*.

4.4 Meio Social

Um agente deve estar habilitado a comunicar de forma a atingir, o mais eficientemente possível, os objectivos do sistema ou sociedade onde está inserido. A comunicação é devida em parte à função de percepção (a recepção de mensagens) e à função de actuação (envio de mensagens). Deste modo os agentes de um *SPD* poderão coordenar as suas acções e comportamentos, tornando o sistema mais coerente[5].

4.4.1 Coordenação

A coordenação de actividades entre agentes, embora podendo ser obtida de várias formas é, na maioria dos casos, concretizada através da cooperação ou competição entre os agentes do sistema. A cooperação pode-se desenvolver através de

procedimentos de coordenação entre agentes ditos benevolentes, e pressupõe a existência de sociedades organizadas. Na competição, por outro lado, os agentes olham apenas ao seu bem-estar e interesses pessoais; o todo é sacrificado às partes.

No sistema *SADMED* os agentes de conhecimento comunicam entre si e agem de uma forma cooperativa, na medida em que contribuem activamente para a elaboração de um diagnóstico.

O modelo de coordenação incorporado no sistema *SADMED* passa fundamentalmente pela sincronização das acções dos agentes e, desenvolve-se segundo quatro eixos base [Ferber 1999], os quais são enunciados no que se segue:

Características temporais, rapidez, adaptabilidade e antecipação são os adjectivos que permitem tipificar a coordenação de um sistema com a coordenada tempo. Neste capítulo o sistema *SADMED* pode ser considerado:

- *Rápido*, na medida em que possui capacidades mínimas de raciocínio, evitando assim situações em que possa ficar "pendurado". A rapidez de reacção, naturalmente, também depende de outros factores como a complexidade computacional inerente às acções a executar, o número de agentes presentes no sistema e o número de tarefas;
- *Não adaptativo*, por não se adaptar a situações novas uma vez que o problema é conhecido à priori; e
- *Não antecipatório*, não consegue prever situações futuras para agir por antecipação (trata-se de um sistema essencialmente reactivo).

Características organizativas, descrevem a forma como a coordenação está organizada e o seu grau de centralização, podendo distinguir-se, entre outras:

[5] Coerência denota uma medida de eficiência do sistema quando visto como um todo.

- *A Estrutura da organização*, é definida segundo um eixo de centralização/distribuição. Num extremo posicionam-se as estruturas de coordenação centralizadas, mais simples de implementar e mais coerentes (i.e., o sistema, como um todo, é mais eficiente), no outro as estruturas distribuídas mais habilitadas à adaptação a modificações do ambiente e até ao mau funcionamento de alguns agentes. A filosofia adoptada no sistema pende para o lado distribuído do eixo na medida em que o comportamento dos agentes é ditada, na sua quase totalidade, pelas modificações no ambiente, através dos agentes servidor ou através de mensagens colocadas no *QN* pelos outros agentes. Os agentes de conhecimento podem em algumas circunstâncias coordenar as suas actividades, o que acontece quando se olha a um tipo de cooperação como a horizontal;

- *O Modo de comunicação*, no caso em que os agentes têm que endereçar problemas comuns, como é o dia-a-dia do sistema *SADMED*, o objectivo da comunicação passa por manter um bom desempenho dos agentes do sistema sem violar as suas prerrogativas autonómicas. A adopção do *QN* como meio de comunicação vai neste sentido; e

- *A Liberdade de acção***,** está por definição ligada ao grau de independência que um agente possui, em relação ao seu comportamento, condicionado pelas instruções que pode receber ou não por parte de um coordenador ou em relação ao(s) protocolo(os) de coordenação. Neste campo os agentes do sistema *SADMED* apresentam um baixo grau de independência por estarem fortemente confinados às tarefas inerentes a um *SADM* e, por isso, sincronizados com o seu ciclo de vida.

Características de qualidade e eficiência, a qualidade e eficiência que é possível aferir pelos resultados obtidos na aplicação de um sistema e que estão necessariamente dependentes de aspectos como o comportamento dos agentes, a gestão dos recursos, o desempenho, o número de agentes e a minimização de conflitos (situações de estrangulamento e contenção). No desenvolvimento do sistema *SADMED* houve uma preocupação, logo à partida, de se conseguir um sistema com capacidade de resposta em tempo real, optando-se desde então por uma organização social estática e pré-definida, e uma forma de coordenação linear. Os resultados obtidos no Capítulo 5 revelam os níveis de eficiência associados ao sistema *SADMED*.

Características de realização, apontam para os meios requeridos na concretização do sistema de coordenação:

- *Grau e quantidade de dados*, volume de informação que os agentes devem trocar entre si para efeitos de coordenação foi mantida no mínimo possível, reduzindo-se assim as comunicações necessárias;
- *Grau de representações mútuas*, a informação necessária acerca dos agentes é mantida em estruturas de dados e de conhecimento no seio do *QN*. Os agentes consultam-nas sempre que necessitam, sem ter que manter cópias das mesmas; e
- *Dificuldade de implementação*, o sistema *SADMED* obedece a uma especificação formal, fácil de implementar e manter.

Características de generalização, neste ponto avalia-se o grau de heterogeneidade do sistema. O processo de coordenação do sistema *SADMED* foi totalmente orientado para o problema dos *SADM*, integrando os diversos tipos de agentes necessários à concretização/realização de um processo de diagnóstico.

4.4.2 Comunicação

A linguagem utilizada na comunicação inter-agente, no caso do sistema *SADMED*, permite o envio e recepção de mensagens simples cuja sintaxe se apresenta na Tabela 4.3 e cujo conteúdo semântico é dado pela função,

demo(Contexto, Mensagem)→ {verdadeiro, falso, desconhecido},

em termos das produções:

demo(Contexto, Mensagem, verdadeiro) ← Mensagem;
demo(Contexto, Mensagem, falso) ← ¬ Mensagem; e
demo(Contexto, Mensagem, desconhecido) ← não Mensagem, não ¬ Mensagem.

em que "*{*" e "*}*" é a notação para conjuntos, "*¬*" denota "*negação forte*", "*não*" denota negação por falha na prova, e "*verdadeiro*", "*falso*", "*desconhecido*" denotam respectivamente as constantes lógicas *verdadeiro*, *falso* e *desconhecido* e, "→" denota "→".

A comunicação entre os agentes do sistema segue a classificação sugerida por Ferber [Ferber 1999], e que é dada nos termos expressos pelo texto que se segue:

Ligação entre o remetente e o endereço, a comunicação entre dois agentes pode desenvolver-se de duas formas distintas (os agentes do sistema *SADMED* utilizam ambas):

- *Ponto a ponto ou personalizada*, o agente remetente envia uma mensagem a um dado agente, o destinatário, identificando-o na mensagem; e

- *Difusa*, o agente remetente envia uma mensagem, sendo, neste caso, potenciais destinatários todos os agentes activos no sistema.

Natureza do meio ou canal de comunicação, podem identificar-se quatro tipos diferentes de canais de comunicação:

- *Reencaminhamento directo*, quando um agente envia uma mensagem a terceiros passa-a ao canal de comunicação, o qual se encarrega, por sua vez, de a fazer chegar directamente ao destinatário (análogo ao processo de entrega de uma carta ou a um processo de correio electrónico);
- *Reencaminhamento por propagação de sinal*, um agente envia uma mensagem (sinal) que é difundida pelo ambiente e cuja intensidade vai decrescendo à medida que o tempo avança; e
- *Reencaminhamento por anúncio público*, um agente, quando pretende comunicar, coloca a mensagem num espaço comum (e.g., o *QN*), que é visível por todos os restantes agentes do sistema. É este o tipo de comunicação incorporado no sistema *SADMED*.

Intenção da comunicação, considera-se intencional todo o processo de comunicação encetado voluntariamente por um agente, por oposição à comunicação não intencional, que se dá de uma forma independente do emissor (e.g., a linguagem corporal). No sistema *SADMED* a comunicação é sempre intencional na medida em que os agentes não possuem conhecimento acerca da dinâmica e informação residente nos outros agentes.

A linguagem de comunicação aqui utilizada (i.e., produções em *PLE*) permite que os agentes troquem mensagens entre si, o que não torna desnecessário, porém, a existência de um protocolo que governe a troca de mensagens ao nível global; i.e., entre todos os agentes do sistema. No sistema *SADMED*, tal protocolo é

naturalmente incorporado por via da utilização da filosofia *Sistema Baseado em Quadros Negros* (*SBQN*). O *QN* disponibiliza uma zona de memória onde os agentes colocam as mensagens que pretendem enviar e onde consultam as mensagens que lhes estão destinadas. Uma vez que este não interfere no processo de escrita-envio e leitura-recepção de mensagens entre os agentes, não se estão a utilizar processos de comunicação do tipo assistido [O'Hare & Jennings 1996].

Tabela 4.3 – Exemplos de Mensagens.

Mensagem	Descrição
qn(Directiva)	- A mensagem *Directiva* é enviada ao *QN* para que seja consultada por todos os agentes.
qn(agc(R,Directiva))	- A mensagem *Directiva* é enviada ao *QN* para que seja consultada pelo agente de conhecimento identificado por *R*.
qn(ags(R,Directiva))	- A mensagem *Directiva* é enviada ao *QN* para que seja consultada pelo agente servidor identificado por *R*.
qn(aga(R,Directiva))	- A mensagem *Directiva* é enviada ao *QN* para que seja consultada pelo agente de apoio médico identificado por *R*.
qn(agr(Directiva))	- A mensagem *Directiva* é enviada ao *QN* para que seja consultada pelo agente recursos.
qn(agm(Janela,Descrição))	- A mensagem *Descrição* é enviada ao *QN* para que seja consultada pelo agente monitor a fim de a colocar no quadro identificado por *Janela*.

No que concerne ao relacionamento inter-agentes, identificam-se diversas formas distintas de o concretizar, as quais são enunciados no texto que se segue:

Agentes Servidor/Agentes de Conhecimento, com meios de comunicação destinados a enviar dados para treino das *RNAs*, ou para gerar o diagnóstico para os agentes de apoio médico;

Agentes Servidor/Agente Monitor, Agentes de Apoio ao Diagnóstico Médico/Agente Monitor, Agente Recursos/Agente Monitor, e Agentes de

Conhecimento/Agente Monitor, com meios de comunicação destinados a enviar dados para serem visualizados e anotados através do agente monitor;

Agente Monitor/Agentes Servidor, Agente Monitor/Agentes de Apoio ao Diagnóstico Médico, Agente Monitor/Agente Recursos e Agente Monitor/Agentes de Conhecimento, com meios de comunicação destinados a enviar ordens oriundas do administrador do sistema;

Agentes de Conhecimento/Agente Recursos e Agentes Servidor/Agente Recursos, com meios de comunicação destinados a enviar dados relativos à utilização de recursos;

Agente Recursos/Agentes Servidor e Agente Recursos/Agentes de Conhecimento, com meios de comunicação associados à gestão dos recursos do sistema;

Agentes de Conhecimento/Agentes de Conhecimento, com meios de comunicação que permitem aos agentes do sistema cooperar e coordenar-se entre si;

Agentes de Conhecimento/Agentes de Apoio ao Diagnóstico Médico, com meios de comunicação que permitem que os agentes de conhecimento do sistema dêem a conhecer os diagnósticos entretanto realizados;

Agentes de Conhecimento/Agentes Servidor, com meios de comunicação destinados a solicitar dados para treino das *RNAs*, ou para gerar o diagnóstico para os agentes de apoio médico; e

Agentes de Apoio ao Diagnóstico Médico/Agentes de Conhecimento, com meios de comunicação destinados a facilitar a apresentação de um diagnóstico médico ao(s) utilizador(es) do sistema.

Na Figura 4.6 apresenta-se a geometria do sistema *SADMED*, com um significado: o da interacção possível entre os seus agentes e o seu ambiente (i.e., uma medida da interpretação a associar a cada agente bem como aos (seus) possíveis relacionamentos).

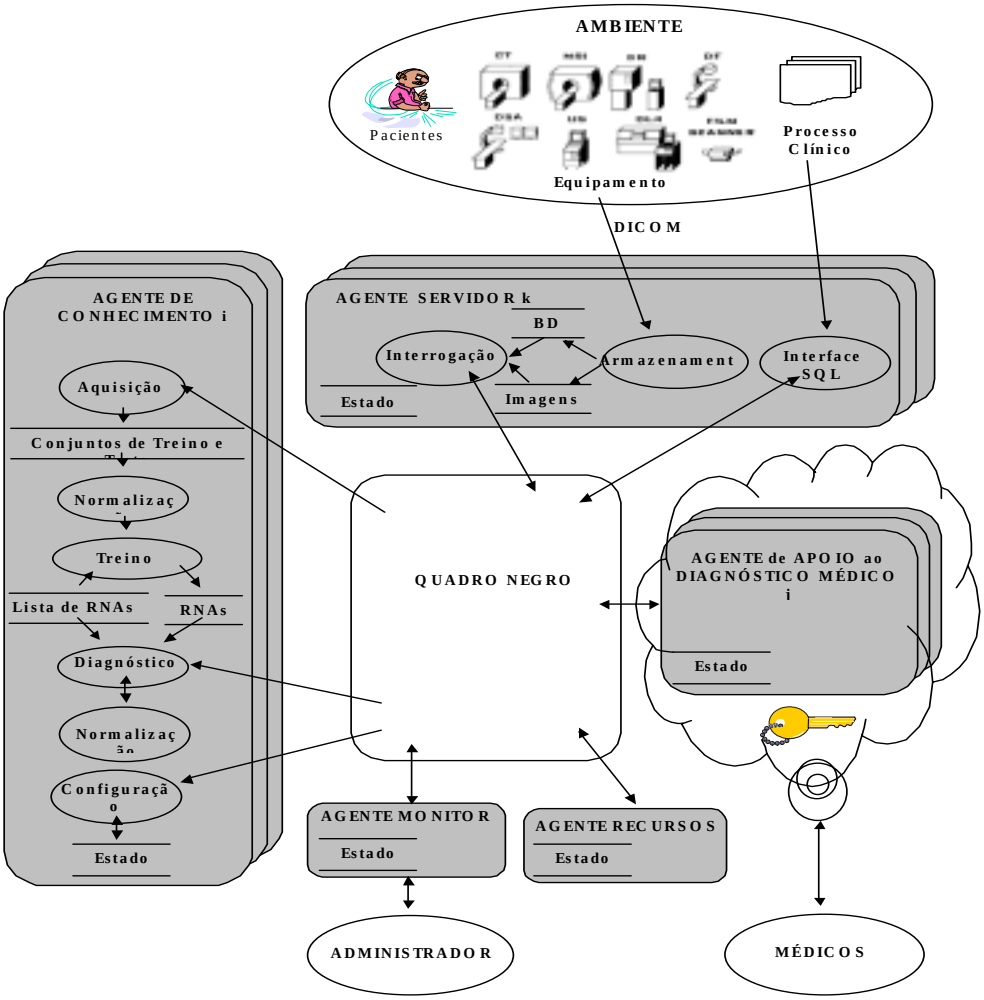


Figura 4.6 – Geometria do sistema *SADMED*.

A geometria do sistema é retratada através da utilização de regras-ponte [Cavedon & Tilhar 1995] [Denti et al. 1995] [Santos & Neves 1998] (i.e., Δ_{sadmed} é agora definido em termos das regras-ponte referidas na Tabela 4.4). A coordenação a estabelecer entre os diferentes agentes que corporizam o sistema é obtida através da definição de um conjunto de relacionamentos entre estes face aos acontecimentos que os próprios desencadeiam.

Tabela 4.4 – Regras-ponte para o sistema *SADMED*.

	Regra-Ponte	**Descrição**
1)	C_{agm}*: ocorre(activa)* ——— *ocorre(activa)*	Quando o agente monitor é activado, também todos os outros agentes do sistema devem ser activados.
2)	C_{agsi}*: ocorre(exame_dicom)* ——— C_{agr}*: ocorre(local_armazenamento)*	Quando é efectuado um novo exame médico num dos equipamentos *DICOM*, o agente servidor correspondente procede à classificação e armazenamento desse exame. Uma mensagem é enviada ao agente recursos com vista a saber qual o(s) local(ais) onde esse exame deve ser armazenado. Após a recepção da resposta por parte do agente recursos é efectuado a gravação do exame.
3)	C_{agr}*: ocorre(local_armazenamento)* ——— C_{agsi}*: ocorre(local_exame)*	Quando o agente recursos recebe uma mensagem relativa ao armazenamento de exames, envia a resposta tendo em atenção a sua base de conhecimento para armazenamento do exame.
4)	C_{agak}*: ocorre(pedido_diagnóstico)* ——— C_{agsi}*: ocorre(pedido_informação)* $\wedge$ C_{agcj}*: ocorre(pedido_diagnóstico)*	Quando é solicitado ao agente de apoio ao diagnóstico médico um diagnóstico relativo a um determinado paciente, uma mensagem é enviada ao(s) agente(s) servidor solicitando toda a informação disponível relativa a esse paciente; uma mensagem é enviada ao(s) agente(s) de conhecimento solicitando um diagnóstico para o referido paciente. Recebidas as respostas, toda a informação recolhida é apresentada ao médico.

	Regra-Ponte	Descrição
5)	C_{agcj}: *ocorre(pedido_diagnóstico)* ——— C_{agsi}: *ocorre(pedido_informação)* $\wedge$ C_{agak}: *ocorre(recebe_diagnóstico)*	Quando o agente de conhecimento recebe um pedido de diagnóstico para um paciente, envia uma mensagem ao(s) agente(s) servidor solicitando toda a informação disponível relativa a esse paciente; logo que recebe a resposta elabora o diagnóstico e envia a resposta ao agente de apoio ao diagnóstico médico que lhe fez a solicitação.
6)	C_{agsi}: *ocorre(pedido_informação)* ——— C_{agcj}: *ocorre(recebe_informação)* $\vee$ C_{aga}: *ocorre(recebe_informação)*	Quando um agente servidor recebe um pedido de informação referente a um paciente, responde, enviando essa mesma informação numa mensagem.
7)	C_{agm}:*ocorre(treino)* ——— C_{agcj}: *ocorre(treino)*	Quando o administrador solicita ao agente monitor o treino de um agente de conhecimento, comunica-lhe os casos de treino e de teste, junta-lhe os respectivos diagnósticos e envia ao agente de conhecimento uma mensagem com toda essa informação, solicitando o treino do agente.
8)	C_{agci}:*ocorre(treino)* ——— C_{agm}: *ocorre(recebe_resultados)*	Quando o agente de conhecimento termina um processo de treino do seu sistema de aprendizagem, envia os resultados do processo de aprendizagem ao agente monitor.
9)	C_{agm}: *ocorre(desactiva)* ——— *ocorre(desactiva)*	Quando o agente monitor é desactivado, também todos os outros agentes do sistema devem ser desactivados.

4.5 A Estrutura Lógica que suporta o *SADMED*

Como foi dito anteriormente, o conjunto de funcionalidades que corporizam a estrutura lógica do *SADMED* serão agora objecto de estudo. Isto inclui ou passa pela caracterização formal dos agentes do sistema (i.e., agentes de conhecimento, agentes servidores, agentes de apoio ao diagnóstico médico, agente monitor e agente recursos), do meio envolvente. Através da introspecção um agente avalia o estado da sua base de conhecimento, a sua credibilidade temporal, prioridades (e.g.,

aptidão) de modo a se posicionar para a obtenção de um diagnóstico credível e sustentável.

4.5.1 O Agente de Apoio ao Diagnóstico Médico aga_k

O agente de apoio ao diagnóstico médico encarrega-se da interface do sistema com os médicos, sendo a entidade responsável por receber as solicitações relativas aos pacientes e apresentar, quer os meios de auxílio ao diagnóstico (e.g., exames médicos, processos clínicos) quer os diagnósticos elaborados pelo sistema. É caracterizado pelos eventos e pelas seguintes propriedades (fluentes): *ini, aut, inf, diag* e *pare*, definidas para o intervalo de valores *{V,F}*(verdadeiro e falso).

Os acontecimentos associados ao agente de apoio ao diagnóstico médico são descritos na Tabela 4.5.

Tabela 4.5 - Acontecimentos (ou eventos) do agente de apoio ao diagnóstico médico.

Evento	**Descrição**	**Pré-condição**	**Pós-condição**
activa	- Activação do agente.	*ini:F*	*ini:V*
autentica	- Validação do utilizador (i.e., agente de saúde), através da execução dos procedimentos da autenticação, sendo aberta uma sessão de trabalho e efectuado o registo de todas as operações efectuadas na sessão.	*aut:F*	*aut:V*
pedido_diagnóstico	- Pedido de diagnóstico para um paciente.		*inf:F* *diag:F*
recebe_informação	- Recebe informação relativa a um paciente (e.g., imagens médicas, informação clínica).	*inf:F*	*inf:V*
recebe_diagnóstico	- Recebe o diagnóstico relativo ao paciente elaborado pelos agentes de conhecimento.	*diag:F*	*diag:V*

Evento	Descrição	Pré-condição	Pós-condição
apresenta_diagnóstico	- Apresenta a informação recolhida relativamente ao paciente e o respectivo diagnóstico.	*inf:T* *diag:T*	
desactiva	- Desactivação do agente.	*pare:T*	

A activação de um agente de apoio ao diagnóstico médico passa pela consulta da sua base de conhecimento, pelo estabelecer de um canal de comunicação com o *QN* e por indicar ao agente monitor que há um novo agente de apoio ao diagnóstico médico no sistema.

A desactivação de um agente de apoio ao diagnóstico médico passa pelo envio de uma mensagem ao agente monitor informando da sua desactivação e pela desactivação do canal de comunicação com o *QN*.

A validação do utilizador (i.e., agente de saúde), que pretende utilizar o agente de apoio ao diagnóstico médico pode ser descrita pela regra:

$$\frac{\begin{array}{c} identifica(M) \wedge \\ procedimento\_autenticação(M) \wedge \\ inicia\_sessão(M,S) \wedge \\ cria\_log\_sessão(M,S) \end{array}}{autentica(M)}\ autentica$$

onde o significado de cada termo é dado na Tabela 4.6, sendo a produção $\frac{condição}{acção}\ operação$ auto explicativa.

Tabela 4.6 - Construção da operação de validação.

Termo	Significado
identifica(M)	- Coloca em *M* os dados para autenticação do utilizador.

Termo	Significado
procedimento_autenticação(M)	- Executa os procedimentos de autenticação.
inicia_sessão(M,S)	- Inicia a sessão *S* para o utilizador *M*. Esta sessão quando inactiva por um período de tempo pré-estabelecido na base de conhecimento do agente fecha automaticamente.
cria_log_sessão(M,S)	- Inicia a criação de um registo de toda a actividade do utilizador *M* na sessão *S*.
autentica(M)	- Autentica o utilizador *M*.

O pedido de um diagnóstico ao agente de apoio ao diagnóstico médico por parte do utilizador para um determinado paciente pode ser descrita pela regra:

$$\frac{\begin{array}{c} identifica(P) \wedge \\ constrói\_pedido\_de\_diagnóstico(P,PD) \wedge \\ pedido\_de\_informação(P) \wedge \\ pedido\_de\_diagnóstico\_agente\_conhecimento(P) \wedge \\ recebe\_informação(P,I) \wedge \\ recebe\_diagnóstico(P,D) \wedge \\ apresenta\_diagnóstico(P,I,D) \wedge \end{array}}{pedido\_diagnóstico(P,D)} \; pedido\_diagnóstico$$

onde o significado de cada termo é dado na Tabela 4.7.

Tabela 4.7 - Construção da operação de pedido de um diagnóstico.

Termo	Significado
identifica(P)	- Coloca em *P* a identificação do paciente.
constrói_pedido_de_diagnóstico(P,PD)	- Constrói o pedido de diagnóstico *PD*.
pedido_de_informação(P)	- Envia uma mensagem aos agentes servidores com um pedido de informações relativamente ao paciente *P*.
pedido_de_diagnóstico_agente_conhecimento(P)	- Envia uma mensagem aos agentes de conhecimento com um pedido de diagnóstico relativamente ao paciente *P*.
recebe_informação(P,I)	- Recebe a informação *I* relativamente ao paciente *P* dos agentes servidores.
recebe_diagnóstico(P,D)	- Recebe o diagnóstico *D* relativamente ao paciente *P* dos agentes de conhecimento.

Termo	Significado
apresenta_diagnóstico(P,I,D)	- Apresenta a informação *I*, juntamente com o diagnóstico *D* ao utilizador.
pedido_diagnóstico(P,D)	- Diagnóstico *D* do paciente *P*.

4.5.2 O Agente Monitor *agm*

O agente monitor encarrega-se da monitorização do funcionamento do sistema, sendo a entidade responsável por reportar, para o exterior, o que acontece no interior deste, desde que activado, fazendo uso de:

- Uma janela de entrada no sistema, onde são apresentadas as mensagens oriundas do ambiente;
- Uma janela de saída do sistema, onde é colocada informação acerca das acções desenvolvidas pelo sistema; e
- Uma janela relatório, onde são presentes as mensagens enviadas pelos vários agentes constituintes do sistema acerca de ocorrências e estado de conhecimento destes.

Como entidade interlocutora com o administrador do sistema, para além de permitir a visualização de toda a informação referida, possibilita ainda, a partir de uma ordem do administrador, iniciar ou dar por encerrado o funcionamento do sistema e proceder ao treino dos agentes de conhecimento.

Toda a actividade e estado do sistema são monitorizados; i.e., dados de saída, mensagens enviadas pelos agentes e acções do sistema são também registadas na base de conhecimento.

Os acontecimentos associados ao agente monitor são descritos na Tabela 4.8.

Tabela 4.8 - Acontecimentos (ou eventos) associados ao agente monitor.

Evento	Descrição
activa	- Activação do agente *monitor* e do restante sistema.
treino	- Envia uma mensagem com a informação referente aos casos de treino e de teste e respectivos diagnósticos para o agente de conhecimento seleccionado.
recebe_resultados	- Recepção dos resultados do processo de aprendizagem.
monitoriza	- Monitorização do sistema com envio mensagens para a janela e ficheiro correspondente.
desactiva	- Desactivação do agente e restante sistema.

4.5.2.1 Iniciação ou Activação do Sistema

A operação de *iniciação* ou de activação do sistema desenrola-se ao nível do agente *agm* e passa pela realização de um certo número de operações, que são caracterizadas no texto que se segue:

- Criar as bases de conhecimento que irão registar as acções propostas pelo sistema em termos de diagnósticos, ou as medidas do seu desempenho, entre outras;
- Consultar a base de conhecimentos que contém a descrição dos parâmetros de configuração do sistema;
- Criar o *QN* e estabelecer com este um canal de comunicação;
- Activar os agentes de conhecimento;
- Activar os agentes servidores; e
- Activar o agente recursos e enviar-lhe uma mensagem indicando os recursos disponíveis, as regras de gestão de recursos e as regras de segurança do sistema.

4.5.2.2 A Desactivação do Sistema

A operação de desactivação do sistema desenrola-se ao nível do agente *agm* e passa pela realização de um certo número de operações, que são caracterizadas no texto que se segue:

- Difundir, através do *QN*, uma ordem de desactivação aos agentes de conhecimento;
- Difundir, através do *QN*, uma ordem de desactivação aos agentes servidores;
- Enviar através do QN, uma ordem de desactivação aos agentes de apoio ao diagnóstico médico;
- Enviar através do *QN*, uma ordem de desactivação ao agente recursos;
- Encerrar as sequências de saída e entrada de mensagens para o sistema;
- Esperar que os agentes se retirem do sistema;
- Terminar o processo associado ao *QN*; e
- Auto desactivar-se dando por terminada a execução do sistema *SADMED*.

4.5.2.3 A Operação de Treino

O treino e parametrização dos agentes de conhecimento desenrola-se ao nível do agente *agm*. Quando o administrador solicita ao agente *agm* o treino de um determinado agente de conhecimento, fornece-lhe como dados os casos de treino, assim como os casos de teste, juntamente com os respectivos diagnósticos. O agente *agm* constrói uma mensagem com essa informação e envia-a ao agente de conhecimento, solicitando o seu treino com a informação que juntou à mensagem.

A operação de treino pode ser descrita pela regra:

$$\frac{\begin{array}{c} recolhe\_casos\_treino(CTR) \wedge \\ recolhe\_casos\_teste(CTE) \wedge \\ recolhe\_diagnósticos(D) \wedge \\ constroí\_pedido\_treino(CTR,CTE,D,PT) \wedge \\ solicita\_treino(K,PT) \wedge \\ recebe\_resultados(K,R) \end{array}}{treino(K)}\ treino$$

onde o significado de cada termo é dado na Tabela 4.9.

Tabela 4.9 - Construção da operação de treino.

Termo	Significado
recolhe_casos_treino(CTR)	- Recolha de informação relativa aos casos de treino *CTR*.
recolhe_casos_teste(CTE)	- Recolha de informação relativa aos casos de teste *CTE*.
recolhe_diagnósticos(D)	- Recolha dos diagnósticos *D*.
constroí_pedido_treino(CTR,CTE,D,PT)	- Construção do pedido de treino *PT* a partir da informação recolhida.
solicita_treino(K,PT)	- Envio da mensagem *PT* ao agente *K* a solicitar o seu treino.
recebe_resultados(K,R)	- Recepção dos resultados nos casos de teste do processo de aprendizagem.
treino(K)	- Treino do agente *K*.

4.5.2.4 Operação de Monitorização

A operação de monitorização passa pelo agrupamento das mensagens que são enviadas pelos restantes agentes do sistema através do *QN*, e na sua posterior concentração na janela correspondente. Por outro lado, estas mensagens também são registadas nas bases de conhecimento criadas para o efeito, como foi referido em epígrafe.

A operação de monitorização pode ser descrita pela regra:

$$\frac{\begin{array}{c}recolha\_mensagens(M) \wedge \\ grava\_mensagens(M) \wedge \\ apresenta\_mensagens(M,J)\end{array}}{monitorização}\ monitorização$$

onde o significado de cada termo é dado na Tabela 4.10.

Tabela 4.10 - Construção da operação de monitorização.

Termo	Significado
recolha_mensagens(M)	- Recolha das mensagens *M* no *QN*.
grava_mensagens(M)	- Gravação das mensagens *M* na base de conhecimento.
apresenta_mensagens(M,J)	- Apresentação das mensagens *M* na janela *J*.

4.5.3 Os Agentes Servidores ags_j

Os agentes servidores são uma forma de entidades semi-autónomas que têm a seu cargo a ligação do sistema ao ambiente, fonte de dados para o sistema (e.g., dos pacientes, dos equipamentos médicos, do sistema de informação da unidade de saúde). Os agentes servidores possuem autonomia no sentido em que têm algum controlo sobre as acções que despoletam.

4.5.3.1 O Agente Servidor *DICOM*

O agente servidor *DICOM* implementa os protocolos de comunicação e tratamento de imagem médica segundo a norma *DICOM*.

Os acontecimentos associados ao agente servidor *DICOM* são descritos na Tabela 4.11.

Tabela 4.11 - Acontecimentos (ou eventos) associados ao agente servidor *DICOM*.

Evento	Descrição	Pré-condição	Pós-condição
activa	- Activação do agente.	*ini:F*	*ini:V*
exame_dicom	- Recepção de informação (e.g., imagens) correspondentes a um exame vindos de um equipamento *DICOM* (serviço *DICOM Storage*).		*local:F*
local_exame	- Recepção de localização para gravação do exame.		*local:V*
grava_exame	- Gravação de um exame.	*local:V*	
pedido_informação	- Disponibilização de informação relativamente a um paciente através de uma mensagem para o agente requerente.		
desactiva	- Desactivação do agente.	*pare:V*	

A activação do agente servidor *DICOM* passa pela consulta à sua base de conhecimento, pelo estabelecer de um canal de comunicação com o *QN* e por indicar ao agente monitor que o agente servidor *DICOM* está activo no sistema.

A desactivação do agente servidor *DICOM* passa pelo envio de uma mensagem ao agente monitor informando da sua desactivação e pela desactivação do canal de comunicação com o *QN*.

Quando se realiza um exame médico num equipamento que respeite a norma *DICOM*, a informação por este gerada é automaticamente enviada aos servidores *DICOM* com quem pode estabelecer ligação. O agente servidor encontra-se num estado de auto-suspensão à espera do envio desses exames, procedendo de seguida à sua classificação e armazenamento de acordo com as indicações do agente recursos. Este processo pode ser descrito pela regra:

$$exame\_dicom(E) \wedge$$
$$local\_armazenamento(R,T) \wedge$$
$$classifica\_exame(E,C) \wedge$$
$$local\_exame(R,L) \wedge$$

$$\frac{grava\_exame(E,L,C)}{gravação\_exame(E)}\ gravação\_exame$$

onde o significado de cada termo é dado na Tabela 4.12.

Tabela 4.12 - Construção da operação de pedido de um diagnóstico.

Termo	Significado
exame_dicom(E)	- Recepção do exame *E* de um equipamento *DICOM*.
local_armazenamento(R,T)	- Pedido de um local para armazenamento do exame com referência *R* e tamanho *T* ao agente recursos.
classifica_exame(E,C)	- Classificação do exame *E* para registo na sua base de conhecimento.
local_exame(R,L)	- Recepção da informação respeitante ao local de armazenamento *L* para o exame de referência *R* por parte do agente recursos.
grava_exame(E,L,C)	- Gravação do exame *E* na localização *L* e correspondente classificação *C* na sua base de conhecimento.
gravação_exame(E)	- Gravação e classificação do exame *E*.

Quando há um pedido de informação para um determinado paciente, o agente servidor consulta a sua base de conhecimento e disponibiliza toda a informação de que dispõe relativamente a esse paciente ao agente que lhe fez a solicitação. Este processo pode ser descrita pela regra:

$$\frac{pedido\_informação(P,A) \wedge prepara\_informação(P,I) \wedge envia\_informação(I,A)}{fornece\_informação(P)}\ fornece\_informação$$

onde o significado de cada termo é dado na Tabela 4.13.

Tabela 4.13 - Construção da operação de fornecimento de informação.

Termo	Significado
pedido_informação(P,A)	- Recepção do pedido de informação relativa ao paciente *P* do agente *A*.
prepara_informação(P,I)	- Preparação de toda a informação conhecida relativamente ao paciente *P*.
envia_informação(I,A)	- Envio de uma mensagem para o agente *A* com a informação *I* relativamente ao paciente referido no pedido de informação.
fornece_informação(P)	- Fornecimento de informação relativa ao paciente *P*.

4.5.3.2 O Agente Servidor do Processo Clínico

O agente servidor do processo clínico implementa uma interface de acesso ao sistema de informação da unidade de saúde. Os acontecimentos associados ao agente servidor do processo clínico são descritos na Tabela 4.14.

Tabela 4.14 - Acontecimentos (ou eventos) associados ao agente servidor do processo clínico.

Evento	Descrição
activa	- Activação do agente.
pedido_informação	- Disponibilização de informação relativamente a um paciente após consulta ao sistema de informação da unidade de saúde através de uma mensagem para o agente requerente.
desactiva	- Desactivação do agente.

Quando ocorre um pedido de informação relativa a um determinado paciente, o agente servidor consulta a sua base de conhecimento e constrói o pedido de informação para ser dirigido ao sistema de informação da unidade de saúde, e disponibiliza toda a informação obtida ao agente que lhe fez a solicitação. Este processo pode ser descrita pela regra:

$$pedido\_informação(P,A) \wedge$$
$$elabora\_pedido\_de\_informação(P,Q) \wedge$$

$$\frac{\textit{interroga\_sistema\_de\_informação}(Q,R) \wedge \textit{envia\_informação}(R,A)}{\textit{fornece\_informação}(P)} \; \textit{fornece\_informação}$$

onde o significado de cada termo é dado na Tabela 4.15.

Tabela 4.15 - Construção da operação de fornecimento de informação.

Termo	Significado
pedido_informação(P,A)	- Recepção do pedido de informação relativa ao paciente *P* do agente *A*.
elabora_pedido_de_informação(P,Q)	- Elaboração da interrogação *Q* a ser dirigida ao sistema de informação da unidade de saúde.
interroga_sistema_de_informação(Q,R)	- Interrogação do sistema de informação da unidade de saúde obtendo o resultado *R.*.
envia_informação(R,A)	- Envio de uma mensagem para o agente *A* com a informação *R* relativamente ao paciente referido no pedido de informação.
fornece_informação(P)	- Fornecimento de informação relativa ao paciente *P*.

4.5.4 Os Agentes de Conhecimento agc_i

Os agentes de conhecimento são uma forma de entidades semi-autónomas, cooperativas e competitivas que interactuam através do *QN* (Figura 4.6). Os agentes de conhecimento possuem autonomia no sentido em que exercem algum controlo sobre as suas acções; i.e., são cooperativos na medida em que contribuem para um mesmo diagnóstico. Na sua base de conhecimento, como já foi referido, encontram-se as *RNAs*, responsáveis pelo diagnóstico.

Os acontecimentos associados aos agentes de conhecimento são descritos na Tabela 4.16.

Tabela 4.16 - Acontecimentos (ou eventos) associados aos agentes de conhecimento.

Evento	Descrição
activa	- Activação do agente.
pedido_diagnóstico	- Elaboração de um diagnóstico para um determinado paciente.
treino	- Treino e teste das *RNAs*.
desactiva	- Desactivação do agente.

A activação de um agente de conhecimento passa pela consulta à sua base de conhecimento, pelo estabelecer de um canal de comunicação com o *QN* e por indicar ao agente monitor que se encontra activo no sistema.

A desactivação de um agente de conhecimento passa pelo envio de uma mensagem ao agente monitor a informar que se desligou do sistema e pela desactivação do canal de comunicação com o *QN*.

Quando há um pedido de diagnóstico para um determinado paciente por parte de um agente de apoio ao diagnóstico médico, é recolhida junto dos agentes servidores toda a informação disponível relativamente a esse paciente e essa informação dá entrada nos sistemas de aprendizagem, sendo o diagnóstico construído posteriormente a partir das saídas desses sistemas. Este processo pode ser descrito pela regra:

$$\frac{\begin{array}{c} pedido\_diagnóstico(P,A) \wedge \\ pedido\_informação(P,SDICOM) \wedge \\ pedido\_informação(P,SPC) \wedge \\ recebe\_informação(SDICOM,I) \wedge \\ recebe\_informação(SPC,R) \wedge \\ filtragem\_e\_normalização(I,R,N) \wedge \\ geração\_diagnóstico(N,D) \wedge \\ envia\_diagnóstico(D,A) \end{array}}{fornece\_diagnóstico} \\ fornece\_diagnóstico(P)$$

onde o significado de cada termo é dado na Tabela 4.17.

Tabela 4.17 - Construção da operação de elaboração de diagnóstico.

Termo	Significado
pedido_diagnóstico(P,A)	- Recepção de pedido de diagnóstico relativamente ao paciente *P* do agente *A*.
pedido_informação(P,SDICOM)	- Envio de uma mensagem com um pedido de informação ao agente servidor *DICOM*.
pedido_informação(P,SPC)	- Envio de uma mensagem com um pedido de informação ao agente servidor do Processo Clínico.
recebe_informação(SDICOM,I)	- Recepção da informação do agente servidor *DICOM*.
recebe_informação(SPC,R)	- Recepção da informação do agente servidor do Processo Clínico.
filtragem_e_normalização(I,R,N)	- Preparação da informação recolhida para submeter aos sistemas de aprendizagem.
geração_diagnóstico(N,D)	- Elaboração do diagnóstico a partir dos resultados apresentados pelos sistemas de aprendizagem.
envia_diagnóstico(D,A)	- Envio de uma mensagem para o agente *A* com o diagnóstico *D* relativamente ao paciente referido no pedido de informação.
fornece_diagnóstico(P)	- Fornecimento do diagnóstico relativo ao paciente *P*.

Quando surge uma mensagem do agente monitor a solicitar o treino das *RNAs* de um determinado agente de conhecimento, o agente de conhecimento procede ao treino do seu sistema de aprendizagem usando para tal a informação recebida (i.e., os casos de treino, os casos de teste e os diagnósticos.). No final é enviada uma mensagem com os resultados do processo de aprendizagem ao agente monitor. Este processo pode ser descrito pela regra:

$$\frac{\begin{array}{c} solicita\_treino(K,PT) \wedge \\ prepara\_treino(PT,CTR,CTE,D) \wedge \\ treina(CTR,CTE,D,R) \wedge \\ envia\_resultados(R) \end{array}}{treino(K)}\ treino$$

onde o significado de cada termo é dado na Tabela 4.18.

Tabela 4.18 - Construção da operação de treino.

Termo	Significado
solicita_treino(K,PT)	- Recepção de pedido para treino por parte do agente *K*.
prepara_treino(PT,CTR,CTE,D)	- Preparação da informação para se proceder ao processo de treino dos sistemas de aprendizagem.
treina(CTR,CTE,D,R)	- Processo de treino dos sistemas de aprendizagem.
envia_resultados(R)	- Envio dos resultados do processo de treino.
treino(K)	- Treino do agente *K*.

4.5.5 O Agente de Recursos *agr*

O agente recursos encarrega-se de monitorizar e gerir os recursos do sistema, além de proceder às cópias de segurança da informação existente no sistema. Os acontecimentos associados ao agente recursos são descritos na Tabela 4.19.

Tabela 4.19 - Acontecimentos (ou eventos) associados ao agente recursos.

Evento	Descrição
activa	- Activação do agente.
local_armazenamento	- Determinação do melhor local para armazenar a informação correspondente a um exame e envio de uma mensagem com essa informação para o agente servidor requerente.
gestão	- Periodicamente este agente activa esta regra que tem como consequência que se faça uma análise a todos os recursos disponíveis, e graus de utilização.
segurança	- Periodicamente este agente activa esta regra(s) que têm como consequência a implementação dos procedimentos de segurança.
desactiva	- Desactivação do agente.

A activação do agente recursos passa pela consulta da sua base de conhecimento, pelo estabelecer de um canal de comunicação com o *QN* e por indicar ao agente monitor que se encontra activo no sistema.

A desactivação do agente recursos passa pelo envio de uma mensagem ao agente monitor a informar que se desligou do sistema e pela desactivação do canal de comunicação com o *QN*.

Periodicamente, e de acordo com as regras de gestão de recursos, o agente recursos procede a uma análise da utilização destes por parte dos agentes do sistema. O mesmo procedimento é seguido com as regras de segurança.

Quando surge uma mensagem dum agente servidor a solicitar o local para armazenamento de informação, o agente recursos determina esse local recorrendo à sua base de conhecimento e envia uma mensagem ao agente servidor com essa informação. Este processo pode ser descrito pela regra:

$$\frac{\begin{array}{c} local\_armazenamento(R,T) \wedge \\ determina\_local(T,L) \wedge \\ local\_exame(R,L) \end{array}}{local\_de\_armazenamento(T)} \; local\_de\_armazenamento$$

onde o significado de cada termo é dado na Tabela 4.20.

Tabela 4.20 - Construção da operação de determinação do local de armazenamento.

Termo	Significado
local_armazenamento(R,T)	- Recepção de pedido para local de armazenamento dum exame de referência *R* e tamanho *T*.
determina_local(T,L)	- Determinação do local onde se poderá efectuar o armazenamento.
local_exame(R,L)	- Envio das coordenadas do local ao agente servidor.
local_de_armazenamento(T)	- Local de armazenamento para um exame de tamanho *T*.

4.5.6 O Ambiente *qenv*

A interacção do sistema *SADMED* com o seu meio ambiente (i.e., os equipamentos, o sistema de informação da unidade de saúde, os utilizadores em geral) é protagonizada pelo agente monitor, pelos agentes de apoio ao diagnóstico médico e pelos agentes servidores. Ora estes implementam três tipos de procedimentos, típicos de um sistema de aprendizagem reforçada e que são referidos no texto que se segue:

- Receber do ambiente do sistema as mensagens para serem processadas (i.e., um problema);
- Enviar para o ambiente do sistema as acções que através de uma atitude conjunta os agentes de conhecimento propõem (i.e., uma solução); e
- Receber desse mesmo ambiente informação que mede o impacto que a solução teve (i.e., o reforço).

O sistema e a sua envolvente deverão fazer uso de uma linguagem passível de ser interpretada de um modo correcto por ambos os interlocutores. O ambiente pode, no caso presente, tomar uma das seguintes formas:

- um agente (humano ou de *software*); ou
- um periférico (e.g., um computador, um equipamento médico, um autómato); ou
- uma base de conhecimento.

Cada uma destas situações condiciona o comportamento do sistema [Russell & Norvig 1995].

4.5 Conclusões

Um protótipo de um *SADMED – Ambiente Computacional para Sistemas de Apoio ao Diagnóstico Médico* foi objecto de teste em várias situações, sendo os resultados obtidos deveras encorajadores (a primeira aplicação comercial do sistema encontra-se já no terreno, e a ser utilizada pelos serviços de Imagiologia do Hospital Geral de Santo António, no Porto, em Portugal). Não se pode porém, fechar o capítulo, sem se dizer ainda algo quanto à problemática associada à escalabilidade do sistema. Este foi projectado tendo-se particular atenção a tal pressuposto, de tal modo que *SADMED* pode comportar um numero qualquer de processadores, desde que enumerável. Sistema este cuja espinha dorsal se alicerça na Lógica Matemática, e em duas das suas variantes computacionais, a Programação em Lógica Estendida (*PLE*) e a Programação em Lógica Contextual (*PLC*).

Ora, tendo-se em linha de conta os avanços em Ciência Fundamental na área da Programação em Lógica, este paradigma afirma-se não só como uma ferramenta para o cálculo automático, mas também como dando corpo a uma nova tecnologia para a resolução de problemas; i.e., passa-se a resolver problemas através da demonstração de problemas. Por conseguinte, ferramentas especializadas, técnicas e ambientes são necessários para apoiar os engenheiros de software no desenvolvimento de sistemas em tamanho real baseados em *PLE* e *PLC*. Este foi o caso neste trabalho, em que a área de impacto (ou de referência) foi a área médica e as tecnologias utilizadas as associadas à prestação de cuidados de saúde em tempo útil e de qualidade ao comum dos cidadãos, independentemente da sua localização geográfica ou condição social.

Capítulo 5
Diagnóstico Médico Baseado em Tomografia Computorizada via *PLE*, *PLC* e *RNAs*

Apresenta-se um subsistema de apoio ao diagnóstico baseado em tomografia computorizada utilizando programação em lógica estendida e redes neuronais artificiais.

À medida que aumenta o investimento em registos médicos computorizados por parte dos fornecedores de cuidados de saúde fica, por um lado, mais disponível informação sobre o acto médico, por outro, os conhecimentos no sector tornam-se mais fiáveis. De facto, Sistemas Inteligentes de Diagnóstico (*SID*) dotados de funções de mineração de dados e descoberta de conhecimento a partir destes enormes repositórios de informação estão-se a tornar cada vez mais comuns, tornando possível uma melhor oferta de serviços ou cuidados de saúde. Em particular o embeber desta tecnologia em sistemas *SID* aparenta ser particularmente adequado ao diagnóstico médico e em domínios afins.

A informação tornou-se um dos bens mais valiosos nos cuidados de saúde, mas o problema da diversidade de tratamento tem de ser considerado; o modo como se

elimina o erro está na homogeneização e padronização, actuando do mesmo modo, através dos tempos. Transformando estas pressuposições em axiomas, foi desenvolvido um ambiente computacional para suporte a sistemas de apoio ao diagnóstico médico. Permitindo a amálgama da mineração de dados e da descoberta de conhecimento via um demonstrador de teoremas que combina o potencial de uma extensão à programação em lógica com as funcionalidades de uma abordagem conexionista à resolução de problemas utilizando Redes Neuronais Artificiais (*RNAs*).

A emulação do conhecimento do radiologista foi obtida através da identificação de patologias utilizando imagens de foro médico usando *RNAs* e sistemas simbólicos baseados numa extensão à linguagem de programação em lógica. O objectivo passa por melhorar a fiabilidade e consistência do diagnóstico médico e marcar novas fronteiras nas aplicações das *RNAs*.

5.1 Introdução

A maioria das aplicações que envolvem computação visual e tratamento de imagens do foro médico usadas no processo de diagnóstico fazem-no em tempo real a partir de sequências de imagens. A visão e o tratamento de imagem vem-se a assumir como uma das principais áreas de aplicação de técnicas de *IA*, lidando com problemas relacionados com o reconhecimento, avaliação e manipulação de descrições conceptuais de diferentes universos de discurso.

Neste capítulo será referido como é que a Programação em Lógica Estendida (*PLE*) [Neves 1984] [Traylor e Gelfond 1993] [Neves et al. 1997] e a Programação em Lógica Contextual (*PLC*) [Cavedon & Tilhar 1995] [Denti et al. 1995] [Santos & Neves 1998] são utilizadas na avaliação e manipulação de descrições conceptuais a

partir de sequências de imagens, e como estas podem ser traduzidas em programas em *PLE* e *PLC*, reduzindo assim o problema de reconhecimento de descrições conceptuais à prova de objectivos em *PLE* e *PLC*.

De facto estes programas definem a extensão de um mapeamento de relações entrada-saída, que constituem os blocos constituintes de *RNAs*; i.e., pela prova de objectivos em *PLE* e *PLC* podem-se definir topologias de *RNAs*. *RNAs* que apresentam mais valias como a aprendizagem não linear, o mapeamento entrada-saída, a adaptabilidade e tolerância a ruído, e que por conseguinte são tidas como candidatas naturais a ferramentas de apoio ao diagnóstico. O reconhecimento de descrições conceptuais em termos abstractos passa pelo tratamento de eventos em termos de programas em *PLE* e *PLC*, os quais são considerados como primitivas do sistema computacional e que podem ser aceites directamente pela estrutura objecto de um sistema de avaliação de imagens de foro médico em termos de *RNAs*.

A contribuição deste capítulo para o trabalho aqui apresentado desenvolve-se segundo duas vertentes; por um lado foi estendido o formalismo associado a programas em *PLE* e *PLC* de modo a lidar com descrições conceptuais de imagens de foro médico, por outro o uso de programas em *PLE* e *PLC* nestas situações abre o caminho para a sua integração com sistemas conexionistas, como as definidas por *RNAs*, na resolução de problemas em outras áreas do conhecimento, em especial as que se situam na intersecção das Ciências da Computação e da Ciência Médica.

Em termos de representação de conhecimento e raciocínio esta abordagem tem como objectivo a aplicação da *PLE* e *PLC* à representação de conhecimento em *IA* e bases de dados, através da introdução da negação clássica, dados disjuntivos, meta-raciocínio e raciocínio em domínios abertos, o que estende o poder de representação das linguagens.

5.2 Representação da imagem

A tomografia foi introduzida na radiologia durante os anos 30. Enquanto as técnicas radiológicas convencionais produziam várias imagens de um objecto, os leitores tomográficos rodavam para dividir o objecto de estudo e organizá-lo em secções de imagem paralelas e parcialmente consecutivas. O processo, que inicialmente era totalmente mecânico, foi evoluindo face ao advento de novas tecnologias. Na Tomografia Computorizada (*TC*), o computador armazena um grande numero de dados de uma determinada região do corpo, fazendo com que seja possível determinar a relação da radiação *X* (*raios X*) absorvida pelas diferentes partes do corpo humano. A imagem *TC* consiste numa matriz de valores de atenuação de *raios X* corporizados por diferentes níveis de cinzento, criando assim uma imagem do item que foi objecto de estudo.

O constituinte atómico de uma imagem *TC* é o pixel. Dependendo do tamanho da região do corpo em estudo e da matriz de imagem, o pixel denota uma certa parcela da área do corpo humano que está a ser objecto de estudo. O pixel também se refere a um tecido em particular, em que o seu volume é determinado pela espessura da fatia, pelo tamanho da matriz e pelo diâmetro da região do corpo objecto de atenção. Assim, é possível e razoável afirmar que uma imagem é uma matriz de voxeis (Figura 5.1).

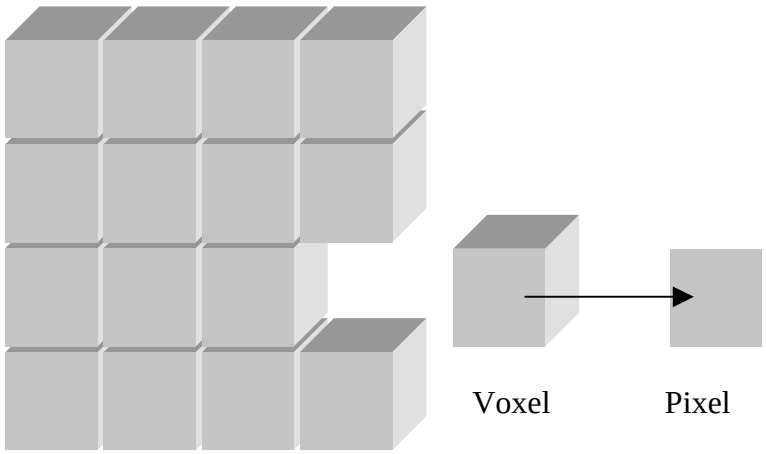


Figura 5.1 – Pixel com volume (Voxel)

A cada voxel é associado valor numérico, isto é, uma medida da degradação da radiação absorvida pelo tecido que está a ser objecto de estudo ou observação. O coeficiente de atenuação quantifica o grau de absorção de radiações pelo tecido em estudo. Após calibração interna da máquina, a densidade da água deverá ter o valor 0 (zero), e o ar o valor de –1000 Unidades Hounsfield (*UH*) (poderá haver variações em alguns equipamentos).

A escala de densidades de Hounsfield referida anteriormente corresponde ao valor das degradações na radiação ou atenuações. A atenuação do ar e da água (definidos como -1000 *UH* e 0 *UH*, respectivamente) representam pontos fixos na escala de densidade da *TC*, que não se alteram devido a variações de radiação emitida. Dependendo da radiação emitida pela máquina, a relação entre os graus de absorção de radiação entre diferentes tipos de tecido e a atenuação em termos do elemento água irá variar. Na Tabela 5.1 são apresentados alguns valores de referência.

Tabela 5.1 – Valor das degradações ou atenuações da radiação emitida para vários tecidos e fluidos

Tipos de Tecido ou fluido	Valores Padrão (*UH*)
Osso (Compacto)	> 250
Osso (Esponjoso)	130±100
Tiróide	70±10
Fígado	65±5
Músculo	45±5
Baço	45±5
Linfoma	45±10
Pâncreas	40±10
Rim	30±10
Gordura	-65±10
Tipos de Fluidos	
Sangue (Coagulado)	80±10
Sangue (Sangue venoso)	55±5
Plasma	27±2
Enxudado	>18±2
Transxudado	<18±2
Solução *Ringer*	12±2

Os valores de degradação ou atenuação da radiação emitida e absorvida por um certo tecido, são fundamentais para a reconstrução de imagem, variam entre os -1000 *UH* e os 1000 *UH*, a que correspondem várias zonas de cinzento. Contudo, o olho humano apenas consegue distinguir entre 15 a 20 desses níveis de cinzento. No caso em que a escala completa de densidade de 2000 *UH* fosse apresentada numa única imagem, o utilizador apenas conseguiria distinguir uma única sombra de cinzento no diagnóstico da zona do tecido mole.

Assim a janela da imagem (*image window*) foi desenvolvida como meio de produzir contrastes suficientemente representativos para diferenças de densidade de *UH* mínimas. O conceito de janela faz com que seja possível *expandir* a escala de cinzentos (W*indow Width*)(*WW*) de acordo com uma gama arbitrária de valores. Os

valores da atenuação acima do limite superior da janela irão aparecer na imagem com a cor branca, e aqueles que estão abaixo do limite inferior irão aparecer na imagem com a cor preta. O nível de janela (*Window Level*)(*WL*) assume-se como o centro da escala dos valores de atenuação da radiação (i.e., denota quais as estruturas orgânicas que são representadas nas sombras médias de cinzento). O ajuste da janela tem de estar em concordância com o tecido que está a ser diagnosticado. Janelas de pequena abertura fornecem imagens de alto contraste, contudo existe o perigo de as estruturas fora do limite da janela poderem ser negligenciadas. Com uma vasta abertura de janela as pequenas diferenças de densidade parecem homogéneas e são mascaradas, ficando a resolução reduzida. A Figura 5.2 descreve o procedimento de ajuste para a visualização da imagem.

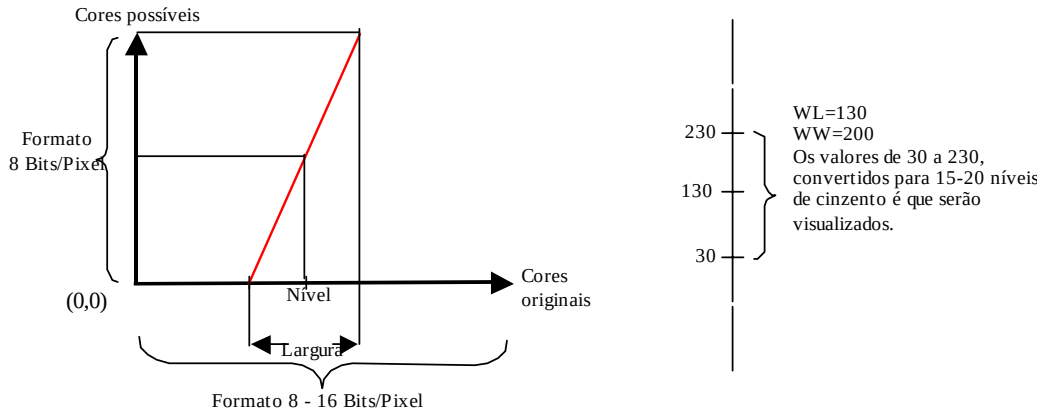


Figura 5.2 – Ajuste dos parâmetros de nível e largura de janela.

5.3 Desenvolvimento e Análise do Sistema

O processo de desenvolvimento, análise e utilização do sistema de apoio ao diagnóstico em *TC* é agora descrito em termos de procedimentos:

- *Selecção* – selecção e segmentação dos dados (i.e., imagem de cada exame em formato *DICOM*) de acordo com a secção do corpo humano seleccionada para diagnóstico;
- *Pré-processamento, transformação e normalização da informação* – esta é a fase em que os dados são limpos, em que alguma informação é removida (e.g., parte da imagem que diz respeito ao suporte de cabeça, mesa do equipamento de *TC*, tiras de fixação), uma vez que é interpretada como ruído. Posteriormente procede-se à transformação e normalização dos dados tendo em atenção o significado de cada *voxel* em termos da escala de Hounsfield. Os procedimentos desta fase são dados na forma de teoremas a demonstrar e definidos em *PLE* e *PLC*, na forma:

$(\forall \Pi_{imagem\text{-}1})\ (\forall A_1) \ldots\ (\forall A_8),\ \neg\, demo(\Pi_{imagem\text{-}1},\ imagem(A_1,\ldots,A_8), verdadeiro);$

$(\forall \Pi_{imagem\text{-}1})\ (\forall A_1) \ldots\ (\forall A_8),\ \neg\, demo(\Pi_{imagem\text{-}1},\ imagem(A_1,\ldots,A_8), falso);$

$(\forall \Pi_{imagem\text{-}1})\ (\forall A_1) \ldots\ (\forall A_8),\ \neg\, demo(\Pi_{imagem\text{-}1},\ imagem(A_1,\ldots,A_8), \perp);$

$(\forall \Pi_{imagem\text{-}2})\ (\forall A_1) \ldots\ (\forall A_8),\ \neg\, demo(\Pi_{imagem\text{-}2},\ imagem(A_1,\ldots,A_8), verdadeiro);$

$(\forall \Pi_{imagem\text{-}2})\ (\forall A_1) \ldots\ (\forall A_8),\ \neg\, demo(\Pi_{imagem\text{-}2},\ imagem(A_1,\ldots,A_8), falso);$

$(\forall \Pi_{imagem\text{-}2})\ (\forall A_1) \ldots\ (\forall A_8),\ \neg\, demo(\Pi_{imagem\text{-}2},\ imagem(A_1,\ldots,A_8), \perp);$

$(\forall \Pi_{imagem\text{-}3}),\ (\forall A_1) \ldots\ (\forall A_8),\ \neg\, demo(\Pi_{imagem\text{-}3},\ imagem(A_1,\ldots,A_8), verdadeiro);$

$(\forall \Pi_{imagem\text{-}3}),\ (\forall A_1) \ldots\ (\forall A_8),\ \neg\, demo(\Pi_{imagem\text{-}3},\ imagem(A_1,\ldots,A_8), falso);$ *e*

$(\forall \Pi_{imagem\text{-}3}),\ (\forall A_1) \ldots\ (\forall A_8),\ \neg\, demo(\Pi_{imagem\text{-}3},\ imagem(A_1,\ldots,A_8), \perp);$

em que $\perp$ denota um conjunto não enumerável de diagnósticos (i.e., numa escala 0...1, $0 = \lim_{\perp \to \infty} (ite) \frac{1}{\perp}$ denota a ausência de informação). Por razões que se prendem com os objectivos deste trabalho, apenas valores tomados como *verdadeiros* serão objecto de estudo. A teoria em lógica é então dada em termos da extensão do predicado *imagem: valores normalizados do voxel* $\rightarrow$ *{verdadeiro, falso}*, na forma:

imagem(-0.032, 0.794, ..., -0.148).

imagem(0.632, 0.326, ..., -0.034).

imagem(0.130, -0.480, ..., 0.301).

$\neg$ *imagem(A1, A2,..., A8)* $\leftarrow$ *não imagem(A1, A2,..., A8).*

- *Extracção* – esta fase está relacionada com a extracção de padrões a partir dos dados utilizando programas *PLE* e *PLC* [Neves et al. 2000a]; e
- *Interpretação* e *evolução* – os padrões identificados pelo sistema traduzem-se em teoremas a demonstrar, sendo as instanciações das variáveis usadas no suporte à tomada de decisão através de *RNAs* (i.e., as entradas são, por conseguinte, constituídas pelos valores normalizados dos *voxeis* de cada imagem).

A arquitectura utilizada para dar suporte ao sistema computacional e descrito no capítulo anterior é um modelo de um sistema de processamento de informação inteligente, dos seus principais subsistemas, dos seus papéis funcionais, e do fluir de informação e controlo entre estes. De facto, muitos sistemas complexos são constituídos por subsistemas especializados que interactuam de um modo circunscrito entre si. Os benefícios desta arquitectura também se aplicam na concepção de agentes inteligentes, entidades que podem ser vistas como entidades que interactuam com os seus ambientes de um modo flexível e orientado ao objectivo, e interpretadas como teorias [Neves 1984] [Neves et al. 1997]. A inteligência do sistema desenvolve-se a partir da contribuição de cada elemento do colectivo.

Para a implementação deste sistema, distribuído por natureza, utilizou-se tecnologia de Internet no lado do utilizador final (e.g., o radiologista, os técnicos), assim como aplicações a correr no sistema operativo *LINUX* no lado do processamento de dados.

A intranet foi implementada usando um *PC* (*Personal Computer*) com o *LINUX* como sistema operativo e o *Apache* como servidor de páginas. As ligações do exterior foram implementadas usando um *router RDIS* com um serviço *RAS*

(*Remote Access Service*). O servidor de imagem usa um *PC* com o *LINUX* como sistema operativo, e sistema de gestão de base de dados relacional *Postgres* para o suporte da base de dados. A navegação é efectuada utilizando o *browser* da *Netscape* com um *plug-in* para visualização de imagens no formato *DICOM* a correr em computadores tipo *PC* com o Windows como sistema operativo.

O servidor de imagem *DICOM* dá suporte à *interface médica* – esta janela permite a visualização e exploração de dados segundo a norma *DICOM*, oriundos de equipamentos de *TC*, *RM* (Ressonância Magnética), etc. Fornece ao utilizador formas de visualização interactiva, como o nível e largura de janela e o registo dos dados do paciente (Figura 5.3). Tem praticamente as mesmas funcionalidades de uma consola de *TC* ou *RM*.

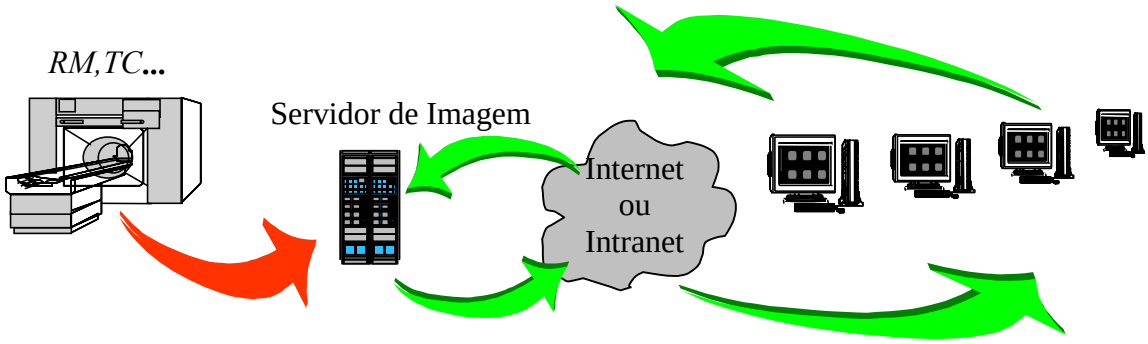


Figura 5.3 – Servidor de Imagem *DICOM*

A arquitectura do sistema é dada na Figura 5.4 (uma cópia da Figura 4.6, que aqui se reproduz dada o seu natural interesse e oportunidade) em termos dos seus elementos constituintes:

- Os *Agentes Servidores*, os quais foram desenvolvidos utilizando a linguagem de programação *C (GTK)* para o sistema operativo *LINUX*, sendo responsáveis pela aquisição de informação a partir dos dispositivos *DICOM* (*servidor DICOM*) e a partir do *Sistema de Informação* da unidade de saúde;

- Os *Agentes de Conhecimento*, os quais se definem como aplicações a correr sobre o sistema operativo *LINUX*, tendo sido desenvolvidos utilizando a linguagem de programação *C (GTK)*, e responsáveis pelas *RNAs* do sistema;
- O *Agente Monitor*, o qual foi desenvolvido usando *CGI* (*Common Gateway Interface*) e a linguagem *PERL*. Este módulo disponibiliza o interface para o administrador do sistema, sendo utilizado quando há que configurar o sistema;
- Os *Agentes de Apoio ao Diagnóstico Médico*, os quais foram desenvolvidos usando *CGI* e a linguagem *PERL*. Este subsistema implementa a interface do sistema com o médico, tanto para visualização de imagem médica, como para consulta dos diagnósticos gerados pelo sistema; e
- O *Agente Recursos*, o qual foi desenvolvido utilizando a linguagem de programação *C (GTK)* para o sistema operativo *LINUX,* sendo responsável pela manutenção dos recursos do sistema a níveis considerados aceitáveis pelos seus utilizadores.

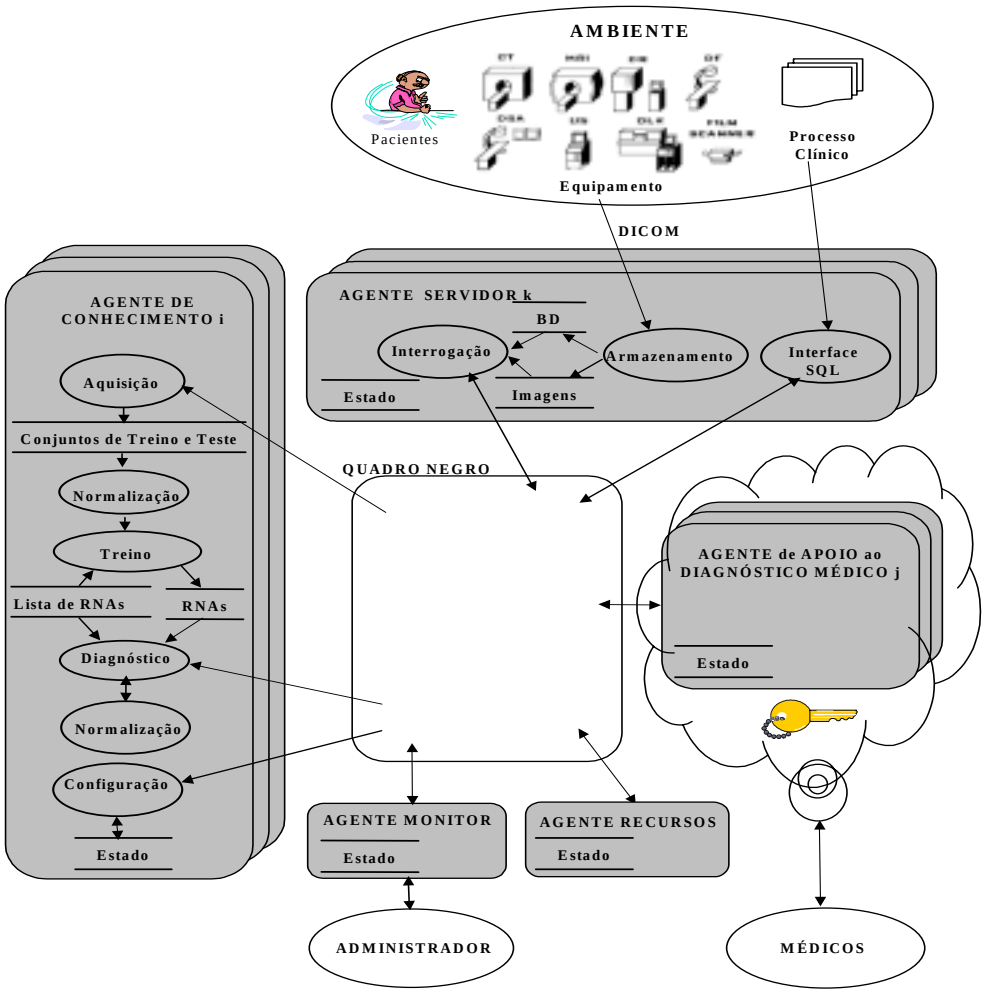


Figura 5.4 – A arquitectura do sistema *SADMED*

Através desta abordagem pode oferecer-se um sistema de apoio à decisão na área médica, com os clínicos a conduzirem uma espécie de diálogo com os agentes de conhecimento para interrogação de bases de dados/conhecimento, com a criação de cenários.

O objectivo a atingir através do trabalho descrito neste capítulo passa pela emulação do conhecimento do radiologista no processo de detecção de alguma patologia em imagem médica (e.g., em *Tomografia Computorizada* (*TC*) através da utilização de

uma combinação de sistemas simbólicos com técnicas computacionais de reconhecimento de padrões baseadas em *RNAs* [Neves et al. 1997]). Como segundo objectivo pretende-se estudar o comportamento do sistema *SADMED* no terreno.

5.4 Um sistema de diagnóstico para Tomografia Computorizada

A *TC* apresenta algumas vantagens sobre outras modalidades igualmente utilizadas no tratamento de imagem médica, uma vez que disponibiliza imagens de tecidos com uma grande variedade de contrastes e níveis, obtidos através de um simples ajustamento do nível e largura de janela da *raw data*[6] da imagem; i.e., fornece informação que não é visualizada em película (Figura 5.5) (e.g., nesta figura mostra-se que através da selecção de diferentes larguras e níveis de janela, o que se visualiza em ***A*** aparenta ser um espessamento do osso, enquanto que o que é dado em ***B*** refere-se provavelmente a um hematoma). Outras situações existem como a regulação de nível e largura de janela de imagens num joelho de modo a mostrar fracturas meniscais reais mas não de modo a inventá-las. Assim, esta *afinação* pode ser efectuada do melhor modo pelo radiologista antes de efectuar a leitura do exame. Ou ainda na situação em que parte da pélvis e tecidos moles são habitualmente deixados de fora da imagem pelos técnicos de radiologia aquando da execução de uma *RM* (a *RM* utiliza o mesmo formato de imagem do *CT*) da coluna lombar. Através de uma reconstrução mais larga (i.e., *Field of View – FOV*) pelo radiologista é possível ver os tecidos moles da parte inferior das costas e pélvis, o que é impossível de realizar de outro modo.

[6] Informação para reconstrução da imagem dos equipamentos de *TC* nos objectos *DICOM.*

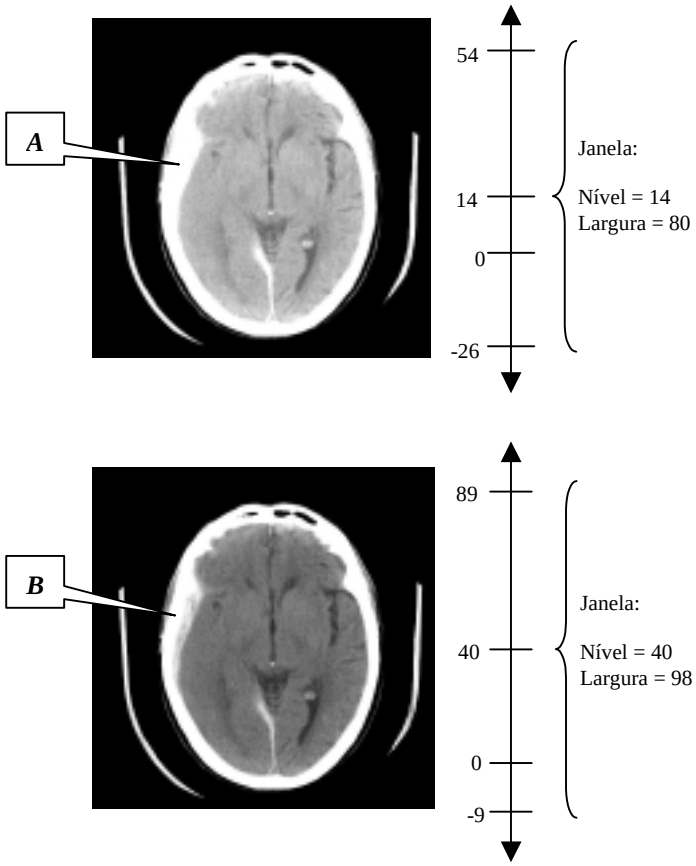


Figura 5.5 – O processo de regulação do nível e largura de janela de uma imagem.

O processo de desenvolvimento, análise e utilização do ambiente computacional para suporte de diagnóstico médico para *TC* é agora descrito.

A estrutura do sistema é representada na Figura 5.6 em termos dos respectivos módulos funcionais que se passam a enumerar:

- Módulo de Aquisição de Informação;
- Módulo de Normalização;
- Módulo de Treino da *RNA*;
- Módulo de Configuração das *RNAs*; e
- Módulo de Diagnóstico.

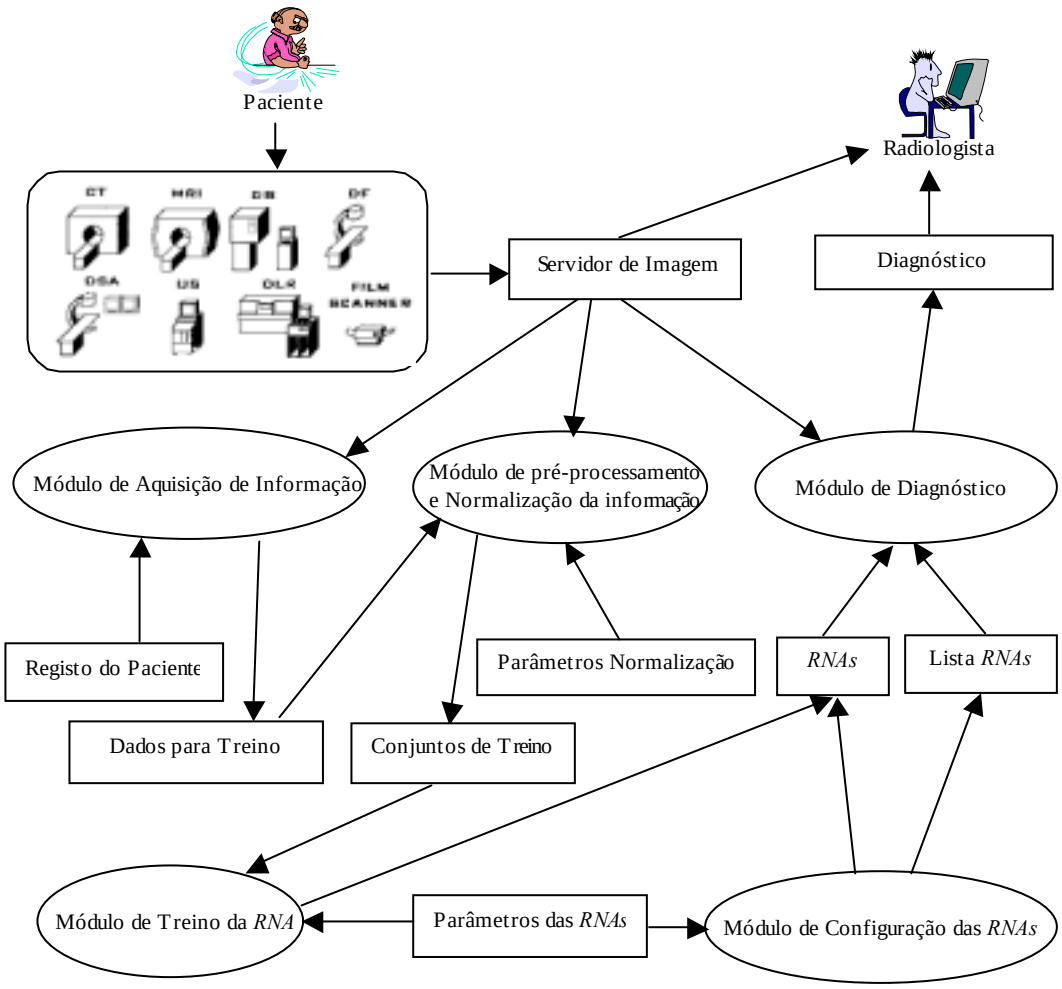


Figura 5.6 – Estrutura do sistema *SADMED*

5.4.1 Modulo para a Configuração das *RNAs* (*MCR*)

A implementação deste subsistema foi conseguida através de uma aplicação *WEB* desenvolvida usando *CGIs* e a linguagem *PERL* (Figura 5.7). Nestas páginas é então possível criar a lista de *RNAs* disponíveis, seus parâmetros de configuração e suas saídas (i.e., a(s) patologia(s) associadas a um dado exame).

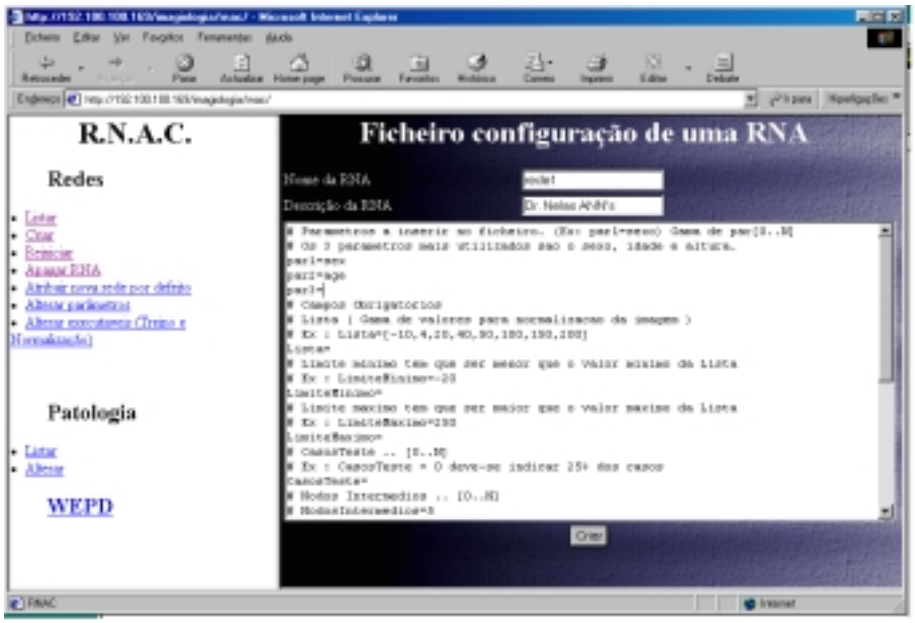


Figura 5.7 – Interface do Subsistema *MCR* – configuração de uma *RNA*

Trata-se de um conjunto de aplicações desenvolvidas para criar e parametrizar as *RNAs* a serem utilizadas pelo sistema *SADMED*. Nesta unidade funcional é possível criar, alterar, remover, listar, configurar, e reiniciar uma rede (Figura 5.8). Cada rede é associada a uma determinada área anatómica, a um determinado médico especialista e a uma determinada parametrização.

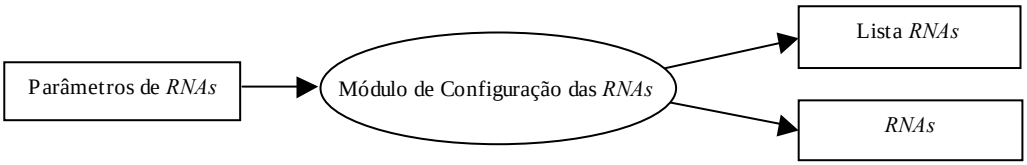


Figura 5.8 – Estrutura do subsistema *MCR*

5.4.2 Módulo de Aquisição de Informação (*MAI*), Módulo de Pré-Processamento e Normalização de Dados (*MPN*), e Módulo de Treino da *RNA*.

A aplicação que integra estes subsistemas foi desenvolvida em *PERL* para ser usada via *WEB*. A sua principal característica passa por disponibilizar a interface para a preparação dos conjuntos de casos de teste e de treino para as *RNAs*, bem como o pré-processamento, transformação e normalização dos conjuntos de casos de teste e de treino para as mesmas *RNAs.* Por outro lado, existe a possibilidade de efectuar estas operações relativamente não só a imagens individuais mas também a conjuntos de imagens (i.e., a um exame). Para a primeira situação tem-se que ao utilizador é permitida a apresentação dos dados de duas formas distintas:

- Através da indicação de um ficheiro contendo a localização da directoria que contem os exames (Figura 5.9); e
- Através do próprio browser, preenchendo um formulário onde deve ser indicado a directoria onde se encontram os exames e uma listagem de pares imagem/diagnóstico (Figura 5.10).

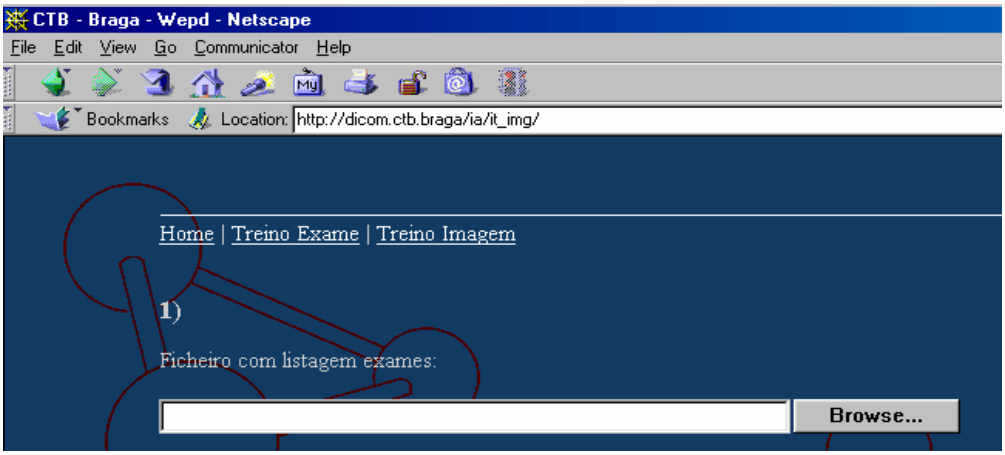


Figura 5.9 – Interface do subsistema *MAI* – Indicação dos dados por ficheiro.

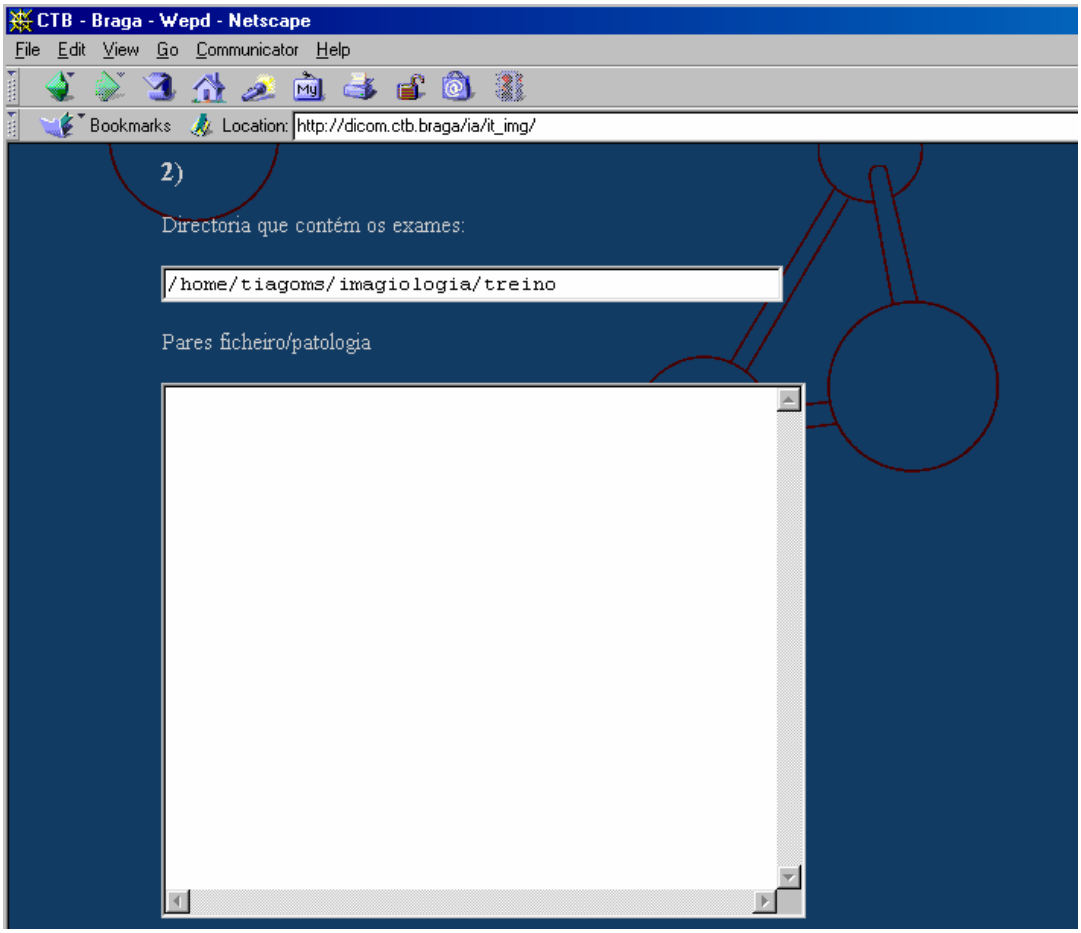


Figura 5.10 – Interface do subsistema *MAI* – Indicação dos dados no próprio *browser.*

Trata-se de um conjunto de aplicações que permitem que um técnico de radiologia reúna não só as imagens em formato *DICOM*, mas também os respectivos diagnósticos junto dos médicos especialistas, assim como informação complementar com relevância para o diagnóstico. Esta informação formará o conjunto de casos de treino da *RNA*.

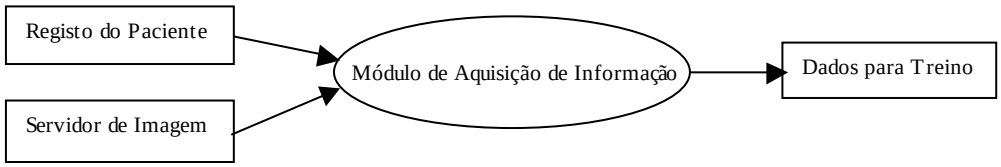


Figura 5.11 – Estrutura do subsistema *MAI*

Para cada exame de um *TC* cerebral é recolhida a informação:

- imagens *DICOM* (i.e., identificação dos ficheiros com as imagens);
- idade;
- naturalidade;
- sexo;
- traumatizado;
- contraste; e
- patologia (e.g., normal, hidrocefalia, atrofia, enfarte, hemorragia, lesões tumorais primitivas, lesões tumorais secundárias, ectopias, distrofias, malformações).

De forma a efectuar o treino da *RNA* com os dados disponíveis é pedido ainda na mesma página o tipo de *RNA* a utilizar, de entre as várias disponibilizadas pelo subsistema (Figura 5.12).

Figura 5.12 – Interface do subsistema *MPN* – Especificação da *RNA* a utilizar.

Estando presentes todos os dados necessários, procede-se então à normalização das imagens. Note-se que os dados pessoais do paciente são obtidos directamente do ficheiro *DICOM*.

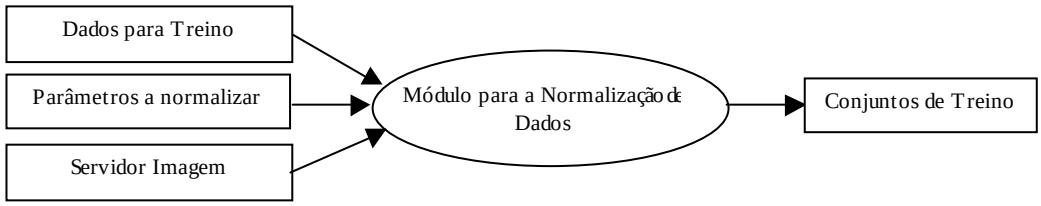


Figura 5.13 – Estrutura do subsistema *MPN*

Na figura que se segue observa-se o resultado da operação de normalização, em conjunção com os dados pessoais do paciente, patologia, etc.

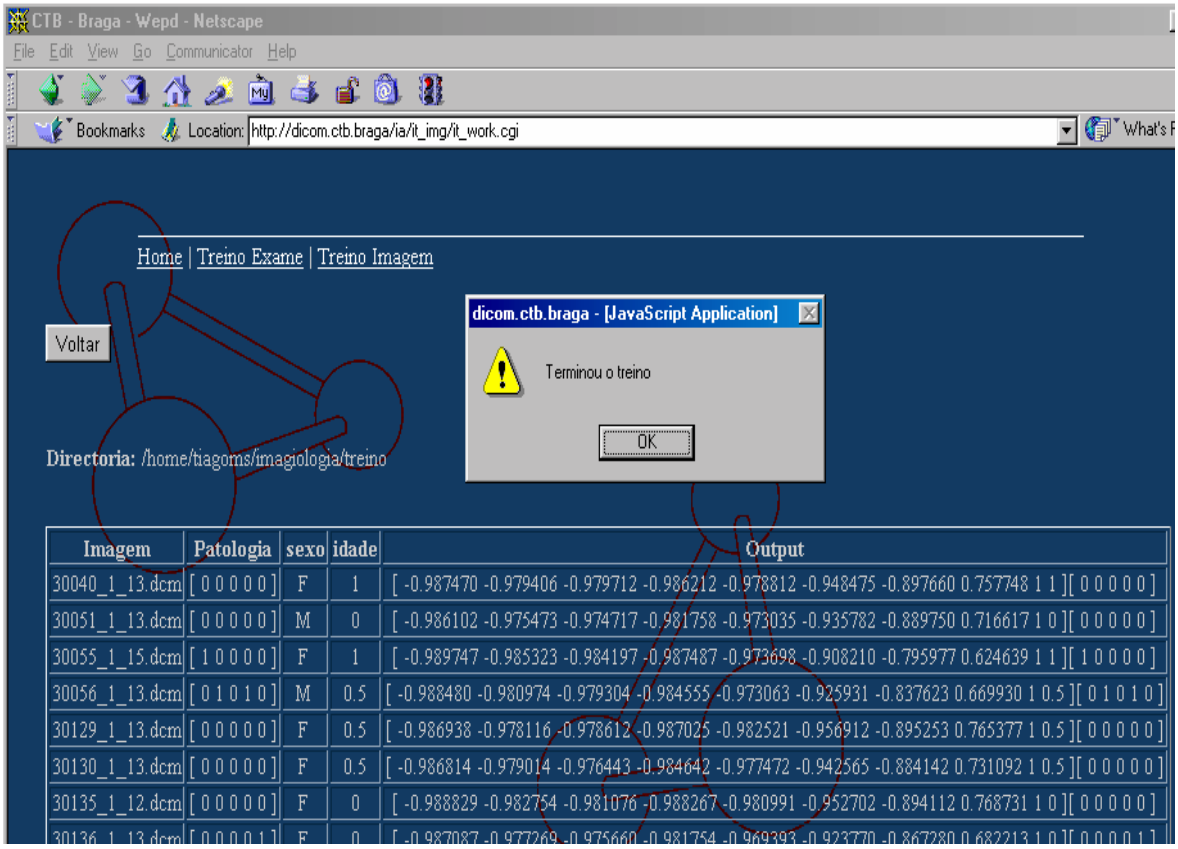


Figura 5.14 – Interface do subsistema *MTR* – Parte final do treino da *RNA*

Nesta altura já foi realizado o treino da rede com os dados entretanto disponibilizados.

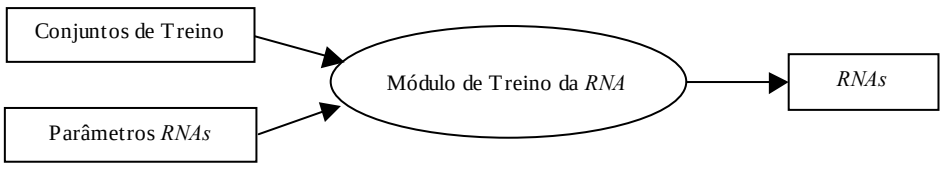


Figura 5.15 – Estrutura do subsistema *MTR*

No caso em que há que processar exames, o processo de treino da(s) *RNA(s)* é em tudo semelhante ao definido em epígrafe (Figuras 5.16, 5.17).

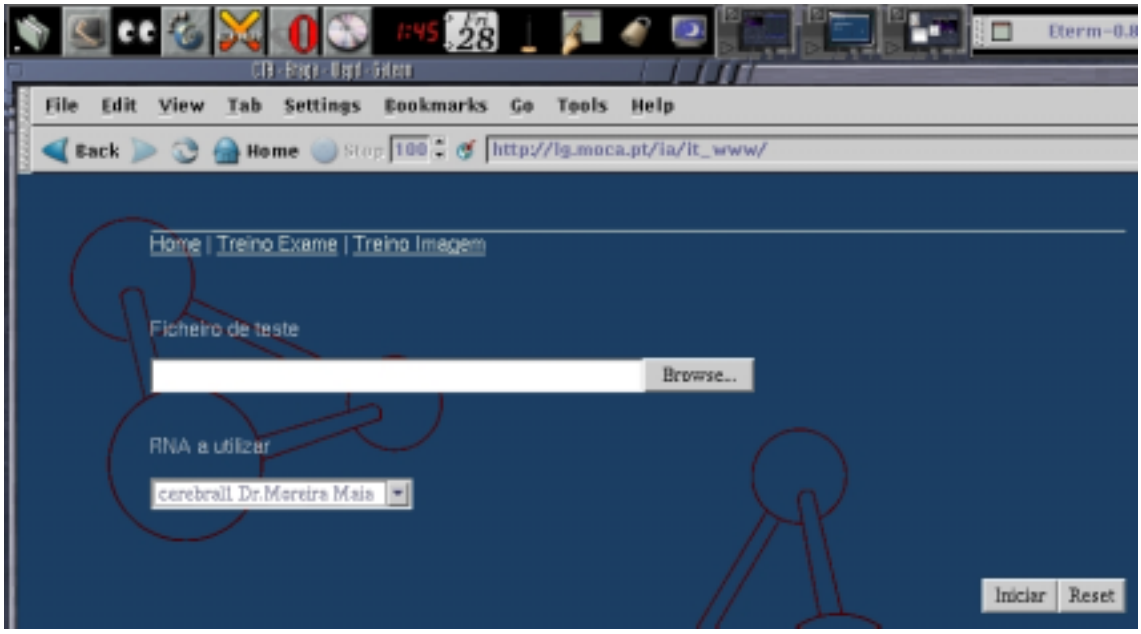


Figura 5.16 – Treino por exame

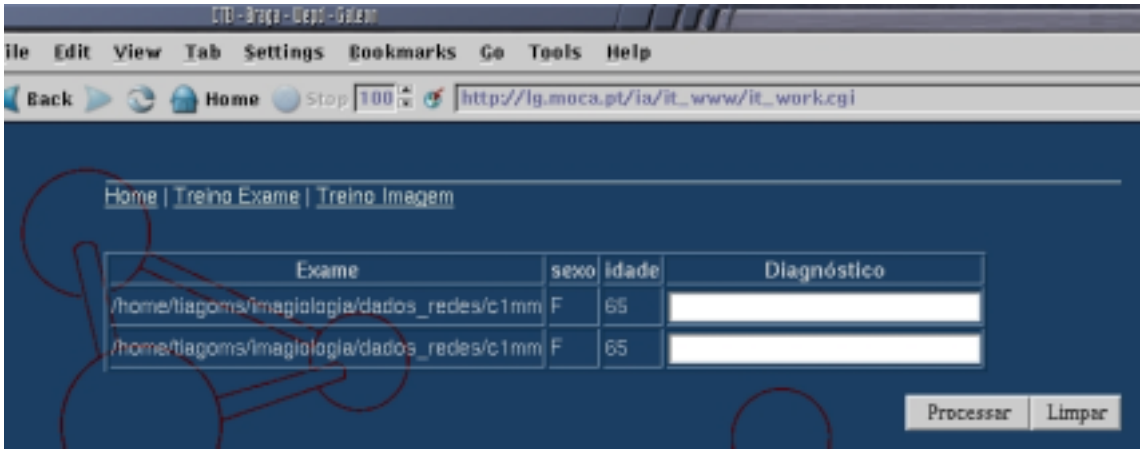


Figura 5.17 – Treino por exame

5.4.2.1 Pré-Processamento e Normalização da Imagem

Na norma *DICOM* a representação digital de imagens é constituída por uma matriz de Voxeis. Dado que as imagens que se seleccionaram têm uma resolução de 512x512, o número de elementos passíveis de serem incluídos no vector de entrada da *RNA* referente à imagem é de 262144. Ora, esta cardinalidade do vector de entrada para a *RNA* a ser submetida a treino, tem como senãos uma aprendizagem difícil e tempos de resposta excessivamente dilatados. No entanto este número baixa consideravelmente se levarmos em conta que da totalidade de valores lidos pelo equipamento, grande parte corresponde a elementos sem qualquer interferência no diagnóstico (e.g., ar, apoio de cabeça, mesa do equipamento), pelo que poderão ser eliminados. Serão por conseguinte apenas relevantes para o diagnóstico os valores que pertençam ao intervalo [–20, +250] da escala de Hounsfield (e.g., a cardinalidade do vector de entrada para a *RNA* terá o valor aproximado de 50000 num *TC* cerebral).

Para a extracção de dados do ficheiro de imagem, em termos de uma matriz de voxeis, foi desenvolvido o programa *ExtDicom*, que considera valores da escala de Hounsfield pertencente ao intervalo [–20, +250], e que é invocando na forma:

```
ExtDicom –l <n.º de intervalos> <imagem> <resultado>
```

fazendo com que o procedimento de pré-processamento de dados que corporizam as imagens que materializam os conjuntos de treino das *RNAs* seja dado pelas produções em linguagem de programação *C*, que a seguir se explicitam:

```
private void por_limites(int32* buff,int width,int height,int
highbit,FILE* fp)
{long i, maxval=LONG_MIN, minval=999999;
 float aux=0.0;
 int amplitude;
```

```
int cont[nr_intervalos];
for (i=0;i<nr_intervalos;i++) cont[i]=0;
int antes=0;
int depois=0;
assert(fp != NULL);
aux = pow(2, highbit)/1000.0;
amplitude = (limite_max-limite_min)/nr_intervalos;
fputs("Método dos intervalos constantes\n", fp);
fprintf(fp,"\n==> Tamanho: %dx%d\n", width, height);
for (i=0; i < width*height; i++)
       {if (buff[i] > maxval) maxval = buff[i];
  else if (buff[i] < minval)  minval = buff[i];}
if (int(maxval/aux) > 500)
       {for (i=0; i < width*height; i++)
   {buff[i] = int(buff[i]/aux);}}
fprintf(fp,"==> Valor maximo: %d\n", maxval);
fprintf(fp,"==> Valor minimo: %d\n", minval);
fprintf(fp,"==> Limite minimo: %d\n", limite_min);
fprintf(fp,"==> Limite maximo: %d\n", limite_max);
for (i=0; i < width*height; i++)
  {if (buff[i]>=limite_max) depois++;
   if (buff[i]<=limite_min) antes ++;
   if (buff[i]>limite_min && buff[i]<limite_max)
     cont[(buff[i]-limite_min)/amplitude]++; }
fprintf(fp,"[ ");
fprintf(fp,"%d ", antes);
for (i=0; i<nr_intervalos; i++)
  {fprintf(fp, "%d ", cont[i]);}
fprintf(fp,"%d ", depois);
fprintf(fp," ]");}
```

Este procedimento escrito na linguagem de programação *C* divide o intervalo [–20, +250] em *n* subintervalos, sendo esse número dado como parâmetro para o sistema global.

Uma segunda abordagem, um pouco mais elaborada, passa por invocar o programa de extracção de dados do ficheiro de imagem na forma:

```
Dicom –p <[lista de valores]> <imagem> <resultado>
```

o que fará com que o procedimento de pré-processamento invocado seja dado pelas produções na linguagem de programação *C*:

```
private void por_percentagem (int32* buff, int width, int
height, int highbit, FILE* fp )
{long i, c, maxval=LONG_MIN, minval=999999;
 float aux=0.0;
 char lista_aux[254];
 char *token;
 char *tok2;
 int n=0;
 int antes=0;
 int depois=0;
 float buffer[width*height];
 assert(fp != NULL);
 token = strtok(lista, "[");
 token = strtok(token, "]");
 sprintf(lista_aux, "%s", token);
 while ((tok2=strsep(&token, ",")) && (tok2 != NULL)) n++;
 int array[n];
 float result[n];
 for (i=0;i<n;i++) result[i]=0.0;
 n=0;
 token = lista_aux;
 while ((tok2=strsep(&token, ",")) && (tok2 != NULL))
   {array[n] = atoi(tok2);
    n++;}
 aux = pow(2, highbit)/1000.0;
 for (i=0; i < width*height; i++)
 {if (buff[i] > maxval)  maxval = buff[i];
        else if (buff[i] < minval)  minval = buff[i];}
 if (int(maxval/aux) > 500)
   {for (i=0; i < width*height; i++)
     {buffer[i] = float(buff[i]/aux);}}
 for (i=0; i < width*height; i++)
```

```
   {if (buffer[i]<=limite_min) antes++;
    if (buffer[i]>=limite_max) depois++;
    if ( (buffer[i]<limite_max) && (buffer[i]>limite_min) )
       {for (c=0; c<n; c++)
    {if ((c==0)&&(buffer[i]<=array[c]))
 {result[c]=result[c]+1.0; }
          else if ((c==n-1)&&(buffer[i]>=array[c]))
 {result[c]=result[c]+1.0;}
          else if
((buffer[i]>=array[c])&&(buffer[i]<array[c+1]))
 {result[c]=result[c]+(-((buffer[i]-array[c+1])
/(array[c+1]-array[c])));
             result[c+1]=result[c+1]+(1-((array[c+1]-
buffer[i])
/(array[c+1]-array[c])));}}}}
 fprintf(fp, "[ ");
 fprintf(fp, "%d ", antes);
 for (i=0; i<n; i++)
   {fprintf(fp, "%d ", int(result[i]));}
 fprintf(fp, "%d ", depois);
 fprintf(fp, " ]");}
```

De notar que o termo *lista de valores* no procedimento de invocação do programa é constituída por uma lista de valores de Hounsfield que há que pré-processar (e.g., [-20,0,27,45,55,80,130,250], a serem interpretados na forma:

< ***-20***	– a que correspondem valores residuais;
0	– esta gama de valores denota a quantidade de Água;
27	– esta gama de valores refere-se à quantidade de Plasma;
45	– esta gama de valores traduz a quantidade de Limfoma, Músculo,..;
55	- esta gama de valores indica a quantidade de Sangue Venoso;
80	- esta gama de valores corporiza a quantidade de Sangue Coagulado;
130	– esta gama de valores representa a quantidade de Osso Esponjoso; e
> ***250***	– valor residual, que pode também denotar Osso Compacto).

Esta abordagem à extracção de dados do ficheiro de imagem à semelhança da anterior divide o intervalo de valores de Hounsfield com interesse para a determinação da patologia em subintervalos, onde para cada valor de Hounsfield é feita a contagem do número de voxeis com esse valor (de Hounsfield), o que dá o volume desse elemento na imagem tendo em conta a proximidade relativa a cada um dos valores (os quais correspondem a elementos com significado para o diagnóstico). Um exemplo de diagnóstico segundo estas linhas é dado no texto que se segue, na forma:

Dados:

- A lista de valores: [-10,0,30,50,100]; e
- O valor do voxel é de 25 na escala de valores de Hounsfield.

Resolução:

- A lista de valores de Hounsfield denota os pontos da imagem que se apresentam relevantes para a resolução do problema (i.e., estabelecer o diagnóstico);
- São seleccionados os pontos da lista que enquadram o valor do voxel. Neste caso, são seleccionados os pontos 0 e 30, uma vez que o valor do voxel é de 25 na escala de valores de Hounsfield; e
- É calculada a *proximidade relativa* (em percentagem) do valor do voxel em relação aos dois pontos seleccionados, de acordo com as expressões:

$$proximidade\_ao\_PontoInferior = -\frac{Voxel - LimiteSuperior}{LimiteSuperior - LimiteInferior}$$

$$proximidade\_ao\_PontoSuperior = 1 - \frac{LimiteSuperior - Voxel}{LimiteSuperior - LimiteInferior}$$

  i.e., ao ponto inferior (0) é adicionado o valor 0,166(6), enquanto que ao ponto superior (30) é adicionado o valor 0,833(3). Os restantes elementos da lista de valores de Hounsfield, referidos em epígrafe, permanecem inalterados. São também contabilizados os elementos fora dos limites, pelas

mesmas razões apontadas anteriormente (i.e., para se conhecer o volume desses elementos na imagem).

Uma outra questão a considerar tem a ver com o facto de que existe uma gama de valores de Hounsfield que denotam um bem determinado tipo de elementos na imagem (e.g., gordura ou osso). Tendo em atenção que se trata de uma gama de valores e não apenas de um valor único, podem-se incluir, na lista de valores de Hounsfield para o problema, os que são mais relevantes para o diagnóstico e, desta forma, a contagem será feita tendo em conta a proximidade relativa dos valores dos voxel com os valores dados.

Nesta fase do processo de conceptualização do problema em equação, pode-se dar início à normalização dos dados, o que passa por se efectuar uma transformação nos dados do problema de modo a acelerar e melhorar o processo de aprendizagem das *RNAs*. Este escalonamento é necessário, uma vez que certas entradas para as *RNAs* podem apresentar valores que rodam a centena de milhar, enquanto que outras rondam apenas a centena de unidades. Ora, caso os dados fossem submetidos à *RNA* sem qualquer alteração, a *RNA* atribuiria às suas ligações entre nodos pesos cujos módulos não se afastariam muito de zero (i.e., $|\bullet| \geq 0$) para valores de entrada da *RNA* cujos $|\bullet| >> 0$ e, pesos cujos $|\bullet| >> 0$ para valores de entrada na *RNA* com $|\bullet| \geq 0$. Significa isto que uma pequena alteração num desses pequenos valores influenciaria de uma forma esmagadora o resultado obtido, e por outro lado, uma variação na ordem das dezenas de milhar nos valores mais elevados não traria alterações significativas aos resultados. Tal situação não é admissível, pois pode-se estar a não considerar no diagnóstico dados importantes do problema em equação. Para se ultrapassar este tipo de senão, os dados de entrada para a *RNA* são normalizados de acordo com a função:

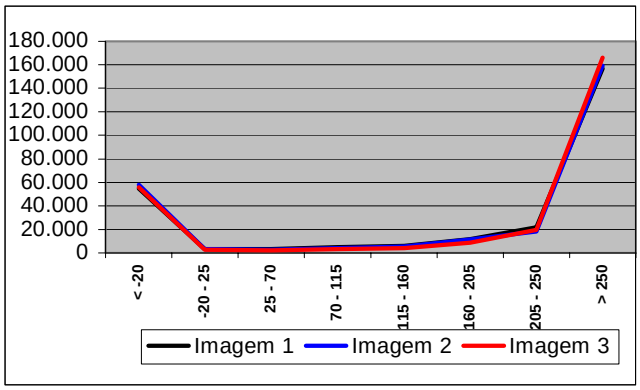
$$y = \frac{(x - min) * 2}{(max - min)} - 1$$

onde o *max* e *min* denotam, respectivamente, o valor máximo e o valor mínimo da variável *x*, a normalizar. Esta função tem como contradomínio o intervalo [-1,1], o que se revela ideal para o uso de funções de activação das *RNAs* como a *sigmoid*, para além do facto de que os algoritmos de aprendizagem de gradiente descendente, tal como o *Backpropagation*, são extremamente sensíveis a este tipo de escalonamento.

Atenda-se ao exemplo:

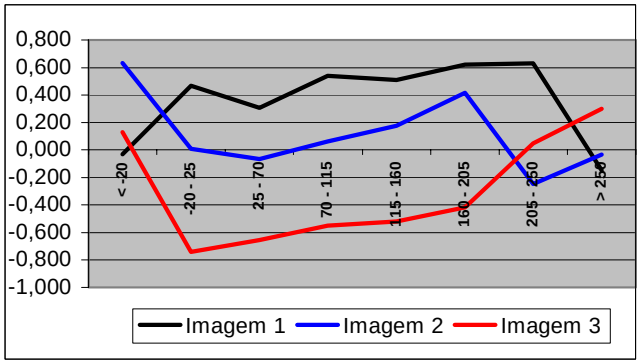
Valores antes do processamento

Valor do Voxel	Imagem 1	Imagem 2	Imagem 3
< -20	54.840	58.161	55.649
-20 - 25	3.234	3.005	2.629
25 - 70	2.980	2.701	2.258
70 - 115	4.810	4.092	3.173
115 - 160	5.764	5.261	4.220
160 - 205	11.859	11.251	8.740
205 - 250	21.626	18.350	19.451
> 250	157.031	159.323	166.024



Valores após processamento

Valor do Voxel	Imagem 1	Imagem 2	Imagem 3
< -20	-0,032	0,632	0,130
-20 - 25	0,468	0,010	-0,742
25 - 70	0,307	-0,065	-0,656
70 - 115	0,540	0,061	-0,551
115 - 160	0,509	0,174	-0,520
160 - 205	0,620	0,417	-0,420
205 - 250	0,631	-0,249	0,047
> 250	-0,148	-0,034	0,301

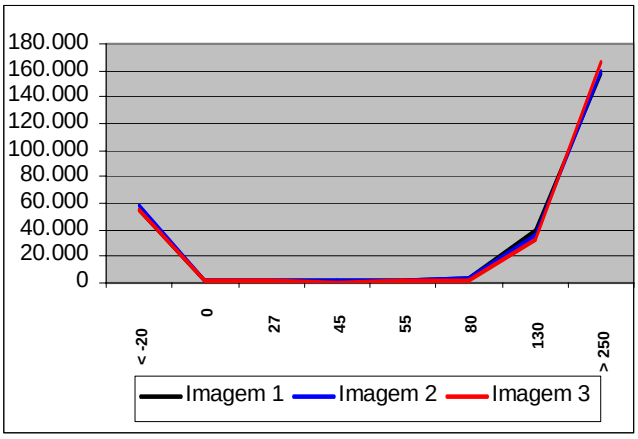


	< -20	-20 - 25	25 - 70	70 - 115	115 - 160	160 - 205	205 - 250	> 250
Valor Max	60.000	3.500	3.500	5.500	6.500	13.000	23.000	180.000
Valor Min	50.000	2.500	2.000	2.500	3.500	7.000	15.555	140.000

Figura 5.18 – Pré-processamento e normalização usando o método dos intervalos constantes.

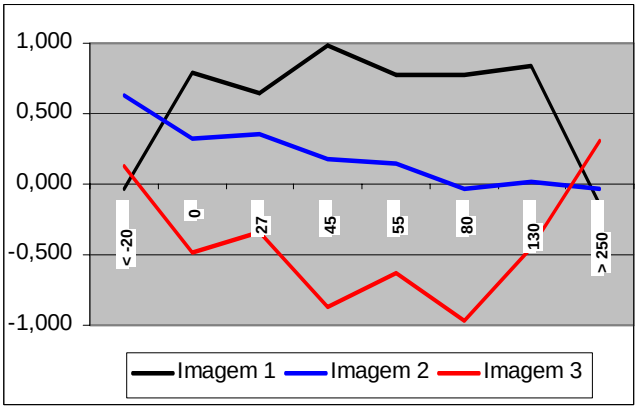
Valores antes do processamento

Ponto de referência	Imagem 1	Imagem 2	Imagem 3
< -20	54.840	58.161	55.649
0	2.328	2.164	1.882
27	1.494	1.407	1.198
45	997	897	767
55	1.533	1.345	1.112
80	3.831	3.220	2.522
130	40.086	35.624	32.987
> 250	157.031	159.323	166.024



Valores após processamento

Valor do Voxel	Imagem 1	Imagem 2	Imagem 3
< -20	-0,032	0,632	0,130
0	0,794	0,326	-0,480
27	0,647	0,357	-0,340
45	0,976	0,176	-0,864
55	0,777	0,150	-0,627
80	0,775	-0,040	-0,971
130	0,834	0,023	-0,457
> 250	-0,148	-0,034	0,301



	< -20	0	27	45	55	80	130	> 250
Valor Max	60.000	2.400	1.600	1.000	1.600	4.000	41.000	180.000
Valor Min	50.000	1.700	1.000	750	1.000	2.500	30.000	140.000

Figura 5.19 – Pré-processamento e normalização usando o método dos pontos de referência.

Como facilmente se pode verificar através dos gráficos das Figuras 5.18 e 5.19, os dados das três imagens antes da normalização dão origem a curvas praticamente idênticas (as linhas do gráfico estão praticamente sobrepostas). Após a

normalização, as diferenças entre estas são evidentes. Ora, são estas diferenças que importa realçar para que a *RNA* possa facilmente assimilá-las. Por outro lado, será importante salientar que estas imagens se reportam a partes distintas do cérebro de um paciente, pelo que as diferenças tinham de ser necessariamente evidentes.

5.4.3 Módulo de Diagnóstico (*MD*)

A implementação das funcionalidades deste módulo foi obtida através de uma aplicação *WEB* desenvolvida usando *CGI* e a linguagem *PERL*. Com este tipo de páginas na *WEB* é possível facultar ao médico o acesso a exames com diagnóstico fornecido pela *RNA* (Figuras 5.20, 5.21, 5.22, 5.23, 5.24 e 5.25).

Figura 5.20 – Interface do Módulo *MD*: Lista de Exames

Figura 5.21 – Interface do Módulo *MD*: Imagens de um Exame

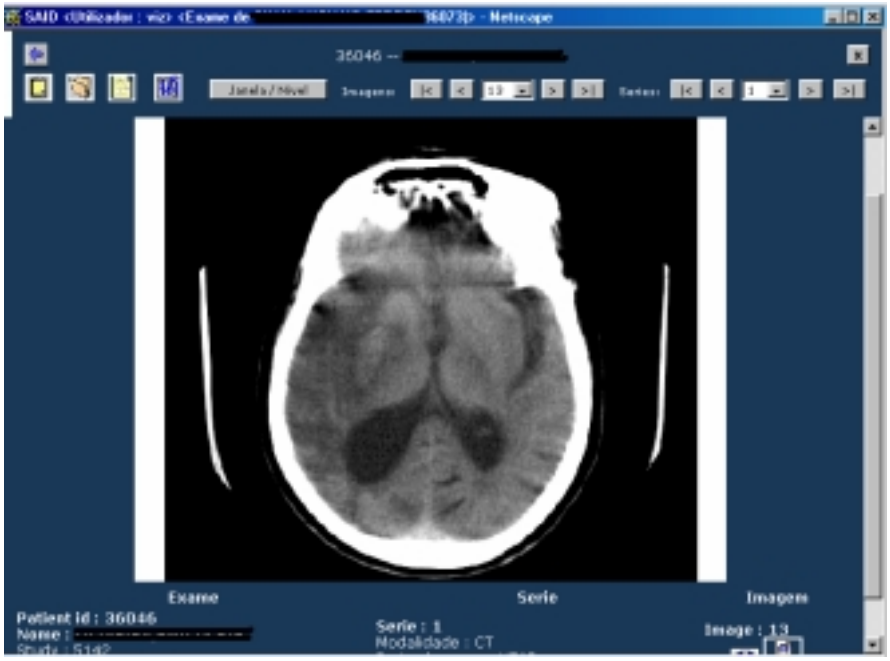


Figura 5.22 – Interface do Módulo *MD*: Imagem em Análise

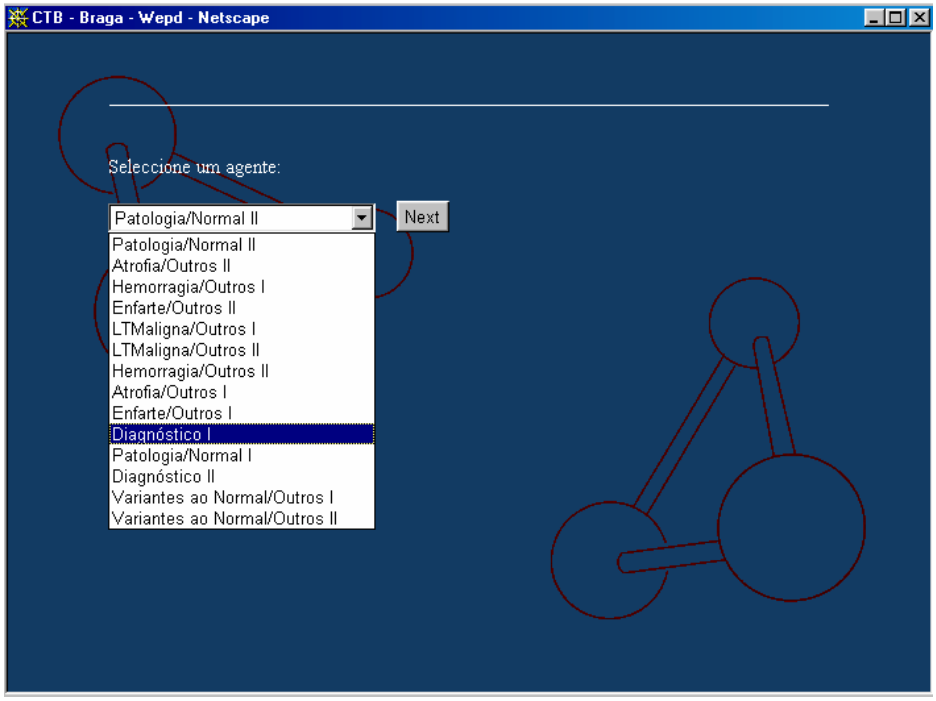


Figura 5.23 – Interface do Módulo *MD*: Análise de uma Imagem

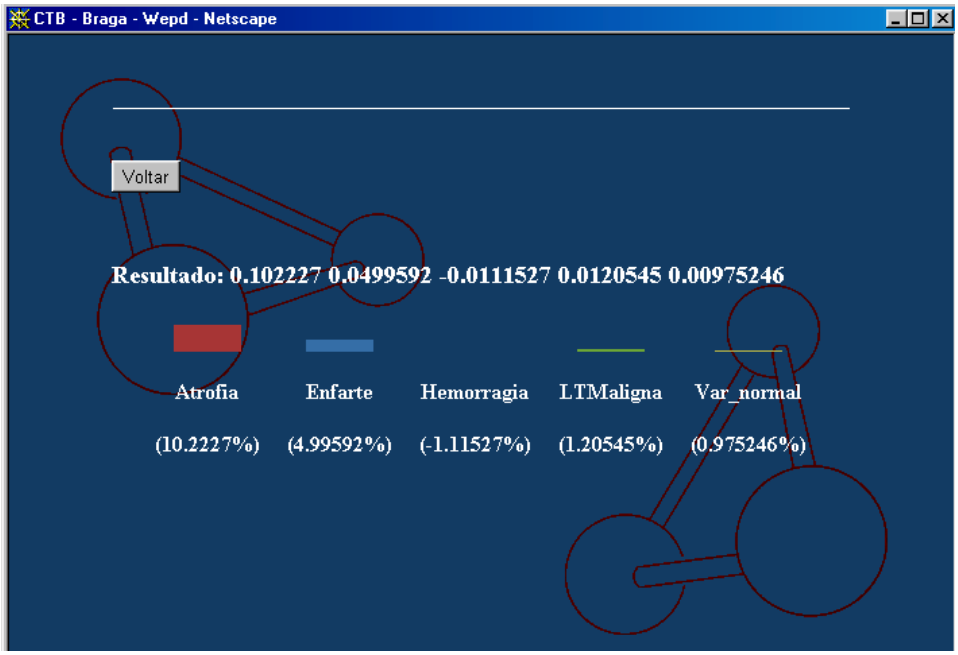


Figura 5.24 – Interface do Módulo *MD*: Diagnóstico através de uma Imagem

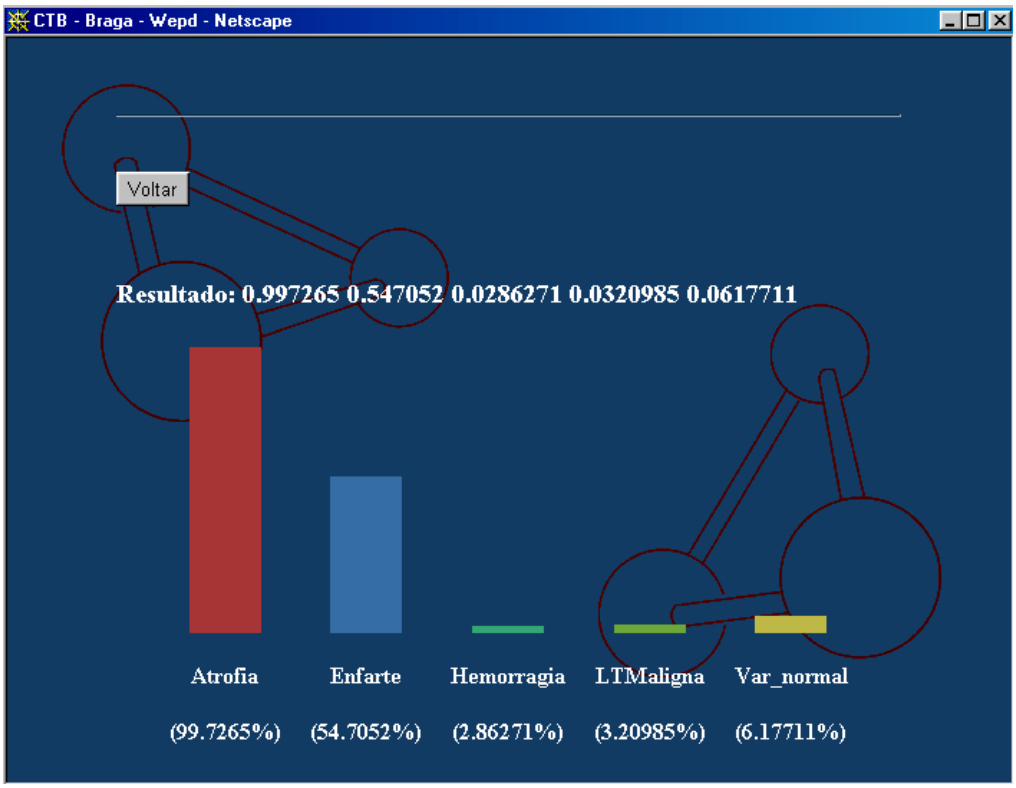


Figura 5.25 – Interface do Módulo *MD*: Diagnóstico através de uma Imagem

A partir de um conjunto de funcionalidades presentes no Módulo de Diagnóstico (*MD*), é possível submeter a uma *RNA* previamente treinada, um exame, sendo obtido uma sugestão de diagnóstico (Figura 5.26).

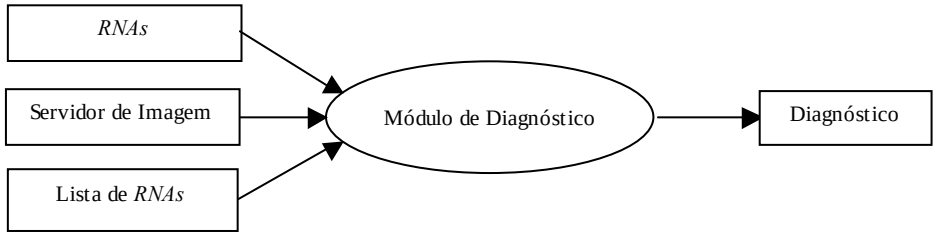


Figura 5.26 – Estrutura do módulo *MD*

5.5 Caso prático

A modalidade usada para este estudo foi a de *Tomografia Computorizada* (*TC*) num equipamento da *GE* (*General Electric*) modelo *GEprospeed*. As imagens estavam em formato *RAW*, tendo sido utilizadas um total de 188 imagens de outros tantos exames (aos pacientes não foram aplicados produtos de contraste). O exame seleccionado para análise foi o *TC* cerebral devido ao facto de os médicos que colaboraram neste estudo serem todos neuroradiologistas. A imagem escolhida de cada exame corresponde ao corte: OM – órbito-meatal, supratentorial a passar nos plexus, cornos anteriores dos ventrículos laterais, núcleos da base, cápsula interna (Figura 5.27). Esta escolha deveu-se ao facto de a imagem correspondente a este corte apresentar algumas características interessantes, enunciadas a seguir:

- Poucos artefactos;
- Área do cérebro com uma grande variedade de níveis de densidade de parenquima *UH* (Unidades de Hounsfield), correspondentes a substâncias tais como liquor céfalo-raquidiano, substância cinzenta, substância branca, osso;
- Disponibilidade de uma grande quantidade de exames com este tipo de imagem; e
- Imagem onde normalmente surge uma grande variedade de patologias, tais como tumores malignos, tumores benignos, enfartes ou hemorragias.

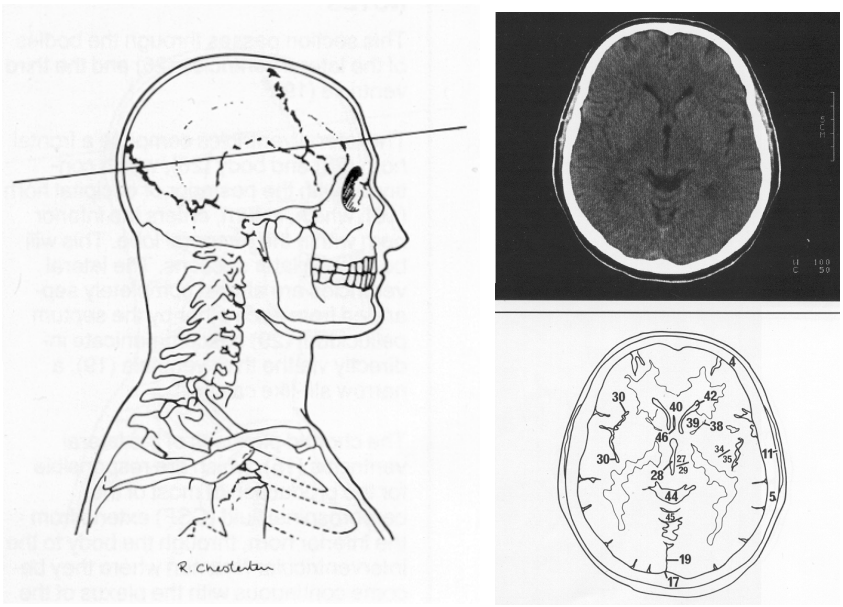


Figura 5.27 – Secção do corpo escolhida para o estudo

O agente de conhecimento incorpora *RNAs* multicamada, *feedforward*, completamente interligadas com apenas uma camada intermédia, ligação de *bias* e sem ligações de atalho, função de activação *sigmoid* e algoritmo de treino *RPROP*. Das imagens disponíveis, 25% foram usadas como casos de teste.

A camada de entrada da *RNA* é constituída pelos valores normalizados de cada imagem, juntamente com alguns dados do paciente (e.g., idade e sexo neste caso). A camada de saída é dada por um vector cujos valores apontam para um diagnóstico (Figura 5.28).

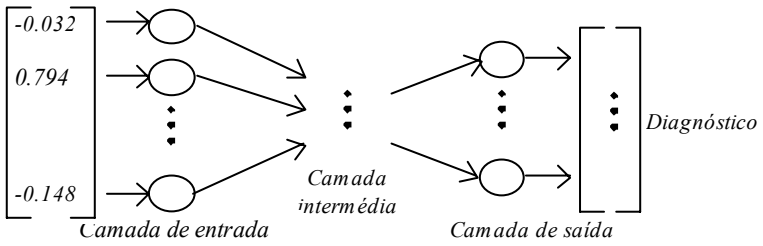


Figura 5.28 – Esquema da *RNA*

As imagens assim como os dados do paciente (e.g., idade e sexo, neste caso) foram fornecidos a dois neuroradiologistas. Foi-lhes pedido um diagnóstico, tendo em atenção somente a informação fornecida. O resultado é apresentado na Tabela 5.2 (note-se que para algumas das imagens foi diagnosticado mais do que uma patologia).

Utilizando parte das aplicações presentes no módulo *MAI* e a informação fornecida pelos neuroradiologistas, é gerado para cada médico um ficheiro (*testexxx.conf*) que servirá para treino das *RNAs*. O ficheiro *testexxx.conf* terá tantas linhas quantas as imagens classificadas, de acordo com a sintaxe:

#IMAGEM:PATOLOGIA:SEXO:IDADE

em que *IMAGEM* denota o nome do ficheiro com a imagem em formato *DICOM*, *PATOLOGIA* denota a(s) patologia(s) associada(s) à imagem e, *SEXO* e *IDADE* denotam respectivamente, o sexo e a idade do paciente.

Tabela 5.2 – As avaliações dos Neuroradiologistas.

<table>
<tr><th></th><th colspan="2">Médico A</th><th colspan="2">Médico B</th></tr>
<tr><td>Normal</td><td>125</td><td>125</td><td>111</td><td>111</td></tr>
<tr><td>Atrofia</td><td>48</td><td rowspan="5">73</td><td>62</td><td rowspan="5">101</td></tr>
<tr><td>Enfarte</td><td>12</td><td>24</td></tr>
<tr><td>Hemorragia</td><td>6</td><td>7</td></tr>
<tr><td>Lesão Tumoral Maligna</td><td>3</td><td>3</td></tr>
<tr><td>Variantes ao Normal</td><td>4</td><td>5</td></tr>
</table>

É interessante notar que nas mesmas circunstâncias e baseados na mesma informação, a opinião dos dois neuroradiologistas apenas coincidiu em 78% dos casos (Tabela 5.3). Refira-se a bem da verdade que isto não significa que qualquer um dos médicos tenha errado, pois os dados disponíveis eram claramente insuficientes para efectuar um bom diagnóstico. O que se afirma é que a análise que cada médico faz à imagem não é coincidente, pelo que um sistema de apoio ao diagnóstico colocado junto do visualizador poderá fornecer um complemento de informação ao especialista, o que pode estar a um passo utilizando sistemas médicos inteligentes do tipo *SADMED*.

Tabela 5.3 – Concordância entre as avaliações dos dois médicos.

	Casos	%
De acordo	147	78
Parcialmente de acordo	15	8
Desacordo	26	14

As imagens foram pré-processadas e normalizadas (e.g., a Tabela 5.4 apresenta o resultado do pré-processamento e normalização das imagens das Figuras 5.29 e 5.30). Na Figura 5.31 esta informação é dada em modo gráfico. Note-se a diferença entre as imagens que não apresentam qualquer patologia e as imagens que revelaram atrofia.

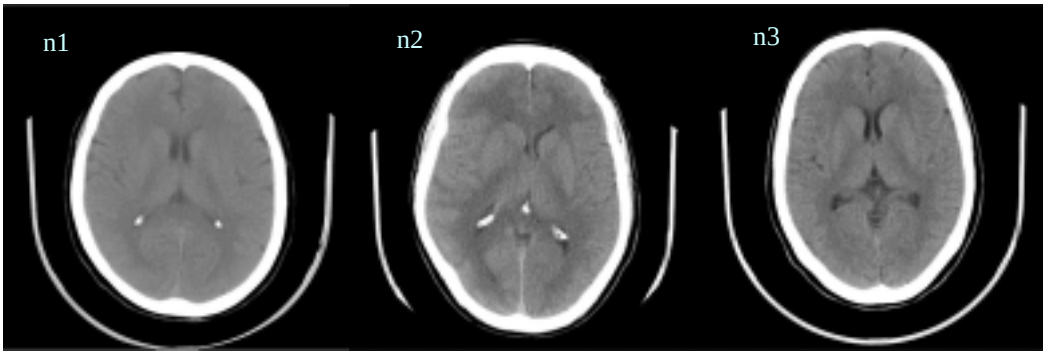


Figura 5.29 – Imagens que não revelam qualquer patologia: Normais

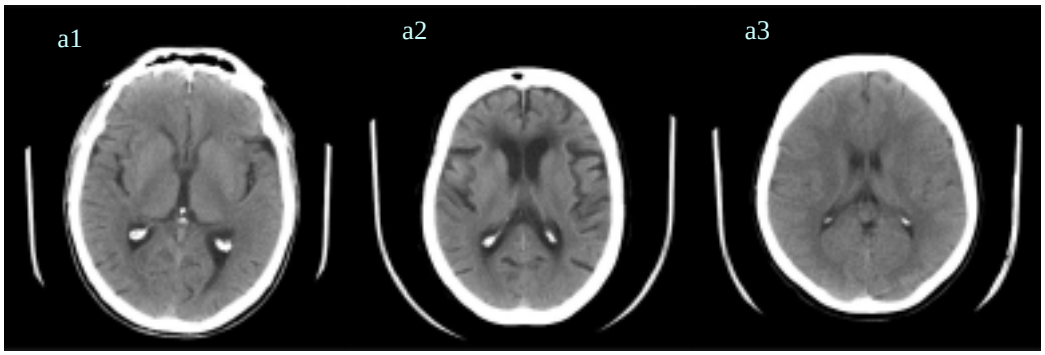


Figura 5.30 – Imagens que apresentam uma patologia: Atrofia

Tabela 5.4 – Valores normalizados referentes às imagens das Figuras 5.29 e 5.30.

ref	-10	0- water	27-plasma	45-lymphoma	55-blood	80-blood coagulated	130-bone (soft)	180-bone (hard)
n1	0.31823	0.45625	0.03514	-0.35506	-0.51878	-0.22025	-0.43859	-0.05544
n2	0.44538	0.63513	0.15467	-0.38964	-0.63793	-0.42789	-0.34840	-0.03514
n3	0.55308	0.63520	0.40038	-0.18085	-0.41580	0.05167	-0.03659	-0.45693
a1	-0.61198	-0.53500	-0.57255	-0.66888	-0.62671	-0.00147	0.50762	-0.29031
a2	-0.88692	-0.90335	-0.88370	-0.72303	-0.52083	0.03958	0.08898	-0.03677
a3	-0.60022	-0.59190	-0.59918	-0.69120	-0.76469	-0.49353	-0.34096	0.31703

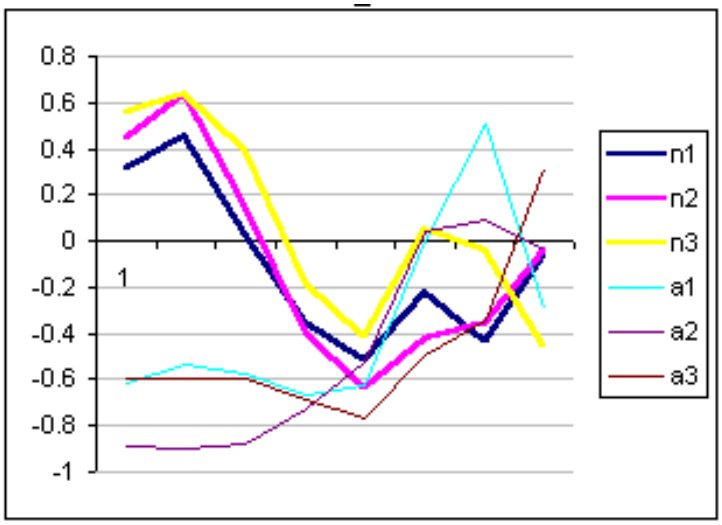


Figura 5.31 – Modo gráfico da Tabela 5.4

Os resultados obtidos com recurso a *RNAs* utilizando três técnicas diferentes de pré-processamento e normalização de dados, e consequentemente a três topologias para as *RNAs* (i.e., 10-5-5, 20-5-5 e 8-5-5 em termos do formato nodos_da_camada_de_entrada - nodos_da_camada_de_intermédia - nodos_da_camada_de_saída) e dando como saída uma ou mais das cinco patologias possíveis como diagnóstico, são apresentados na Tabela 5.5. Note-se que, quando se

utiliza a primeira técnica, em 47 casos de teste o sistema acertou em 31, com um desvio padrão de 2.46 (Tabela 5.5).

Alterando a configuração da *RNA* de modo a que a saída só dê como indicação se estamos perante uma imagem que revela ou não qualquer patologia, então os resultados melhoram consideravelmente (i.e., em 47 casos de teste o sistema acertou em 39, com um desvio padrão de 1.5 (Tabela 5.6).

Tabela 5.5 – Resultados obtidos pela *RNA* com discriminação de patologia.

casos de treino: 141 casos de teste: 47		Médico A		Médico B	
		acerto médio	std	acerto médio	std
pn3 10 - 5 - 5	RMSE(treino):	0.133471+-0.00368172	0.0102886	0.150552+-0.00427486	0.0119461
	correcto(treino):	127.1+-1.27248	3.5559600	123.1+-1.34009	3.7448800
	correcto(treino):	0.901418+-0.0090247	0.0252196	0.87305+-0.00950417	0.0265594
	RMSE(teste):	0.262203+-0.00870008	0.0243124	0.30215+-0.00683658	0.0191048
	correcto(teste):	31.2333+-0.879974	2.4590900	27.7+-0.746789	2.0869000
	correcto(teste):	0.664539+-0.0187229	0.0523211	0.589362+-0.0158891	0.0444022
pso2 20 - 5 - 5	RMSE(treino):	0.102474+-0.0055619	0.0155427	0.108804+-0.00641998	0.0179407
	correcto(treino):	134.3+-0.830761	2.3215600	132.233+-1.01512	2.8367500
	correcto(treino):	0.952482+-0.00589192	0.0164650	0.937825+-0.00719941	0.0201188
	RMSE(teste):	0.292564+-0.00656773	0.0183535	0.333025+-0.00726028	0.0202889
	correcto(teste):	30.4333+-0.767397	2.1444900	26.1667+-0.60963	1.7036100
	correcto(teste):	0.647518+-0.0163276	0.0456275	0.556738+-0.0129709	0.0362471
pso5 8 - 5 - 5	RMSE(treino):	0.143975+-0.00373412	0.0104350	0.175185+-0.00290799	0.0081264
	correcto(treino):	125.467+-1.42127	3.9717400	112.833+-2.27117	6.3467800
	correcto(treino):	0.889835+-0.0100799	0.0281684	0.800236+-0.0161076	0.0450126
	RMSE(teste):	0.274923+-0.00625132	0.0174693	0.297292+-0.00735091	0.0205421
	correcto(teste):	29.9+-0.847599	2.3686200	25.6+-0.707	1.9757100
	correcto(teste):	0.63617+-0.018034	0.0503961	0.544681+-0.0150426	0.0420365

Tabela 5.6 – Resultados obtidos pela *RNA* com uma só saída.

casos treino: 141 casos teste: 47		Médico A acerto médio	Médico B acerto médio
pn3 10 - 5 - 5 (todas)	correcto(treino):	127.1+-1.27248	123.1+-1.34009
	correcto(treino):	0.901418+-0.0090247	0.87305+-0.00950417
	correcto(teste):	31.2333+-0.879974	27.7+-0.746789
	correcto(teste):	0.664539+-0.0187229	0.589362+-0.0158891
pn3 10 - 5 - 1 (patologia)	correcto(treino):	135.8+-0.626185	134.2+-0.646995
	correcto(treino):	0.963121+-0.00444103	0.951773+-0.00458861
	correcto(teste):	38.5+-0.682531	38.8333+-0.665056
	correcto(teste):	0.819149+-0.0145219	0.826241+-0.0141501
pn3 10 - 5 - 1 (atrofia)	correcto(treino):	134.3+-1.71527	133.433+-1.44576
	correcto(treino):	0.952482+-0.012165	0.946336+-0.0102536
	correcto(teste):	37.6333+-0.582972	40.9+-0.558735
	correcto(teste):	0.800709+-0.0124037	0.870213+-0.011888
pn3 10 - 5 - 1 (enfarte)	correcto(treino):	139.033+-0.404019	138.433+-0.395179
	correcto(treino):	0.986052+-0.00286538	0.981797+-0.00280269
	correcto(teste):	43.3+-0.471282	36.6667+-0.482241
	correcto(teste):	0.921277+-0.0100273	0.780142+-0.0102605
pn3 10 - 5 - 1 (hemorragia)	correcto(treino):	138.633+-0.575348	136.933+-0.939435
	correcto(treino):	0.983215+-0.00408048	0.971158+-0.00666266
	correcto(teste):	44.3+-0.62442	42.9667+-1.0055
	correcto(teste):	0.942553+-0.0132855	0.914184+-0.0213936

Uma vez que a *RNA* apresenta o resultado em termos de uma percentagem, o sistema não só fornece um diagnóstico mas também dá uma indicação da confiança que se tem nesse diagnóstico (Figuras 5.24 e 5.25).

5.6 Conclusões

Mostrou-se aqui que é razoável continuar a apostar em procurar aumentar a aceitabilidade da *Análise de Dados Inteligente* (*ADI*) por parte da comunidade médica, através do desenvolvimento, adaptação e reutilização de sistemas baseados em conhecimento para resolução de problemas na área médica. É também convicção de quem esteve envolvido neste trabalho que para potenciar a utilização destas ferramentas na prática, os processos de *ADI* só têm a ganhar se contarem com

os médicos em todas as suas fases, começando nos procedimentos de recolha e tratamento dos dados, até à concretização e exploração dos resultados. De facto, um dos aspectos interessantes deste trabalho esteve precisamente no envolvimento dos médicos na preparação dos dados para os processos de *ADI* (e.g., representação de dados, modelação, pré-processamento, selecção ou transformação). Há que referir que cada *RNA* para auxílio ao diagnóstico é treinada pelo próprio médico, aprendendo com este. Tudo o que até aqui foi afirmado é somente o começo de uma longa caminhada para os sistemas *ADI* nas instituições de cuidados de saúde nacionais.

5.7 Trabalho Futuro

Uma outra abordagem, à qual apenas se faz referência em termos de trabalho futuro, seria a de dividir a imagem em *ROIs* (*Regions of Interest*) sendo, cada uma tratada individualmente pela sua própria *RNA*. Desta forma introduz-se o conceito de localização (i.e., dependendo da área de cada *ROI*, a quantidade de informação a ser processada diminui drasticamente e, por conseguinte, é facilitado substancialmente o treino da *RNA*).

Ao introduzir o conceito de localização está-se também a detalhar a análise, uma vez que além de se poder indicar que na imagem se detecta uma patologia também se poderá indicar a sua localização. Após o processamento de cada *ROI* pela *RNA* respectiva, é então possível utilizar os resultados obtidos como entradas numa outra *RNA*, a qual fará o pleno da imagem que está a ser processada.

Nesta abordagem deixa-se de ter apenas uma Rede Neuronal Artificial, mas sim uma rede de *RNAs*, dando lugar a uma arquitectura do tipo apresentado na Figura 5.32.

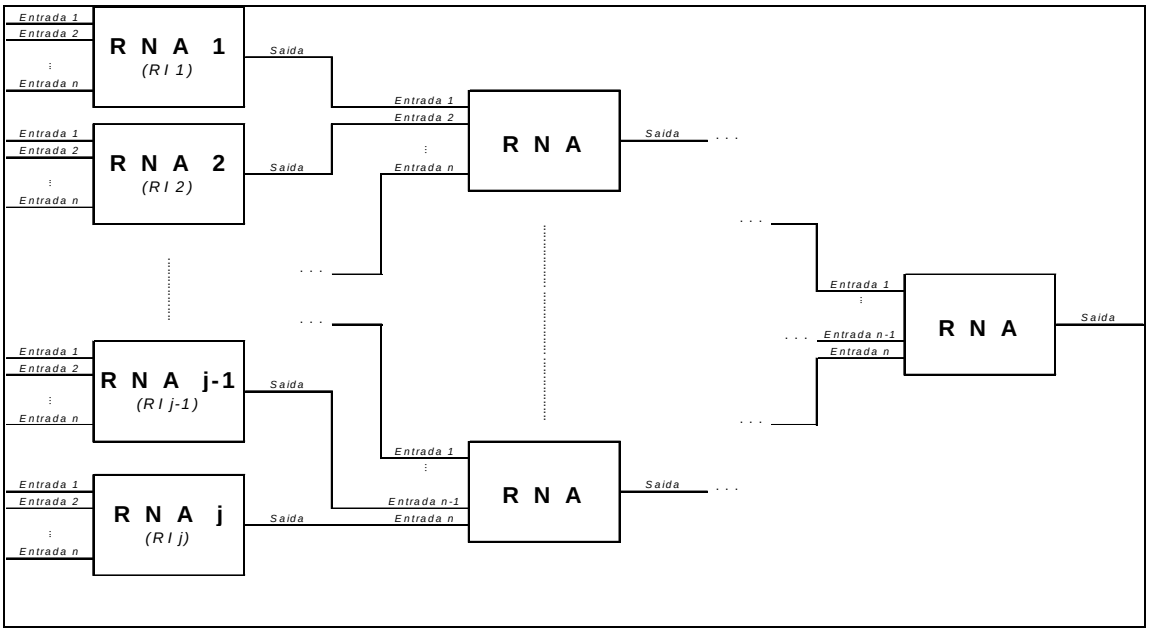


Figura 5.32 – Rede de *RNAs*

Capítulo 6
Conclusões e Trabalho Futuro

Faz-se o fecho do trabalho desenvolvido, lançando-se as pontes para trabalho futuro.

Um percurso como investigador não preconiza um fim em si mesmo; i.e., antes contextualiza o investigador numa amálgama labiríntica de caminhos que o próprio, deixando para trás a sua visão ciclópica do saber, se predispõe a questionar. Deste modo não se fecham portas, delimitando de forma estanque áreas de investigação. Pelo contrário, abrem-se outras que poderão conduzir a novos estádios do saber e proporcionar oportunidades de investigação porventura únicas e deveras compensadoras.

O presente capítulo enquadra e caracteriza de forma sinóptica o trabalho de pesquisa efectuado. Procurou-se contribuir para o aumento do conhecimento científico na área da Inteligência Artificial e da Saúde, mais concretamente na área do diagnóstico através da imagem médica. Não se pretendeu obter uma solução definitiva, mas sim contribuir com algumas soluções pontuais, sendo neste caso o sistema *SADMED* e a sua aplicação no diagnóstico médico pela imagem a face mais visível das contribuições aqui apresentadas; i.e., o percurso a calcorrear clarifica-se para que o estado de conhecimento aqui atingido não permaneça suspenso, mas

antes se afirme como um ponto de partida para novas aventuras no infindável universo do conhecimento.

Um dos objectivos atingidos com o trabalho efectuado teve a ver com aspectos ergonómicos da arquitectura e, dos sistemas desenvolvidos de forma a facilitar a sua utilização por indivíduos não especialistas em informática, o que se conseguiu tornando as ferramentas utilizadas o mais intuitivas possível e equipadas com interfaces visualmente e funcionalmente atractivas. Daí a opção pelas tecnologias da Internet em termos de interfaces médicas. Note-se, também, que nos sistemas desenvolvidos, a necessidade de introdução de dados foi reduzida ao mínimo; daí a ligação directa do material informático aos equipamentos de geração de imagem médica.

6.1 Síntese

A tipificação de um *SADM* surge na década de 70 (no século XX), como resultado do trabalho desenvolvido por Edward Shortliffe [Shortliffe 1976], em termos do sistema *MYCIN*, utilizado na detecção de infecções no sangue; i.e., pode-se afirmar que em termos de uma visão diacrónica, a problemática do desempenho dos *SADM* foi desde os anos setenta objecto de estudo, tendo havido desenvolvimentos significativos quer ao nível da representação do conhecimento, quer ao nível dos algoritmos para o tratamento de imagem médica e/ou outros casos de diagnóstico.

Neste trabalho e, seguindo as linhas referidas em epígrafe, teve-se como objectivo:

- Melhorar o desempenho computacional dos *SADM*; e

- Apetrechar esta tecnologia, olhando a modelos/arquitecturas para o cálculo automático de alto desempenho, em termos de *SPD*/*SM,* de forma a potenciar a utilização dos *SADM* numa nova gama de problemas: os problemas naturalmente distribuídos.

Capítulo 1

Foram apresentadas notas introdutórias relativas à aplicação de técnicas de Inteligência Artificial em diagnóstico médico, foi feita uma resenha da evolução da Inteligência Artificial, motivação, objectivos, e organização da tese.

Capítulo 2

Forneceu-se uma descrição dos Sistemas Multiagentes (*SM*). De forma a compreender o significado de *SM*, apresentou-se a sua constituição e modo de funcionamento.

Foi feita uma abordagem a sistemas de representação de conhecimento que contextualizam situações em que não existe informação completa acerca do universo de discurso.

Faz-se uma abordagem à *PL* como ferramenta para a representação de conhecimento, tornando-se necessário considerar a introdução de uma extensão a esta, de modo a ultrapassar a problemática do tratamento de informação incompleta.

A extensão à *PL* faz-se pela introdução, na linguagem, de dois tipos de negação: a negação por falha na prova, denotada pelo termo *não*, e a negação forte ou clássica, denotada por '¬'. Neste contexto, passa a ser

possível distinguir situações que são falsas de situações para as quais não existe prova de que sejam verdadeiras.

É tratado, ainda, o problema do *PMF* na *PLE*, abordando-se possíveis senãos que a formalização do conceito do *PMF* possa trazer à extensão de predicados de um programa em lógica estendido.

De seguida faz-se uma abordagem a diferentes tipos de valores que podem estar presentes em sistemas onde se verifiquem situações de informação incompleta, designados por valores nulos. São considerados três tipos de valores nulos: do tipo desconhecido, do tipo desconhecido mas de um conjunto determinado de valores e, do tipo não permitido.

Para os valores nulos apresentados, desenvolveu-se um programa em lógica estendida que permite o tratamento desses valores nos termos já enunciados.

Por fim, apresentou-se o desenvolvimento de um meta-predicado, para a manipulação de programas como os mencionados ao longo deste capítulo, tendo-se concretizado a sua utilização através do tratamento de situações introduzidas durante a apresentação dos diversos tipos de valores nulos.

Capítulo 3

Descreveu-se o funcionamento e características dos sistemas para a resolução de problemas usando Redes Neuronais Artificias (*RNAs*). As *RFMCs* têm sido empregues com sucesso em diversas domínios do conhecimento, incluindo as áreas da Engenharia, Direito, Ciências da Computação, Processos de Controlo, Robótica e Automação, Estatística, Medicina, Produção e Economia, entre outros. Esta grande popularidade das *RFMCs* advém das suas capacidades de estabelecimento de

correspondências não linear entre padrões. Normalmente, para um dado conjunto de dados de treino, uma rede com apenas uma (ou quanto muito duas) camada(s) intermédia(s) apreende uma dada tarefa com mestria [Neves & Cortez 1998].

Capítulo 4

Descreveu-se formalmente um ambiente computacional para sistemas de apoio ao diagnóstico médico, o sistema *SADMED*, apresentando-se a sua arquitectura, caracterizando cada constituinte do sistema e explicitando o seu *modus operandi*. Um protótipo do *SADMED – Ambiente Computacional para Sistemas de Apoio ao Diagnóstico Médico* foi objecto de teste com casos reais, sendo os resultados obtidos deveras encorajadores (a primeira aplicação comercial do sistema encontra-se já no terreno, a ser utilizada pelos serviços de Imagiologia do Hospital Geral de Santo António, no Porto, em Portugal). Estes resultados vieram confirmar certas opções consideradas no momento da análise e especificação do sistema, sendo de referir a problemática associada à sua escalabilidade. Este sistema foi projectado dando-se particular atenção a tal pressuposto, de tal modo que *SADMED* pode comportar um número qualquer de processadores, tendo como espinha dorsal a Lógica Matemática, e recorrendo a uma das suas variantes computacionais, a Programação em Lógica Estendida (*PLE*). Paradigma este que, tendo em linha de conta os avanços em Ciência Fundamental na área da Programação em Lógica, afirma-se não só como uma ferramenta para o cálculo automático, mas também como dando corpo a uma nova tecnologia para a resolução de problemas; i.e., passa-se a resolver problemas através da demonstração de teoremas. Por conseguinte, ferramentas especializadas, técnicas e ambientes são necessários para apoiar os engenheiros de software no desenvolvimento de sistemas em tamanho real baseados em *PLE*. Este foi o caso neste trabalho, em que a área de impacto (ou de referência) foi a área

médica e as tecnologias utilizadas as associadas à prestação de cuidados de saúde em tempo útil e de qualidade ao comum dos cidadãos, independentemente da sua localização geográfica ou condição social.

Capítulo 5

Foi exemplificada e avaliada a utilização do sistema *SADMED* no diagnóstico de imagens geradas por equipamentos de tomografia computorizada, tendo sido conseguidos resultados particularmente animadores.

6.2 Estado e Caracterização do Sistema *SADMED*

O desenvolvimento do sistema *SADMED* foi baseado em tecnologias de Internet com o fim último de aliar as potencialidades das tecnologias da Inteligência Artificial à facilidade de utilização e aprendizagem através da Internet. Este sistema apresenta-se como um protótipo, cumprindo criteriosamente a especificação inicial, em que há a destacar:

- É proprietário de um subsistema de coordenação que capta de uma forma eficiente as virtualidades dos *SPD/SM*;
- É um sistema portável, que pode ser executado em redes de computadores de baixo custo, nas plataformas mais comuns (e.g., Unix, Linux e Windows);
- É um sistema dotado de uma interface amigável, com inegáveis qualidades para o desenvolvimento e prototipação de novas aplicações de apoio ao diagnóstico médico, o que passa quer pela declaração das condições de aprendizagem, quer dos parâmetros do sistema (e.g., agentes a considerar, máquinas onde estes se devem alojar);

- A possibilidade de se separar claramente o conhecimento do sistema ao nível das estruturas de controlo, do conhecimento associado ao problema; e
- A capacidade de se operar ao nível da estrutura do sistema e do código, o que permite considerar outras formas de *SADM*, incluir heurísticas de resolução de conflitos, e definir novos tipos de diagnóstico.

6.3 Conclusões e Linhas de Orientação Futuras

O trabalho desenvolvido e aqui apresentado não é ortogonal, resulta da intersecção de vários domínios do saber de onde se recebe inspiração e onde, por seu turno, se criam influências. As contribuições deste projecto são de cariz marcadamente tecnológico, e passíveis de uma leitura dada na forma:

Segundo o Prisma da *IAD* – Inteligência Artificial Distribuída

Do ponto de vista da *IAD*, este trabalho teve como resultado o sistema *SADMED*, uma arquitectura[7] que suporta sistemas para o diagnóstico médico eminentemente distribuído, onde se faz uso de técnicas que se encontram nos *SPD/SM*, e se aplica maquinaria formal para a sua especificação (Capítulo 4). O resultado final é um *SPD/SM* de carácter experimental (um protótipo), que integra nos seus alicerces a filosofia dos *SBQN*. Para a criação deste sistema foram equacionadas alternativas que colocaram em confronto diversas opções, tais como: abordagem *SPD* vs. *SM*; agentes simples de baixa granularidade vs. agentes inteligentes/cognitivos de alta granularidade; controlo centralizado vs. controlo distribuído; comunicação via *QNs* vs. canais directos entre agentes. Foi feita uma aposta clara na simplicidade da arquitectura e na caracterização funcional dos agentes, na simplificação dos

[7] Segundo [Wooldrigde & Jennings 1995] podem identificar-se quatro linhas de acção para o estudo e desenvolvimento dos *SBA*: Teorias; Linguagens; Arquitecturas; e Aplicações.

mecanismos de controlo, coordenação e comunicação interagente. As potencialidades globais do sistema assim como a sua robustez, a sua eficiência e a sua flexibilidade emergem directamente dos processos de sociabilização passíveis de se desenvolver entre agentes. Os resultados apresentados no Capítulo 5 corroboram, em certa medida, as opções tomadas.

Segundo o Prisma dos *SADM*

No que respeita à tecnologia a aplicar nos *SADM*, este trabalho contribuiu para uma sistematização das metodologias de resolução de problemas e técnicas envolvidas na sua concepção e desenvolvimento, na área médica, de onde resultou:

- A definição de uma arquitectura para sistemas de apoio ao diagnóstico médico;
- O desenvolvimento de um protótipo para aplicação ao diagnóstico em tomografia computorizada; e
- A concepção de um algoritmo para o pré-processamento e normalização da informação contida numa imagem *DICOM* de *TC*.

6.4 Tese

O problema que se procurou equacionar e resolver com este trabalho teve a ver com a aplicação das potencialidades da Inteligência Artificial na área da Saúde.

> **Questão** – Qual será o papel, no futuro, para a Inteligência Artificial e os sistemas inteligentes na área da Saúde?

De forma sucinta tudo o que foi dito ao longo deste trabalho traduz-se na tese que se passa a enunciar:

Tese – Os sistemas inteligentes na área da saúde serão embebidos nos sistemas médicos e terão um papel fundamental no diagnóstico médico e na formação de pessoal médico e/ou auxiliar (e.g., técnicos, enfermeiros, pessoal auxiliar, o público em geral).

6.5 Trabalho Futuro

Uma outra abordagem, à qual apenas se faz referência em termos de trabalho futuro, passaria por dividir a imagem em *ROIs* (*Regions of Interest*), sendo cada uma tratada individualmente pela sua própria *RNA*. Desta forma introduz-se o conceito de localização (i.e., dependendo da área de cada *ROI*, a quantidade de informação a ser processada diminui drasticamente e, por conseguinte, é facilitado substancialmente o treino da *RNA*).

Ao introduzir o conceito de localização está-se também a detalhar a análise, uma vez que além de se poder indicar que na imagem se detecta uma patologia, também se poderá indicar a sua localização. Após o processamento de cada *ROI* pela *RNA* respectiva, é então possível utilizar os resultados obtidos como entradas numa outra *RNA*, a qual fará o pleno da imagem que está a ser processada.

Nesta abordagem deixa-se de ter apenas uma *RNA*, mas sim uma rede de *RNAs*, dando lugar a uma arquitectura do tipo apresentado na Figura 5.32.

Não obstante tudo o que foi dito, outro sector a considerar, para trabalho futuro, passa pela utilização do sistema *SADMED* para o diagnóstico em espectroscopia protónica por ressonância magnética. Esta técnica permite avaliar a presença de determinados metabolitos em lesões cerebrais e mediante a sua quantificação torna

possível realizar o diagnóstico dos mesmos. São obtidos gráficos em que se observam os picos correspondentes a esses metabolitos sendo o diagnóstico realizado através da análise desses mesmos gráficos. Esta técnica permite realizar diagnósticos mais precisos de lesões cuja apreciação apenas pela imagem não é possível obter (e.g., o diagnóstico diferencial entre determinados tumores cerebrais e abcessos).

Referências

[Ahuja et al. 1986] Ahuja, S., Carriero, N., Gelernter, D. LINDA and Friends, Computer, Volume 19, Number 8, pages 26-34, August, 1986.

[Altman 1997] Aitman, R.B., . lnformaties in the care of patients: Ten Notable Challenges, Western Journal of Medicine 166(2):118-122, 1997.

[Alves et al. 1993] Alves, V., Ribeiro, A., Neves, J., Distributed Problem Solving – An Universal Computer Architecture, 5th Workshop on Logic Programming Environments, Vancouver, Canada, 1993.

[Alves et al. 2001a] Alves, J. Neves, M. Maia, L. Nelas. Computer Tomography based Diagnosis using Extended logic programming and Artificial Neural Networks. Proceedings of the International NAISO Congress on Information Science Innovations ISI2001, Dubai, U.A.E., March 17-21, 2001.

[Alves et al. 2001b] V. Alves, J. Neves, M. Maia, L. Nelas. Knowledge Discovery and Data Mining in Medical Reports. Proceedings of CIMA2001, Bangor, Wales, United Kingdom, June 19-22, 2001.

[Alves et al. 2001c] V. Alves, J. Neves, M. Maia, L. Nelas. A Computational Environment for Medical Diagnosis Support Systems. Medical Data Analysis, Second International Symposium, ISMDA 2001, Madrid, Spain,

Proceedings LNCS 2199 Springer, October, 2001.

[Analide 1996] Analide, C., Representação de Conhecimento e Raciocínio em Estruturas Hierárquicas, Tese de Dissertação de Mestrado, Departamento de Informática, Universidade do Minho, Braga, Portugal 1996.

[Azoff 1994] E. Azoff. Neural Network Time Series Forecasting of Financial Markets. John Wiley & Sons, 1994.

[Baim 1988] Baim P.W., A Method for Attribute selection in Inductive Learning Systems, IEEE Trans. on PAMI, Vol. 10, Nº 6, pages 888-896, 1988.

[Barlett 1997] P. Barlett. For valid generalization, the size of the weights is more important that the size of the network. *Advances in Neural Information Processing Systems,* 9:134-140, 1997.

[Barnett & Cimino 1987] Barnett, G.O., Cimino, E.P., DXPLAIN- An Evolving Diagnostic Decision-Support System. Journal of the American Medical Association 258(1):67-74, 1987

[Berner et al. 1996] Berner E.S., Jackson J.R. and Algina J., Relationships among Performance Scores of Four Diagnostic Decision Support Systems. Journal of the American Medical Informatics Association 3(3):208-215, 1996.

[Bodenreider et al. 1998] Bodenreider O., Burgun A., Botti G., Fieschi M., LeBeux P. and Kohler F., Evolution of the Unified Medical Language System as a Medical Knowiedge Source. Journal of the American Medical lnformatics Association 5(1):76-87, 1998.

[Bose & Liang 1996] N. Bose and P. Liang. Neural Network Fundamentals with

Graphs, Algorithms and Applications. McGraw-Hill, USA, 1996.

[Boser et al. 1992] B. Boser, I. Guyon, and V. Vanik. A training algorithm for optimal margin classifiers. In Fifth Annual Workshop on Computational Learning Theory, pages 144-152. San Mateo, CA: Morgan Kaufmann, 1992.

[Bratko & Kononenko 1987] Bratko I., Kononenko I., Learning Rules from incomplete and Noisy Data, in B. Phelps (ed.), Interactions in Artificial Intelligence and Statistical Methods, Hampshire: Technical Press, 1987.

[Bratko & Mulec 1980] Bratko I., Mulec P., An Experiment in Automatic Learning of Diagnostic Rules, Informatica, Ljubljana, Vol. 4, Nº4, pages 18-25,1980.

[Bratko et al. 1989] Bratko I., Mozetic I., Lavrac N., KARDIO:A study in Deep and Qualitative Knowledge for Expert Systems, Cambridge, MA: MIT Press, 1989.

[Broomhead & Lowe 1988] D. Broomhead and D. Lowe. Multivariable functional interpolation and adaptative networks. *Complex Systems,* 2:321-355, 1988.

[Carpinter & Grossberg 1988] G. Carpinter and S. Grossberg. The ART of Adaptive Pattern Recognition by a Self-Organizing Neural Network. *Computer,* 21:77-88, 1988.

[Carriero & Gelernter 1989] Carriero, N., Gelernter, D., LINDA in Context, Communications of ACM, vol. 32, nº 4, pages 444-458, 1989.

[Catlett 1991] Catlett J., On changing continuous attributes into ordered discrete attributes, Proc. European Working Session on Learning-91, Porto, Março

4-6, pages 164-178, 1991.

[Cavedon & Tilhar 1995] Cavedon, L., Tilhar, G., A Logical Framework for Multi-Agent Systems and Joint Attitudes, in Proceedings of First Australian Workshop on Distributed Artificial Intelligence, Canberra, Australia, 1995.

[Cestnik et al. 1987] Cestnik B., Kononenko I. e Bratko I., ASSISTENT 86: A knowledge elicitation tool for sophisticated users, in: I. Bratko, N. Lavrac (eds), Progress in Machine learning, Wilmslow Sigma Press, 1987.

[Chan & Wong 1989] Chan K.C.C., Wong A.K.C., Automatic Construction of Expert Systems from Data: A Statistical Approach, Proc. IJCAI Workshop on Knowledge Discovery in Databases, Detroit, Michigan, August, pages 37-48, 1989.

[Cimatti & Serafini 1995] Cimatti, A., Serafini, L., Multiagent Reasoning with Belief Contexts II: Elaboration Tolerance, in Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95), pages 57-64, 1995.

[Clark & Boswell 1991] Clark P., Boswell R., Rule Induction with CN2: Some recent Improvements, proc. european Working Session on Learning –91, Porto, Portugal, Março, pages 151-163, 1991.

[Coelho 1995] Coelho, H., Inteligência Artificial em 25 Lições, Fundação Calouste Gulbenkian, Portugal, 1995.

[Denti et al. 1995] Denti, E., Natali, A., Omicini, A., Robot Control Systems as Contextual Logic Programs, in Logic Programming: Formal Methods and Practical Applications, Studies in Computer Science and Artificial

Intelligence 11, Elsevier, 1995.

[Efron & Tibshirani 1993] B. Efron and R. Tibshirani. An Introduction to the Bootstrap. Chapman & Hall, USA, 1993

[Elliot 1993] D. Elliot. a better activation function for artificial neural networks. Technical report tr 93-8, Institute for Systems Research, University of Maryland, Maryland, USA, January 1993.

[Elomaa & Holskti 1989] Elomaa T., Holsti N., An Experimental Comparison of Inducing Decision Trees and Decision Lists in Noisy Domains, Proc. 4th European Working Session on Learning, Montpeiller, Dec. 4-6, pages 59-69, 1989.

[Engelmore et al. 1988] Engelmore, R. S., Morgan, A., Nii, H. P., HERSAY-II, in Engelmore, R., Morgan, T., eds, Blackboard Systems, Addison-Wesley Publishing Company Inc., USA, 1988.

[Fahlman 1989] S. Fahlman. Faster Learning Variations on Back-Propagation: An Empirical Study. in: [Touretzky et al. 1989], 1989.

[Faraday & Chatfield 1998] J. Faraday and C. Chatfield. Times Series Forecasting with Neural Networks: A Case Study. *Applied Statistics,* 47:231-250, 1998.

[Ferber 1999] Ferber, J., Multi-Agent Systems – An Introduction to Distributed Artificial Intelligence, Addison-Wesley, England, 1999.

[Foster 1987] Foster I., Logic Operating Systems, Design Issues in proceedings of the Fourth Internatinal Conference on Logic Programming, Vol.2, pag. 910-926, MIT Press, May 1987.

[Franklin & Graesser 1996] Franklin, S., Graesser, A., Is It an Agent, or Just a Program? A Taxonomy for Autonomous Agents, in Müller, J. P., Wooldridge, M. J. & Jennings, N. R., eds, Intelligent Agents III – Agent Theories, Architectures, and Languages, Lecture Notes in Artificial Intelligence 11, Springer-Verlag, 1996.

[Freeman 1996] Freeman, E., Linda Group, Department of Computer Science, University of Yale, 1996. http://www.cs.yale.edu/HTML/YALE/CS/Linda/linda.html

[Gallant 1993] Gallant, S., Neural Network Learning and Expert Systems. MIT Press, Cambridge, USA, 1993.

[Gelfond & Lifschitz 1990] Gelfond, M., Lifschitz, V., Logic Programs with Classical Negation. In Proceedings of the International Conference on Logic Programming, 1990.

[Giunchiglia et al. 1993] Giunchiglia, E., Traverso, P., Giunchiglia, F., Multi-Context Systems as a Specification Framework for Complex Reasoning Systems, in J. Teur and T. Wetter, eds, Formal Specification Methods for Complex Reasoning Systems, Ellis Horwood Publishers, England, 1993.

[Goldberg 1989] Goldberg, D. E., Genetic Algorithms in search, optimization, and machine learning. Addison-Wesley Publishing Company, 1989.

[Grimson et al. 1996] Grimson W., Lozano-Perez T., Wells III W., Ettinger G., White S., Kikinis, An Automatic Registration Method for Frameless Stereotaxy, Image Guided Surgery, and Enhanced Reality Visualization, In Transactions on Medical Imaging, 1996.

[Haddadi 1996] Haddadi, A., Communication and Cooperation in Agent Systems: A Pragmatic Theory. Springer-Verlag, Heidelberg, 1996.

[Haykin 1999] S. Haykin. Neural Networks - A Compreensive Foundation. Prentice-Hall, New Jersey, 2nd edition, 1999.

[Hojker et al. 1988] Hojker S., Kononenko I., Jauk A., Fidler V., Porenta M., expert System's Development in the Management of Thyroid Diseases, Proc. European Congress for Nuclear Medicine, Milano, Sept., 1988.

[Holland & Miller 1991] Holland, J. H., Miller, J., Artificially Adaptive Agents in Economic Theory, American Economic Review, 81(2):365-370, 1991.

[Holland 1975] Holland, J. H., Adaptation in Natural and Artificial Systems, Ann Arbor: University of Michigan Press, USA, 1975.

[Holland 1995] Holland, J. H., Adaptation in Natural and Artificial Systems, Ann Arbor: University of Michigan Press, USA, 1995.

[Holland 1997] Holland, J. H., A Ordem Oculta - Como a Adaptação Gera a Complexidade, Gradiva, Portugal, 1997.

[Holmes 1997] Holmes, J. M., Discovering Risk of Disease with a Learning Classifier System, in Bäck, T., ed, Proceedings of the Seventh ICGA, Morgan Kaufmann Publishers, USA, 1997.

[Hopfield 1982] J. Hopfield. Neural Networks and Physical Systems with Emergent Collective Computational Abilities. In *the National Academy of Science,* volume 79, pages 2554-8, 1982.

[Horn et al. 1985] Horn K.A., Compton P., Lazarus L., Quinlan J.R., Na Expert System for the Interpretation of Thyroid Assays in a Clinical Laboratory, The Australian Computer Journel, Vol. 17, Nº 1, pages 7-11,1985.

[Hustadt 1994] Hustadt, U., "Do we need the closed-world assumption in knowledge representation?", in Working Notes of the KI'94 Workshop, pp. 24-26, Baader, Buchheit, Jeusfeld, Nutt (eds.), Saarbrücken, Alemanha, 1994.

[Jagannathan et al. 1989] Jagannathan, V., Dodhiawala, R., Baunm, L. S., Blackboard Architectures and Applications, Academic Press, Inc., 1989.

[Jordan 1995] M. Jordan. Why the logistic function? a tutorial discussion on probabilities and neural networks. 9503, Massachusetts Institute of Technology, USA, August 1995.

[Kanna et al. 2000] Kanna Rajan, Mark Shirley, William Taylor, and Bob Kanefsky, "Ground Tools for the 21st Century", in Proceedings of IEEE Aerospace Conference, Big Sky, MT, 2000.

[Karalic & Pirnat 1990] Karalic A., Pirnat V., Significance Level Based Classification with Multiple trees, Informatica, Ljubljana, Vol. 15, Nº 1, 1991, pages 54-58.

[Kern et al. 1990] Kerm J., Dezelic G., Tezak-Bencic M., Durrigl T., Medical Decision Making Using Inductive Learning Program, Proc 1st Congress on Yugoslav Medical Informatics, Beograd, Dec. 6-8 1990, pp 221-228.

[Kernsley & Martinez 1992] D. Kernsley and T. Martinez. A Survey Of Neural Network Research And Fielded Applications. *International Journal of*

Neural Networks: Research and Applications, 2:123-133, 1992.

[Kohavi 1995] R. Kohavi. A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. In *Proceedings of the International Joint Conference on Artificial Intelligence (1JCAI),* Montreal, Quebec, Canada, August 1995.

[Kohonen 1982] T. Kohonen. Self-organized formation of topologically correct feature maps. Biological Cybernetics, 43:59-69, 1982.

[Kononenko et al. 1984] Knonenko I., Bratko I., Roskar E., Experiments in automatic learning of medical diagnostic rules, International School for the Synthesis of Expert's Knowledge Workshop, Bled, Slovenia, august 1984.

[Kononenko et al. 1988] Knonenko I., Jauk T., Jank T., Induction of reliable decision rules, International School for the Synthesis of Expert's Knowledge Workshop, Udine, Italy, 10-13 sept. 1988.

[Kononenko et al. 2000] Knonenko I., Bratko I., Kukar M., Application of Machine Learning to Medical Diagnosis, em Machine Learning and Data Mining:Methods and Applications, R.S.Michalski, I.Bratko and M.Kubat (ed.), 2000.

[Kosko 1988] B. Kosko. Bidirectional Associative Memories. *IEEE Transactions on Systems, Man and Cybernetics,* SMC-18:49-60, 1988.

[Kowalski 1990] Kowalski, R. A., Problems and Promises of Computational Logic, in Proceedings of the Symposium on Computational Logic, Lloyd (ed.), pp. 80-95, Springer Verlag Lecture Notes in Computer Science, 1990.

[Kowalski 1995] Kowalski, R. A., Using Meta-Logic to Reconcile Reactive with Rational Agents, in Meta-Logic and Logic Programming, Apt and Turini (eds), pages 227-242, MIT Press,1995.

[Kripke 1971] Kripke S., Semantical Considerations on Modal Logic. In Reference and Modality, Oxford University Press, London, UK, 1971.

[Kwok & Yeung 1999] Kwok T., and Yeung D., Constructive algorithms for structure learning in feedforward neural networks for regression problems: A survey. *IEEE Transactions on Neural Networks,* 8(3):630-645, May 1999.

[LeCun 1993] LeCun Y., Efficient Learning and Second-order Methods. A Tutorial at NIPS 93, 1993.

[Lesmo et al. 1982] Lesmo L., Saitta L., Torasso P., Learning of Fuzzy Production Rules for Medical diagnoses, In Gupta M.M., Sanchez E.(eds.) Approximate Reasoning in Decision Analysis, North-Holland, 1982.

[Luger & Stubblefield 1998] Lugger, G. F., Stubblefield, W. A., Artificial Intelligence Structures and Strategies for Complex Problem Solving, Addison-Wesley, 1998.

[Machado 2001] Machado, J., Agentes Como Objectos de um Sistema Distribuído de Realidade Virtual, Tese de Doutoramento, Universidade do Minho, Braga, Portugal, 2001.

[McCarthy 1980] McCarthy, J., Circumscription – A Form of Non-Monotonic Reasoning, in Journal of Artificial Intelligence, 13 (1,2), pp. 27-39, 1980.

[McCulloch & Pits 1943] W. McCulloch and W. Pits. A Logical Calculus of the Ideas Immanent in Nervous Activity. Bulletin of Mathematical Biophysics, 5:115-133, 1943.

[Memmi 1989] D. Memmi. Connectionism and artificial intelligence. In Neuro-Nimes'89 International Workshop on Neural Networks and their Applications, pages 17-34, Nimes, France, 1989.

[Michie et al. 1994] Michie D., Spiegelhalter D.J., Taylor C.C (eds.) Machine Learning, Neural and Statistical Classification, Ellis Horwood, 1994.

[Minker 1982] Minker, J. On indefinite data bases and the closed world assumption, In Proceedings of the 6th Conference on Automated Deduction, Springer Verlag, editors, pages 292-308, 1982.

[Minsky 1990] M. Minsky. Logical vs. Analogical or Symbolic vs. Connectionist or Neat vs. Scruffy. MIT *Press*, 1, 1990.

[Muggleton 1990] Muggleton S., Inductive Acquisition of Expert Knowledge, Turing Institute Press and Addison-Wesley, 1990.

[Nayak & Rajan 1999] Nayak P., Rajan K., Validating the DS-1 Remote Agent Experiment, 5th International Symposium on Artificial Intelligence, Robotics and Automation in Space, ESTEC, Noordwijk, The Netherlands, June 1999

[Neves & Cortez 1998] J. Neves and P. Cortez. Combining Genetic Algorithms, Neural Networks and Data Filtering for Times Series Forecasting. In Nicos E. Mastorakis, editor, *2nd IMACS International Conference on: Circuits, Systems and Computers - IMACS-CSC'98,* volume *2,* pages 933-939*,*

Piraeus, Greece, October 1998.

[Neves & Garrido 1991] Neves J., Garrido P., Amalgamating the Neural and Logic Computing Paradigms, ICANN-91, Finland, 1991.

[Neves & Machado 1997] Neves, J., Machado, J., Formalizing Context in Knowledge Management Systems, in Proceedings of the International Workshop of Distributed Artificial Intelligence and Multi-Agent Systems (DAIMAS'97), St Petersburg, Russia, 1997.

[Neves 1984] Neves J. "A Logic Interpreter to Handle Time and Negation in Logic Data Bases", in Proceedings of the ACM'84 Annual Conference – The Fifth Generation Challenge, San Francisco, California, USA, 1984.

[Neves et al. 1997] Neves J., Machado, J., Analide, C., Novais, P., Abelha, A. "Extended Logic Programming applied to the specification of Multi-agent Systems and their Computing Environment", in Proceedings of the ICIPS'97 (Special Session on Intelligent Network Agent Systems), Beijing, China, 1997.

[Neves et al. 1994] Neves J., Santos M., Alves, V. An Adaptable and Dynamic Architecture for Distributed Problem-Solving Based on The Blackboard Paradigm, in Proceedings of the AI'94 – The Seventh Australian Joint Conference on Artificial Intelligence, Armidale, Australia, 1994.

[Neves et al. 1999a] Neves J., Abelha A., Machado J., Alves V., Rocha M., Cortez P., Basto S. and Botelho H., An Unified Framework for Data Modeling on Medical Information Systems, the XV International Congress of the European Federation for Medical Informatics (MIE-99), Ljubljana,

Slovenia, 1999.

[Neves et al. 1999b] Neves J., Alves V., Nelas L., Romeu A. and Basto S., An Information System That Supports Knowledge Discovery And Data Mining in Medical Imaging, Proceedings of the Workshop on Machine Learning in Medical Applications, Chania, Greece, 1999.

[Neves et al. 2000] Neves, J., Alves V., Nelas L., Maia M., and Cruz R. "A Multi-Feature Image Classification System that Reduces the Cost-of-Quality Expenditure", in Proceedings of the Second ICSC Symposium on Engineering of Intelligent Systems, Paisley, Scotland, UK, pages 594-554, 2000.

[Nii 1986] Nii, P., Blackboard Systems: The Blackboard Model of Problem Solving and the Evolution of Blackboard Architectures, AI Magazine 7(2), 38-53, 1986.

[Nii 1989] Nii, P., Introduction, in Jagannathan V., Dodhiawala, R., Baum, L., eds, Blackboard Architectures and Applications, Academic Press, Inc., 1989.

[Nuñez 1990] Nuñez M., Decision Tree Induction Using Domain Knowledge, In Wielinga B. et al. (eds.), Current Trends in Knowledge acquisition, IOS Press, 1990.

[O'Hare & Jennings 1996] O'Hare, G. M. P., Jennings, N. R., Foundations of Distributed Artificial Intelligence, John Wiley and Sons, eds, 1996.

[Patterson 1996] D. Patterson. *Artificial Neural Networks - Theory and Applications.* Prentice Hall, Singapore, 1996.

[Prechelt 1994] L. Prechelt. PROBEN1 - A Set of Neural Network Benchmark Problems and Benchmarking Rules. Research Report, Fakultät für Informatik, Universität Karlsruhe Germany, 1994.

[Quinlan et al. 1987] Quilan R., Compton P., Horn K.A., Lazarus L., Inductive knowledge acquisicion: A case study, in J.R.Quilan (ed.), Applications of Expert Systems, Turing Institute Press and Addison Wesley, 1987.

[Ramón & Cajal 1990] S. Ramón, Cajal, *New Ideas on the Structure of the Nervous System in Man and Vertebrates.* MIT Press, Cambridge, MA, translation of the French edition of 1894, 1990.

[Rich & Waters 1988] Rich, C. and Waters, R. "The Programmer's Apprentice: a research overview", IEEE Computer, 21(11), pg. 11-24, November 1988.

[Rich & Waters 1990] Rich, C. and Waters, R. "The Programmer's Apprentice", Addison-Wesley, 1990.

[Riedmiller & Braun 1993] M. Riedmiller and H. Braun. A Direct Adaptative Method for Faster Backpropagation Learning: The RPROP Algorithm. In *Proceedings of the IEEE International Conference on Neural Networks,* San Francisco, CA, USA, 1993.

[Riedmiller 1994a] M. Riedmiller. Supervised Learning in Multilayer Perceptrons - from Backpropagation to Adaptive Learning Techniques . *Computer Standards and Interfaces, 16,* 1994.

[Riedmiller 1994b] Riedmiller. M., Rprop - Description and Implementation Details. Technical report, University of Karlsruhe, Karlsruhe, Germany, January

1994.

[Rieter 1980] Reiter R., A Logic for Default Reasoning. In Special Issue on Non-Monotonic Logics, 1994.

[Rojas 1996] R. Rojas. *Neural Networks - A Systematic Introduction.* Springer-Verlag, Germany, 1996.

[Roskar et al. 1986] Roskar E., Abrams P., Bratko I., Kononenko I., Varsek A., MCUDS - An expert system for the diagnostics of lower urinary tract disorders, Journal of Biomedical Measurements, Informatics and Control, Vol. 1 nº4, pages 201-204, 1986.

[Rumelhart et al. 1986] D. Rumelhart, G. Hinton, and R. Williams. Learning Internal Representations by Error Propagation. In D. Rulmelhart and J. McClelland, editors, *Parallel Distributed Processing: Explorations in the Microstructures of Cognition,* volume 1, pages 318-362, MIT Press, Cambridge MA, 1986.

[Russel & Norvig 1995] Russel, S., Norvig, P., Artificial Intelligence - A Modern Approach, Prentice-Hall, New Jersey, USA, 1995.

[Santos & Neves 1993] Santos, M. and Neves, J., Parallel Distributed Processing in Genetic-Based Machine Learning, Relatório Interno. Universidade do Minho, Portugal. 1993.

[Santos & Neves 1998] Santos, M., Neves, J., Modelling Learning Classifiers as Multiagent Systems, in Proceedings of II Iberoamerican Workshop on DAI and MAS, Toledo, Spain, 1998.

[Santos 1999] Santos, M., Sistemas de Classificação em Ambientes Distribuídos, Tese de Doutoramento, Universidade do Minho, Braga, Portugal, 1999.

[Sarle 1995] W. Sarle. Stopped Training and Other Remedies for Overfitting. In *Proceedings of the 27th Symposium on the Interface of Computer Science and Statistics,* pages 352-360*,* 1995.

[Sarle 1999] W. Sarle. Neural network faq. ftp://ftp.sas.com/pub/neural/FAQ.html*,* 1999.

[Shortliffe 1976] Shortliffe, E. H., Computer-Based Medical Consultations: MYCIN. Elsevier / North-Holland, Amsterdam, London, New York, 1976.

[Searle 1990] Searle, J. R., Is the Brain's Mind a Computer Program?, in Scientific America, January, USA, 1990.

[Sharda & Rampal 1996] R. Sharda and R. Rampal. Neural Networks and Management Science/Operations Research: A Bibliographic Essay. *Encyclopedia of Library and Information Science,* 61:247-259, 1996.

[Shinghal 1992] Shinghal, R., Formal Concepts in Artificial Intelligence, Chapman & Hall, 1992.

[Thimm & Fiesler 1997] G. Thimm and E. Fiesler. Pruning of Neural Networks. Research report idiap-rr 97-03, IDIAP, 1997.

[Traylor & Gelfond 1993] Traylor, B., Gelfond, M., Representing Null Values in Logic Programming, in Proceedings of the International Logic Programming Symposium (ILPS'93), Vancouver, British Columbia,

Canada, 1993.

[Weiss 1999] Weiss, G., Multiagent Systems A Modern Approach to Distributed Artificial Intelligence, MIT Press, 1999.

[Witting 1992] Witting, T, ed, ARCHON An Architecture for Multi-agent Systems, Ellis Horwood, 1992.

[Wolfram 1995] Wolfram, D.A., An Appraisal of INTERNIST-1. Artificial Inteligence in Medicine 7(2):93-116, 1995.

[Yaha & Henschen 1985] Yaha, A., and Henschen, L. “Deduction in non-horn databases”, in Journal of Automated Reasoning 1(2), pages 141-160, 1985.

[Zwitter et al. 1983] Zwitter M., Bratko I., Kononenko I., Rational and irrational reservations against the use of computer in medical diagnosis and prognosis, Proceedings of the 3rd Mediterraneam Conference on Medical and Biological Engineering, Portoroz, Slovenia, 1983.

Apêndice A
Créditos

- ***SICStus Prolog*** é uma marca registada do Swedish Institute of Computer Science, Suécia.

- ***LINDA*** é uma biblioteca do SICStus Prolog, uma marca registada da Scientific Computing Associates, New Haven, USA.

- ***Unix*** é uma marca registada da AT&T.

- ***X Windows*** é uma marca registada do MIT.

- ***Windows*** é uma marca registada da Microsoft.

- ***Netscape*** é uma marca registada da Netscape.

Apêndice B
Referencial de Informática Médica

Neste ponto é compilado um conjunto significativo de referências relativas à área da Informática Médica (*IM*), com especial ênfase nas ligadas ao processamento e representação de imagem médica e estruturadas em três grupos distintos: conferências mais importantes; um guia de endereços de sítios na Internet (we*b sites*) de projectos, conferências, pessoas ou instituições relacionados com os temas abordados neste trabalho e, um dicionário de termos relacionados com imagiologia.

Conferências

- IAAI ("*Innovative Applications of Artificial Intelligence"*);
- PAP ("*Practical Applications of Prolog*");
- ACEP ("*Annual Conference on Evolutionary Programming*");
- ISMDA ("*International Symposium on Medical Data Analysis*");
- MIE ("*International Congress of the European Federation for Medical Informatics*);
- ISI ("*Information Science Innovations*");
- EIS ("*Engineering of Intelligent Systems*").

Guia de Recursos Electrónicos

Descrição	URL
AAAI *American Association for Artificial Intelligence*. (Associação Americana para a Inteligência Artificial).	http:www.aaai.org/
AccuSoft AccuSoft - empresa especializada em aplicações de imagem médica.	http://www.accusoft.com/medical_imaging/
Adaptive Computation Group Sediado no Departamento de Ciências da Computação da Universidade do Novo México sob a direcção de Stephanie Forrest.	http://www.cs.unm.edu/~forrest/adaptive-web/ac_main.htm
Auntminnie Portal de Radiologia.	http://www.auntminnie.com
Cortez Página de Redes Neuronais de Paulo Cortez.	http://shiva.di.uminho.pt/~pcortez
Dclunie Recursos DICOM.	http://idt.net/~dclunie/
DICOM Faq's Perguntas e respostas sobre DICOM.	http://www.lib.ox.ac.uk/internet/news/faq/archive/
DICOM draft standards Lista de recursos DICOM.	ftp://dicom.offis.uni-oldenburg.de/pub/dicom
DICOM software Central RSNA.	http://www.erl.wustl.edu
DICOM Software Master Index Lista de software DICOM.	http://www.erl.wustl.edu/Dicom_web/ctn_list.html
DICOM Standard Publisher Publicações sobre DICOM.	http://www.nema.com/nema
EVONET Rede de Excelência em Computação Evolutiva, criada em 1996 sob os auspícios do ESPRIT (programa Europeu de Tecnologias da Informação).	http://www.dcs.napier.ac.uk/evonet/
GEMINA *Conexionist, Evolutionary and Genetic Computational Group*. Sediado no Departamento de Informática da Universidade do Minho, *sob* a direcção de José Neves.	http://www.venus.di.uminho.pt/SI/GEMINA
IAAI Conferência *Innovative Applications of Artificial*	http:www.aaai.org/Conferences/IAAI/

Descrição	URL
Intelligence. (Aplicações inovadoras de Inteligência Artificial).	
Medical Vision Group Grupo de visão médica sediado no MIT AI Lab.	http://www.ai.mit.edu/projects/medical-vision
NCARAI *The Navy Center for Applied Research in Artificial Intelligence*. Sediado na Divisão de Tecnologias da Informação do Laboratório de Investigação Naval, sob a direcção de Alan Meyrowitz.	http://www.aic.nrl.navy.mil/
Neural Network Faq's Perguntas e respostas sobre RNAs.	ftp://ftp.sas.com/pub/neural/FAQ.html
Papyrus 3.0 Formato de ficheiro compatível com DICOM.	http://www.expasy.ch/www/UIN/html1/projects/papyrus/papyrus.html
Pegasus Pegasus Medical Imaging Software – empresa especializada em algoritmos de compressão de imagens.	http://www.jpg.com/medical
Penn State Radiology DICOM Uma introdução à norma DICOM.	http://www.xray.hmc.psu.edu/dicom/dicom_intro/DICOMIntro.html
RSNA Radiological Society of North America. (Associação de Radiologia da América do Norte).	http://www.rsna.org

Dicionário de termos (imagiologia)

Termo	Descrição
ACR-NEMA	*American College of Radiology - National Electrical Manufacturers Association.*
ACSE	Elemento de controle de serviço duma associação; ou tipo de elementos de serviço usados para criar uma associação.
AE	*Application Entity*. (Entidade de aplicação)
ANSI	*American National Standards Institute.* (Instituto nacional americano de normas)
Associação	Contexto para comunicação
ATM	*Asynchronous Transfer Mode*. (Modo de transferência assíncrono)
Atributo	Propriedade de um objecto de informação, tem um nome e um valor que são independentes de qualquer esquema de codificação.
Caixa de filme	Um arquivo, ou um filme
Caixa de imagem	Área de um filme na qual uma imagem aparece.
CDS	*Clinical Display Station*. (Estação de apresentação de dados)
Classe de objecto de informação	Uma descrição formal de um objecto de informação que inclui uma descrição de seu propósito e os atributos que possui. Não inclui valores para estes atributos.
Classe de serviço	Descrição estruturada de um serviço suportada pelas entidades de aplicação *DICOM* cooperantes, usando comandos específicos *DICOM* e agindo numa classe específica de objectos de informação.
Comando	Um comando *DICOM* é um meio genérico para operar sobre objectos de informação através de uma interface com o utilizador ou via rede.
Composto	Tipo de objecto de informação que contém atributos múltiplos, ao contrario de objectos normalizados que só contêm um atributo.
Compressão	Técnica pela qual o tamanho de um arquivo ou imagem fica reduzido, mantendo a qualidade requerida para uma determinada aplicação.
Conjunto de dados	Informação trocada que consiste directamente num conjunto estruturado de valores de atributos ou indirectamente relacionado com os objectos de informação. O valor de cada atributo no conjunto de dados é expresso como um elemento de dados.

Termo	Descrição
Contexto de aplicação	Associado à ligação de transporte. Um único contexto de aplicação é usado para toda a comunicação *DICOM*.
Contexto de apresentação	Traduz-se na forma em como se certifica que; meios de ambos os lados de uma comunicação concordam com o que é que vão fazer e como codificar os dados.
CR	*Computed Radiography*. (Radiografia computorizada)
CRD	*Customer Requirements Document*. (Documento com os pedidos do cliente)
CT	*Computed Tomography*. (Tomografia computorizada)
Declaração de Conformidade	Declaração formal associada a uma implementação específica da norma *DICOM*. Especifica as classes de serviço, objectos de informação, e protocolos de comunicação suportados pela implementação corrente da norma.
DF	*Digital Fluoroscopy*. (Fluoroscopia digital)
Dicionário de dados	Registo da tabela de elementos *DICOM* que associam uma etiqueta única, um nome, as características do valor, e o significado a cada um dos dados.
DICOM	*Digital Imaging and Communications in Medicine* (também denominado ACR-NEMA 3.0)
DIMSE	*DICOM* Message Service Element. (Elemento do serviço de mensagens *DICOM*)
$\boldsymbol{D_{max}}$	Densidade máxima utilizada por um dispositivo para gerar uma imagem.
$\boldsymbol{D_{min}}$	Densidade mínima utilizada por um dispositivo para gerar uma imagem.
Elemento de comando	Codificação de um parâmetro de comando que carrega o valor do parâmetro.
Elemento de dados	Unidade de informação, definida por uma única entrada no dicionário de dados.
Entidade de aplicação	Pode ser um programa, um pedaço dum programa, ou um grupo de programas que cooperam uns com os outros. De um modo abstracto, pode ser entendido como tratando-se de uma única entidade colectiva.
ER	*Emergency Room*. (Sala de emergência)
Escala de tons	Distribuição de valores cinza utilizados para representar uma imagem.
Exame/Estudo	Colecção (conjunto) de uma série de imagens.
FD	*Film Digitizer*. (Digitalizador de películas)

Termo	Descrição
FDDI	*Fiber Distributed Data Interface*. (Interface de dados distribuída em fibra)
Ficheiro DICOM	Um ficheiro ou arquivo *DICOM* é formatado de acordo com a norma *DICOM*. Apresenta-se com um cabeçalho com informação seguido de um conjunto de dados de acordo com o formato *DICOM*.
Fluxo de comando	Resultado da codificação de um conjunto de termos que agrupados por um comando em *DICOM*.
Fluxo de dados	Resultado da codificação de um conjunto de termos em *DICOM*.
Formato do ficheiro DICOM	O formato do ficheiro DICOM disponibiliza os meios para encapsular num ficheiro o conjunto de dados representando uma instância de *SOP* que esteja relacionada com um objecto de informação *DICOM*.
HIS	*Hospital Information System*. (Sistema de informação hospitalar)
I.D. do contexto de apresentação	Identificador único para um contexto de apresentação.
ICU	*Intensive Care Unit*. (Unidade de cuidados intensivos)
IEEE	*Institute of Electrical and Electronics Engineering*. (Instituto de Engenharia de Electricidade e Electrónica)
Informação do paciente	Dados demográficos de um paciente de imagiologia.
Instância de objecto de informação	Representação da ocorrência de uma entidade do mundo real, incluindo valores para os atributos da classe de objectos de informação à qual a entidade pertence.
ISDN	*Integrated Services Digital Network.* (Rede Digital com Integração de Serviços - *RDIS*)
LAN	*Local Area Network.* (Rede de área local)
LDDL	*Local density to display LUT*. (Densidade local para exibir *LUT*)
Ligação de transporte	Modo de mover dados não estruturados através de uma rede (e.g., *socket* ou *stream TCP/IP*).
LUT	*Look-Up Table*. Matriz que mapeia os valores da imagem de um contexto para outro.
MAN	*Metropolitan Area Network.* (Rede de área metropolitana)
Mensagem	Unidade de dados do protocolo de troca de mensagens, trocada entre duas entidades de aplicação *DICOM* que cooperam. Uma mensagem é composta por um fluxo de comandos seguido, opcionalmente de um

Termo	Descrição
	fluxo de dados.
MIP	*Maximum Intensity Projection*. (Projecção de intensidade máxima)
Modalidade	Dispositivo com capacidade para gerar imagens médicas.
Modelo de informação DICOM	Diagrama de entidade/relacionamento que é usado para modelar as relações entre o objecto de definições da informação que representam classes de objectos do mundo real definidas pelo Modelo de Aplicação *DICOM*.
Módulo	Grupo de atributos de objecto de informação relacionados.
MPR	*Multi-planar Reformatting*. (Reformatação multi-planar)
MRA	*Magnetic Resonance Angiography*. (Angiografia por ressonância magnética)
MRI	*Magnetic Resonance Imager*. (Visualizador de ressonância magnética)
NIU	*Network Interface Unit*. (Interface de rede)
NM	Nuclear Medicine. (Medicina nuclear)
Normalizado	Tipo de objecto de informação que contém um só atributo, ao invés de um objecto composto que pode conter múltiplos atributos.
Objeto de informação	Abstracção de uma entidade real de informação (e.g., *CT*, estudo) que é manipulado por um ou mais comandos *DICOM*.
OSI	*Open System Interconnection*. (Interconexão de sistemas abertos)
PACS	*Picture Archiving and Communication System*. (Sistema de arquivo e comunicação de imagem)
Par de objecto de serviço	Combinação de um serviço e o tipo de objecto de informação com o qual pode operar.
PDS	*Personal Display System*. (Sistema de visualização pessoal)
Perfil	Conjunto de operações ou serviços a serem executados. Semelhante ao conceito de *perfil de uma aplicação*, mas inclui a aplicação, a interface da aplicação com a rede, e um conjunto de opções. Um perfil forma a base para avaliar a conformidade *DICOM*. Um produto não esta conforme com a norma *DICOM*, mas sim com um dos perfis *DICOM* disponíveis.
Pilha de impressão	Grupo de imagens que estão em fila de espera para serem impressas.
PPP	*Point-to-point Protocol*. (Protocolo ponto-a-ponto)
Primitivas OSI	Quatro blocos de construção de comunicação *OSI*: *pedido*; *indicação*;

Termo	Descrição
	resposta; e *confirmação*.
RIS	*Radiology Information System*. (Sistema de informação da radiologia)
SCP	*Service Class Provider*. (Classe de serviço fornecedor)
SCSI	*Small Computer System Interface*. (Interface de ligação de um computador a dispositivos)
SCU	*Service Class User*. (Classe de serviço cliente)
SDL	*Source Density LUT*. (Densidade de fonte *LUT*)
Serviço	Entidade que permite comunicação de camada-a-camada.
Sessão de filme	Grupo de filmes, ou "caixas de filme".
Sintaxe abstracta	Equivalente a uma classe *SOP* (par serviço/objecto). Uma combinação da classe de serviço e o tipo de objecto de informação no qual devem operar.
Sintaxe de transferencia	Modo como os objectos devem ser codificados.
SLIP	*Single Line Internet Protocol*. (Protocolo de Internet de linha única)
SOP	*Standard Operating Procedure.* (Procedimento operacional normal)
SRC	*Systems Response Center*. (Centro de resposta dos sistemas)
TCP/IP	*Transmission Control Protocol/Internet Protocol.*
Tipos de elementos	1 - exigido; 1C - exigido se uma condição é conhecida; 2 – diagnóstico primário exigido; 2C – diagnóstico primário, exigido se uma condição é conhecida; 3 – opcional.
TSL	*Tonescale LUT*. (Escala de tons *LUT*)
UMC	Unit Manufacturing Cost. (Custo industrial unitário)
Unidade de dados protocolar	Bloco usado para codificar dados que passam numa associação.
US	*Ultrasound*. (Ultra-som)
V&V	*Verification and Validation*. (Verificação e validação)
WAN	*Wide Area Network*. (Rede de área larga)
Widget	Item do interface de utilizador.
WW/WL	Window Width / Window Level. (Largura da janela / nível da janela)

Apêndice C
A Norma DICOM

DICOM é um acrónimo para *Digital Imaging and Communications in Medicine* e é uma norma utilizada na indústria para adquirir e transferir imagens e informação médica entre dispositivos electrónicos, tendo como objectivo contribuir para o advento de um protocolo universal. Permite que os utilizadores integrem vários e diferentes equipamentos de aquisição e processamento de imagem (e.g., ecografia, tomografia computorizada, ressonância magnética, impressora de películas), de diferentes fabricantes, fazendo com que estes comuniquem entre si.

Esta norma foi desenvolvida pela *American College of Radiology* (*ACR*) e pela *National Electrical Manufactures Association* (*NEMA*). Sendo a *ACR* a responsável pela direcção técnica e supervisão médica, a *NEMA* age como a entidade que publica e fornece supervisão legal, para além de procurar evitar conflitos de interesses entre os seus associados.

Supõem-se que qualquer sistema de *PACS* (*Picture Archiving and Communication Systems*) deve seguir a norma *DICOM*, embora sejam poucos os que a suportam em pleno (e.g., um sistema pode ter em *DICOM* o arquivo de imagem, mas não o diagnóstico primário remoto).

Como se induz a partir do próprio nome, *DICOM* denota o formato em que as imagens médicas, que são obtidas em modalidades como o *TC*, *RM* e *RX*, são armazenadas. É a norma universal adoptada para a representação digital de informação médica. Um ficheiro *DICOM* disponibiliza informação sobre o paciente, o tipo de exame, as características da imagem médica e dados inerentes aos equipamentos geradores dessas imagens.

***ACR-NEMA* Versão 1.0** - O comité *ACR-NEMA* debruçou-se sobre os dispositivos de interface existentes no momento, nenhum sendo considerado completamente satisfatório. A norma, que na altura era oficialmente conhecida como *ACR-NEMA* 300-1985, passa a ser conhecido na indústria como *ACR-NEMA* Versão 1.0.; foi apresentada na reunião anual da Sociedade de Radiologia da América do Norte (*RSNA*) em 1985.

***ACR-NEMA* Versão 2.0** - *ACR-NEMA* 300-1988, que é mais conhecido como *ACR-NEMA* Versão 2.0, foi apresentada na *RSNA* em 1988.

Nesta versão ultrapassaram-se algumas das ambiguidades associados com o protocolo. Esta versão, que também ficou conhecida como o "conector de 50-pinos," foi adoptada apenas por um número limitado de fornecedores: *GE*, *Siemens*, *Philips*, e *Vortech Data Inc.*. E a primeira implementação comercial foi apresentada no *RSNA* de 1990 pela *GE* e pela *Vortech*, as únicas duas companhias a comercializarem-no.

Ficou claro que uma solução de comunicação ponto-a-ponto não foi ao encontro dos fornecedores de soluções e das necessidades futuras da comunidade de utilizadores. Era então necessária uma norma de rede. Coincidente com a apresentação da solução dos "50-pinos", a *Vortech* desenvolveu uma versão da norma que implementa *ACR-NEMA* Versão 2.0 sobre *TCP/IP*. Embora esta solução não

enderece todas as potencialidades da Internet, permitiu que se efectuasse o envio de imagens do foro médico através de uma rede. Ao mesmo tempo, a *Siemens* e a *Philips* desenvolveram o *Standard Product Interconnect* (*SPI*), que era, nem mais do que uma implementação do *ACR-NEMA* Versão 2.0 em rede.

ACR-NEMA* Versão 3.0, *"DICOM" - A norma *DICOM* foi desenvolvida para suportar a interligação das partes em rede, e não só interfaces ponto-a-ponto. Esta norma foca-se na interoperação em rede, e não só a interligação (Figura C.1). A norma é frequentemente chamada de *ACR-NEMA* Versão 3.0, uma vez que substitui *ACR-NEMA* Versão 2.0. O nome foi mudado para *DICOM* para reflectir a contribuição de outras organizações, bem como a capacidade da norma para se expandir além do apoio exclusivo à transmissão de imagens radiológicas.

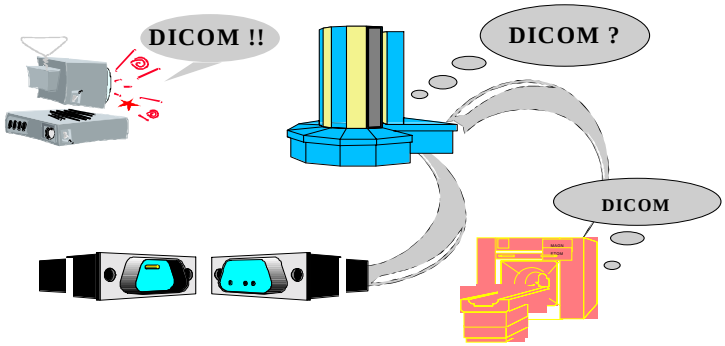


Figura C.1 - *DICOM*, interoperação e interligação em rede

Foi no encontro anual da *RSNA* de 1992, que as partes 1 e 8 da norma *ACR-NEMA DICOM* foram aprovadas, e os seus documentos finais postos em circulação. As restantes partes da 2 à 7 e a parte 9 foram colocadas à discussão nesse encontro.

Resumindo, a *ACR* e *NEMA* desenvolveram a nova versão da norma, tendo como objectivo:

- Estabelecer um padrão para o ambiente em rede (e.g., o sistema anterior apenas contemplava a ligação ponto-a-ponto);
- Definir os mínimos a que uma reivindicação de conformidade com a norma deve obedecer; e
- Permitir a interoperabilidade (não só a interligação) entre equipamentos de diferentes marcas e modelos, introduzindo objectos de informação explícitos (e.g., imagens, estudos, relatórios).

C.1 Objectivos da Norma *DICOM*

O principal objectivo da norma *DICOM* passa por facilitar a interoperabilidade de dispositivos que reivindicam conformidade com a norma. Em particular:

- Atribui um significado aos comandos e dados associados. Para os dispositivos interagirem deve haver normas que explicitam a forma ou o modo de reacção dos dispositivos a comandos e dados associados, e não só à informação que será movida entre dispositivos;
- Define os pressupostos de conformidade para a implementação da norma; i.e., uma reivindicação de conformidade tem que disponibilizar a informação necessária para determinar as funções para as quais pode ser esperada interoperabilidade com outro dispositivo que esteja em conformidade com a norma;
- Facilita as operações num ambiente em rede;
- É estruturada; i.e., permite a introdução de novos serviços (e.g., facilitando a introdução de novas aplicações de visualização de imagem médica); e
- Faz uso de padrões aceites a nível internacional onde quer que estes sejam aplicáveis.

Embora a norma *DICOM* tenha potencial para facilitar a implementação de soluções *PACS*, a norma, per si, não atravessa horizontalmente todas as funcionalidades subjacentes a um *PACS*; i.e., esta norma facilita a interoperabilidade de sistemas que reivindicam conformidade entre si, em ambiente multi-utilizador, mas não garante, por si só, a interoperabilidade.

C.2 Composição do *DICOM*

Como já foi referido anteriormente a norma *DICOM* é uma norma composta por diversas componentes, sendo as componentes já consolidadas as enumeradas a seguir:

Parte 1 – *Introdução* – dados históricos e breve introdução à norma, metas da norma, definições e referências;

Parte 2 – *Conformidade* – o que é estar em conformidade, o propósito de uma declaração de conformidade, qual o aspecto de uma declaração de conformidade;

Parte 3 – *Objecto de Informação* – definições normalizadas e compostas, modelo de informação *DICOM*, tabelas de módulos, esquemas de codificação;

Parte 4 – *Classes de Serviço* – definições e descrição das classes de serviço, descrição de negociação de associação, definições de classe *SOP*;

Parte 5 – *Estrutura de Dados e Semântica* – estrutura e codificação de dados, sua codificação e representação, codificação dos pixels da imagem, caracterização da sintaxe para a transferência de dados;

Parte 6 – *Dicionário de Dados* – registo de identificadores únicos, representações de valores, etiquetas de atributo, multiplicidade de valores;

Parte 7 – *Mensagens* – estrutura das mensagens, serviços *DIMSE*, especificações de serviços e sequências, especificação de protocolos, referência à noção de contexto, codificação de tipos de estado, negociação de associação, dicionário de comandos, *UIDs* de aplicações;

Parte 8 – *Protocolos de Rede* – ambiente de suporte a comunicações, serviço de camada superior *OSI*, perfil de camada superior *OSI*, protocolo de camada superior *TCP/IP*, endereçamento *DICOM*;

Parte 9 – *Protocolo de Ponto-a-Ponto* (*ACR-NEMA 2.0 50-pin*) – suporte para troca de mensagens;

Parte 10 – *Armazenamento de Dados e Formato de Ficheiro* – formatos de ficheiro para quando os dados pertencem a um arquivo;

Parte 11 – *Perfis da Aplicação* – define a aplicação específica, o meio e os objectos, um propósito semelhante ao definido para *SOP*;

Parte 12 – *Representação do Meio Físico* – de acordo com os parâmetros que representam o meio físico (e.g., *CD-R*); e

Parte 13 – *Suporte de Gestão da Impressão em Comunicações Ponto-a-Ponto* – usando o protocolo de controlo *DICOM* para enviar imagens sobre uma interface de vídeo ou uma interface digital de 8-bits, com comandos de controlo do utilizador sobre o *RS232* ou *RS422*.

C.3 Características do *DICOM*

Assim, após uma breve descrição do que é a norma *DICOM*, pode-se olhar às funcionalidades que lhe estão associadas (Figura C.2):

- **Protocolo de Rede**

  - O *DICOM* incorpora a figura do "*negociador*" fazendo com que haja acordo entre os nodos da rede sobre as funcionalidades a implementar;

- **Codificação da mensagem**
  - O *DICOM* define 24 tipos de dados;
  - A codificação das mensagens *DICOM* inclui compressão *JPEG* (17 variantes);
  - O *DICOM* apresenta uma sintaxe para imagens encapsuladas e para múltiplos enquadramentos; e
  - O *DICOM* suporta repertórios de caracteres.
- **Representação de dados através de objectos**
  - O *DICOM* está baseado num modelo de dados que alia a sustentabilidade à fiabilidade;
  - O *DICOM* inclui um mecanismo de *UID* robusto.
- **Dicionário de dados**
  - O *DICOM* inclui um grande número de elementos de dados.
- **Classes de serviço**
  - O *DICOM* define classes de serviço para aplicações específicas (e.g., a administração e impressão de imagens) e níveis de conformidade.
- **Suporte de consulta *off-line***
  - O *DICOM* define uma estrutura de directórios e perfis para o armazenamento de dados.
- **Conformidade**
  - O *DICOM* requer declarações de conformidade com a norma aos fabricantes de equipamento, e especifica de uma forma clara essas exigências.

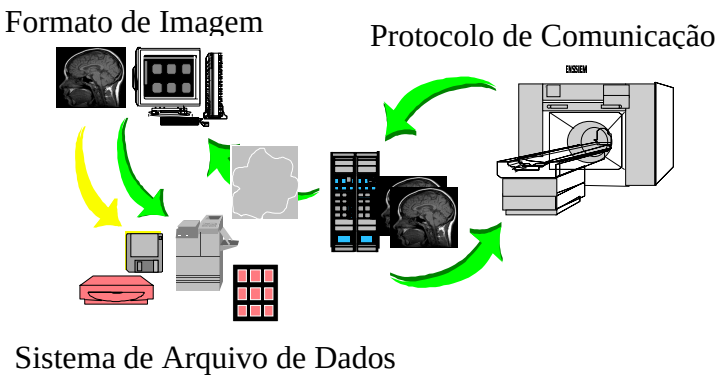


Figura C.2 – Norma *DICOM*

C.4 Declaração de Conformidade

A norma *DICOM* exige que o fabricante publique uma declaração de conformidade para cada dispositivo que reivindica a conformidade com o *DICOM*. O principal propósito das declarações de conformidade com o *DICOM* passa por ajudar os compradores de equipamento médico a determinar se os vários dispositivos que adquirem se interligarão de modo a satisfazer as suas expectativas. Esta determinação é crítica se o comprador planear adquirir equipamentos de mais do que um fabricante, agora ou no futuro.

Cada declaração de conformidade com o *DICOM* tem justamente que descrever como é que o dispositivo está em conformidade com a norma, inclusive as funcionalidades que são suportadas. Por exemplo, uma declaração de conformidade para uma impressora de películas deverá conter informação sobre:

- O tipo de rede na qual a impressora opera (e.g., *TCP/IP* sobre *Ethernet*);

- A classe de serviço para a qual está em conformidade (e.g., administração do processo de impressão, indicando que imprime imagens a preto e branco);
- O alcance de valores suportados (e.g., formatos de 1-*up* a 35-*up*, e resolução de 4096 x 5120 pixels); e
- Funções opcionais, como o suporte ou não à impressão de slides.

Para que dois dispositivos que estejam em conformidade com a norma *DICOM* possam comunicar, têm que se complementar, ou seja, um deve ser um *SCU*, e o outro um *SCP*, (i.e., o que é que um pode pedir e o outro pode fornecer). Ora esta situação pode-se desenvolver desde que se utilizem contextos de apresentação. Ora um contexto de apresentação não é mais do que uma tabela com informação que inclui a lista de características de um dado dispositivo médico, assim como os métodos utilizados para a codificação de dados. Cada declaração de conformidade deve incluir dois contextos de apresentação: os *propostos* e os *aceites*. O contexto de apresentação *propostos* lista as características do equipamento que um dispositivo tem a receber de outros dispositivos que estejam em conformidade com o *DICOM*. O contexto de apresentação *aceites* lista as características do equipamento que um dispositivo pode fornecer a outros dispositivos.

A comparação dos contextos de apresentação de dois dispositivos indicará se estes estão em condições de dialogar e operar conjuntamente. Ler uma declaração de conformidade não é uma tarefa fácil, sendo necessário uma compreensão detalhada da norma *DICOM*.

C.5 Classes de Serviços e Pares Serviço/Objecto

Na terminologia *DICOM* tem-se que:

- Um objecto de informação corresponde a um tipo específico de imagem (e.g., *TC*, *RM*, *RX*);
- Uma classe de serviço define o serviço ou a operação a realizar (e.g., imprimir, armazenar);
- Uma entidade aplicação denota um procedimento que pode operar sobre objectos; e
- Um par serviço/objecto (*SOP*) denota a associação de um objecto de informação e de um serviço (e.g., imprimir um *TC*, armazenar um *RX*).

A unidade básica de conformidade do *DICOM* é o *SOP* ou a classe *SOP*. Um dispositivo em conformidade com o *DICOM* tem que estar em conformidade com um ou mais *SOPs DICOM*, e não só com o *DICOM*. Para estar em conformidade com um *SOP*, o dispositivo tem que suportar uma imagem específica e a correspondente operação que a definição do *SOP* prescreve. (a norma *DICOM* define todos as *SOPs*).

Além disso, o dispositivo tem que estar em conformidade com a classe de serviço, uma classe de serviço de utilizador (*SCU*) ou uma classe de serviço de fornecedor (*SCP*). Basicamente um *SCU* usa os serviços que um *SCP* fornece. Alguns dispositivos podem estar em conformidade com ambas as categorias de classes de serviço. Exemplos de conformidade entre classes de serviço são apresentados a seguir:

- Um dispositivo de modalidade, por exemplo *RM*, estará em conformidade com um *SCU* se usar os serviços de dispositivos como estações de trabalho e impressoras laser para visualizar as imagens (e.g., num monitor ou numa cópia em ficheiro);
- Uma impressora é um *SCP* e fornece serviços de impressão para as modalidades, as estações de trabalho, e outros dispositivos; e

- Uma estação de trabalho pode ser um *SCP* e um *SCU*, tornando possível a visualização de imagens para as modalidades e outros dispositivos, e usando os serviços de certos dispositivos como as impressoras.

Algumas das classes de serviços disponibilizados pela norma *DICOM* são a seguir mencionados:

- Verificação;
- Armazenamento;
- Perguntas / respostas;
- Notificação de contexto;
- Gestão da impressão;
- Gestão de pacientes;
- Gestão de estudos; e
- Gestão de resultados.

C.6 Protocolo de Comunicação *DICOM*

A norma *DICOM* usa uma arquitectura de cliente/servidor, podendo uma dada máquina ser um cliente (*SCU*), um servidor (*SCP*), ou ambos.

Em DICOM toda a informação é agrupada em *objectos* que são chamados *Módulos de Entidade de Informação* (*IE*) e *Definições de Objecto de Informação* (*IOD*). A informação é transferida via *OSI* ou *TCP/IP*. Os objectos *IOD* são constituídos por elementos denominados Objectos de Serviços de Mensagem *DIMSE*. Os objectos *ASSOCIATE-PDU* são responsáveis pela inicialização e desactivação da associação. A transmissão de dados por um meio amovível (e.g., fita magnética, disquete, *CD-R*), e a ligação ponto-a-ponto, que já está obsoleta, são as excepções da transferência de rede, apesar de ambos usarem o mesmo *IOD* como protocolo de transferência de rede.

A Figura C.3 apresenta a sequência de eventos necessários para que se envie uma imagem ou um grupo de imagens, os quais obedecem ao procedimento:

- o *SCU* pede uma ***associação*;**
- envia dados (i.e., uma ou mais imagens neste caso); e
- termina a ***associação*.**

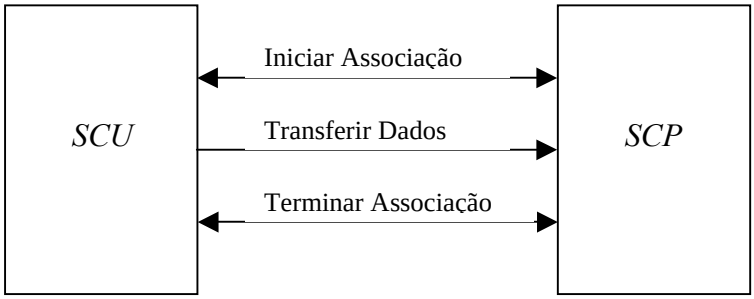


Figura C.3 – Situação Cliente/Servidor

C.7 *DICOM*, informação e seu armazenamento

A norma *DICOM* foi estabelecida com o objectivo de fornecer um formato para a permuta de dados em sistemas de informação médica na área da imagiologia. Uma apresentação habitual de *DICOM* é o de uma imagem de ultra-som com um registo da visita do paciente à unidade prestadora de cuidados de saúde, alguns dados sobre o estado clínico do paciente na altura da visita e, informação detalhada sobre como a imagem foi feita.

O *DICOM* está organizado de forma hierárquica, estruturado em termos de módulos obrigatórios e módulos opcionais. O módulo do paciente não faz parte desta hierarquia, é um módulo opcional. Outros módulos contêm informação relativa ao

equipamento, ao quadro de referência de patologias, e assim por diante. A norma *DICOM* traduz-se facilmente numa implementação em base de dados. O sistema normalmente usado é o de uma base de dados relacional que use a linguagem de comandos *Structured Query Language* (*SQL*) como veículo de interpelação.

C.8 Expectativas de Desempenho

Por si só, a norma *DICOM* não fornece um mecanismo de transferência de informação muito eficiente (i.e., baseia-se nos dispositivos fonte para controlar a transferência de informação, o que nem sempre é mais eficiente do que se alicerçar nos dispositivos de destino que executam essa mesma tarefa). Para superar este possível afunilamento, muitos fornecedores de soluções na área médica estão a desenvolver aplicações que recorrem a um protocolo de transferência de dados de alto débito e, delegam nos dispositivos de destino o controlo desse afunilamento. Seguindo este cenário, tem-se a seguinte sequência de operações:

- A aplicação fonte comunica à aplicação de destino que tem dados para enviar;
- O dispositivo de destino reconhece este pedido;
- A fonte envia os dados;
- O destino comunica à fonte quando recebeu esses dados; e
- Então, e só então, é que a fonte se liberta dos dados que tem armazenados.

Este procedimento assegura a integridade da transferência de informação entre fonte e destino. Actualmente muitos fornecedores de soluções disponibilizam a sua própria versão de alto débito para a transferência de dados e oferecem conformidade *DICOM* via um portal.

C.9 Ponto de Situação da Norma *DICOM*

Nos dias de hoje ainda é comum encontrar na comunidade de utilizadores *DICOM* um pouco de cepticismo relativamente à mesma.

A interoperabilidade entre os diferentes sistemas permanece a preocupação principal: *Será que os equipamentos DICOM comprados a fabricantes diferentes irão interoperar?* Uma outra preocupação tem a ver com o desempenho: *Será que o uso de equipamento de fabricantes diferentes, que estão em conformidade com o DICOM, fornece um nível adequado em termos de desempenho?* Uma terceira preocupação está ligada a problemas de conectividade a sistemas *HIS/RIS* (*Hospital Information System/Radiology Information System*): *Será que o DICOM fornece um mecanismo viável para acesso a sistemas de informação*.

A norma *DICOM* liberta os utilizadores de equipamentos na área da imagiologia de sistemas proprietários. Oferece aos clientes a oportunidade de seleccionarem os produtos mais adequados para servir cada aplicação.

Em princípio, qualquer utilizador da norma *DICOM* tem como objectivo:

- Garantir que não há diminuição das funcionalidades já disponíveis dos seus equipamentos (i.e., há que obter não só interconexão, mas também a interoperação entre equipamentos);
- Garantir a interconexão segura com outros dispositivos de imagem médica. Os utilizadores não querem estar sujeitos à ditadura de um só fornecedor;
- Garantir a integração de produtos *DICOM* de diferentes fornecedores sem perda de produtividade;
- Dispor de interfaces amigáveis para sistemas de *HIS/RIS*; e

- Ter acesso a sistemas de armazenamento de dados fiáveis e de acesso rápido.

Índice Remissivo