Instituto Superior de Tecnologia do Rio de Janeiro
Série Livros Didáticos Digitais
Informática para Todos

Uma Introdução à Inteligência Computacional: Fundamentos, Ferramentas e Aplicações

Uma Introdução à Inteligência Computacional: Fundamentos, Ferramentas e Aplicações

Ronaldo Ribeiro Goldschmidt

1ª Edição

Governo do Estado do Rio de Janeiro
Fundação de Apoio à Escola Técnica
Instituto Superior de Tecnologia do Rio de Janeiro

Série Livros Didáticos Digitais Gratuitos

Uma Introdução à Inteligência Computacional: fundamentos, ferramentas e aplicações

Ronaldo Ribeiro Goldschmidt

Rio de Janeiro
2010

Projeto gráfico e editoração eletrônica: Alexander Daltio Vialli
Capa: Alexander Daltio Vialli
Revisão Textual:

Normalização Técnica: Edirlane Carvalho de Souza

G623i Goldschmidt, Ronaldo Ribeiro. Inteligência Computacional / Ronaldo Ribeiro Goldschmidt. Rio de Janeiro: IST-Rio, 2010.
143p. ; 29cm.
ISBN 978-85-98931-08-1
1. Inteligência Artificial. 2. Sistemas Inteligentes. 3. Sistemas de Apoio à Decisão. 4. Sistemas de Informação. 5. Redes Neurais. 6. Lógica Nebulosa. 7. Algoritmos Genéticos. 8. Sistemas Especialistas.
I. Título.

CDD 621.399

SUMÁRIO:

PREFÁCIO 1

Conheço o Professor Ronaldo Goldschmidt há muitos anos, desde o tempo de seu mestrado no *IME- Instituto Militar de Engenharia,* na área de Inteligência Artificial, como orientador da monografia e Professor de disciplinas. Nesta época já surpreendia pela sua tenacidade em conseguir seu objetivo final. O Professor Ronaldo, formado em matemática, e com muita vontade em aprender, logo nas primeiras aulas mostrava facilidade em escrever bem. Escrevia claro e conciso sobre os temas que lhe eram solicitados. Já era uma promessa de excelente professor e escritor. Sua monografia fluía com facilidade, sem erros, mesmo com um tema inédito no Brasil: Reconhecimento de Palavras Isoladas por Computador. Em seguida foi para a PUC-Rio para o início de seu doutorado, na área de Inteligência Computacional. A tese fluiu com rapidez derivando daí, pela sua facilidade de escrever, um livro publicado pela Editora Campus chamado "Mineração de Dados - Guia Prático'.O conteúdo desse livro reflete parte das aulas de seu doutorado.

Agora, o Professor Ronaldo, nos mostra um outro livro, este eletrônico: "Inteligência Computacional", composto de sete capítulos com ampla teoria e prática. Um com conceitos básicos em Sistemas Especialistas, cuja prática ele adquiriu em empresas. Outro sobre Lógica Nebulosa. Um capítulo bem claro sobre Redes Neurais e outro sobre Algoritmos Genéticos. O Professor Ronaldo conseguiu escrevê-los de forma didática útil para alunos de graduação e de pos graduação que queiram um primeiro contato com essa área.

Trata-se de um livro útil que preencherá lacuna no Brasil nesta área de Inteligência Computacional.

Espero que outros livros surjam escritos pelo autor, como, por exemplo, sobre Mineração de Textos.

Que venham mais livros bem escritos e claros como este!

Não poderia deixar de finalizar este prefácio sem declarar a minha admiração pela energia positiva que o PROFESSOR IRRADIA e pela sua dignidade de grande ser humano que é.

Emmanuel Passos, D.Sc.
Professor da PUC-Rio

PREFÁCIO 2

Adam é um garoto de treze anos que passou boa parte de sua infância rodeado de computadores e aparelhos eletrônicos. Naquele dia ele estava sentado no Grand Auditorium cercado dos maiores cientistas e engenheiros da computação e pronto para presenciar, com seus próprios olhos, o que seria um das maiores avanços dos últimos anos: o computador "Ultronic". Proclamava o engenheiro chefe "...um computador com mais de 10^{17} unidades lógicas. Isto é mais do que o número de neurônios existentes em todos os cérebros de nosso país! Sua inteligência será inimaginável! Felizmente não precisaremos imaginá-la. Em alguns momentos teremos o privilégio de constatar sua inteligência". O computador foi ligado e imediatamente o engenheiro chefe desafiou: "Alguém gostaria de inaugurar o Ultronic Computer Systems fazendo uma primeira pergunta?". Todos se sentiam intimidados... Ficaram em silêncio. Adam, entretanto, estava irrequieto. Levantou a mão. O engenheiro chefe lhe deu a palavra e ele perguntou: "Como ele se sente sendo um computador?".

Esta é a história de abertura do livro The Emperor New Mind, de Roger Penrose, e ilustra com sabedoria os conflitos e desafios da área de inteligência artificial (IA). A área de Inteligência Artificial pode ser definida como uma área de estudo preocupada em construir ou programar computadores de forma a capacitá-los a fazer um conjunto de ações que requerem inteligência para realizá-las. Esta definição assume que o computador é realmente capaz de entender, inferir e aconselhar! Outra definição mais suave, talvez considerasse a área de Inteligência Artificial como aquela capaz de construir programas e computadores cujos comportamentos pudessem ser comparados ao processo mental de um ser humano. Por fim poder-se-ia considerar também como a ciência que estuda a inteligência em geral. Estas definições ilustram um problema filosófico essencial da área: a capacidade de imitar, simular ou recriar o pensamento humano em uma máquina. Seria este um empreendimento meramente ambicioso ou é radicalmente infundado? Talvez, por tudo isso, denominada recentemente e de forma mais abrangente de Inteligência Computacional.

Entretanto, qualquer que seja a resposta, o livro Inteligência Computacional do Prof. Ronaldo Ribeiro Goldschmidt nos convida a mergulhar na área, entender suas características, sua linguagem sua técnica e sua tecnologia. Sistemas Especialistas, Lógica Nebulosa, Redes Neurais Artificiais e Algoritmos Genéticos são os temas tratados neste livro, permitindo ao leitor um entendimento amplo da área, assim como de suas limitações. O estudante interessado na área encontrará neste excelente livro uma bela introdução ao assunto.

Continuando a história acima... O engenheiro chefe encaminhou a pergunta de Adam para o computador Ultronic. O computador respondeu... Respondeu que não a entendia e que não possuía a mínima ideia de seu sentido.

De qualquer forma, de nossa parte, vamos continuar tentando! Boa leitura!

Márcio Francisco Campos, Msc.
Diretor do IST-Rio

Capítulo

1

Introdução

1.1. Visão Geral

A sobrevivência da espécie humana deve-se principalmente pelas capacidades mentais que o ser humano vem desenvolvendo ao longo de sua existência. Em função disso, o entendimento sobre "como o Homem pensa" tem sido objeto de estudo durante milhares de anos, ou seja, como uma pequena porção de matéria pode perceber, compreender, prever e manipular um mundo muito maior e mais complexo que ela própria (Russell e Norvig, 2004).

A Inteligência Computacional (IC), denominada originalmente de Inteligência Artificial (IA), é uma das ciências mais recentes, tendo surgido logo após a Segunda Guerra Mundial e tendo seu nome original cunhado em 1956. Por razões de simplificação, ao longo deste livro, consideraremos indistintas as denominações "Inteligência Computacional" e "Inteligência Artificial".

A Inteligência Computacional vai além da perspectiva de compreensão do pensamento humano, pois também procura construir entidades artificiais inteligentes.

Para conhecer o que vem, então, a ser a área da Inteligência Computacional faz-se necessária uma reflexão sobre o conceito de inteligência. Na realidade, há diversas definições e inúmeras discussões filosóficas sobre o que vem a ser este conceito que fogem do escopo deste livro. No entanto, algumas habilidades que necessariamente envolvem inteligência podem ser citadas:

- Capacidade de raciocínio / dedução / inferência
- Capacidade de aprendizado
- Capacidade de percepção
- Capacidade de evolução e adaptação

Por outro lado, podemos facilmente listar alguns exemplos de tarefas que requerem inteligência para serem realizadas:

- Jogar xadrez
- Entender a linguagem humana
- Decidir diante de incertezas

- Resolver problemas complexos (de difícil formulação matemática envolvendo muitas variáveis)
- Reconhecer objetos pela imagem

As definições de Inteligência Artificial variam ao longo de duas dimensões principais (Russell e Norvig, 2004):

- A que se baseiam em pensamento e raciocínio onde o objetivo é desenvolver sistemas que pensam como seres humanos ou que pensam racionalmente.
- A que se baseia em comportamento onde o objetivo é desenvolver sistemas que atuam como seres humanos ou que atuam racionalmente.

A seguir estão algumas das definições de Inteligência Computacional que podem ser encontradas na literatura especializada. As quatro primeiras definições se enquadram melhor na primeira dimensão. As demais na segunda.

- "O novo e interessante esforço para fazer os computadores pensarem... máquinas com mentes, no sentido total e literal." (Haugeland, 1985)
- "[Automatização de] atividades que associamos ao pensamento humano, atividades como a tomada de decisões, a resolução de problemas, o aprendizado..." (Bellman, 1956)
- "O estudo das faculdades mentais pelo uso de modelos computacionais" (Charniak e McDermott, 1985)
- "O estudo das computações que tornam possível perceber, raciocinar e agir." (Winston, 1992)
- "O estudo sobre como fazer computadores realizarem coisas nas quais, no momento, as pessoas sejam melhores." (Rich e Knight, 1992)
- "A Inteligência Artificial pode ser definida como o ramo da Ciência da Computação que se ocupa da automação do comportamento inteligente". (Luger, 2004)

No contexto deste livro, adotamos a seguinte definição para Inteligência Computacional: "*Ciência multidisciplar que busca desenvolver e aplicar técnicas computacionais que simulem o comportamento humano em atividades específicas.*"

Como mencionado na definição acima, a Inteligência Computacional envolve idéias, pontos de vista, conceitos e técnicas de diversas áreas, dentre as quais podemos citar:

- Filosofia
- Matemática
- Economia
- Neurociência
- Psicologia
- Ciência da Computação

- Linguística

A figura 1.1 apresenta uma taxonomia com os principais paradigmas da Inteligência Computacional. Pode-se perceber que a IC envolve estudos e aplicações de técnicas inspiradas na natureza. Abaixo é apresentada uma visão geral e introdutória sobre cada elemento da taxonomia.

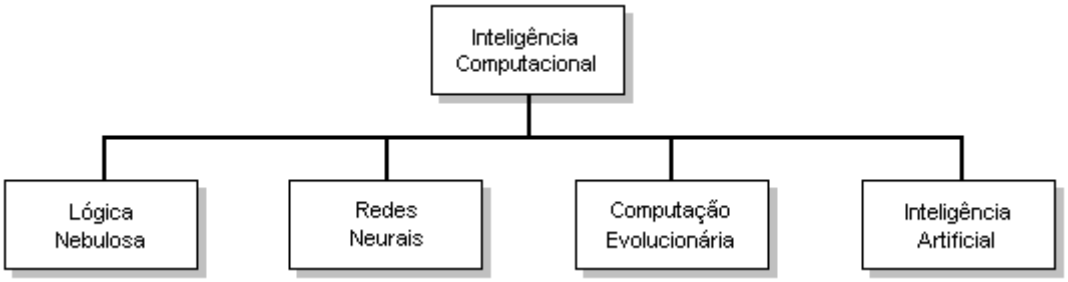


Figura 1.1: Inteligência Computacional – uma taxonomia

- Lógica Nebulosa – Do inglês Fuzzy Logic, este paradigma tem por objetivo modelar o modo aproximado de raciocínio humano, visando criar métodos computacionais capazes de tomar decisões racionais em ambientes de incerteza, subjetividade e imprecisão. A Lógica Nebulosa fornece mecanismos para manipular informações imprecisas e subjetivas, tais como os conceitos: muito, pouco, pequeno, alto, bom, quente, frio, etc, fornecendo uma resposta aproximada para questões baseada em conhecimentos inexatos, incompletos ou não totalmente confiáveis.
- Redes Neurais – São modelos computacionais não lineares, inspirados na estrutura e no funcionamento do cérebro, que procuram reproduzir características humanas, tais como: aprendizado, associação, generalização e abstração. Devido à sua estrutura, as Redes Neurais são bastante efetivas no aprendizado de padrões a partir de dados históricos não lineares, incompletos, com ruído e até compostos de exemplos contraditórios.
- Computação Evolucionária – É uma área de pesquisa interdisciplinar que compreende diversos paradigmas inspirados no princípio da evolução natural das espécies proposto por Charles Darwin e na recombinação genética. Desdobra-se nas seguintes especialidades:
    - Algoritmos Genéticos – Fornecem um mecanismo de busca adaptativa que se baseia no princípio da sobrevivência dos mais aptos. Isto é obtido a partir de uma população de indivíduos (soluções), representados por cromossomas (palavras binárias), cada um associado a uma aptidão (avaliação da solução frente ao problema), que são submetidos a um processo de evolução (seleção e reprodução) por vários ciclos.

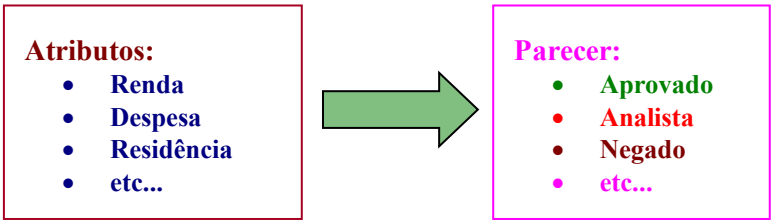
- Programação Genética – É uma técnica automática de programação que propicia a evolução de programas de computadores que resolvem problemas de maneira exata ou aproximada.
- Hardware Evolucionário – É uma extensão do modelo genético de aprendizado no espaço de estruturas complexas como circuitos eletrônicos. Utiliza conceitos dos sistemas evolucionários naturais no projeto automático de circuitos, hardware auto-reparável, projeto de robôs e projeto de circuitos VLSI. Esta área vem ganhando cada vez mais aplicabilidade em função dos recentes avanços em nanotecnologia.

- Inteligência Artificial – Nome original da área de estudo do presente texto, restringe-se ao processamento simbólico do conhecimento, criando programas que fazem os computadores parecerem inteligentes. As soluções dos problemas são heurísticas e respostas satisfatórias são aceitas. A IA é constituída de técnicas próprias para a solução de problemas, com destaque para os chamados sistemas especialistas, que são programas computacionais destinados a solucionar problemas em campos especializados do conhecimento humano. Usam técnicas de IA, bases de conhecimento e raciocínio inferencial.

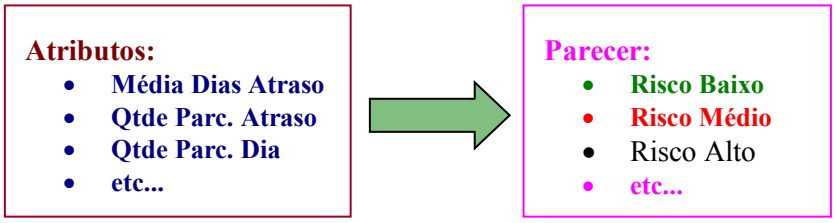
Convém mencionar que muitos esforços na área de IC buscam integrar várias das técnicas dos paradigmas mencionados acima procurando construir os chamados Sistemas Híbridos. Espera-se que, por meio da combinação de técnicas, deficiências individuais destas técnicas possam ser supridas, obtendo modelos mais robustos e completos.

É importante mencionar que a taxonomia acima pode ser interpretada segundo dois paradigmas: clássico e distribuído. No paradigma clássico, modelos inteligentes são projetados e aplicados individual e isoladamente. Por outro lado, no paradigma distribuído, busca-se explorar as potencialidades da inteligência grupal, voltada para a interação social e o comportamento cognitivo dela emergente. Alguns autores utilizam as denominações IA Clássica e IA Distribuída para distinguir entre os dois paradigmas.

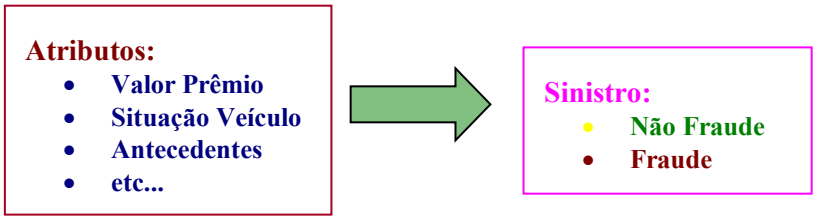
Diante do exposto anteriormente, pode-se dizer que a Inteligência Computacional possui entre seus propósitos o suprimento de meios para a construção dos chamados métodos de apoio à decisão. Um método de apoio à decisão é considerado qualquer instrumento que auxilie o Homem no processo de tomada de decisão em alguma área do conhecimento humano. Diversos exemplos de métodos de apoio à decisão serão apresentados ao longo do texto, mas a título ilustrativo neste primeiro momento, a figura 1.2 relaciona exemplos de métodos de apoio à decisão, identificando algumas de suas entradas e saídas:



(a) - Área Financeira – Análise e Concessão (CRED Análise)



(b) - Área Financeira – Cobrança de Clientes em Atraso (CRED Cobrança)



(c) - Área de Seguros – Detecção de Fraudes (SAF – Sistema de Análise de Fraudes)

(d) Sistema Especialista na área Médica – Clínico Geral (SEAM)

Figura 1.2 - Exemplos de métodos de apoio à decisão (com algums entradas e saídas)

Embora existam algumas divergências procedentes na literatura, para fins de simplificação, serão tratados como sinônimos e de forma indistinta ao longo deste livro as expressões indicadas a seguir. Cada uma delas segue acompanhada de uma definição a título meramente ilustrativo.

- Método de Apoio à Decisão (MAD) - qualquer instrumento que auxilie o Homem no processo de tomada de decisão em alguma área do conhecimento humano.
- Sistema de Apoio à Decisão (SAD) - Ferramentas computacionais que auxiliam na conjugação de diversas informações de forma direcionada à tomada de decisão em alguma área do conhecimento
- Sistema Inteligente (SI) - SADs que incorporam conhecimento sobre alguma área e que procuram simular o comportamento humano na tomada de decisão, sugerindo alternativas de ação.

- Sistema Baseado em Conhecimento (SBC) - São programas de computador que usam o conhecimento representado explicitamente para resolver problemas (Rezende, 2003).

1.2. Um Breve Histórico

No atualmente reconhecido como primeiro trabalho pertencente à área da Inteligência Artificial, Warren McCulloch e Walter Pitts propuseram em 1943 um modelo de neurônios artificiais de dois estados (ligado ou desligado) no qual a troca de estados em um neurônio ocorre em função dos estados de neurônios suficientemente próximos. McCulloch e Pitts também sugeriram que redes de neurônios artificiais adeqadamente compostas seriam capazes de aprender.

Em 1949, Donald Hebb demonstrou a chamada regra de aprendizagem de hebb, segundo a qual ocorrem modificações nas intensidades das conexões entre neurônios artificiais em uma mesma rede.

Nos anos seguintes surgiram diversos trabalhos que hoje podem ser caracterizados como sendo de IA, mas foi Alan Turing quem primeiro apresentou uma visão mais concreta e completa da IA em seu artigo de 1950 intitulado "Computing Machinery and Intelligency". Neste trabalho, Turing apresentou o chamado Teste de Turing cuja idéia consiste basicamente em:

- *Introduzir um juiz humano que deverá conversar via terminal de computador com outro ser humano e com uma máquina, sem saber quem é quem. A máquina será considerada inteligente se após toda a conversa, o juiz não puder identificar com certeza quem é máquina e quem é o humano.*

Em 1951, Marvin Minsky e Dean Edmonds construíram o SNARC, primeiro computador de rede neural artificial. Mais tarde, Minsky acabou provando teoremas importantes que mostravam limitações dos modelos de redes neurais existentes até então e que foram os responsáveis pelo esmorecimento das pesquisas na área.

Em 1955 ocorreu o seminário de Dartmouth, onde pesquisadores como John McCarthy, Allen Newell e Herbert Simon apresentaram idéias e aplicações de IA na área de jogos. A partir deste seminário foi cunhado o nome Inteligência Artificial.

Os primeiros anos da IA foram repletos de sucessos, muitos ainda limitados por restrições computacionais de processamento e de memória, características da época.

Newell e Simon propuseram o General Problem Solver (GPS), ou solucionador de problemas gerais, programa projetado para reproduzir regras humanas na solução de problemas. Embora tenha tido sucesso com uma classe restrita de problemas, o GPS veio mais tarde a enfrentar suas limitações diante de novos problemas.

Pesquisas em Prova Automática de Teoremas e em Compreensão de Linguagem Natural surgiram em instituições de renome como IBM, MIT e Stanford. Entre elas, surgiu a primeira linguagem de programação da IA: LISP.

Durante os anos subseqüentes, muitos pesquisadores da IA eram ousados nos prognósticos otimistas sobre os avanços na área. Um exemplo disso pode ser ilustrado pela declaração de 1957 de Herbert Simon:

- *Não é meu objetivo surpreendê-los ou chocá-los – mas o modo simples de resumir tudo isso é dizer que agora existem no mundo máquinas que pensam, aprendem e criam. Além disso, sua capacidade de realizar essas atividades está crescendo rapidamente até o ponto – em um futuro visível – no qual a variedade de problemas com que elas poderão lidar será correspondente à variedade de problemas com os quais lida a mente humana.*

O entusiamo exacerbado dos pesquisadores em IA e ilusão de poder computacional ilimitado da IA foram, aos poucos, sendo dominados pela razão e sensatez, essenciais ao pensamento científico.

Umas dificuldades enfrentadas foi o fato de que a maioria dos programas desenvolvidos à época continha pouco ou nenhum conhecimento sobre o domínio da aplicação. Eram dependentes ou limitados a situações específicas.

Em 1966, a publicação de um relatório elaborado por um comitê consultivo evidenciando a inexistência de um sistema genérico de tradução automática para textos científicos levou o governo americano cancelar todas as subvenções a projetos de tradução automática (Russell e Norvig, 2004). Atualmente, sabe-se que a traduação automática ainda é imperfeita, mas amplamente utilizada em diversos tipos de documentos.

Outro problema encontrado foi a dificuldade de operacionalizar programas em aplicações reais onde o volume de dados, variáveis, parâmetros e fatos a serem considerados ou experimentados era muito grande. Espaços de busca extensos, muitas vezes infinitos, representavam uma séria limitação aos programas de IA. Antes do desenvolvimento das teorias da complexidade e computabilidade computacionais, acreditava-se que para lidar com o aumento da escala do volume a ser processado era apenas uma questão de disponibilidade de memória e velocidade de processamento.

A incapacidade de lidar com a explosão combinatória de grande parte dos problemas reais foi uma dos principais críticas relacionadas à IA dos anos 60. Novas subvenções foram canceladas.

Uma terceira dificuldade enfrentada pela IA foi a limitação de representação das estruturas que vinham sendo utilizadas para reproduzir comportamento inteligente. Minsky e Papert demonstraram no livro Perceptrons (1969) as limitações das redes neurais da época em representar problemas não lineares. Como a maioria dos problemas reais relevantes é não linear, tal publicação teve forte impacto na comunidade científica da época, reduzindo drasticamente os esforços de pesquisa na área da IA.

O sistema DENDRAL, divulgado por Bruce Buchanan e sua equipe em 1969, foi um dos primeiros exemplos de métodos de apoio à decisão que procuravam incorporar conhecimento específico do domínio da aplicação. Este sistema tinha como objetivo resolver o problema de inferir a estrutura molecular a partir das informaçõs fornecidas por um espectrômetro de massa.

O MYCIN também foi considerado um marco no desenvolvimento dos chamados Sistemas Especialistas, pois, assim como o DENDRAL, também incorporava conhecimento específico do contexto do problema. No caso do MYCIN, o propósito era diagnosticas infecções sangüíneas. Com cerca de 450 regras, o MYCIN apresentava desempenho comparável ao de especialistas sobre o assunto. Para o seu desenvolvimento, foram necessárias diversas e extensas entrevistas com especialistas na área. Consultas a livros didáticos e estudos de casos anteriores também foram necessários.

Com o surgimento do DENDRAL e do MYCIN, houve uma retomada da IA com forte crescimento da demanda por aplicações para resolução de problemas reais. Foram, então, desenvolvidas linguagens e ambientes de programação voltadas à representação do conhecimento e raciocínio. Dentre elas, surgiu a linguagem PROLOG (Programação em Lógica), muito utilizada na época para a construção de Sistemas Especialistas e aplicações de Processamento de Linguagem Natural. Em 1975, Minsky propôs a utilização de quadros e roteiros (frames e scripts) como forma de representação do conhecimento sobre tipos específicos de objetos e eventos, organizados em estruturas tononômicas.

O primeiro sistema especialista comercial bem sucedido, o RI começou a ser utilizado pela Digital Equipament Corporation em 1982 no apoio à configuração de novos sistemas de computadores. Em 1986, o RI proporcionava à Digital uma economia anual de cerca de 40 milhões de dólares (Russell e Norvig, 2004).

A indústria da IA viveu na década de 80 seus anos de glória, chegando a movimentar em 1988 alguns bilhões de dólares da economia mundial. No entanto, em função do não cumprimento de promessas ambiciosas, a área da IA viveu nos anos subseqüentes um encolhimento em suas pesquisas e atividades.

No final da década de 80 houve uma retomada das pesquisas em Redes Neurais com a criação de sistemas conexionistas e o algoritmo de retropropagação de erro capazes de lidar de forma bem sucedida com problemas não lineares. Rumelhart e McClelland, dois dos responsáveis por esta retomada, publicaram em 1986 diversos trabalhos afins em uma coletânea denominada Parallel Distributed Processing. Nesta época, alguns pesquisadores começaram a utilizar a expressão Inteligência Computacional como uma extensão à Inteligência Artificial.

Desde então a IA, ou IC, vem se consolidando como ciência, tanto no conteúdo quanto na metodologia. Atualmente é muito comum as pesquisas em IA utilizarem teorias existentes como base, ao invés de propor teorias inteiramente novas. Adicionalmente, o progresso recente na compreensão das bases teóricas da Inteligência Artificial caminha lado a lado com o avanço na capacidade dos sistemas reais. As subáreas da IA se tornaram mais integradas, assim como a própria IA vem se integrando a outras áreas e disciplinas do conhecimento humano na formulação de diversas aplicações. Um exemplo disso é a aplicação de recursos de IA na indústria da chamada mineração de dados. Na mineração de dados, o objetivo é identificar conhecimento útil a partir de grandes bases de dados. Para tanto, técnicas de IA são conjugadas a diversas tecnologias de outras áreas tais como estatística, banco de dados, reconhecimento de padrões, interface humano-máquina.

1.3. Organização do Texto

O presente texto está organizado em mais seis capítulos que procuram apresentar de forma introdutória os principais conceitos, recursos e técnicas da IC em seus diversos paradigmas. A inspiração na natureza em que se baseia cada paradigma da IC também é oportunamente comentada em cada capítulo.

O capítulo 2 apresenta alguns dos principais conceitos básicos da área preparando o leitor para uma melhor compreensão dos capítulos subseqüentes.

Um detalhamento dos chamados sistemas especialistas pode ser obtido no capítulo 3. Técnicas de aquisição e representação simbólica de conhecimento também são descritas neste capítulo.

A Lógica Nebulosa que introduz flexibilidade aos sistemas especialistas está descrita no capítulo 4. Orientações sobre como construir os chamados sistemas de inferência nebulosa também são fornecidas.

As Redes Neurais e os Algoritmos Genéticos são apresentados nos capítulos 5 e 6, respectivamente. Exemplos sobre aplicações destas tecnologias também foram incorporados ao longo do texto.

As considerações finais no capítulo 7 compreendem alguns exemplos de desafios e problemas em aberto tanto na própria IC quanto em áreas em que a IC pode ser utilizada como ferramenta de aplicação.

Em todos os capítulos, o texto traz uma reflexão sobre as vantagens e desvantagens de cada tecnologia apresentada. Espera-se que, desta forma, o leitor desenvolva um senso crítico voltado à identificação de possibilidades de aplicação de recursos da IA diante dos mais diversificados tipos de problemas.

Um apêndice ilustrando o funcionamento de algumas ferramentas de IC pode ser consultado como uma espécie de manual para usuários iniciantes. Além das dicas de utilização, o apêndica indica sites de onde as ferramentas podem ser gratuitamente obtidas.

Capítulo

2

Conceitos Básicos

2.1. Considerações Iniciais

Conforme já comentado, este capítulo tem como objetivo apresentar alguns conceitos básicos úteis e necessários ao entendimento do restante do texto. Assim sendo, ele foi estruturado em diversas seções, cada uma delas tratando de um conceito específico. Procurou-se apresentar os conceitos em uma sequência dedutiva e incremental, de forma que cada conceito apresentado utilize somente conceitos apresentados anteriormente.

2.2. Estrutura Genérica de um Sistema Baseado em Conhecimento

Os chamados sistemas baseados em conhecimento (SBCs) apresentam uma estrutura geral baseada em módulos, como mostra a figura 2.1 retirada de (Rezende, 2003):

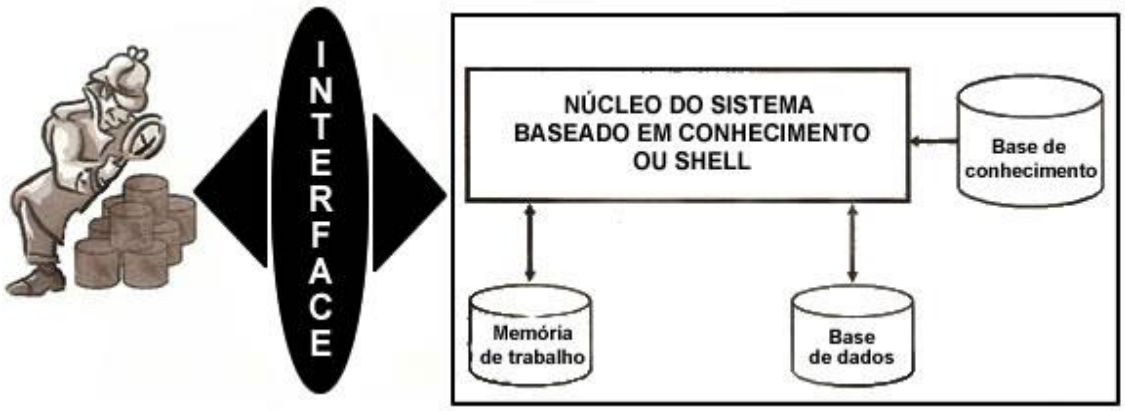


Figura 2.1 Estrutura de um Sistema Baseado em Conhecimento

- Núcleo do Sistema Baseado em Conhecimento ou Shell: Parte do SBC que desempenha as principais funções do sistema, entre elas a interação com o usuário e a execução dos mecanismos de inferência.
- Base de Conhecimento: Parte responsável por armazenar o conhecimento do SBC sobre o domínio da aplicação. As informações contidas neste módulo do sistema devem estar em um formato reconhecido pelo Núcleo do SBC.

- Memória de Trabalho: Local onde são armazenadas as respostas do usuário e as conclusões intermediárias em um processo de raciocínio (Rezende, 2003).
- Base de Dados: Alguns SBCs podem se comunicar com um repositório de onde os dados possam ser obtidos e onde possam ser guardados de forma persistente.
- Interface com o usuário: Meio de contato direto do sistema com o usuário, é realiza interações com o utilizador por meio de perguntas e da apresentação de resultados.

2.2.1. Núcleo do Sistema Baseado em Conhecimento

O Núcleo é parte fundamental em SBC. É nele que são realizadas as principais atividades do sistema. O núcleo é dividido em três sub-módulos interdependentes. Estes módulos podem funcionar em conjunto ou separados, dependendo apenas da implementação do sistema.

- Módulo Coletor de Dados: É responsável pela interação com o usuário, obtendo informações do problema em questão, através da formulação de sucessivas perguntas ao usuário. Quando ativado pelo motor de Inferência, o Coletor de Dados faz as perguntas necessárias e valida as respostas do usuário baseando-se em críticas preestabelecidas. Tais funções verificam a validade das respostas (Rezende, 2003).
- Motor de Inferência: Módulo responsável pelo desenvolvimento do raciocínio a partir das informações obtidas pelo Coletor de Dados e do conhecimento representado e disponibilizado na Base de Conhecimento (Rezende, 2003).
- Módulo de Explicações: Responsável pela explicação, ou justificativa, das conclusões obtidas e dos motivos pelos quais o SBC fez determinadas perguntas e chegou a determinados resultados (Rezende, 2003).

Durante a execução de suas atividades o Núcleo do SBC interage com a Memória de Trabalho, que manipula os dados temporários utilizados no processo. Além de acessar a Base de Conhecimento a fim de obter o conhecimento necessário ao processamento das informações coletadas.

Um *shell* pode ser visto como um ambiente para facilitar a construção e a execução de um SBC. Por trazer embutidos o motor de inferência e a interface com o usuário, o custo de produção do SBC se torna relativamente baixo.

Os shells são destinados a permitir que pessoas sem conhecimento em programação possam criar SBCs se aproveitando do esforço já realizado por outros que tiveram problemas semelhantes. Um exemplo de Shell é o ambiente SINTA desenvolvido pela Universidade Federal do Ceará. O apêndice apresenta algumas dicas para a sua utilização.

2.2.2. Base de Conhecimento

A Base de Conhecimento é a parte do SBC responsável por armazenar o conhecimento necessário para resolução do problema abordado pela aplicação. Ela pode conter asserções sobre o domínio de conhecimento, regras que descrevem relações nesse domínio, heurísticas e métodos de resolução de problemas.

Uma Base de Conhecimento é um conjunto de representações de ações e acontecimentos do mundo. As representações individuais são conhecidas como sentenças. As sentenças são expressas em uma linguagem especifica, chamada linguagem de Representação do Conhecimento (RUSSEL e NORVIG, 2004). As linguagens de representação de conhecimento baseiam-se em diferentes técnicas de representação, como: regras de produção, redes semânticas, frames e lógica. Os SBCs também podem usar combinações entre essas técnicas. Sistemas que se utilizam dessa metodologia são conhecidos como Sistemas Híbridos.

2.2.3. Memória de Trabalho

A Memória de Trabalho representa a área de trabalho de um SBC, na qual são registradas todas as respostas fornecidas pelo usuário durante as interações realizadas com o sistema, evitando que o usuário responda à mesma questão mais de uma vez. Nela também podem ser registradas as conclusões intermediárias e seqüências de passos de raciocínio realizados durante a execução dos programas (Rezende, 2003).

2.2.4. Interface

A interface realiza a comunicação entre o SBC e o usuário. Pode ser elaborada genericamente ou pode ser construída em função do domínio da aplicação. A linguagem de programação utilizada na construção das interfaces pode ser diferente das linguagens utilizadas na representação do conhecimento.

2.3. Como resolver um problema utilizando Inteligência Computacional?

Para decisar se a IC pode ser utilizada para resolver um determinado problema, algumas questões precisam ser consideradas:

- Primeiro é importante estar clara a necessidade de que a solução para o problema em questão requer a incorporação de conhecimento especializado sobre o domínio da aplicação.

- Em seguida, deve-se investigar qual o tipo de conhecimento envolvido e quais as principais idéias a serem consideradas.

- Simultaneamente ao item anterior, deve-se procurar idenficar as fontes do conhecimento a ser utilizado. E como essas fontes podem ser acessadas.

- A partir de então, deve-se refletir sobre a melhor forma de representar do conhecimento e sobre como poderia ser realizado processo de aquisição e representação. Mais a frente serão apresentadas algumas formas de representação do conhecimento.

Em geral, problemas cuja solução demanda uma sequencia de passos rígida, bem definida não são adequados para serem resolvidos com recursos de IC. Nestes casos deve-se pensar na possibilidade de codificação da solução utilizando alguma técnica tradicional e alguma linguagem de programação orientada a objetos ou mesmo estruturada.

2.4. Hierarquia do Dado – Informação – Conhecimento

Uma vez que a IC lida com conhecimento, informação e dados, faz-se necessária uma distinção entre estes conceitos. A figura 2.2 apresenta um exemplo mostrando a hierarquia entre eles.

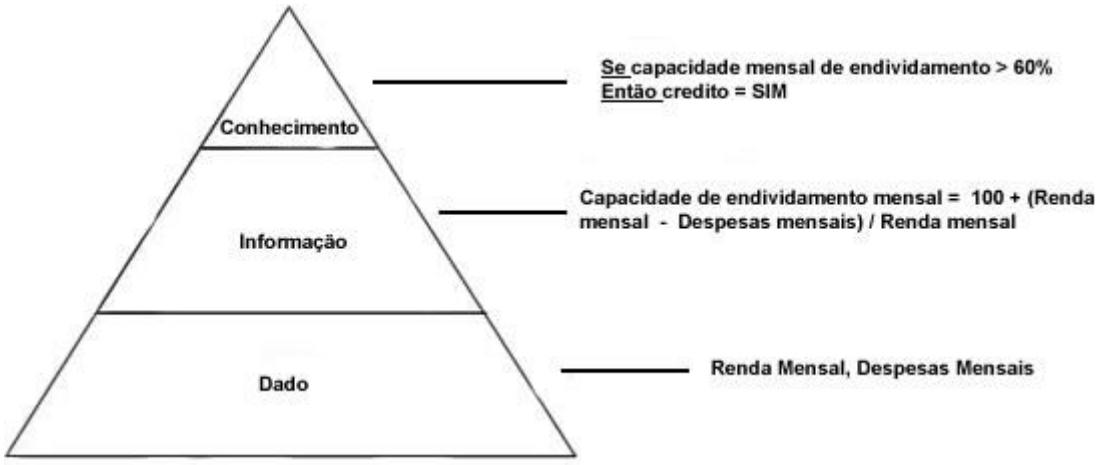


Figura 2.2 Hierarquia do Dado – Informação – Conhecimento

- Dados: representações sintáticas sobre fatos, mas sem semântica ou significado atribuído. Por exemplo, os valores 5.000, Silva, 10101990. Na figura 2.2, os valores 5.300 e 3.247 são exemplos de dados.
- Informações: São dados munidos de significado, possivelmente processados. Envolve interpretação de um conteúdo a partir do estabelecimento de um contexto. No exemplo da figura 2.2 uma informação (capacidade de endividamento) foi calculada a partir de outras (Renda e Despesa Mensais).
- Conhecimento: Inclui, mas não está limitado, às descrições, hipóteses, conceitos, teorias, princípios e procedimentos que são ou úteis ou verdadeiros. Possui diversas definições cujo debate foge ao escopo deste livro. Na figura 2.2, segue o exemplo de um conhecimento representado como uma regra da forma SE-ENTÃO. Envolve a relação entre informações.

Platão, em sua definição clássica, estabelece que conhecimento é um conjunto de crenças verdadeiras e justificadas, conforme ilustra a figura 2.3. A epistemologia é a ciência que estuda o conhecimento.

O conhecimento pode ser dividido em duas categorias:

- Conhecimento explícito – Aquele formal, claro, regrado, fácil de ser descrito e comunicado a outras pessoas. É o conhecimento que está registrado em livros, revistas, artigos, diagramas, desenhos, figuras e documentos, dentre outros. Esse conhecimento é de fácil articulação, manipulação e transmissão. A palavra explícito vem do latim *explicitus* que significa “formal, explicado, declarado"

- Conhecimento tácito – é o conhecimento que existe na cabeça das pessoas, formulado a partir de experiências que cada um adquiriu ao longo de sua vida. Geralmente é difícil de ser formalizado ou explicado a outra pessoa, pois é subjetivo e inerente às habilidades de um indivíduo, como "know-how". A palavra tácito vem do latim *tacitus* que significa "não expresso por palavras".

Conforme comentado, o conhecimento tácito é o mais difícil de se capturar e transmitir. Já o conhecimento explícito pode ser facilmente adquirido e até mesmo confundido com informação pura e simples.

Portanto, é, em geral, um desafio adquirir o conhecimento tácito, visto que este está na mente das pessoas. Como este conhecimento está vinculado a uma pessoa em particular, as grandes organizações possuem a dificuldade de transmitir a outros este tipo de conhecimento. Devido à importância desta forma de conhecimento, as grandes organizações deveriam investir fortemente em maneiras de manter seus funcionários, tentando encontrar formas de diminuir a rotatividade de pessoal e de aumentar a interação, facilitando assim a disseminação de conhecimento e da informação.

Organizações ocidentais, principalmente americanas e européias, têm investido esforços em converter conhecimento tácito em explícito – em documentos, processos, bases de dados, etc. Esse esforço é freqüentemente chamado de "transformação do capital humano no capital estrutural de uma organização".

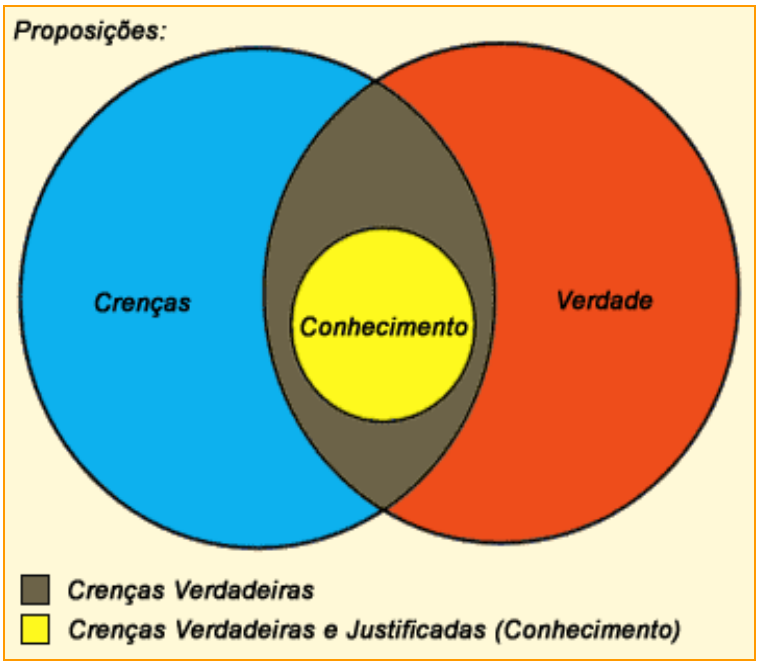


Figura 2.3 Ilustração gráfica da definição de conhecimento, segundo Platão

2.5. O que é Representação do Conhecimento?

Davis *et al.* (1993) *apud* (Rezende, 2003) definem representação do conhecimento (RC) como algo que substitui o objeto ou fenômeno real, de modo a permitir a uma entidade determinar as conseqüências de um ato pelo pensamento ao invés de sua realização.

Representação de Conhecimento (RC) pode ser interpretada como a aplicação de linguagens formais usadas para expressar os conhecimentos de especialistas em algum campo, de forma eficiente, e colocá-los prontos para serem acessados pelo usuário de um sistema inteligente. Uma representação é um conjunto de combinações sintáticas e semânticas que nos possibilitam descrever uma determinada aplicação.

A representação sintática especifica os símbolos que podem ser usados na representação do conhecimento e as maneiras como estes símbolos podem ser conseguidos.

A representação semântica, por outro lado, especifica que significado está incorporado naqueles símbolos representados pela sintaxe.

Qualquer que seja a forma de representação do conhecimento, esta deve dispor de algum mecanismo computacional que possa processar o conhecimento representado.

2.5.1. Exemplos de Formas de Representação do Conhecimento:

2.5.1.1. Lógica Matemática (Cálculo de Predicados)

A Lógica Matemática é uma linguagem formal, sendo o estudo matemátivo mais antigo sobre a natureza do raciocínio e do conhecimento. Foi um dos primeiros esquemas de representação usados em Inteligência Artificial.

A lógica matemática possui diversas regras de dedução, que são formas de se realizar inferências dedutivas a partir das expressões da linguagem, sem influência de idéias extras ou intuições. Existem vários tipos de lógicas que são utilizadas para realizar deduções automáticas. No entanto, o presente texto compreende apenas o cálculo de predicados (ou lógica de primeira ordem). Neste cálculo, as proposições são formadas por predicados, argumentos, quantificadores e variáveis. A lógica de primeira ordem é de grande importância para a representação do conhecimento e tem sido o instrumento mais utilizado para a formalização do conhecimento no desenvolvimento de um SBC.

Consideremos o exemplo de sentença que estabelece que: *Todo amigo de Paulo é amigo de Pedro.*

Em sua representação no Cálculo de Predicados, temos:

$$\forall X(\text{amigo}(X, \text{Paulo}) \rightarrow \text{amigo}(X, \text{Pedro}))$$

Onde:

Predicado: amigo(X,Y)

Variáveis: X, Y

Constantes: Paulo, Pedro

A Lógica de Predicados envolve sentenças cujo valor lógico deve ser apurado após a instanciação de variáveis. No exemplo, os predicados amigo(X, Paulo) e amigo(X,

Pedro) só assumem valor lógico (verdadeiro ou falso) após a variável X ter recebido algum valor (instanciada).

2.5.1.2. Regras de Produção

Muitos sistemas se inspiram na idéia de que a tomada de decisão humana pode ser moldada por meio de regras de condição do tipo SE condição ENTÃO conclusão e ações. Portanto, as regras podem expressar relacionamentos lógicos e equivalentes de definições para simular o raciocínio humano. Um exemplo simples pode ser ilustrado pela afirmativa: "SE está chovendo ENTÃO carregue uma sombrinha". Assim, dado o fato de "estar chovendo" pode-se inferir ou derivar que se deve "carregar uma sombrinha".

Portanto, como comentado anteriormente, regras de produção são estruturas no seguinte tipo:

SE <condições> ENTÃO <conclusões> FAÇA <ações>

Onde:

SE é uma lista de condições a serem satisfeitas (chamado de antecendente ou LHS – Left Hand Side, lado esquerdo – da regra; ENTÃO é uma lista de conclusões; e FAÇA são as ações a serem executadas. Conclusões e ações são chamadas de conseqüente ou RHS - Right Hand Side, lado direito – da regra.

Em um processo de inferência, cada uma das condições da lista de condições deve ser verificada. Caso todas sejam satisfeitas, as conclusões serão consideradas verdadeiras e as ações serão executadas.

2.5.1.3. Redes Semânticas

Uma rede semântica é um grafo rotulado e direcionado formado por um conjunto de nós representando os objetos (indivíduos, coisas, conceitos, situações em um domínio) e por um conjunto de arcos representando as relações entre os objetos (Rezende, 2003) *apud* (Rich e Knight, 1993; Branchman, 1983). Um arco é classificado de acordo com o nome da relação representada por ele. Os diversos arcos podem possuir o mesmo rótulo, contudo, cada objeto relacionado é representado por um único nó.

Os objetos podem ser classificados como complexos ou simples. Os objetos complexos muitas vezes podem ser decompostos em objetos mais simples e através dessas composições produzirem dois tipos de relações:

- Classe-de (is-a) – as relações entre os objetos estão em uma taxonomia hierárquica;
- Faz-parte (part-of) – as relações entre os objetos satisfazer a um tipo de composição, ou seja, o objeto é um elemento de outro, não havendo uma relação de herança.

Uma propriedade muito importante dessa relação é a transitividade, que permite declarar de forma concisa a propriedade nos objetos mais gerais. Mecanismos de inferência podem ser utilizados para derivar essas propriedades para os objetos mais específicos. A esse procedimento é denominado herança de propriedades. A principal razão das redes semânticas serem bem aceitas na representação do conhecimento é a

possibilidade de visualização gráfica das estruturas de conhecimento, porém essas possuem limitações expressivas que restringem o uso deste tipo de linguagem.

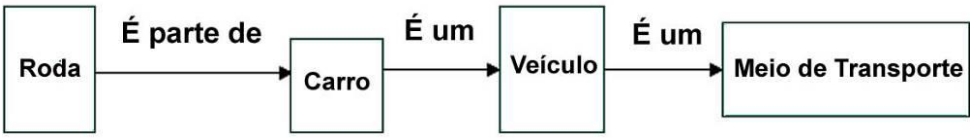


Figura 2.4 (a) Exemplo de uma rede semântica

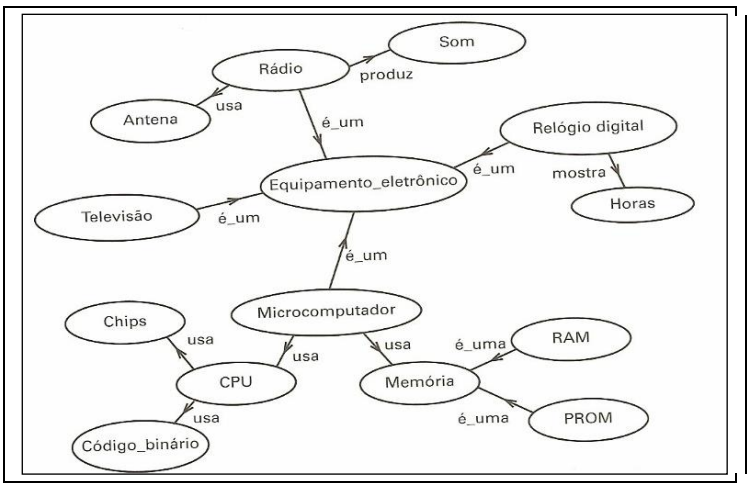


Figura 2.4 (b) Outro exemplo de uma rede semântica

2.5.1.4. Quadros (frames) e Roteiros (scripts)

O modelo de *frames* e *scripts* foi introduzido em 1975 na representação do conhecimento por Marvin Minsky. Quadros e Roteiros são usualmente aplicados para modelar situações estereotipadas (Quadros) para as quais podem ser definidas ações padronizadas (Scripts).

Um frame é um termo usado para designar um agrupamento de conhecimentos relevantes a uma coisa, um indivíduo, uma situação ou um conceito (Rezende, 2003) *apud* (Maida, 1987; Minsky, 1975). Os *frames* integram conhecimento declarativo sobre objetos e eventos e conhecimento procedimental sobre como recuperar informações ou calcular valores. Um frame possui um nome que identifica o conceito por ele definido e consiste de um conjunto de atributos, chamados *slots*. São estruturas de dados complexas e análogas a registros em bases de dados que modelam objetos do mundo real de forma mais poderosa e expressiva. Cada frame possui um nome que o referencia e detalhes da herança por ele submetida por seus *frames-pai*, junto com uma coleção de *slots* que contêm valores ou ponteiros para valores. A figura 2.5 apresenta um exemplo contendo alguns quadros.

Cada *slot* possui um nome que o identifica e um conjunto de atributos que apresentam propriedades que dizem respeito ao tipo de valores e às restrições de número que podem ser associadas a cada atributo. Essas propriedades são chamadas *facetas*.

As facetas contêm informações que descrevem os *slots.* Essas informações definem os valores que o *slot* pode assumir, ou indicam a maneira de calcular ou deduzir o seu valor (procedimentos). Exemplos de facetas são: tipo, domínio, valor *default*, etc.

Os valores dos *slots* podem ser definidos explicitamente ou herdados de um de seus ancestrais. Já os valores default são associados a objetos ou a classes gerais, enquanto os valores correntes são associados a instâncias específicas. Observação: Os valores default são usados somente quando o valor corrente não está disponível. A principal característica desse tipo de modelo é a relação de herança de propriedades, na qual uma classe especifica chamada de *frame-filho* pode herdar propriedades da classe mais geral (*frame-pai*).
Um *frame-filho* pode herdar valores (*default* ou correntes) de qualquer um de seus *frames-pai*, que por sua vez, herdaram de seus pais, e assim por diante, o que permite que não ocorra a duplicação de informação.
Instâncias aparecem com nós-folhas na hierarquia de *frames* e podem ter apenas um pai. Elas podem conter apenas valores correntes em seus *slots*. Em geral, a informação flui na hierarquia de *frames* a partir dos *frames* do topo para os *frames* das extremidades, o que caracteriza a relação de herança. Sempre que ocorrer um pedido de algum valor de um *slot*, o algoritmo de herança é automaticamente chamado. O valor é primeiramente procurado no frame original, e apenas se tal valor não existir localmente será necessário procurar em outros lugares, para isso será necessário saber onde procurar e quando parar a procura.
A característica de herança permite que os dados sejam armazenados de forma abstrata e aninhados com propriedades comuns herdadas. Desta forma, se evita a duplicação de informação, simplifica o código e proporciona a criação de sistemas de fácil leitura e manutenção.

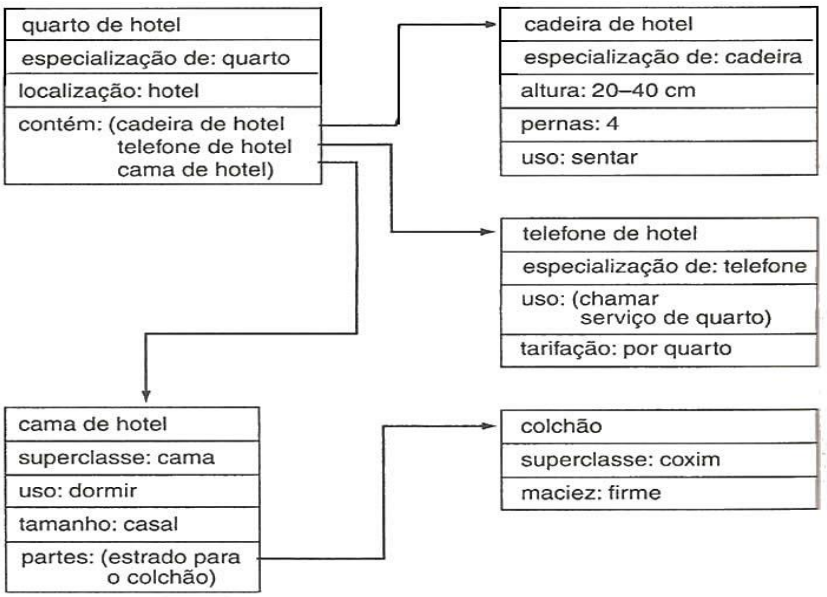


Figura 2.5 Exemplo de Quadros (Frames)

Os *scripts* são estruturas de informação que auxiliam na compreensão de situações de comportamento padronizado. Eles inspiraram o estudo de sistemas de Raciocínio

Baseado em Casos em que a busca por situações análogas é fundamental para solucionar um novo problema. Os *scripts* são úteis porque, no mundo real há padrões para a ocorrência de eventos. Entretanto, a definição de um *script* não é necessariamente compartilhada por todos, já que cada memória compreende um *script* sobre uma experiência, a partir do próprio ponto de vista. Os *scripts* possuem o conhecimento normativo, mas não o conhecimento da experiência. A figura 2.6 ilustra um exemplo de roteiros, definidos em função de cenários.

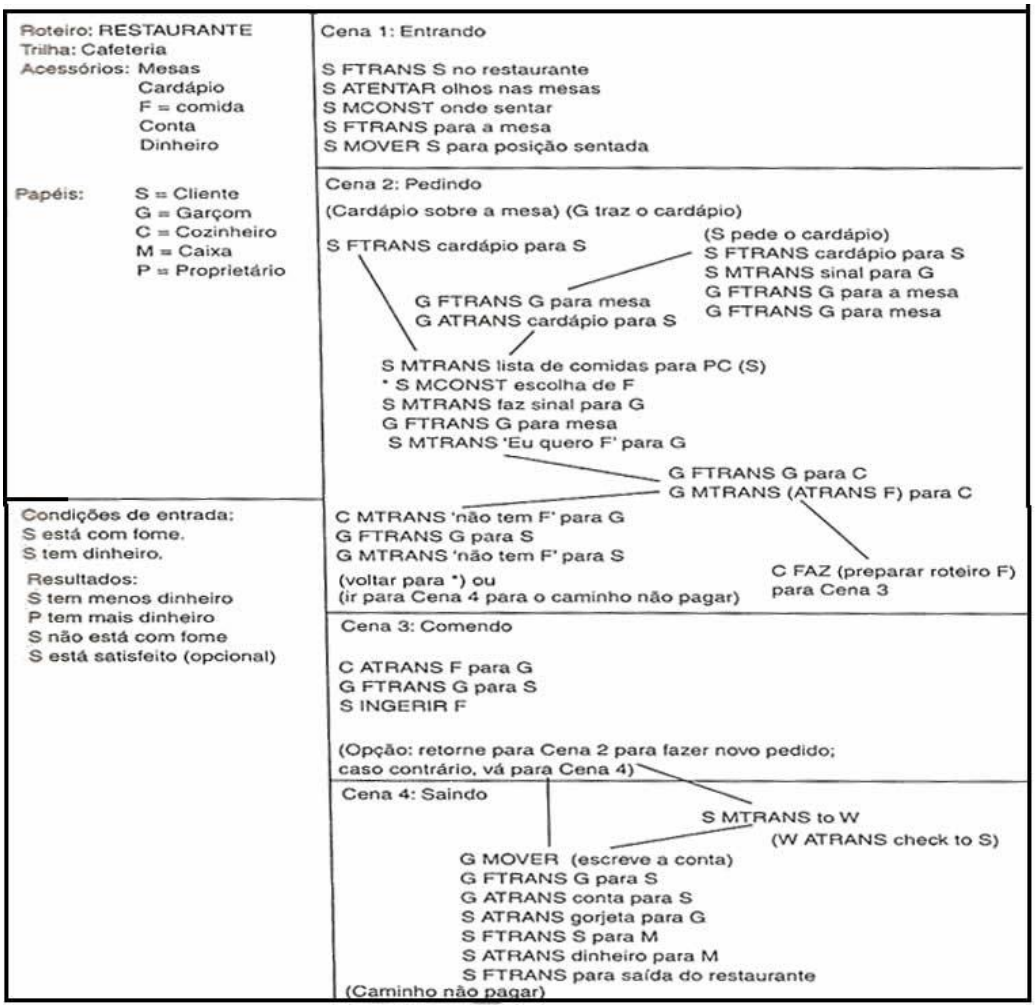


Figura 2.6 Exemplo de Roteiros (Scripts)

2.5.1.5. Grafos Conceituais

De forma análoga às Redes Semânticas e conforme o póprio nome sugere, os grafos conceituais modelam graficamente o conhecimento como um grafo: vértices são itens (não necessariamente nomeados) e arestas expressam a relação entre os itens.

Os componentes de um grafo conceitual podem ser descritos como:

a) os conceitos, representados por retângulos ou por colchetes [CONCEITO], expressam ações, estados ou entidades em um domínio específico de conhecimento;

b) as relações conceituais podem ser abreviadas como *relações*, sendo divididas em dois tipos:

1. Representada por círculo ou parênteses (RELAÇÃO) identificando as ligações existentes entre dois conceitos, caracterizando o relacionamento entre eles;
2. Representada por losango ou entre os símbolos de maior e menor RELAÇÃO>, identifica a hierarquia ou a precedência entre os conceitos.

As ligações entre os conceitos são expressas por meio das relações e formam pares ordenados. Os arcos que constituem as ligações têm a finalidade de definir o "sentido da leitura". Este sentido é identificado pela seta presente no arco pertencente à relação. Caso uma relação entre conceitos seja desfeita, o arco que a representa no grafo conceitual precisa ser removido.

A representação gráfica contribui para o melhor entendimento das relações entre os conceitos. A fim de ilustrar as definições acima, a figura 2.7 mostra um grafo conceitual que representa a frase "Brasil produz açúcar e vende à Argentina".

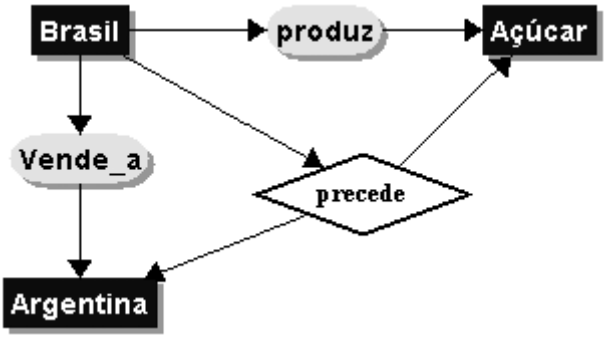


Figura 2.7 Grafo Conceitual da frase "Brasil produz açúcar e vende à Argentina"

O grafo da figura 2.7 é lido respeitando os sentidos das setas e observando a precedência colocada pela relação <precede>. Seguindo estas orientações, é possível obter duas interpretações do exemplo proposto:

1º O Brasil produz açúcar e vende à Argentina

2º O Brasil vende à Argentina açúcar que produz

A relação <precede> possui função ressaltar qual o conceito principal e quais dependem dele. Neste exemplo, os conceitos açúcar e Argentina são diretamente dependentes do conceito Brasil.

O grafo conceitual da figura 2.8 elimina a ambigüidade da sentença original "O cão coça a sua orelha com a sua pata" ao indicar por meio de uma variável que a orelha e a pata pertencem ao mesmo cão responsável pela ação de coçar. Além disso, o grafo explicita que o instrumento para a ação de coça é a pata e que o objeto coçado é a orelha.

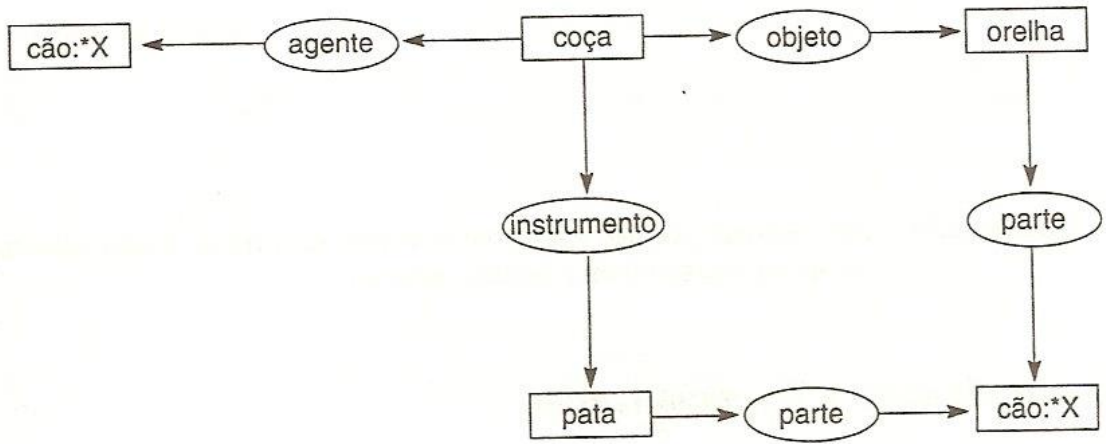


Figura 2.8: Grafo Conceitual da sentence "O cão coça a sua orelha com sua pata".

2.5.1.6. Ontologias

O acentuado e constante crescimento do volume de dados disponíveis nas empresas e no cotidiando tem se tornado um grande problema, em função da dificuldade natural de encontrar, organizar, associar, acessar e manter as mais variadas informações solicitadas pelos usuários. As ontologias proporcionam um meio de lidar com este tipo de problema, fornecendo meios de representação de conhecimento que contêm informações a cerca de itens e sobre como estes se relacionam entre si. Como conseqüência, as ontologias vêm conquistando cada vez mais o seu espaço e promovendo um entendimento comum e compartilhado sobre o domínio onde as informações se encontram organizadas.

Embora existam diversas definições para ontologia, aquela mais amplamente utilizada pela área da Computação estabelece que: "Uma especificação formal e explícita de uma conceitualização compartilhada." (Gruber, 1993).

Pode-se dizer, a grosso modo, que ontologias são modelos de dados que permitem definir categorias para as conceitos que existem em um mesmo domínio e que possuem o objetivo de promover um entendimento comum e compartilhado sobre um propósito específico. Ou ainda, em outras palavras: uma ontologia é um corpo de conhecimento que contém informações a cerca de itens e sobre como estes itens se relacionam entre si.

A figura 2.9 apresenta um exemplo clássico de ontologia onde as associações entre os conceitos representam especialização entre eles.

Ontologias são utilizadas para a recuperação de informações, gestão de conhecimento, processamento de linguagem natural e outros, trabalhando de forma a definir um vocabulário específico usado para descrever uma certa realidade e um conjunto de decisões explícitas, fixando de forma rigorosa, o significado pretendido para o vocabulário, procurando reduzir assim, as ambigüidades, melhorando a precisão das busca e reduzindo o tempo gasto. Normalmente as ontologias procuram descrever indivíduos, classes, atributos e relacionamentos.

São exemplos de linguagens para definição de ontologias: OIL, RDF, OWL, entre outras...

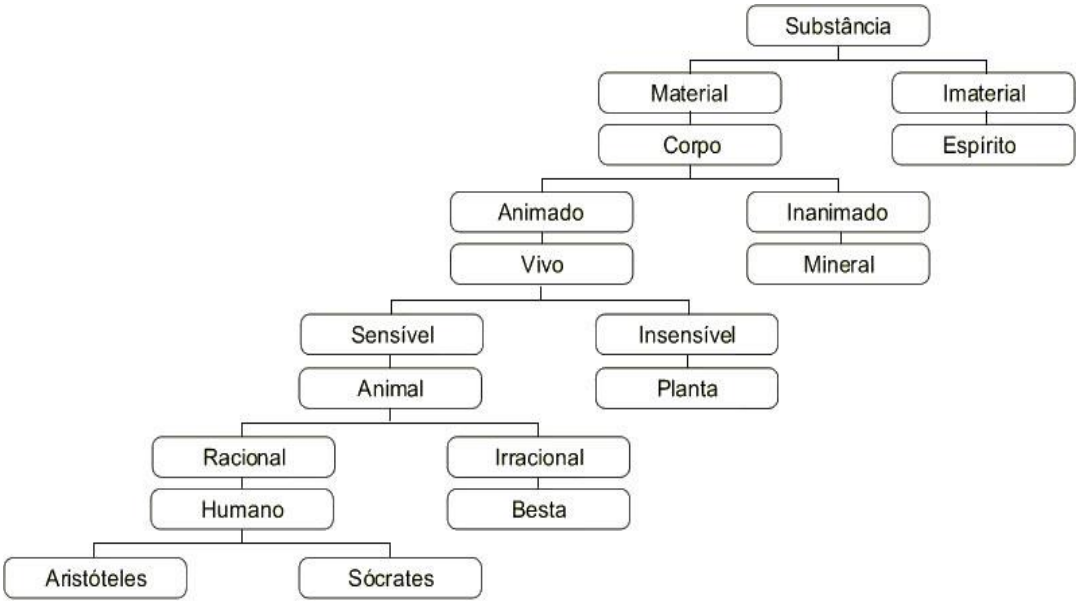


Figura 2.9 Exemplo de Ontologia – Árvore de Porfírio

2.6. Como escolher qual modelo de Representação do Conhecimento deve ser utilizado?

Uma das perguntas mais difíceis de serem respondidas é que tipo de Representação do Conhecimento utilizar, visto que não existe uma teoria geral sobre o assunto. Sabe-se que toda forma de representação do conhecimento deve dispor de algum mecanismo computacional que possa processar o conhecimento representado e que essa representação deve obedecer alguns princípios básicos, como os já mencionados nas seções acima. Sendo assim, a melhor representação do conhecimento a ser escolhida dependerá do contexto no qual esta será aplicada, levando-se em conta as suas características para se definir bem a representação que atenderá as necessidades envolvidas. No entanto, algumas questões devem ser consideradas:

(i) No problema a ser tratado, há a necessidade de se adquirir mais conhecimento? Em caso positivo, modelos conexionistas (envolvendo redes neurais) podem ser necessários. Caso seja utilizada alguma forma de representação de conhecimento simbólica (conforme as apresentadas nas seções anteriores), a incorporação de novos conhecimentos poderá não ser automática, requerendo, portanto, a intervenção humana no processo.

(ii) Caso a recuperação de conhecimento seja necessária, representações simbólicas se mostram bastante interessantes, uma vez que facilitam o entendimento e compreensão do homem.

(iii) Nos casos em que haja necessidade de processar o conhecimento para obter soluções para situações que se apresentem, mecanismos de inferência que permitam a dedução de novos fatos a partir de fatos existentes são necessários.

2.7. Etapas do Processo de Desenvolvimento de um SBC

Segundo (Rezende, 2003), o processo de desenvolvimento de um SBC está dividido em quatro fases, conforme mostra a figura 2.10. A fase 1 é realizada apenas uma vez, e as fases 2, 3, 4 compõem uma etapa contínua de melhoramento do sistema.

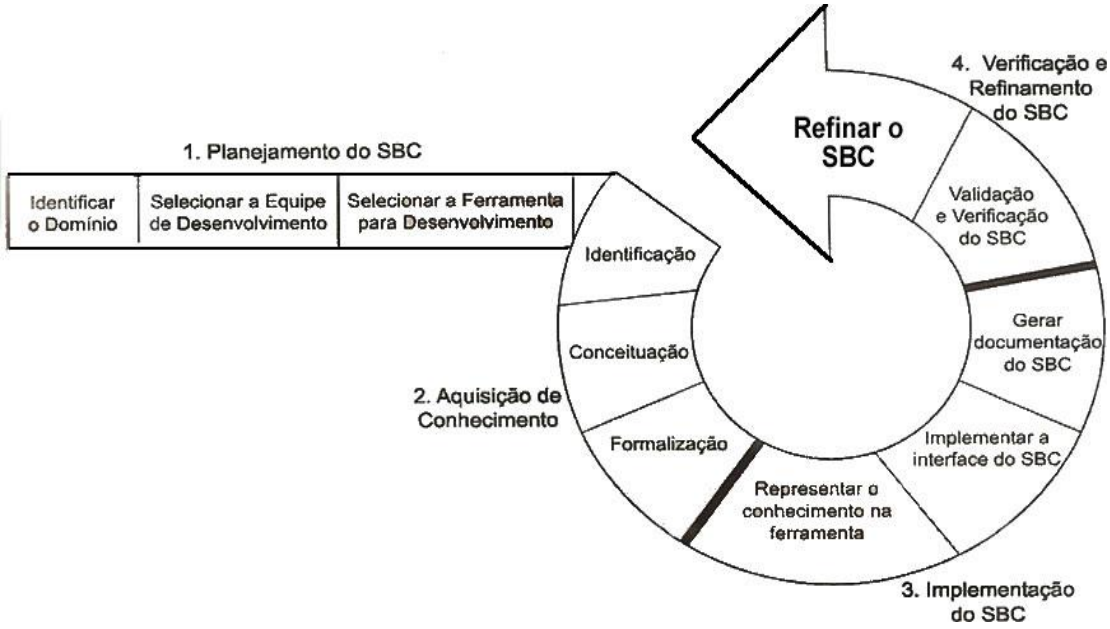


Figura 2.10 Fases do desenvolvimento de um SBC. Fonte: (Rezende, 2003)

Fase 1 – Planejamento do SBC: Tem como objetivo descreve o domínio de conhecimento, termos chaves e referências. Também identifica um resumo simplificado dos conceitos relacionados ao domínio de conhecimento, para que as pessoas que interagirem com o processo de desenvolvimento do SBC possam compreendê-lo melhor. Nessa fase, é realizada a análise funcional, responsável por identificar módulos, entradas e saídas necessários. Ainda nesta fase, são selecionadas a equipe de desenvolvimento do SBC e a ferramenta a ser utilizada no desenvolvimento do sistema. Compreende também a especificação da linguagem a ser usada na representação do conhecimento do domínio.

Fase 2 – Aquisição do Conhecimento: Esta fase tem como objetivo adquirir os conhecimentos que serão armazenados na Base de Conhecimento, ou seja, é a fase de execução do planejamento realizado na fase anterior. Esta fase refere-se à identificação, conceitualização e formalização do conhecimento.

Fase 3 – Implementação do SBC: Nesta fase, o conhecimento adquirido deve ser implementado. Para isso, utiliza-se a estrutura de Representação do Conhecimento selecionada na Fase 1 deste processo. Ainda nesta fase é realizada a codificação do sistema por meio de linguagens ou ferramentas adequadas. Compreende também a documentação do sistema, geração de manuais e implementação da interface.

Fase 4 – Validação e Refinamento do SBC: Esta fase envolve a validação e verificação do sistema e é considerada um processo contínuo, pois é necessário

assegurar que o sistema funcione corretamente, forneça resultados verdadeiros (corretos) e satisfaça os requisitos do cliente. Além disso, realiza eventuais mudanças nos requisitos do sistema, enfatizando a aquisição contínua do conhecimento e a avaliação do sistema em andamento.

Um dos estágios mais complexos no desenvolvimento de um Sistema Baseado em Conhecimento é a Aquisição de Conhecimento, que visa identificar e modelar o conhecimento que será utilizado na solução genérica de problemas em um domínio de aplicação.

2.8. Técnicas para Aquisição de Conhecimento (AC) e Esforços na Sistematização do Processo de AC

2.8.1. O que é?

Aquisição de conhecimento é o processo utilizado para se obter "conhecimento", de fontes diversas. Buchanan *et al.* (1983) *apud* (Rezende, 2003) definem AC como a transferência e transformação do conhecimento especializado com potencial para a resolução de problemas de alguma fonte de conhecimento para um programa. Aquisição de conhecimento é um processo feito geralmente por um especialista, ou seja, aquele que tem o conhecimento técnico acerca do assunto no qual o Sistema Inteligente (SI) será baseado e um engenheiro de conhecimento responsável por codificar as informações extraídas do especialista para a construção das Bases de Conhecimento (BC). As bases de conhecimento têm por finalidade servir de apoio ao (SI) fornecendo o "conhecimento apropriado" sempre que necessário.

Em função da complexidade do processo de aquisição de conhecimento, alguns especialistas chegam até a chamá-lo de gargalo da construção dos Sistemas Inteligentes. Essa dificuldade advem do fato da inexistência de uma metodologia padronizada e confiável para a extração do "conhecimento". Não raro o processo de aquisição de conhecimento é feito de maneira "artesanal", e se dá por meio de brainstorms, entrevistas estruturadas com o(s) especialista(s), e com a observância do(s) mesmo(s) trabalhando.

Na maioria dos casos, o engenheiro de conhecimento não consegue extrair do especialista todas as informações de que ele precisa, e às vezes as informações perdidas são realmente necessárias para a construção de um Sistema Inteligente confiável. Esse problema se dá devido ao fato de alguns conhecimentos estarem tão naturalmente solidificados na mente do especialista que eles só são ativados em uma situação real de trabalho. Alguns conhecimentos são tão naturais para o especialista que eles permanecem "escondidos" em um ponto da memória e só são recuperados com estímulos. Como já mencionado, são situações reias de trabalho que requerem o uso desse conhecimento específico.

Outro problema que ocorre com freqüência é a dificuldade de verbalização de uma resolução de problema por parte do especialista, ou seja, ele sabe resolver determinada questão, mas não sabe explicar para o engenheiro de conhecimento como realmente ele faz isso. Há ainda a questão de alguns profissionais da empresa serem tão importantes e requisitados que é difícil conseguir um tempo com eles para a realização das entrevistas e aplicação dos questionários.

2.8.2. Técnicas de Aquisição de conhecimento

Devido à necessidade de fazer a aquisição de conhecimento de forma clara e efetiva, várias técnicas têm sido desenvolvidas para ajudar nesse processo. Elas são classificadas em manuais, semi-automáticas e automáticas. As técnicas manuais são as mais utilizadas, sendo comandadas inteiramente pelo engenheiro de conhecimento. As semi-automáticas são realizadas junto com as manuais. Elas proporcionam ao especialista, ferramentas para ajudar na tarefa da criação dos sistemas diminuindo a participação do engenheiro de conhecimento. As automáticas visam minimizar ao máximo a participação humana. Elas utilizam aprendizado de máquina para fazer a mineração de conhecimento a partir de grandes fontes de informação, sendo, por esse motivo, mais complexas.

2.8.2.1. Técnicas Manuais

2.8.2.1.1. Baseadas em Descrições ou em Literaturas

Nesta técnica, o engenheiro de conhecimento realiza um estudo sobre o assunto que o sistema se propõe a auxiliar com o intuito de adquirir um conhecimento sobre o domínio. Fazer esse estudo prévio é importante para que as entrevistas com o especialista possam ocorrer dar de forma mais natural, sem que o especialista precise explicar tudo sobre o assunto em questão. O engenheiro que estudou previamente o assunto pode ir levantando questionamentos, por exemplo, sobre termos técnicos. Isso faz com que a conversa flua de forma mais natural e em um nível mais adiantado. Desta forma, o número de entrevistas pode ser reduzido, o que demanda uma despesa menor para a empresa, visto que ela cederá menos vezes o funcionário para as entrevistas.

Convém mencionar, no entanto, que nem sempre existem referências homologadas sobre o assunto em questão. Além disso, muitos assuntos demandam um conhecimento prévio para entendimento dos textos disponíveis para estudos.

2.8.2.1.2. Baseadas em Entrevistas

Nesta técnica são realizadas entrevistas com o especialista. As informações podem ser coletadas com o auxílio de gravadores ou filmadoras. Essas informações são depois estudadas para se extrair delas o conhecimento desejado. No entanto, o uso de quaisquer destes recursos deve ser previamente acordado junto ao especialista, evitando possíveis constrangimentos.

Recomenda-se que pelo menos dois engenheiros de conhecimento participem do processo. Um questiona e o outro toma nota dos pontos principais de forma a facilitar o posterior levantamento dos assuntos discutidos.

Outro procedimento desejável é a elaboração prévia de questionários que auxiliem a condução do processo de entrevista. Neste caso, diz-se que a entrevista está estruturada. Recomenda-se opcionalmente a divulgação prévia do questionário. Entrevistas estruturadas são, em geral, mais produtivas. Baseiam-se em um processo sistemático orientado a objetivo, facilitando a comunicação entre os envolvidos. Ajudam a evitar distorções decorrentes da subjetividade.

Nos casos em que as perguntas surgem em decorrência da evolução da conversa, as entrevistas são denominadas de entrevistas não estruturadas. Este tipo de abordagem

pode ser útil para realizar prospecção de detalhes sobre o assunto que não tenham sido mencionados anteriormente. Por outro lado, pode trazer as seguintes desvantagens:

- Especialistas podem não se preparar para a entrevista
- Engenheiros de conhecimento com pouca experiência podem se desorientar
- Ocorrência de dificuldades para o especialista organizar idéias
- Demanda de preparação prévia dos envolvidos (complexidade)
- Dificuldades para interpretação e integração da informação

2.8.2.1.3. Baseadas em Acompanhamento

Esta técnica visa acompanhar o processo de raciocínio do especialista em casos reais, ou seja, acompanhá-lo em seu local de trabalho. Isto faz com que ele seja mais natural e espontâneo. Em geral, com esta técnica, as informações surgem mais facilmente e o engenheiro faz as devidas anotações e esclarece dúvidas com o especialista na medida em que elas se apresentem.

Por usar casos reais, esta técnica evita que o especialista seja direcionado a responder questões irrelevantes ao sistema. Cabe ressaltar, no entanto, que com esta técnica, nem sempre se consegue uma amostragem de casos realmente representativa.

Opcionalmente pode ser utilizada uma abordagem em que o especialista analisa situações anteriores, explicando como determinadas soluções foram obtidas.

2.8.2.2. Técnicas Semi-automáticas

As técnicas semi-automáticas foram criadas para suprir as falhas que podem ocorrer quando se utilizam técnicas manuais, pois essas são mais suscetíveis a erros devido ao número de pessoas envolvidas – especialistas, engenheiros de conhecimento e programadores.

O processo de obter o conhecimento do especialista e repassá-lo posteriormente ao programador acaba gerando ruídos de comunicação entre as partes envolvidas. Uma possível solução para resolver essa deficiência é a aquisição de conhecimento semi-automática. Esta forma de aquisição consiste na utilização de ferramentas computacionais que ajudam o engenheiro de conhecimento a codificar melhor o conhecimento a ser incorporado no sistema. Auxiliam na formalização e edição do conhecimento, evitando tanto erros de sintaxe quanto erros lógicos na estruturação do modelo em construção.

KESAQ, PATERAQ, SEGSE, dentre outras, são exemplos de ferramentas de aquisição de conhecimento semi-automáticas.

As técnicas semi-automáticas, em geral, proporcionam uma redução do número de pessoas envolvidas no processo, e conseqüentemente os problemas de comunicação também são reduzidos. A aquisição de conhecimento semi-automática também agiliza o processo de construção das bases de conhecimento, visto que permite que o engenheiro e o especialista consigam obter respostas mais rápidas, uma vez que as bases podem ser

testadas enquanto vão sendo estruturadas. Isso faz com que possíveis erros de modelagem ou interpretação apareçam prematuramente.

2.8.2.3. Técnicas Baseadas em Aprendizado de Máquina

O Aprendizado de Máquina é uma área da Inteligência Computacional que investiga meios para tornar computadores capazes de aprender a partir da experiência. Este tópico encontra-se melhor descrito mais à frente neste mesmo capítulo (seção 2.12).

2.8.2.4. Técnicas Baseadas em Mineração de Dados e Mineração de Textos

Muitas vezes, o conhecimento a ser adquirido encontra-se embutido em dados históricos disponíveis em grandes bases de dados. A área da Mineração de Dados tem como objetivo abstrair padrões úteis (conhecimento) a partir de grandes bases de dados estruturados. Por outro lado, a Mineração de Textos volta-se para dados semi-estruturados ou mesmo desestruturados. Embora, sejam temas de crescente interesse na atualidade e estarem fortemente relacionados à Inteligência Computacional, fogem do escopo deste livro. Maiores detalhes podem ser obtidos em literatura especializada tais como, por exemplo, (Goldschmidt e Passos, 2005), (Carvalho, 2001).

2.9. Problemas e Espaço de Soluções/Estados

Dado um problema cuja solução seja aderente às tecnologias proporcionadas pela Inteligência Computacional, uma questão que surge é saber qual o número de soluções possíveis. Um único problema pode admitir infinitas soluções e a cada passo o problema pode situar-se em um estado diferente. Entende-se por estado de um problema como sendo a posição ou condição em que tal problema se encontra em um determinado instante.

Em geral, o espaço de solução de um problema, também chamado espaço de estados ou espaço de busca, é representado computacionalmente por meio de um grafo no qual cada vértice corresponde a um estado e as arestas indicam transições entre dois estados. A busca pela solução de um problema compreende a navegação pelos vértices do grafo de estados associado a fim de encontrar o(s) vértice(s) cujo(s) estado(s) corresponde(m) a solução de um problema. A figura 2.11 apresenta um trecho do espaço de solução do jogo da velha.

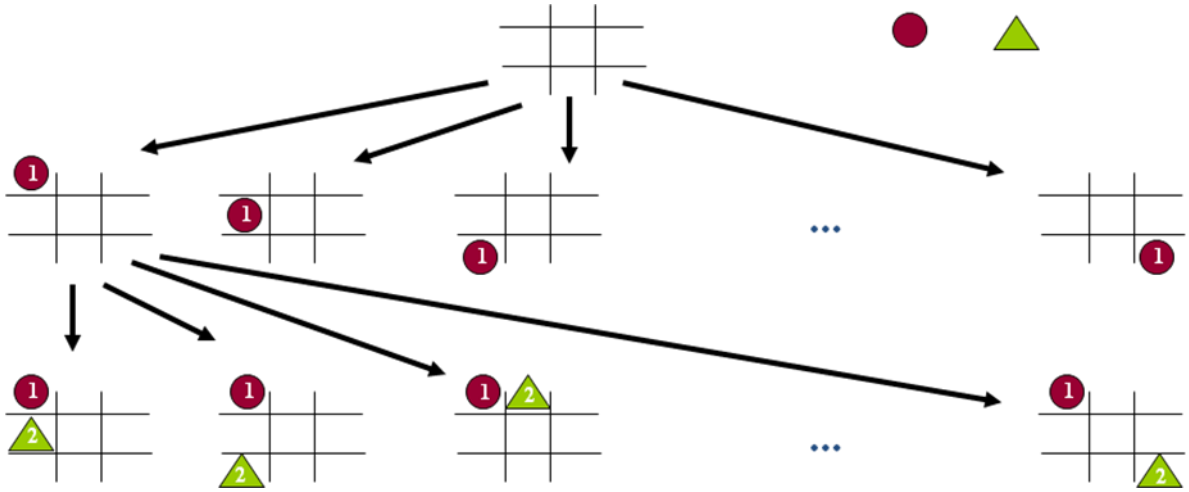


Figura 2.11 – Trecho do espaço de solução do jogo da velha

Há várias estratégias para percorrer o grafo de estados em busca da solução de um problema. Em muitas delas faz-se necessária a utilização de uma boa função heurística. (Russel e Norvig, 2004).

2.10. O que é Heurística?

A palavra heurística está relacionada ao verbo grego original, "eurisco", que significa "eu descubro". É uma junção de intuição, lógica e conhecimento prévio sobre determinado assunto, formalizado em regras e/ou métodos, que podem ajudar na invenção, descoberta e solução de problemas.

Em outras palavras uma heurística é um conhecimento prévio sobre um determinado problema que auxilia na busca por soluções, simplificando o espaço de busca associado.

Na busca em espaço de estados, heurísticas são formalizadas como regras que determinam a escolha dos ramos que possuem a maior probabilidade de levarem a uma solução aceitável para o problema, podendo este possuir a probabilidade de falhar, pois são apenas hipóteses informadas sobre o próximo passo a ser dado, baseado em experiências e na intuição.

Os processos de busca baseados em heurísticas exigem, em geral, menos tempo que os processos algorítmicos na busca por soluções eficientes.

A seguir na figura 2.12 encontram-se ilustradas três heurísticas para o conhecido jogo-dos-oito. A figura 2.13 mostra um trecho do espaço de busca do jogo dos oito.

Considere a fórmula genérica para apuração do valor herístico f(n) para cada estado como sendo expressa por: f(n) = g(n) + h(n), onde:

g(n) é o comprimento real do caminho de um estado ***n*** qualquer até o estado inicial.

h(n) é uma estimativa heurística da distância entre o estado ***n*** e o objetivo.

Dentre dois estados com mesmo ***h(n)*** deve ser escolhido aquele com menor ***g(n)*** (menor caminho)

Cada uma das heurísticas do exemplo ignora uma parcela crítica de informação, podendo ser melhorada.

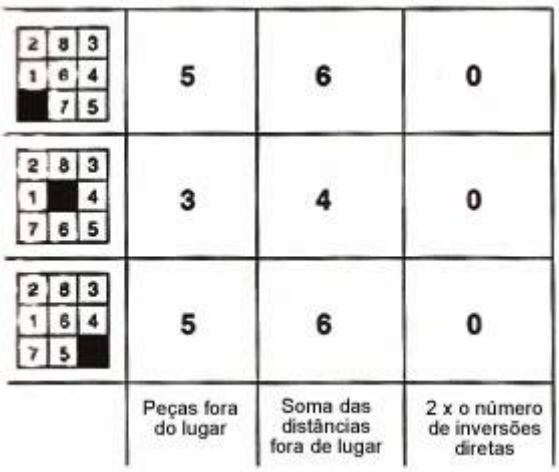


Figura 2.12 Três heurísticas aplicadas a estados no jogo-dos-oito.

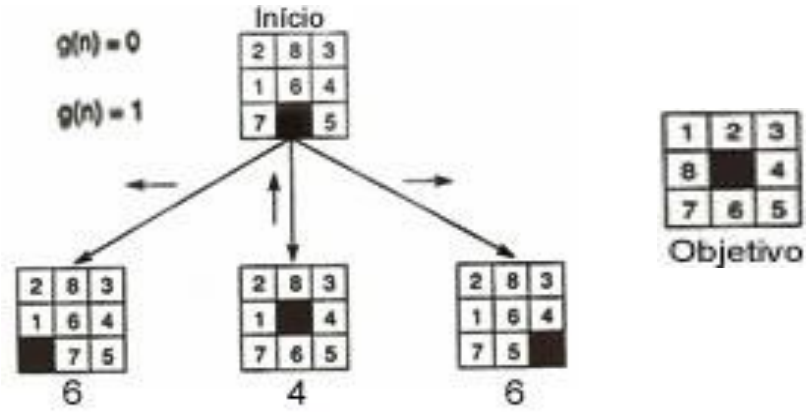


Figura 2.13 Trecho do grafo de busca do jogo-dos-oito

A figura 2.14 ilustra uma heurística para o jogo da velha que se baseia no maior número de vitórias possíveis de serem alcançadas a partir do estado em que o jogo se encontra em um determinado momento.

Figura 2.14 Heurística do Jogo da Velha considerando o estado inicial (tabuleiro vazio)

2.11. Comparação entre Programas de IA e Programas Convencionais

A tabela 2.1 apresenta uma comparação entre características de programas elaborados com recursos de Inteligência Artificial e programas convencionais:

Tabela 2.1 Vantagens e desvantagens de SEs e especialistas humanos

Programas com IA	Programas Convencionais
Processamento simbólico	Processamento numérico
Soluções heurísticas (passos da solução estão implícitos)	Soluções algorítmicas (passos da solução estão explícitos)
Estrutura de controle do programa independente do domínio do conhecimento	Estrutura de controle e informações (muitas vezes) integrados
Alteração do conhecimento em geral não requer alterações no programa	Alteração do conhecimento muitas vezes requer alterações no programa
Fácil de modificar e atualizar.	Modificações são, em geral, trabalhosas.
Respostas satisfatórias são aceitas.	Em geral, busca-se a melhor resposta.

2.12. Aprendizado de Máquina (AM)

2.12.1. O que é?

Aprendizado de máquina é uma área da inteligência artificial que tem por objetivo desenvolver algoritmos e técnicas computacionais que permitam que o computador seja capaz de aprender, ou seja, façam com que o computador consiga adquirir conhecimento de forma automática a partir de exemplos históricos, e assim, aperfeiçoar seu desempenho em determinada tarefa.

O que torna esse aprendizado possível são os sistemas de aprendizado. Esses sistemas são programas de computador capazes de tomar decisões baseados em resoluções bem sucedidas de problemas (exemplos históricos) já vistos por eles em experiências anteriores. Os sistemas de aprendizado possuem características bastante comuns e que possibilitam sua classificação quanto à linguagem de descrição, paradigma, modo e forma de aprendizado utilizado.

Os exemplos históricos utilizados pelos algoritmos de Aprendizado de Máquina são representados por meio de um conjunto de características, também denominadas de atributos, que procura descrever o problema em questão e que, portanto, varia em função do contexto de aplicação. Para cada situação ou caso de um problema, os atributos contêm valores que descrevem tal situação. A escolha dos atributos para descrever os exemplos de uma aplicação possui grande influência na qualidade do conhecimento adquirido pelos algoritmos de aprendizado.

De forma análoga, a representatividade dos exemplos disponíveis para o processo de aprendizado também interfere no desempenho dos Algoritmos de Aprendizado e na qualidade do conhecimento por eles abstraído. Isto porque o aprendizado ocorre de forma indutiva. Os algoritmos partem de exemplos específicos, procurando construir

modelos de conhecimento que sejam compatíveis, não só com os exemplos utilizados no aprendizado, mas também com possíveis novos exemplos que possam surgir. A capacidade de um modelo de conhecimento gerado por um algoritmo de aprendizado ser compatível com novos exemplos é denominada generalização. Quanto maior a capacidade de generalização de um modelo de conhecimento mais útil e desejável tal modelo pode se mostrar em aplicações práticas reais.

Recordando alguns fundamentos dedutivos da Matemática, a indução é a forma de inferência lógica que permite conclusões genéricas a partir de um conjunto particular de exemplos. Ela se caracterizada pelo raciocínio originado a partir de um conceito específico que é generalizado (Rezende, 2003).

No aprendizado indutivo, que é base para os principais algoritmos de aprendizado de máquina da atualidade, é comum que o modelo de conhecimento seja abstraído a partir de sucessivas iterações sobre o conjunto de exemplos históricos disponíveis. Pode ser subdividido em:

- Aprendizado Supervisionado – Nesta forma de aprendizado indutivo, os exemplos históricos disponíveis devem conter qual a informação esperada a ser produzida pelo modelo de conhecimento que está sendo construído.
- Aprendizado Não Supervisionado – Nesta abordagem de aprendizado, os algoritmos procuram agrupar os exemplos históricos em função da similaridade que eles apresentem entre si. Desta forma, exemplos mais similares tendem a ficar em um mesmo grupo, enquanto que exemplos diferentes tendem a ser organizados em grupos distintos.

2.12.2. Paradigmas de Aprendizado

Um paradigma de aprendizado diz respeito à forma com que o espaço de busca por um modelo de conhecimento (que represente os dados históricos disponíveis) deve ser percorrido. A seguir encontram-se os principais paradigmas de aprendizado sobre os quais os algoritmos de aprendizado de máquina se baseiam. Podem ser utilizados tanto para predição quanto para descrição do conjunto de dados existente.

- Simbólico: Compreende a construção da representação de conceitos a partir da análise de exemplos. Neste paradigma o modelo de conhecimento construído pode ser representado por meio de expressões lógicas, árvores, regras, redes semânticas, dentre outras.
- Estatístico: Neste paradigma, métodos estatísticos (em geral, paramétricos) são utilizados para encontrar boas aproximações do modelo de conhecimento que esteja sendo induzido.
- Baseado em Exemplos: Envolve a busca de casos existentes similares ao novo exemplo a ser analisado para deduzir a saída do sistema. Neste paradigma, é ideal a utilização de casos anteriores representativos.
- Conexionista: Utiliza modelos matemáticos simplificados inspirados no modelo biológico do sistema nervoso para tentar abstrair mapeamentos de novos exemplos nas saídas desejadas ou agrupamentos de exemplos similares.

- Evolucionário: Baseia-se nos modelos biológicos da evolução natural e da reprodução genética para evoluir soluções que competem entre si a fim de abstrair um modelo que solucione o problema em questão.

Capítulo

3

Sistemas Especialistas

3.1. Introdução

Sistemas Especialistas são sistemas que armazenam e processam conhecimento adquirido de especialistas em uma área de conhecimento. São sistemas de apoio à decisão que reúnem conhecimentos acerca de áreas específicas e que são capazes de simular o comportamento humano diante de situações a eles apresentadas. Utilizam conhecimentos e procedimentos inferenciais para resolver problemas não triviais que requerem para sua solução alguma ou muita perícia humana.

São geralmente desenvolvidos para atender a uma aplicação determinada e limitada do conhecimento humano. São também, capazes de emitir uma decisão e flexíveis para incorporação de novos conhecimentos para melhorar seu raciocínio. Utilizam conhecimento justificado e bases de informações, tal qual um especialista humano de determinada área do conhecimento.

A partir do conhecimento nele incororado, um Sistema Especialista pode tomar decisões para proporcionar respostas a questões utilizando um processo de tomada de decisão, ou dividindo esse processo por meio de interações com o especialista humano.

De um modo geral, sempre que um problema não pode ser algoritmizado, ou sua solução conduza a um processamento muito complexo e demorado, os Sistemas Especialistas podem ser uma saída, pois possuem seu mecanismo de inferência apoiado em processos heurísticos.

Uma característica importante dos Sistemas Especialistas é que tais sistemas não são influenciados por elementos externos a eles, como ocorre com o especialista humano, sujeito a emoções, pressões, dentre outros fatores. Para as mesmas condições de entreda, um Sistema Especialista deve fornecer sempre o mesmo conjunto de decisões.

A tabela 3.1 apresenta uma pequena comparação entre sistemas especialistas e especialistas humanos.

Para tomar uma decisão sobre um determinado assunto, um especialista o faz a partir de fatos que encontra e de hipóteses que formula, buscando em sua memória um conhecimento prévio armazenado durante anos, no período de sua formação ou no decorrer de sua vida profissional, sobre esses fatos e hipóteses. E o faz de acordo com a sua experiência, isto é, com o seu conhecimento acumulado sobre o assunto. Com esses fatos e hipóteses, emite a decisão.

Tabela 3.1 Vantagens e desvantagens de SEs e especialistas humanos

Especialista Humano	Sistemas Especialista
Perecível	Permanente
Difícil de transferir	fácil de ser transferido
Difícil de documentar	fácil de documentar
Imprevisível	Consistente
Caro	viável economicamente
Criativo	sem inspiração
Adaptável	deve ser atualizado
Sensorial	alimentado com dados simbólicos
Visão ampla	visão estreita
Bom senso	conhecimento técnico

Durante esse processo, são formuladas novas hipóteses e novos fatos são verificados. Esses irão influenciar no processo de raciocínio. Este raciocínio é sempre baseado no conhecimento prévio acumulado. Com esse processo de raciocínio, um sistema especialista pode não chegar a uma decisão se os fatos de que dispõe para aplicar o seu conhecimento prévio não forem suficientes. Pode, por este motivo, inclusive chegar a uma conclusão errada; mas este erro é justificado em função dos fatos que encontrou e do seu conhecimento acumulado previamente.

Para a construção bem sucedida de um sistema especialista, se fazem necessários alguns pré-requisitos, tais como:

- A definição de uma área do conhecimento humano, caracterizando assim o aspecto da especialização sobre um determinado assunto.
- A existência de um analista de conhecimento, pessoa responsável pelo processo de aquisição e formalização do conhecimento e construção do sistema.
- A existência de um ou mais especialistas na referida área, que deverão ser os interlocutores do analista de conhecimento na formulação do sistema especialista.

Além disso, segundo Lucena (1987):

- "Deve existir pelo menos um especialista humano sobre o qual se possa dizer que ele é capaz de se desempenhar bem uma tarefa considerada."
- "As principais fontes do desempenho excepcional do especialista devem ser conhecimento especializado, julgamento e experiência."

3.2. Principais Características

Para entender quais são as principais características comuns aos Sistemas Especialistas, basta examinar o que estes fazem:

- Utilizam raciocínio inferencial;
- Armazenam conhecimentos de forma permanente;
- Resolvem problemas muito complexos tão bem quanto, e às vezes melhor que especialistas humanos;
- Raciocinam heuristicamente, usando o que os peritos consideram efetivamente regras práticas;
- Podem interagir com usuários humanos utilizando inclusive linguagem natural;
- Manipulam e raciocinam sobre descrições simbólicas;
- Funcionam com dados errados e regras incertas de julgamento;
- Contemplam hipóteses múltiplas simultaneamente;
- Explicam porque estão fazendo determinada pergunta;
- São tolerantes a erros, podendo chegar a respostas não ótimas, porém aceitáveis;
- São de fácil manutenção;
- São de fácil documentação;
- Agem sem influência de fatores emocionais, stress ou pressões;
- Apresentam baixo custo operacional;
- Em geral, oferecem segurança;
- São estáveis;
- Requerem um número reduzido de pessoas para interação;
- Justificam suas conclusões, explicando como chegaram a um resultado.

O núcleo de um Sistema Especialista é a potência do corpo de conhecimento acumulado durante sua construção. Este conhecimento é explícito e organizado de forma a simplificar o processo de decisão. A relevância desta característica deve ser suficientemente enfatizada: 'A acumulação e codificação de conhecimento é um dos mais importantes aspectos de um Sistema Especialista'.

Uma das mais importantes características de um Sistema Especialista consiste em sua especialidade de alto nível que auxilia na solução de problemas. Este conhecimento especializado pode representar a experiência dos melhores peritos no campo. Sua especialização de alto nível, juntamente com a habilidade de aplicação, torna seu custo competitivo e apto a ganhar espaço no mercado comercial. A flexibilidade do sistema também auxilia aqui: ele pode crescer incrementalmente segundo as necessidades do negócio ou organização. Isto significa que pode se iniciar com um investimento relativamente modesto expandindo-o de acordo com as necessidades.

O corpo de conhecimento do sistema que define a proficiência de um Sistema Especialista pode também oferecer uma capacidade adicional: a memória institucional. Se a base de conhecimento foi desenvolvida através de interação de pessoas chave da

organização, isto representa a política corrente do grupo. Esta compilação de conhecimento vem a ser o consenso de opiniões de alto nível de um registro permanente das melhores estratégias utilizadas. Quando pessoas chaves desligam-se da organização, suas experiências permanecem.

3.3. Estrutura Geral

Nem todos os SEs apresentam a mesma estrutura, no entanto, em sua grande maioria são constituídos por três elementos fundamentais: base de conhecimento, motor de inferência e interface com o usuário, conforme pode ser observado na figura 3.1:

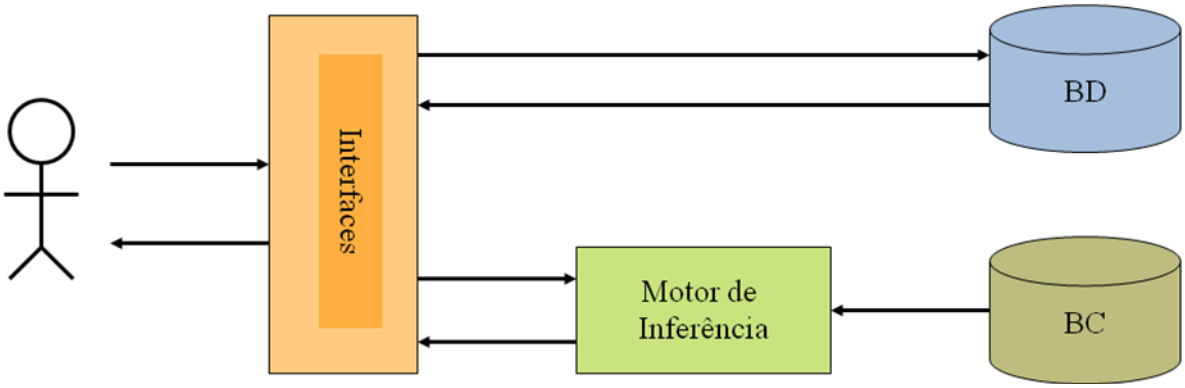


Figura 3.1 Estrutura básica de um SE

3.3.1. A Base de Conhecimento

A base do conhecimento não é uma simples coleção de informações. A tradicional base de dados com dados, arquivos, registros e seus relacionamentos estáticos é aqui representada por uma base de regras e fatos e também heurísticas que correspondem ao conhecimento do especialista, ou dos especialistas do domínio sobre o qual foi construído o sistema. Convém destacar que muitos sistemas especialistas possuem além da base de conhecimento, a tradicional de base de dados de onde alguns fatos são extraídos e outros armazenados.

Esta base de regras, assim como a base de fatos, é processada pelo motor de inferência, permitindo identificar as possibilidades de solução e o processo de raciocínio e inferência que levam a conclusões sobre o problema submetido ao sistema.

Na interação com a base de fatos e regras e com o usuário, obtêm-se as informações necessárias para a resolução do problema. Devido à utilização de heurísticas, o usuário é requerido pelo sistema para prestar informações adicionais e, a cada pergunta respondida pelo usuário ou a cada nova informação, reduz-se o espaço de busca a ser percorrido pelo sistema, encurtando-se o caminho entre o problema e sua solução.

Inicialmente uma base de conhecimento pode ser construída com poucas regras, mas, dependendo da complexidade do ambiente e das necessidades de informações variadas, esta base poderá eventualmente crescer para milhares de regras e fatos. Assim, é preciso

que se tenha o cuidado de implementar instrumentos internos de refinamento que possibilitem cortes e ajustes na base de conhecimento, para que o processo de busca localize segmentos cujas regras e fatos contemplem situações e circunstâncias que conduzam à solução dos problemas em questão.

Nos Sistemas Especialistas, o aprendizado ocorre com a incorporação de novas regras nas bases de conhecimento. A cada processamento da base de conhecimento, o sistema especialista assume como verdade o conhecimento ali formalizado. Isto é possível em virtude da estrutura modular da base de conhecimento, permitindo a adição ou deleção de novos elementos sem alterar a lógica global do sistema.

Em geral, o conhecimento formalizado em bases de conhecimento pode ser representado por meio de regras de produção. O conjunto de regras de produção pode ser mapeado em árores de decisão.

Amplamente utilizadas em algoritmos de classificação, as árvores de decisão são representações simples do conhecimento e, um meio eficiente de construir classificadores que predizem classes baseadas nos valores de atributos de um conjunto de dados.

As árvores de decisão são grafos acíclicos que consistem de nodos que representam os atributos, de arcos, provenientes destes nodos e que recebem os valores possíveis para estes atributos, e de nodos folha, que representam as diferentes classes de um problema.

Uma árvore de decisão tem a função de particionar recursivamente um conjunto de dados, até que cada subconjunto obtido deste particionamento contenha casos de uma única classe. Para atingir esta meta, a técnica de árvores de decisão examina e compara a distribuição de classes durante a construção da árvore. Os resultados obtidos, após a construção de uma árvore de decisão, são os dados organizados de maneira compacta, que são utilizados para classificar novos casos.

A figura 3.2 apresenta um exemplo de árvore de decisão. Neste exemplo, são trabalhados objetos que relatam as condições propícias de uma pessoa receber ou não um empréstimo. É considerada a possibilidades do montante do empréstimo ser médio, baixo ou alto. Alguns objetos são exemplos positivos de uma classe sim, ou seja, os requisitos exigidos a uma pessoa, por um banco, são satisfatórios à concessão de um empréstimo, e outros são negativos, onde os requisitos exigidos não são satisfatórios à concessão de um empréstimo. Classificação, neste caso, é a construção de uma estrutura de árvore, que pode ser usada para classificar corretamente todos os objetos do conjunto.

Muitos são os algoritmos de classificação que elaboram árvores de decisão. Não há uma forma de determinar qual é o melhor algoritmo, um pode ter melhor desempenho em determinada situação e outro algoritmo pode ser mais eficiente em outros tipos de situações.

O algoritmo ID3 foi um dos primeiros algoritmos para construção de árvore de decisão, tendo sua elaboração baseada em sistemas de inferência e em conceitos de sistemas de aprendizagem. Logo após foram elaborados diversos algoritmos, sendo os mais

conhecidos: C4.5, CART (Classification and Regression Trees), CHAID (Chi Square Automatic Interaction Detection), entre outros.

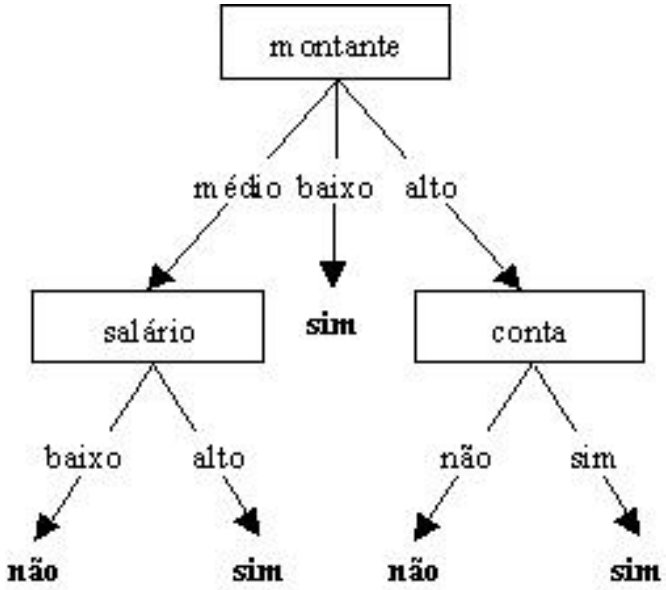


Figura 3.2 Exemplo de árvore de decisão

Após a construção de uma árvore de decisão é importante avaliá-la. Esta avaliação é realizada através da utilização de dados que não tenham sido usados na etapa de construção da árvore. Esta estratégia permite estimar como a árvore generaliza os dados e se adapta a novas situações, podendo, também, se estimar a proporção de erros e acertos ocorridos na construção da árvore.

A partir de uma árvore de decisão, é possível derivar regras. As regras são escritas considerando o trajeto do nodo raiz até uma folha da árvore. Devido ao fato das árvores de decisão tenderem a crescer muito, de acordo com algumas aplicações, elas são muitas vezes substituídas pelas regras. Isto acontece em virtude das regras poderem ser facilmente modularizadas. Uma regra pode ser compreendida sem que haja a necessidade de se referenciar outras regras.

Com base na árvore de decisão apresentada na figura 3.2, pode-se exemplificar a derivação de regras. Dois exemplos de regras obtidas a partir desta árvore são mostrados a seguir:

* Se montante = médio e salário = baixo

  então classe = não

* Se montante = médio e salário = alto

  então classe = sim

3.3.2. O Motor de Inferência

O motor de inferência é um elemento essencial para a existência de um sistema especialista. É o núcleo do sistema. É por intermédio dele que os fatos e regras de heurística que compõem a base de conhecimento são aplicados no processo de resolução do problema.

A capacidade do motor de inferência é baseada em uma combinação de procedimentos de raciocínios que se processam de forma regressiva e progressiva.

Na forma de raciocínio progressivo, também denominado encadeamento para frente, as informações são fornecidas ao sistema pelo usuário, que com suas respostas, estimulam o desencadeamento do processo de busca, navegando através da base de conhecimento, procurando pelos fatos, regras e heurísticas que melhor se aplicam a cada situação. O sistema continua nesta interação com o usuário, até esgotar todas as possibilidades de resposta para o problema.

No modelo de raciocínio regressivo, também denominado encadeamento para trás, os procedimentos de inferência dão- se de forma inversa. O sistema parte de uma opinião conclusiva sobre o assunto, podendo ser inclusive oriunda do próprio usuário, e inicia uma pesquisa pelas informações por meio das regras e fatos da base de conhecimento, procurando provar se aquela conclusão é a mais adequada solução para o problema analisado.

Uma importante característica dos Sistemas Especialistas é o reuso do motor de inferência. Os demais componentes variam em função da aplicação. O motor de inferência é o módulo do sistema responsável pelo processo de inferência que é aplicado sobre o conhecimento disponível na base de conhecimento em função dos fatos informados via interface ou obtidos da base de dados.

3.3.2.1. Modo de Raciocínio

Conforme comentado na seção anterior, como componentes fundamentais de sistemas baseados em regras, os motores de inferência podem utilizar, basicamente, dois modos de raciocínio para determinar um resultado: encadeamento progressivo ou encadeamento para frente (do inglês, “forward chaining”), e encadeamento regressivo ou encadeamento para trás (do inglês, “backward chaining”).

A base de conhecimento contém muitas regras SE/ENTÃO e fatos que são baseados nos resultados das regras. Em geral, os motores de inferência são desenvolvidos para trabalhar com um dos modos de raciocínio. Nos casos em que o motor de inferência pode processar os dois modos, há a necessidade de que seja especificado qual o tipo de encadeamento a ser utilizado.

Consideremos como exemplo um sistema especialista para conserto de casas. Um encanador (especialista) foi entrevistado para obter o conhecimento formalizado nas regras abaixo.

Regra 1:

Se você tem uma torneira mal vedada

E

Se a torneira é uma torneira de pressão

E

Se o vazamento está na manivela

Então

Aperte a porca de vedação.

Regra 2:

Se você apertou a porca de vedação.

E

Se o vazamento persiste

Então

Substitua a vedação.

Estas regras seriam armazenadas na base de conhecimento com os relacionamentos SE/ENTÃO e E/OU mostrados acima.

A máquina de inferência pode processar estas regras usando encadeamento para trás, se começar com a solução de substituir a vedação, e então verificar tal solução fazendo perguntas ao usuário para verificar o resultado. As perguntas feitas ao usuário são apenas para obter informações que não estão na base de conhecimento.

A máquina de inferência pode usar encadeamento para frente perguntando primeiro ao usuário toda a informação necessária. Então ela pega esta informação e tenta aplicá-la a diferentes regras para ver se ela se ajusta a quaisquer dos problemas previamente descritos. Se ajustar, então ela repassa a possível solução ao usuário.

Uma questão que surge naturalmente é qual o melhor modo de encadeamento. A resposta para tal questão pode ser dada com base nas seguintes heurísticas:

- Se o número de premissas (condições) no antecedente da regra for pequeno, comparado com o número de conclusões então use o encadeamento para frente.
- Se o número de conclusões for pequeno, comparado com o número de premissas então use o encadeamento para trás.

Há, no entanto, situações em que se faz necessário combinar os dois tipos de encadeamento. Por exemplo, em diagnósticos médicos, pois é comum um médico observar o paciente, para depois criar uma hipótese que precisa ser confirmada com exames complementares.

A tabela 3.2 apresenta uma comparação entre os dois tipos de encadeamento abordados, mostrando algumas características em cada um deles.

Tabela 3.2 Comparação entre Forward e Backward Chaining

Encadeamento para Frente	**Encadeamento para trás**
Dirigido aos dados	Dirigido às metas
Planejamento, monitorização e controle.	Diagnósticos
Presente para o futuro	Presente para o passado
Antecedente de uma regra para o conseqüente.	Do conseqüente de uma regra para o antecedente.
Trabalha para frente para encontrar soluções, partindo dos fatos.	Trabalha para trás para encontrar fatos que suportem as hipóteses levantadas.
Os antecedentes das regras determinam a busca.	Os conseqüentes da regra determinam a busca.

3.3.3. A Interface com o Usuário

A Interface com o usuário final é talvez o elemento em que os desenvolvedores de sistemas especialistas dedicam mais tempo projetando e implementando.

Um problema submetido a um sistema especialista é endereçado por estratégias de busca. O sistema sempre retém elementos de memória que permitam o encaixe e o encadeamento com outra estratégia, sempre marcando o caminho percorrido.

Para que isto ocorra, é necessário que a interface com o usuário seja bastante flexível. Assim, a interação entre sistema especialista e usuário conduz um processo de navegação, eficiente, na base de conhecimento, durante o processamento das heurísticas.

Uma interface com o usuário flexível permite que o usuário descreva o problema ou os objetivos que deseja alcançar. Permite, ainda, que usuário e sistema adotem um modelo estruturado de consultas.

Isto facilita o processo de recuperação do caminho percorrido pelo sistema em tentativas de solucionar o problema. Este caminho, denominado *trace* (trilha) é muito importante, pois é a base de pesquisa para o desenvolvimento do processo de explanação.

O processo de explanação consiste na explicação, quando requerida pelo usuário, sobre o "porquê" e o "como" o sistema chegou a determinada conclusão, rumo à solução do problema analisado. Neste momento, o sistema realiza um processo inverso de busca, percorrendo as trilhas utilizadas e marcadas durante a sessão de consulta e apresentando

todos os argumentos que o levaram à solução apresentada. Este processo é muito importante e proporciona ao usuário subsídios para julgar se adota ou não a solução apresentada pelo sistema especialista.

Ainda, pode-se considerar o processo de explanação como importante instrumento que poderá ser utilizado para o treinamento do usuário, uma vez que apresenta conceitos teóricos e aplicações práticas, reforçando a tese da importância das interfaces na aplicação de sistemas especialistas.

A interface com o usuário pode assumir formas variadas, dependendo de como foi implementado o sistema especialista. De qualquer forma, tem como principal objetivo procurar tornar o uso do sistema fácil e agradável, eliminando-se as complexidades.

3.4. Etapas de desenvolvimento de sistemas especialistas

Conforme descrito em (Passos, 1997), o desenvolvimento de um sistema especialista é normalmente realizado nas seguintes etapas:

- Reconhecimento do Problema – Como na Engenharia de Software tradicional, o reconhecimento do problema é a primeira etapa do processo de construção de um sistema de informação. Envolve a percepção da necessidade de se dispor de um sistema computacional, possivelmente dotado de inteligência, para auxiliar na realização de tarefas em um determinado ambiente.
- Estudo de Viabilidade – O Estudo de Viabilidade consiste, entre outras funções, em verificar a aderência da utilização de um sistema especialista na solução do problema. Existem problemas que são melhores solucionados utilizando-se outras tecnologias. Por exemplo, se o problema a ser solucionado envolver passos fixos pré-definidos, se não for para tomada de decisão ou for ser utilizado na camada operacional de uma empresa, existe uma grande probabilidade de um sistema especialista não ser a melhor solução.
- Aquisição e Formalização do Conhecimento – Durante essa etapa, o analista de conhecimento procura compreender o problema, identificar as variáveis normalmente analisadas na solução do problema e construir um modelo de inferência que permita, a partir das variáveis fornecidas, obter respostas sobre o referido problema. As variáveis, também denominadas atributos do problema, são propriedades mensuráveis inerentes ao problema, como por exemplo, para saber se uma pessoa está ou não com febre é preciso saber a sua temperatura e comparar a uma escala conhecida para fins de classificação. A combinação desses atributos deve conduzir o sistema ao objetivo final que corresponde à solução do problema. Ainda nesta etapa é necessário também identificar a classe ou tipo de cada atributo, indicando, portanto, se este é numérico ou categórico. Em geral, o modelo de conhecimento é composto por uma base de conhecimento. Uma das formas de representação de conhecimento mais utilizadas é a de regras de produção. A união das regras de produção de um SE constitui a base de conhecimento desse sistema. Tal base pode ser criada por meio de exemplos nos quais o especialista da área fim indica, para cada situação, a sua provável conclusão. Algoritmos como o ID3 (a ser comentado

mais adiante) permitem induzir bases de conhecimento a partir de situações de exemplo.

- Projeto – O projeto de um SE, assim como dos demais tipos de sistemas de informação, consiste na definição de modelos que norteiem a implementação do sistema. Envolve ainda a escolha de ambientes e linguagens com os quais o SE deverá ser codificado.
- Implementação – A implementação de um SE difere da implementação de um sistema de informação convencional no que se refere à incorporação de um módulo de processamento do conhecimento responsável pela inferência de novos fatos a partir dos fatos associados a cada situação e do conhecimento incorporado na base.
- Testes e validação – Esta etapa compreende, dentre outros aspectos normalmente envolvidos na validação de um sistema de informação, a avaliação crítica do conhecimento formalizado. Diversos casos de teste são apresentados de forma a verificar a consistência do modelo de conhecimento incorporado ao sistema.
- Treinamento e Implantação – Correspondem às etapas clássicas de ensino aos futuros usuários do sistema quanto à sua manipulação assim como a colocação desse sistema em ambiente operacional.

3.5. Ferramentas

Nesta seção, são descritas algumas ferramentas de apoio à construção de sistemas especialistas.

3.5.1. Ferramentas de Aquisição de Conhecimento

A pesquisa na área de aquisição de conhecimento tem focalizado o desenvolvimento em ferramentas de aquisição de conhecimento automatizadas. Essas ferramentas são projetadas para serem utilizadas diretamente pelo especialista do domínio, ajudando-o a estruturar o conhecimento, com o objetivo de minimizar as intervenções do engenheiro do conhecimento. O engenheiro do conhecimento tem atuado como um facilitador no processo. Suas responsabilidades passam então a ser:

- Aconselhar os especialistas no processo de elicitação interativa do conhecimento,
- Estabelecer e gerenciar apropriadamente as ferramentas interativas de aquisição do conhecimento,
- Editar a base de conhecimentos codificada com a colaboração dos especialistas,
- Validar a aplicação da base de conhecimentos com a colaboração dos especialistas,
- Estabelecer a interface com usuários em colaboração com os especialistas e usuários,

- Treinando os usuários no uso efetivo da base de conhecimentos em colaboração com o especialista, através do desenvolvimento de procedimentos de treinamento e operacionais.

Além do exposto, métodos automatizados podem ajudar a padronizar o processo de aquisição do conhecimento, e o uso de técnicas específicas para certos tipos de conhecimento e metodologias em geral.

O uso de ferramentas automatizadas de aquisição de conhecimento pode trazer os seguintes benefícios:

- Aumento da qualidade da base de conhecimentos;
- Uma base de conhecimentos que reflete melhor o modelo do especialista do domínio;
- Redução do período de familiarização do engenheiro do conhecimento com o domínio.

Sem o intermediário entre o especialista e o sistema, há menos possibilidades de captação de informação errônea ou incompleta.

Para que o especialista possa codificar diretamente o seu conhecimento, sem muitos riscos, é necessário que ele esteja ciente do problema, do modo que ele vai abordar a solução e da sua capacidade de conceitualização sobre o domínio, além de ser capaz de analisar seu próprio conhecimento, de estar motivado para usar a ferramenta de forma consciente e de assegurar o desempenho do modelo que ele codifica.

É importante levantar a questão da dificuldade de reunir essas características em um especialista de determinado domínio. A aquisição interativa pode ser combinada com a aquisição manual e com o uso de ferramentas interativas por engenheiros ao invés, ou com a cooperação, dos especialistas.

Sendo assim, pode-se dizer que um ambiente de aquisição do conhecimento não fornecerá soluções mágicas, mas poderá facilitar bastante a tarefa do especialista do domínio e do engenheiro do conhecimento.

Seguem abaixo alguns exemplos de ferramentas de aquisição de conhecimento:

Ferramenta KSSO (Knowledge Support System Zero)

KSSO é um ambiente gráfico e interativo de aquisição do conhecimento, permitindo a construção de bases de conhecimento orientadas a objetos, na qual o conhecimento é formalmente representado como uma estrutura de heranças múltiplas de classes, objetos, propriedades, valores e relações.

Este ambiente é composto das seguintes ferramentas: Elicit (Elicitação Interativa), FOCUS (Agrupamento Hierárquco, PrinCom (Agrupamento Espacial), Sócio (Relações de conceitos em um grupo), Induct (Análise lógica) e Export (Transferência para shells).

O especialista pode então especificar o domínio e o contexto do problema, e entrar com atributos ainda confusos. Ao trabalhar com caso, de preferência um que represente características críticas do problema, ele descobre que os atributos definidos não são

suficientes. Aí então, com a ajuda da ferramenta, ele pode visulizar o conjunto de dados da base de conhecimento e perceber o que está faltando. A aquisição do conhecimento se dá por um processo de tentativa e erro, já que é difícil para o especialista transmitir rápida e resumidamente, de maneira organizada e bem estruturada, todo o conhecimento que levou anos e anos para acumular.

Em desenvolvimento de sistemas em larga escala, estruturas de conhecimento para diferentes sub-domínios devem ser construídas. A ferramenta de aquisição pode ser usada para desenvolver e validar essas estruturas de conhecimento. As bases geradas podem então ser combinadas através de estruturas de controle apropriadas.

O ambiente KESSO está implementado na linguagem Pascal e roda em computadores Apple Macintosh.

Ferramenta KMC (The Knowledge Mining Center)

KMC é um ambiente integrado para aquisição de conhecimento, para suporte ao processo de entrevistas, utilizando tecnologia de multimídia. Ele processa um documento de entrevista estruturada, armazenando tanto as anotações do engenheiro do conhecimento, como o registro em áudio da sessão. Esses registros são segmentados e anexados à questão associada.

A utilização do KMC é feita em três passos: preparação do documento de entrevista, realização da entrevista, com possibilidade de modificação do documento, avaliação do documento de entrevista modificado.

Originalmente o KMC oferecia suporte somente à etapa de realização da entrevista, mas também pode ser utilizada na preparação e avaliação da mesma.

No início da realização de uma entrevista, o KMC lê um documento estruturado de entrevistas, disponibilizando informações na tela através do navegador de entrevistas. O usuário pode acessar o navegador, ou o painel de controle de entrevistas, ou ainda digitar comandos rápidos. Desta forma ele pode selecionar questões ou informações adicionais. Todas as funcionalidades estão disponíveis tanto no modo navegador como no modo de entrevista. Essa ferramenta foi implementada em Emacs-Lisp.

Ferramenta KESAQ (Knowledge Expert System Aquisition)

O KESAQ é uma ferramenta brasileira gerada a partir de uma dissertação de mestrado do Instituto Militar de Engenharia.

Permite a construção de bases de conhecimento a partir de planilhas preenchidas pelo especialista. Para tanto, devem se informados os atributos (questões) do problema e o objetivo a ser concluído. Todos os valores possíveis dos atributos e do objetivo devem ser informados. Com estes dados, o KESAQ gera uma planilha com todas as combinações possíveis entre os valores dos atributos identificados. O especialista preenche a planilha informando para cada situação, qual a resposta julga mais adequada. Uma vez preenchida a planilha, o KESAQ permite a geração automática da base de conhecimento, no formato de regras de produção e de árvore de decisão. Utiliza para tanto, o algoritmo ID3, versão preliminar o algoritmo C4.5. Na tabela 3.3 segue um exemplo simplificado e fictício de uma planilha no cenário da análise de crédito. As

figuras 3.3 e 3.4 mostram, respectivamente a árvore de decisão e as regras de produção geradas pelo KESAQ por meio do ID3. Convém destacar que o algoritmo simplificou a base de conhecimento gerando apenas 3 regras ao invés das 4, naturalmente esperadas a partir da planilha de 4 linhas (um regra por linha da planilha). Observe que, nos casos em que a renda for considerada baixa, a resposta é sempre a mesma, independente do valor assumida pelo atributo despesa. Tal situação permitiu a simplificação. Maiores detalhes sobre o algoritmo ID3 podem ser obtidos em referências especializadas sobre o tema como, por exemplo, (Quinlan, 1993).

Tabela 3.3: Exemplo de planilha fictícia preenchida por um especialista

Renda	Despesa	Resposta (AP, NG, ES)
Alta	Baixa	AP
Alta	Alta	ES
Baixa	Baixa	NG
Baixa	Alta	NG

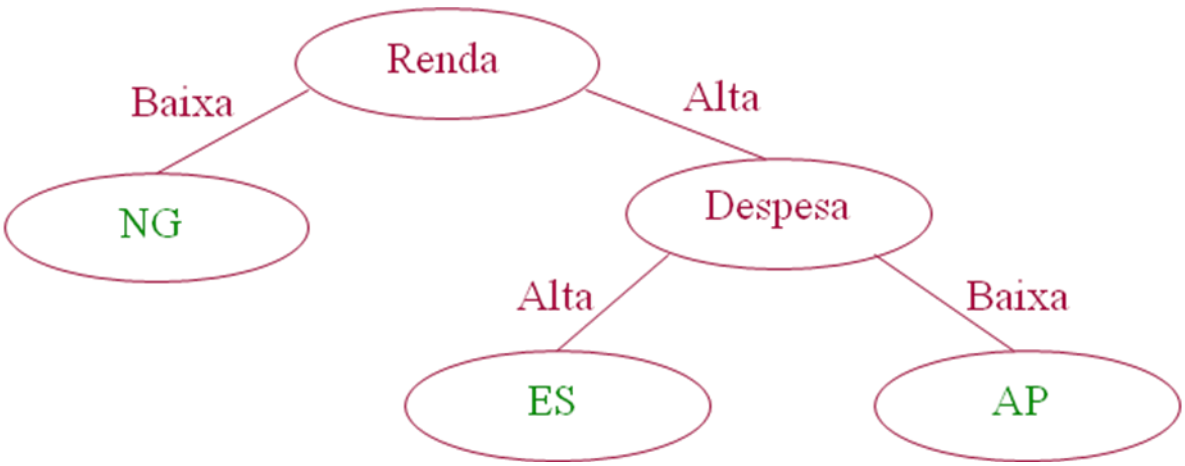


Figura 3.3: Árvore de decisão associada à planilha da Tabela 3.3

Regras de Produção:

1) Se Renda = Baixa Então Resultado = NG

2) Se Renda = Alta E Despesa = Alta Então Resultado = ES

3) Se Renda = Alta E Despesa = Baixa Então Resultado = AP

Figura 3.4: Regras de Produção associadas à árvore de decisão da figura 3.3

3.5.2. Ferramentas para Construção de Sistemas Especialistas

Expert SINTA (Uma ferramenta visual para criação de sistemas especialistas)

O Expert SINTA é uma ferramenta computacional que utiliza técnicas de Inteligência Artificial para construção de sistemas especialistas. Esta ferramenta utiliza um modelo de representação do conhecimento baseado em regras de produção e probabilidades, tendo como objetivo principal simplificar o trabalho de implementação de sistemas especialistas através do uso de uma máquina de inferência compartilhada do tratamento probabilístico das regras de produção e da utilização de explicações sensíveis ao contexto da base de conhecimento modelada. Um sistema especialista baseado em tal tipo de modelo é bastante útil em problemas de classificação. O usuário responde a uma seqüência de menus, e o sistema se encarregará de fornecer respostas que se encaixem no quadro apontado pelo usuário. Entre outras características inerentes ao Expert SINTA, podem ser destacadas:

- Utilização do encadeamento para trás (backward chaining);
- Utilização de fatores de confiança tanto para regras quanto para as respostas fornecidas.
- Ferramentas de depuração;
- Possibilidade de incluir ajudas on-line para cada base.

SEGSE (Sistema Especialista na Geração de Sistemas Especialistas)

O SEGSE, produto de um trabalho de conclusão do curso de Ciência da Computação do Centro Univesitário da Cidade (Mendes, 2005), foi concebido de forma fornecer apoio nas etapas de Estudo de Viabilidade, Análise e Modelagem do Conhecimento, Projeto e Implementação do Sistema Especialista. Incorpora conhecimento técnico para auxiliar na realização de cada uma dessas etapas. Esse conhecimento pode ser configurado por um ou mais analistas de conhecimento, especialistas no desenvolvimento de SEs. A figura 3.5 apresenta o diagrama de casos de uso com as principais funcionalidades do SEGSE.

Existem dois atores distintos na utilização do SEGSE. O analista de conhecimento é o principal operador do sistema. Responde a diversas perguntas formuladas pelo SEGSE, que fornece pareceres sobre o desenvolvimento do SE em questão. O especialista na área em que se deseja criar o SE interage com SEGSE de forma pontual durante a etapa de Aquisição e Modelagem do Conhecimento.

Embora recomendado para analistas de conhecimento aprendizes na área de sistemas especialistas, o SEGSE pode ser utilizado como instrumento para geração rápida de protótipos de SEs para testes e validação.

A seguir encontram-se comentados os casos de uso evidenciados na figura 3.5.

- Validar Aderência do Projeto – Este caso de uso tem como objetivo auxiliar a etapa de Estudo de Viabilidade. Para tanto, procura verificar se o contexto onde se deseja aplicar o SE é, de fato, um problema ao qual a solução por sistemas especialistas se mostra adequada. Este processamento é realizado sobre uma base de conhecimento construída com o objetivo de classificar problemas como aderentes ou não à tecnologia de sistemas especialistas.

- Definir atributos – Nesta funcionalidade, o SEGSE procura apoiar a etapa de Aquisição e Formalização do Conhecimento na definição de quais devem ser as questões consideradas no SE em construção. De forma análoga ao caso de uso anterior, o SEGSE busca validar a aderência de cada candidato a atributo como uma variável efetiva do problema em questão. Também neste processamento o SEGSE possui uma base de conhecimento própria para a classificação de um atributo como sendo de natureza compatível com aquela esperada para atributos de SEs.

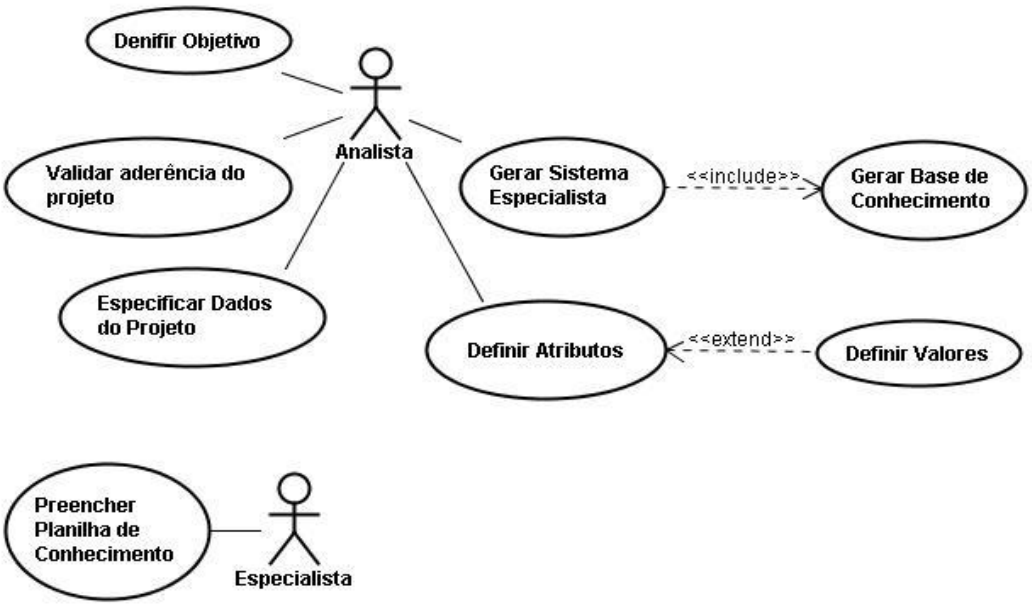


Figura 3.5 – Diagrama de Casos de Uso do SEGSE

- Definir Valores – Após a classificação de cada atributo como uma variável efetiva para utilização no sistema especialista em construção, o SEGSE procura auxiliar na validação dos prováveis valores do referido atributo. Também de forma análoga aos casos de uso anteriores, o SEGSE possui uma base de conhecimento cujo objetivo é classificar cada candidato a valor de

atributo como um valor de natureza compatível com a esperada para valores de atributos de SEs.

- Definir Objetivo – A realização deste caso de uso apóia a etapa de Aquisição e Formalização do Conhecimento e requer que pelo menos dois atributos do sistema especialista em construção tenham sido especificados. O processamento deste caso de uso é realizado sobre uma base de conhecimento própria, desenvolvida com o objetivo de auxiliar na classificação, dentre os atributos do sistema, quais podem ser eventuais objetivos do SE. Para tanto, questões sobre cada atributo devem ser previamente respondidas pelo analista de conhecimento usuário do SEGSE.
- Especificar Dados do Projeto – Na especificação dos dados do projeto o analista de conhecimento deve preencher informações sobre projeto do SE em desenvolvimento, tais como nome, descrição e autor.
- Preencher Planilha de Conhecimento – Ainda como apoio à etapa de Aquisição e Formalização do Conhecimento, este caso de uso tem como objetivo permitir que o especialista sobre o domínio da aplicação do SE em construção indique seu parecer em exemplos gerados pelo SEGSE. Os exemplos são apresentados em uma planilha, formada por todas as combinações entre os valores de todos os atributos. Cada linha da planilha é uma combinação única dos valores dos atributos. Sendo assim, o número de linhas da planilha vai depender exclusivamente da quantidade de valores dos atributos existentes. O especialista deve selecionar um valor do atributo objetivo para cada linha da planilha. Ele deve escolher o valor que expressa a melhor resposta para a situação encontrada no conjunto de valores dos atributos em cada linha.
- Gerar Sistema Especialista – Este caso de uso tem como objetivo apoiar as etapas de Projeto e de Implementação do SE em construção. A geração do SE depende da realização de todos os casos de uso anteriores e inclui o caso de uso "Gerar Base de Conhecimento". De forma análoga ao KESAQ, a geração da base de conhecimento produz um conjunto de regras induzido a partir da planilha de conhecimento utilizando o algoritmo clássico ID3. Como saída, o SEGSE produz o código que implementa a arquitetura genérica de SEs apresentada na figura 3.1. O sistema produzido pode estão ser executado como protótipo do SE desejado.

As interfaces do SEGSE foram implementadas no mesmo padrão das interfaces dos sistemas especialistas gerados com auxílio da ferramenta.

Convém enfatizar que o conhecimento incorporado ao SEGSE para realização dos casos de uso mencionados anteriormente é configurável. Isto significa que diferentes versões do SEGSE podem ser produzidas por diferentes especialistas no desenvolvimento de sistemas de especialistas. Cada especialista pode, em função de suas experiências prévias, configurar atributos, valores e bases de conhecimentos manipulados pelo SEGSE. Essa configuração pode ser criada ou mesmo editada a partir

do próprio ambiente do SEGSE, uma vez que as estruturas de entrada e saída utilizadas pela ferramenta e pelos sistemas gerados são as mesmas.

3.6. Alguns Exemplos de SEs

Nesta seção, são descritas algumas aplicações de sistemas especialistas.

Conforme comentado no capítulo 1, o MYCIN foi um dos primeiros e mais conhecidos sistemas especialistas, sendo considerado um marco na IA. Foi criado inicialmente para diagnosticar e sugerir tratamento para doenças no sangue e desenvolvido na Universidade de Stanford, a partir de 1972. A principio era essencialmente um projeto acadêmico, mas influenciou enormemente a maioria dos sistemas especialistas que estavam por vir. Utilizava o encadeamento para trás (backward) a fim de descobrir que organismos estavam presentes no sangue, para, em seguida, aplicar o encadeamento para frente (forward) de forma a inferir sobre o regime de tratamento.

O META-DENDRAL, criado em 1978, foi o primeiro programa a usar técnicas de aprendizagem para construir automaticamente regras para um sistema especialista. Ele criava regras para serem usadas pelo DENDRAL, cuja tarefa, também comentada no capítulo1, era determinar a estrutura de compostos químicos complexos. O META-DENDRAL era capaz de induzir suas regras com base em um conjunto de dados sobre espectrometria de massa. Desse modo, ele conseguia identificar as estruturas moleculares com precisão bastante alta.

O PROSPECTOR foi um sistema especialista voltado ao aconselhamento sobre a exploração de minerais. Foi criado no final da década de 70 pela SRI International. É um sistema digno de ser mencionado pelas seguintes razões:

- Primeiro, ele se propunha a resolver problemas de extrema dificuldade, os quais até mesmo os melhores especialistas disponíveis (geólogos principalmente) não eram capazes de solucionar com um alto grau de certeza;
- Segundo, ele ilustra uma abordagem bastante diferente para sistemas especialistas pois envolvia a manipulação de probabilidades de ocorrência dos fatos.

São muitas as aplicações reais de sistemas especialistas, o que fornece um indicativo da tradição e da aplicabilidade desse tipo de tecnologia. Outros exemplos de sitemas especialistas podem ser obtidos, por exemplo, em (Chappetta, 2004), (Cazarotti e Menezes, 2004), (Goldschmidt, 2004), (Nunes, 2003), (Baião, 2002).

Capítulo

4

Lógica Nebulosa

4.1. Considerações Iniciais

Transformar conhecimento tácito em explícito, capturando a experiência e o conhecimento do especialista humano é o desafio da inteligência computacional. Colocar as regras de um negócio em uma Base de Conhecimento e explicar detalhadamente o raciocínio obtido para se chegar a um resultado é o objetivo de um sistema inteligente. A tarefa de aquisição de conhecimento voltada à construção de sistemas de apoio à decisão certamente não é uma tarefa trivial enfrentada por profissionais da área de Inteligência Artificial. Muitas vezes, o raciocínio dos especialistas envolve conceitos subjetivos, abstratos e imprecisos que os sistemas especialistas têm dificuldade de expressar.

A Lógica Nebulosa ou Difusa (Fuzzy Logic, em inglês) é uma teoria matemática que tem como principal objetivo permitir a modelagem do modo aproximado de raciocínio, imitando a habilidade humana de tomar decisões em ambientes de incerteza e imprecisão. Com conceitos e recursos da Lógica Nebulosa pode-se construir sistemas inteligentes de controle e suporte à decisão que lidem com informações imprecisas e subjetivas, tais como:

- Investimento de alto risco
- Pressão média
- Fluxo muito intenso
- Temperatura alta
- Muito jovem

São todas expressões lingüísticas cuja interpretação pode variar de um indivíduo para outro, sendo portanto, expressões nebulosas.

Como exemplos de aplicações industriais e comerciais da Lógica Nebulosa, podem ser citados:

- Aparelhos de Refrigeração
- Filmadoras
- Freios Antiderrapantes
- Sistema de Análise de Crédito
- Detecção de Fraude em Seguradoras
- Sistema de Análise de Investimentos
- Dentre muitos outros...

Para uma melhor compreensão dos fundamentos e das aplicações da Lógica Nebulosa, os principais conceitos serão definidos e comparados com conceitos da Lógica Clássica e da tradicional Teoria de Conjuntos.

4.2. Conjuntos Nebulosos

Existem três formas de se definir conjuntos na Teoria de Conjuntos:

a) Representação Explícita – Enumerando todos os elementos que pertencem ao conjunto.

Por exemplo: A= {-2,-1, 0, 1, 2}

b) Representação Implícita – Descrevendo uma lei de formação do conjunto, ou seja, indicando as propriedades dos elementos que pertencem ao conjunto.

Por exemplo, A= {X € Z / |x| ≤ 2}

c) Representação pela Função Característica – A função característica de um conjunto A, XA : U → {0,1}, é uma função que associa a cada elemento do universo de discurso U um valor binário. Este valor indica se o elemento pertence (1) ou não pertence (0) ao conjunto A.

$$X_A(x) \begin{cases} 0 \text{ se } x \notin A \\ 1 \text{ se } x \in A \end{cases}$$

O gráfico de função característica associada ao conjunto A dos exemplos anteriores encontra-se ilustrado na figura 4.1.

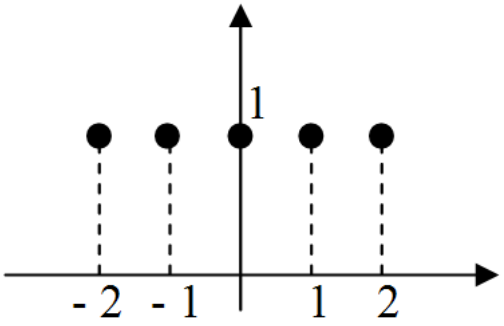


Figura 4.1. Gráfico da função característica do conjunto A

Segundo a Lógica Clássica, também denominada "crisp", uma sentença só pode assumir um dentre os valores verdade: Verdadeiro ou Falso. Analogamente, um elemento pertence ou não pertence a um conjunto. Não existe situação intermediária. Nesta teoria, os conjuntos e as regras são rígidos.

Consideremos os seguintes conjuntos como exemplos:

Muito Jovem= {x é pessoa/ idade (x) < 10}

Jovem= {x é pessoa/ 10 ≤ idade (x) < 40}

Velho= {x é pessoa/ 40 ≤ idade (x) < 60}

Muito Velho= {x é pessoa/ idade (x) ≥ 60}

José e João têm respectivamente 39 e 40 anos. Segundo os conceitos definidos pelos conjuntos acima, José é considerado jovem e João velho. Há uma mudança muito abrupta de um conceito (jovem) para o outro (velho).

A Lógica Nebulosa, assim como a Teoria dos Conjuntos Nebulosos, busca introduzir mecanismos que tornem mais suave a transição entre conceitos. Um deles é a função de pertinência μ_A: x →[0,1] que expressa o quanto um elemento "x" pertence ao conjunto "A". μ_A (x_0) é o grau de pertinência do elemento x_0 ao conjunto A. O grau de pertinência μ_A (x_0) indica o quão o valor x_0 é compatível com o conceito representado pelo conjunto A. Quanto mais próximo μ_A (x_0) for de 1, significa maior compatibilidade de x_0 com o conceito representado pelo conjunto A. Por outro lado, quanto menos compatível x_0 for em relação ao conceito A, menor será μ_A (x_0). A é um conjunto nebuloso.

A figura 4.2 mostra quatro exemplos de conjunto nebulosos construídos em função da variável "Idade". Todos os conjuntos apresentam formatos de trapézio.

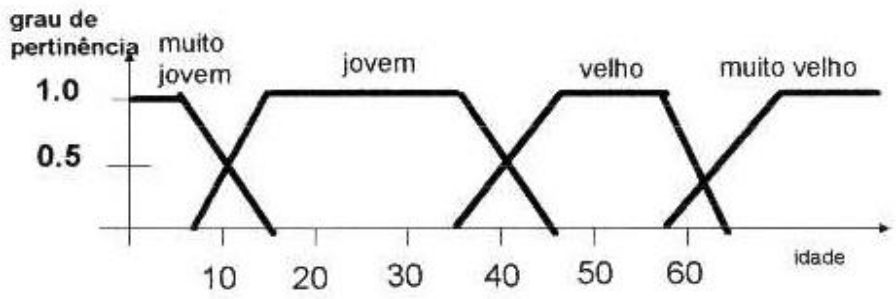


Figura 4.2. Exemplos de conjuntos nebulosos associados à variável Idade

Pode-se perceber pelo exemplo que uma pessoa que possua 40 anos, por exemplo, pertence ao mesmo tempo aos conjuntos jovem e velho, com graus de pertinência 0,65 e 0,45, respectivamente.

$\mu_{velho}(40) = 0{,}45$ $\mu_{muito\ jovem}(40) = 0$

$\mu_{jovem}(40) = 0{,}65$ $\mu_{muito\ velho}(40) = 0$

A pessoa em questão é simultaneamente jovem e velha. Pelos graus de pertinência, observa-se que ela não é nem tão jovem e nem tão velha.

4.3. Representação de Conjuntos Nebulosos

Existem diversos formatos para a representação de conjuntos nebulosos. A escolha de um determinado formato deve ser norteada pela compatibilidade do formato com o conceito que se deseja representar. Merecem destaque, no entanto, os formatos triangular e trapezoidal, que são muito utilizados em diversas aplicações da Lógica Nebulosa.

Uma função de pertinência triangular é expressa pela fórmula abaixo e representada graficamente na figura 4.3.

$$\mu(x) = \begin{cases} 0, & \text{se } x < a \\ (x-a)/(m-a), & \text{se } x \in [a, m[\\ (b-x)/(b-m), & \text{se } x \in [m, b] \\ 0, & \text{se } x > b \end{cases}$$

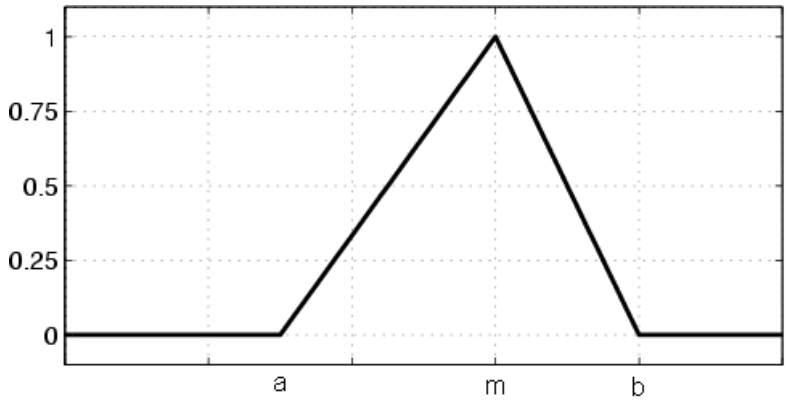


Figura 4.3. Função de pertinência triangular

Abaixo encontra-se a forma geral de uma função de pertinência trapezoidal, ilustrada na figura 4.4.

$$\mu(x) = \begin{cases} 0, se..x \le a \\ (x-a)/(b-a), se..x \in]a,b[\\ .....1, se..x \in .[b,c] \\ (d-x)/(d-c), se..x \in]c,d[\\ 0, se..x \ge d \end{cases}$$

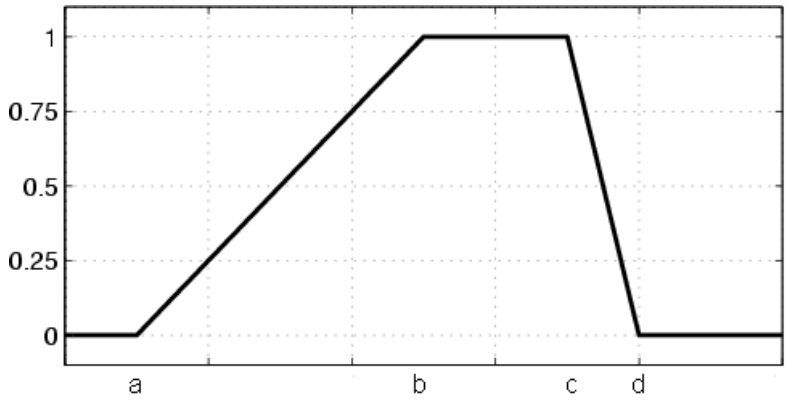


Figura 4.4. Gráfico de representação da pertinência trapezoidal

Uma função de pertinência gaussiana é expressa pela fórmula abaixo e representada graficamente na figura 4.5.

$$\mu(x) = e^{-\left(\frac{x-m}{2v}\right)^2}$$

Onde m é a média e v o desvio padrão da distribuição

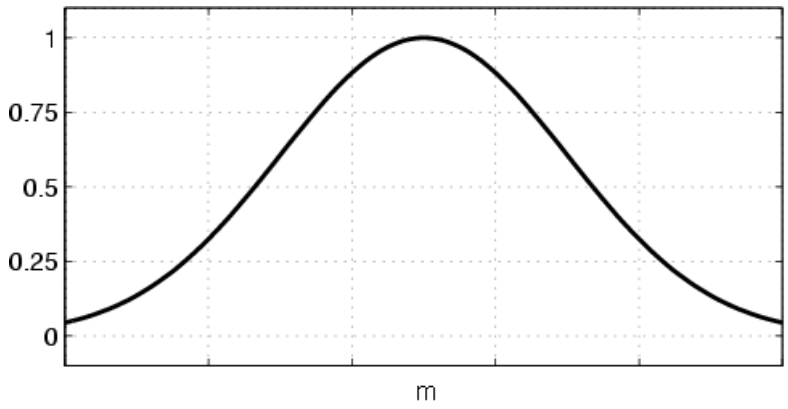


Figura 4.5. Função de pertinência gaussiana

Uma função de pertinência do tipo sino é expressa pela fórmula abaixo e representada graficamente na figura 4.6.

$$\mu(x) = \frac{1}{1 + \left| \frac{x - c}{a} \right|^{2b}}$$

Onde c é o centro da curva

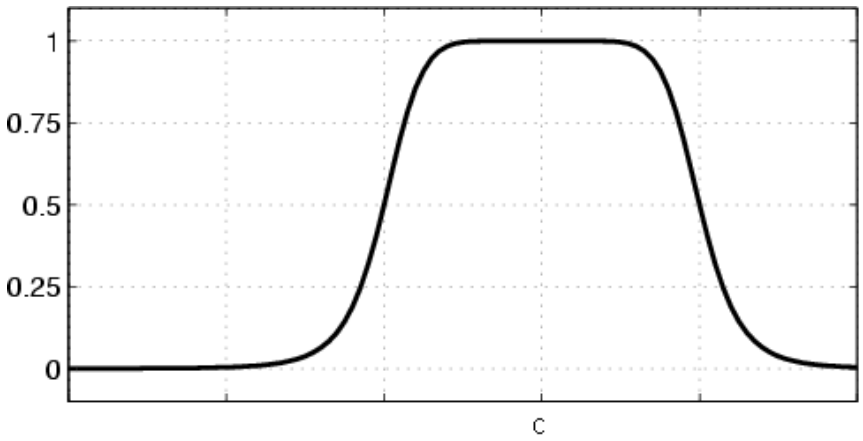


Figura 4.6. Função de pertinência do tipo sino

Uma função de pertinência sigmoidal é expressa pela fórmula abaixo e representada graficamente na figura 4.7.

$$\mu(x) = \frac{1}{1 + e^{-a(x-c)}}$$

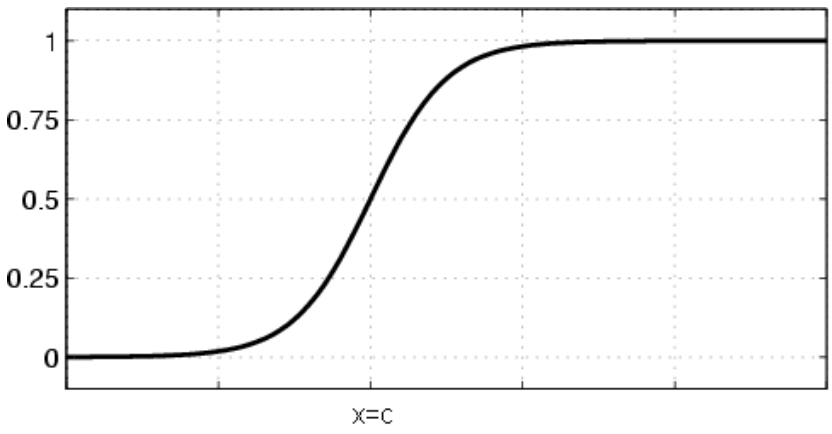


Figura 4.7. Função de pertinência sigmoidal

Uma função de pertinência singleton apenas um valor possui grau de pertinência 1. Os demais possuem grau de pertinência 0. A figura 4.8 ilustra graficamente essa função.

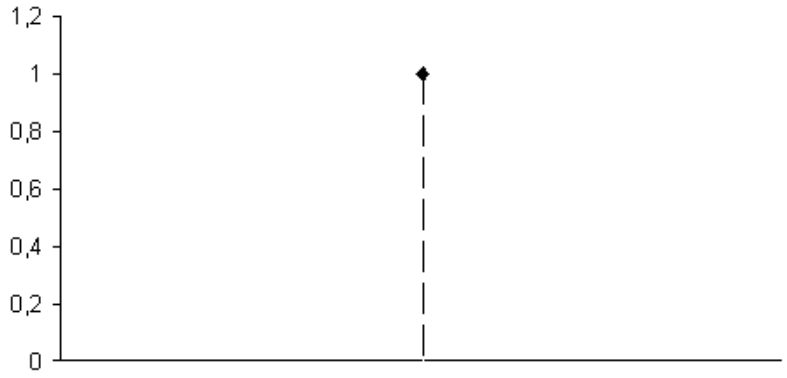


Figura 4.8. Função de pertinência singleton

4.4. Medidas de Imprecisão

Antes de prosseguirmos, torna-se necessário esclarecer a diferença entre probabilidade e grau de pertinência, duas medidas frequentemente utilizadas para expressar imprecisão.

Para facilitar a compreensão, consideremos a seguinte situação ilustrada graficamente na figura 4.9: Um turista deseja continuar seu caminho de modo seguro até a cidade mais próxima. Ao chegar em uma bifurcação encontra um poste com duas placas. A placa que aponta para o lado esquerdo tem escrito: "Caminho seguro com grau de pertinência 0.95"; e a placa que aponta para o lado direto tem os seguintes dizeres: "Caminho seguro com probabilidade de 95%". Pergunta-se: Qual o caminho que esse turista deve realmente escolher? Por que?

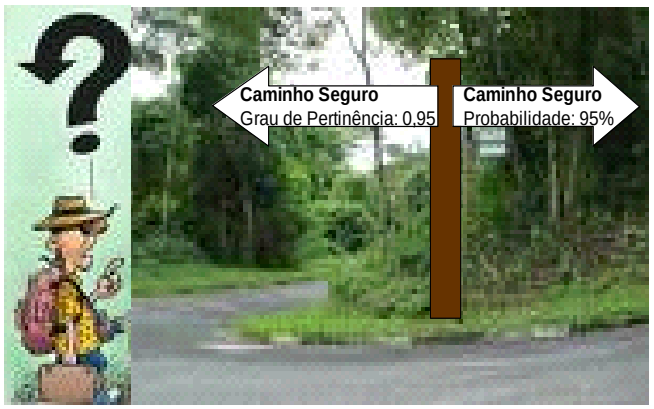


Figura 4.9. Qual deve ser a escolha do turista?

A escolha do turista deve ser pelo caminho da esquerda, conforme ilustra a figura 4.10. A justificativa para tal escolha é a seguinte: A probabilidade expressa as chances de que o caminho seja seguro, mas não se tem idéia do quanto seguro ele é. Por outro lado, o grau de pertinência expressa com certeza absoluta o quanto o caminho é seguro. Em um intervalo de 0 até 1, um grau de pertinência de 0,95 é um grau de altíssima compatibilidade com o conceito de segurança. Assim, considerando a margem de incerteza da probabilidade e a certeza da segurança expressa pelo alto grau de pertinência, conclui-se que, de fato, o caminho da esquerda é a escolha mais sensata.

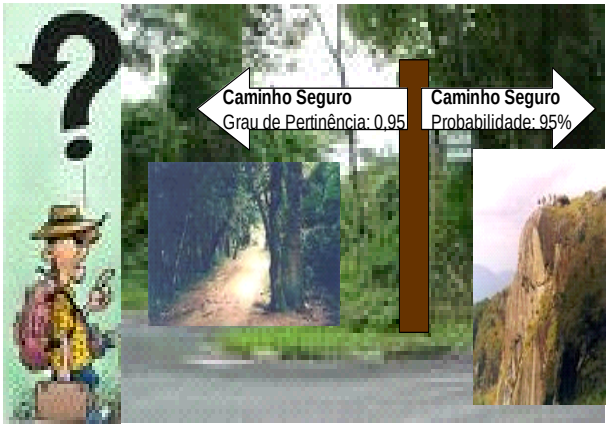


Figura 4.10. A escolha certa: o caminho da esquerda

4.5. Variáveis Lingüísticas

Uma variável lingüística é um objeto utilizado para representar de modo impreciso um conceito em um determinado problema. Seus valores são expressões lingüísticas subjetivas tais como quente, alto, jovem. Variáveis lingüísticas diferem das variáveis numéricas, que admitem apenas valores precisos. Os valores de uma variável lingüística são conjuntos nebulosos.

Exemplo: Variável Numérica : Temperatura = 40º

Variável Lingüística: Temperatura Alta

Os valores de uma variável lingüística definem uma estrutura de conhecimento denominada de partição nebulosa desta variável.

A figura 4.11 apresenta um exemplo de partição nebulosa possível para a variável temperatura.

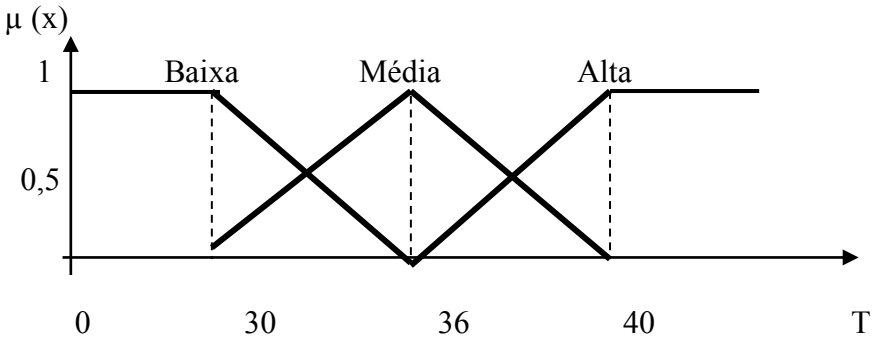


Figura 4.11. Exemplo de variável lingüística

4.6. Operadores Nebulosos

Nesta seção são apresentadas algumas operações sobre conjuntos nebulosos. Todas possuem correspondentes na clássica Teoria de Conjuntos.

Sejam A e B dois conjuntos nebulosos:

4.6.1. Pertinência do operador nebuloso de interseção

A função de pertinência do operador nebuloso de interseção, denominado T-Norm, pode ser definida de diversas maneiras. Abaixo estão três dos exemplos mais usuais:

$\mu_{A \cap B}(x) = \min \{\mu_A(x), \mu_B(x)\}$ (Mínimo – Zadeh)

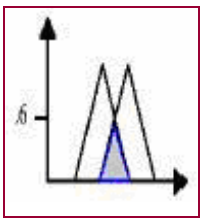


Figura 4.12. Ilustração gráfica do T-Norm de Zadeh

$\mu_{A \cap B}(x) = \mu_A(x) * \mu_B(x)$ (Produto)

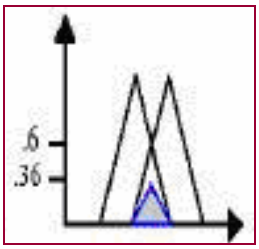


Figura 4.13. Ilustração gráfica do T-Norm do Produto

$$\mu_{A \cap B}(x) = \max\{0, \mu_A(x) + \mu_B(x) - 1\} \qquad \text{(Lukasiewicz)}$$

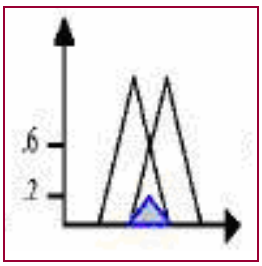


Figura 4.14. Ilustração gráfica do T-Norm de Lukasiewicz

Assim como na Lógica Clássica, a operação de interseção corresponde ao operador lógico de conjunção "E".

A idade de uma pessoa é de 40 anos, isso representa que ela é jovem, com grau de pertinência de 0.6 e velha com grau de pertinência de 0.4. Logo, utilizando o T-Norm de produto, tem-se que:

$$\mu_{jovem \cap velho}(40) = \mu_{jovem}(40) * \mu_{velho}(40) = 0.6 * 0.4 = 0.24$$

Estes valores representam que a pessoa em questão pertence ao conjunto de pessoas jovens e velhas com um grau de pertinência de 0.24.

Analogamente, utilizando o T-Norm de mínimo, tem-se que:

$$\mu_{jovem \cap velho}(40) = \min\{\mu_{jovem}(40), \mu_{velho}(40)\} = \min\{0.6, 0.4\} = 0.4$$

E, por fim, utilizando o T-Norm de Lukasiewicz, tem-se que:

$$\mu_{jovem \cap velho}(40) = \max\{0, \mu_A(40) + \mu_B(40) - 1\} = \max\{0, 0.6 + 0.4 - 1\} = 0$$

Generalizando, sejam a e b graus de pertinência a conjuntos nebulosos e T qualquer função T-Norm. Então T deve dispor das seguintes propriedades:

Comutatividade: T(a,b) =T(b,a)

Associatividade: T(a,T(b,c))=T(T(a,b),c)

Monotonicidade: $a \leq b$ e $c \leq d$, então $T(a,c) \leq T(b,d)$

Coerência nos contornos: $T(a,1) = a$ e $T(a,0) = 0$

4.6.2. Pertinência do operador nebuloso de união

A função de pertinência do operador nebuloso de união, denominado T-Conorm (ou S – norm), pode ser definida, por exemplo, como:

$\mu_{A \cup B}(x) = \max \{\mu_A(x), \mu_B(x)\}$ (Máximo)

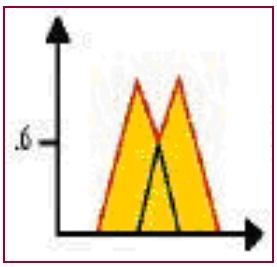


Figura 4.15. Ilustração gráfica do T-Conorm de Máximo

$\mu_{A \cup B}(x) = \min \{1, \mu_A(x) + \mu_B(x)\}$ (Soma Limitada)

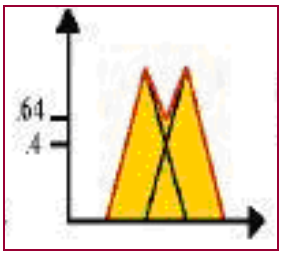


Figura 4.16. Ilustração gráfica do T-Conorm de Soma Limitada

Assim como na Lógica Clássica, a operação de união corresponde ao operador lógico de disjunção "OU".

Prosseguindo com o exemplo apresentado anteriormente em que uma pessoa de 40 anos é jovem, com grau de pertinência de 0.6 e velha com grau de pertinência de 0.4. Logo, utilizando o T-Conorm de máximo, tem-se que:

$$\mu_{jovem \cup velho}(40) = \max \{\mu_{jovem}(40), \mu_{velho}(40)\} = \max \{0.6, 0.4\} = 0.6$$

Estes valores representam que a pessoa em questão pertence ao conjunto de pessoas que são jovens ou velhas com um grau de pertinência de 0.6.

Analogamente, utilizando o T-Conorm de soma limitada, tem-se que:

$$\mu_{jovem \cup velho}(40) = \min \{1, \mu_{jovem}(40) + \mu_{velho}(40)\} = 1$$

Generalizando, sejam a e b graus de pertinência a conjuntos nebulosos e S qualquer função T-Conorm (ou S-Norm). Então devem ser válidas as seguintes propriedades:

Comutatividade: $S(a,b) = S(b,a)$

Associatividade: $S(a,S(b,c))=S(S(a,b),c)$

Monotonicidade: $a \leq b$ e $c \leq d$, então $S(a,c) \leq S(b,d)$

Coerência nos contornos: $S(a,1) = 1$ e $S(a,0) = a$

4.6.3. Pertinência do operador nebuloso de complemento

A função do operador nebuloso de complemento pode ser definida por:

$$\mu_{A'}(x)= 1 - \mu_A(x)$$

Utilizando ainda o exemplo anterior, o grau de pertinência ao conjunto não jovem de uma pessoa de 40 anos seria calculado por:

$$\mu_{\text{não jovem}}(40) = 1 - \mu_{\text{jovem}}(40) = 1 - 0.6 = 0.4$$

A operação de complemento corresponde ao operador lógico de negação "Não".

4.7. Regras Nebulosas

Uma das formas de representação nebulosa do conhecimento mais comum é a representação por meio de regras de produção nebulosas, ou simplesmente regras nebulosas. A estrutura geral de uma regra nebulosa é expressa por uma implicação do tipo:

SE < antecedente nebuloso> ENTÃO <conseqüente nebuloso>

O antecedente nebuloso é formado por condições nebulosas que, quando satisfeitas, determinam o processamento do conseqüente por um mecanismo de inferência nebulosa.

O conseqüente nebuloso, por sua vez, é composto por um conjunto de ações nebulosas ou diagnósticos nebulosos que são gerados a partir do disparo da regra.

Uma condição nebulosa, assim como uma ação ou diagnóstico nebuloso, é um predicado que envolve uma variável lingüística e um conjunto nebuloso.

Alguns exemplos de regras nebulosas:

SE *idade é meia-idade* E *pressão é baixa* ENTÃO *seguro é baixo*.

SE *idade é jovem* E *pressão é alta* ENTÃO *seguro é alto*.

4.8. Arquitetura Funcional de um Sistema de Inferência Nebuloso

Embora existam vários modelos de Inferência Nebulosa, por limitações de espaço será descrito aqui apenas o modelo Mamdani que foi, durante muitos anos, um padrão para a aplicação dos conceitos de Lógica Nebulosa em processamento de conhecimento.

Abaixo encontra-se a forma geral de uma regra típica do modelo Mamdani :

SE X1 = A1 e ... e Xn = An

ENTÃO Y1 = B1 e ... e Ym = Bm

A figura 4.17 apresenta a arquitetura funcional genérica de um sistema de inferência nebuloso.

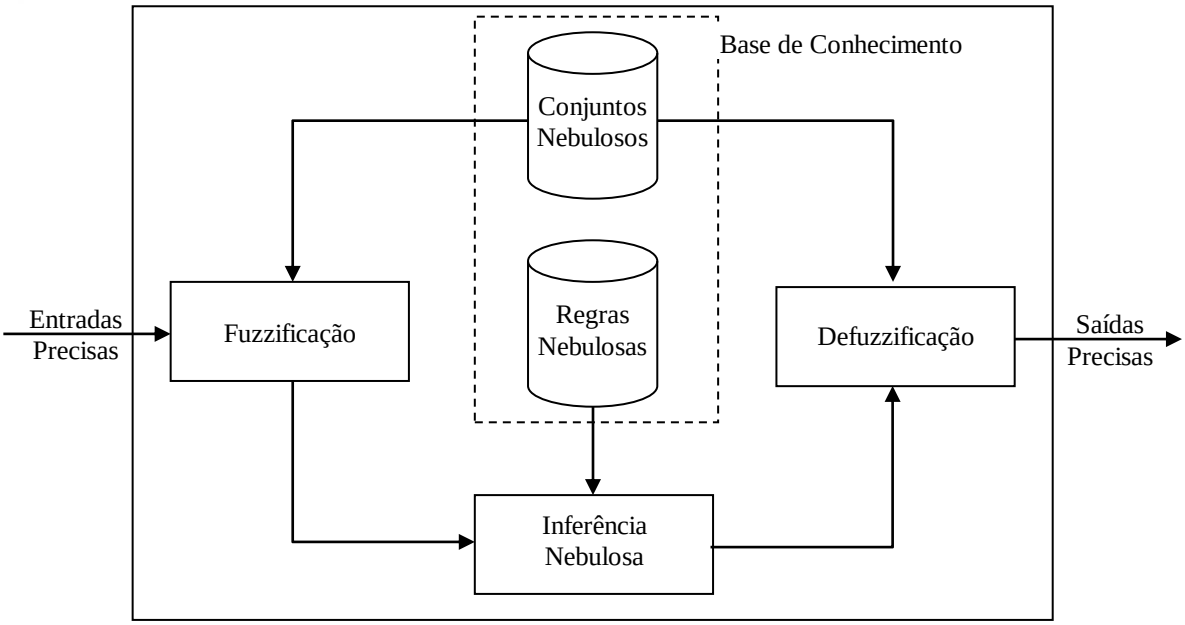


Figura 4.17. Arquitetura Funcional Genérica de um Sistema Nebuloso

Para melhor compreensão do funcionamento desta arquitetura, considere o seguinte exemplo de um sistema nebuloso para definição do valor de apólice de seguro de vida de clientes de uma seguradora.

a) Variáveis lingüísticas e respectivos conjuntos nebulosos:

Tabela 4.1. Variáveis lingüísticas do exemplo

Variáveis	Tipo	Conjuntos Nebulosos
Idade	Entrada	Meia Idade / Jovem
Pressão	Entrada	Alta / Baixa
Seguro	Saída	Alta / Baixa

Tabela 4.2. Conjuntos nebulosos (com representação tabular) do exemplo.

Idade	20	25	30	35	40	45	50	55	60	65
Meia-Idade	0.3	0.4	0.6	0.8	0.9	1.0	0.8	0.6	0.3	0.1
Jovem	0.9	0.8	0.7	0.6	0.4	0.3	0.1	0.0	0.0	0.0

Pressão Máx.	95	100	110	120	130	140	150	160	170	175
Pressão Mín.	50	55	60	65	70	75	80	85	90	100
Alta	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
Baixa	1.0	0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.2	0.1

Seguro	300	500	700	800	900	1000	1200
Alto	0.1	0.3	0.4	0.5	0.8	0.9	1.0
Baixo	1.0	0.9	0.6	0.5	0.3	0.1	0.1

b) Regras Nebulosas (apenas duas para simplificar o exemplo)

SE idade é meia-idade E pressão é baixa ENTÃO seguro é baixo.

SE idade é jovem E pressão é alta ENTÃO seguro é alto.

Deseja-se neste exemplo, determinar qual o valor da apólice de seguro a ser pago pelo cliente a partir dos valores de idade e de pressão deste cliente. As entradas e as saídas do sistema devem ser valores escalares, precisos. Considere que desejamos saber qual o valor da apólice de seguro a ser paga pelo cliente João de 35 anos e pressão (130,70).

O primeiro módulo da arquitetura (fuzzificação) é responsável por identificar os graus de pertinência dos valores de entrada em relação aos conjuntos nebulosos correspondentes.

No exemplo:

Idade	Pressão
$\mu_{\text{meia idade}}(35) = 0.8$	$\mu_{\text{Alta}}(130,70) = 0.5$
$\mu_{\text{jovem}}(35) = 0.6$	$\mu_{\text{Baixa}}(130,70) = 0.6$

Em seguida, o módulo de inferência nebulosa, baseado nos graus de pertinência identificados anteriormente, processa as regras nebulosas existentes na base de conhecimento.

O processamento de cada regra envolve o cálculo da interseção entre os conjuntos indicados no antecedente da regra. Este processo gera um grau de pertinência de disparo para cada regra.

No exemplo:

Regra 1:

SE idade é meia-idade (0.8) e pressão é baixa (0.6) ENTÃO seguro é baixo (grau de disparo da regra = Min {0,8;0,6}= 0,6)

Assim, por esta regra o seguro é baixo com grau de pertinência 0.6

Regra 2:

SE idade é jovem (0.6) e pressão é alta (0.5) ENTÃO seguro é alto

(grau de disparo da regra = Min{0.6, 0.5}=0.5)

Assim, por esta regra, o seguro é alto com grau de pertinência 0.5.

Finalizando, o processo de "defuzzificação" transforma os conjuntos acima em uma saída precisa. Para tanto, primeiro identifica o valor de seguro associado ao grau de disparo de cada regra. Convém destacar que o processo resultou em apenas um valor por conjunto pois as funções de pertinência de ambos os conjuntos são monotônicas, ou seja, um grau de pertinência não se encontra associado a mais do que um valor de seguro por conjunto.

Sendo: $\mu_{alto}(X)= 0.5$ e $\mu_{baixo}(Y)= 0.6$

Portanto: X=800 e Y=700

Em seguida, aplica-se a média ponderada aos valores de seguro indicados por cada regra e os respectivos graus de disparo:

$$\text{Seguro}= \frac{(0.5\text{x}800)+(0.6\text{x}700)}{0.5+0.6} = \frac{400+420}{1.1} = \frac{820}{1.1} = 745.45$$

Assim sendo, pelo sistema nebuloso exemplificado acima, um cliente de 35 anos com pressão 130x70 deve pagar uma apólice de seguro de R$ 745,45 (saída precisa).

Caso o leitor deseje, maiores detalhes sobre Lógica Nebulosa e Sistemas Nebulosos podem ser obtidos nas seguintes referências recomendadas: (Bojadziev & Bojadziev, 1997), (Pedrycz & Gomide, 1998) e (Rezende, 2003).

Capítulo

5

Redes Neurais Artificiais

5.1. Introdução

O estudo das redes neurais artificiais (RNAs) é algo fascinante e esse fascínio aumenta à medida que se tem mais a respeito do assunto. Trata-se de uma área de grande importância para a computação e suas aplicações, responsável pela solução de inúmeros problemas complexos. Este capítulo apresenta alguns dos principais conceitos básicos sobre o assunto.

5.1.1. Um breve histórico

Conforme comentado no capítulo 1, as primeiras informações sobre neurocomputação surgiram em 1943, em artigos do neurofisiologista Warren McCulloch, do MIT, e do matemático Walter Pitts da Universidade de Illinois. Em um primeiro trabalho sobre "neurônios formais", eles apresentaram um modelo matemático de simulação do processo biológico que ocorre em células nervosas vivas. O modelo procurava simular artificialmente o comportamento de um neurônio natural. Neste modelo, o neurônio artificial possuía apenas uma saída, produzida em função da soma dos valores de suas diversas entradas. Em ternos físicos, o trabalho consistia num modelo de resistores variáveis e amplificadores representando conexões sinápticas (sinapse é o nome dado à conexão existente entre neurônios) de um neurônio biológico.

O motivo pelo qual máquinas inspiradas na biologia são diferentes das máquinas atuais se encontra no fato de que as máquinas atuais baseiam seu processamento explicitamente em modelos algoritmicos, em que o comportamento diante de todas as situações possíveis deve ser previamente programado. Mecanismos de controle baseado em mecanismos neurais, entretanto, não são baseados em modelos algorítmicos. Utilizam cálculos matemáticos para efetuar suas operações, porém podem coordenar diversos graus de liberdade durante a execução de tarefas manipulativas e em ambientes desestruturados. Eles são capazes de lidar com tarefas complicadas sem que tenham que desenvolver um algoritmo específico para cada problema nem um modelo do ambiente em que operam.

Baseando-se nas características de seres biológicos, as pesquisas em RNAs e em Inteligência Computacional buscam por gerações completas de novos sistemas computacionais, muito mais eficientes e inteligentes que os sistemas atuais.

5.1.2. O que são redes neurais artificiais (RNAs)?

Em termos intuitivos, redes neurais artificiais (RNAs) são modelos matemáticos inspirados nos princípios de funcionamento dos neurônios biológicos e na estrutura do cérebro. Estes modelos têm capacidade de adquirir, armazenar e utilizar conhecimento

experimental e buscam simular computacionalmente habilidades humanas tais como aprendizado, generalização, associação e abstração.

O cérebro é tido como um processador altamente complexo e que realiza processamentos de maneira paralela. Para isso, ele organiza sua estrutura, ou seja, os neurônios, de forma que eles efetuem o processamento necessário. Isso é feito numa velocidade extremamente alta e ainda não existe um único computador no mundo capaz de realizar todas as tarefas que o cérebro humano desempenha.

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

A tabela 5.1 resume a relação entre características e comportamentos de RNAs com elementos da natureza.

Tabela 5.1 Comparação entre modelo natural e artificial

Modelo Natural	**Modelo Artificial**
Cérebro	RNA
Neurônio Biológico	Neurônio artificial/ elementos processadores
Rede de Neurônios	Estrutura em camadas
10 bilhões de Neurônios	Centenas/ milhares de neurônios
Aprendizado	Aprendizado
Generalização	Generalização
Associação	Associação
Reconhecimento de Padrões	Reconhecimento de Padrões

Segundo Hecht-Nielsen (1990), uma RNA pode ser formalmente definida como: *"uma estrutura que processa informação de forma paralela e distribuída e que consiste de unidades computacionais (as quais podem possuir memória local e podem executar operações locais) interconectadas por canais unidirecionais chamados de conexões. Cada unidade possui uma única conexão de saída, que pode ser dividida em quantas conexões laterais se fizer necessário, sendo que cada uma destas conexões transporta o mesmo sinal (sinal de saída da unidade). Esse sinal de saída pode ser contínuo ou*

discreto. O processamento executado por cada unidade pode ser definido arbitrariamente, com a restrição de que ele deve ser completamente local, isto é, deve depender somente dos valores atuais dos sinais de entrada que chegam até a unidade via as conexões e dos valores armazenados na memória local da unidade computacional."

A figura 5.1 apresenta uma ilustração comparative entre o modelo biologico e o artificial adotado pelas RNAs. Mais a diante, os elementos de um neurônio artificial serão apresentados com detalhes.

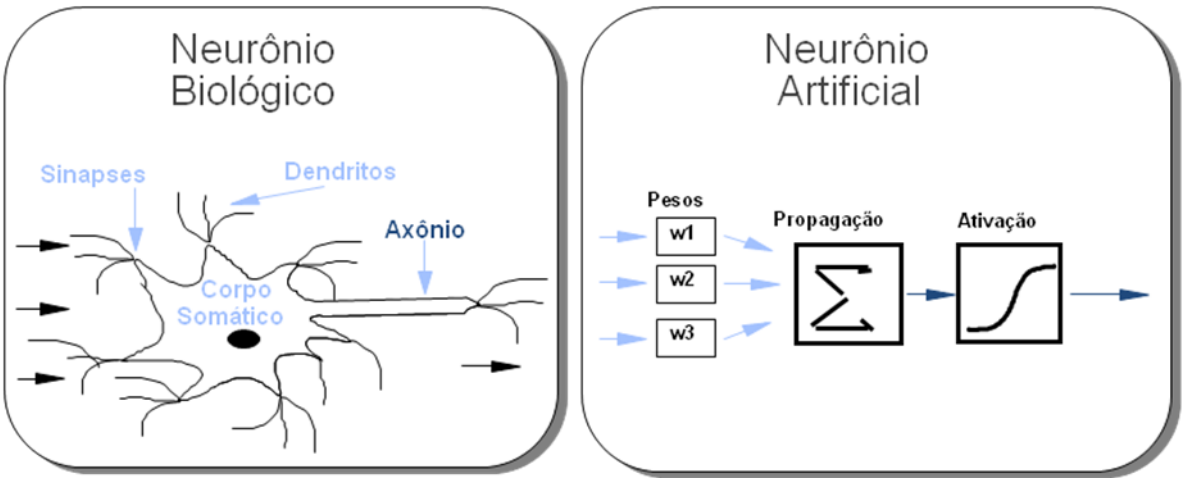


Figura 5.1: Analogia entre os modelos de um neurônio: biológico e artificial

5.1.3. Características de uma RNA

Devido à sua similaridade com a estrutura do cérebro, as RNAs apresentam algumas características similares às do comportamento humano, tais como:

a) Busca paralela e endereçamento pelo conteúdo

O cérebro não possui endereço de memória. Analogamente, nas RNAs o conhecimento fica distribuído pela estrutura das redes, de forma que a procura pela informação ocorre de forma paralela e não seqüencial.

b) Aprendizado por Experiência

As RNAs tentam aprender padrões diretamente a partir dos dados. Para isso, utilizam um processo de repetidas apresentações dos dados à rede que busca abstrair modelos de conhecimento de forma automática. Este processo é denominado *aprendizado*, e é implementado por um *algoritmo de aprendizado*.

c) Generalização

As RNAs são capazes de generalizar seu conhecimento a partir de exemplos anteriores. A capacidade de generalização permite que RNAs lidem com ruídos

e distorções nos dados, respondendo corretamente a novos padrões. A figura 5.2 apresenta um exemplo do conceito de generalização.

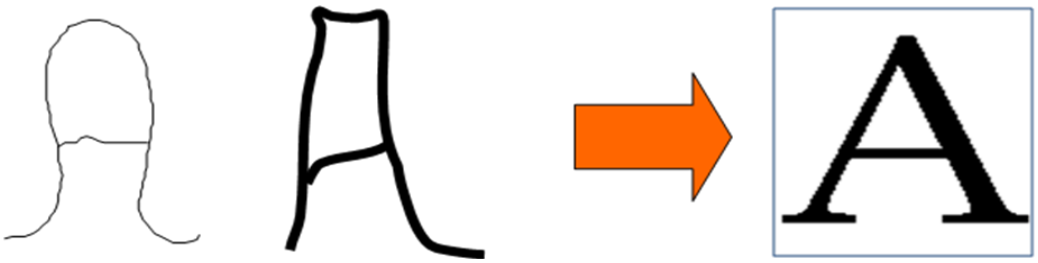


Figura 5.2: Exemplo ilustrativo do conceito de generalização

d) Associação

As RNAS são capazes de estabelecer relações entre padrões de natureza distinta. Por exemplo, identificar pessoas a partir de características da voz destas pessoas.

e) Abstração

Abstração é a capacidade das RNAs em identificar a essência de um conjunto de dados de entrada. Isto significa que as RNAs são capazes de perceber quais as características relevantes em um conjunto de entradas. Assim sendo, a partir de padrões ruidosos as RNAS podem extrair as informações dos padrões sem ruído. A figura 5.3 ilustra o conceito de abstração.

Figura 5.3: Exemplo ilustrativo do conceito de abstração

f) Robustez e Degradação Gradual

Como a informação fica distribuída em uma RNA, a perda de um conjunto de neurônios artificiais não causa necessariamente o mau funcionamento desta rede. Na realidade, o desempenho de uma RNA tende a diminuir gradativamente na medida em que aumenta a quantidade de neurônios artificiais inoperantes.

g) Não Programáveis

As RNAs são construídas e não requerem programação. Dado um problema, a rede deve ser modelada segundo as entradas e saídas envolvidas e um algoritmo de aprendizado, programado previamente é aplicado sobre o modelo e sobre dados históricos, buscando mapear corretamente as entradas da rede nas saídas correspondentes.

h) Soluções Aproximadas

Muitas vezes, as RNAs não produzem a melhor solução para um problema, gerando, no entanto, soluções aproximadas e aceitáveis. Alguns modelos neurais trabalham com o conceito de minimização de erro, nem sempre reduzido ao patamar nulo. Cabe ressaltar que, de forma análoga ao comportamento humano, mesmo treinadas, as RNAs são suscetíveis a geração de soluções incorretas.

Semelhante ao sistema biológico, uma RNA possui, simplificadamente, um sistema de neurônios, e conexões ponderadas por valores reais denominados pesos. Numa RNA os neurônios são arrumados em camadas, com conexões entre elas. A figura 5.2 ilustra graficamente a arquitetura de uma RNA simples. Os círculos representam os neurônios e as linhas representam os pesos das conexões. Por convenção, a camada que recebe os dados é chamada camada de entrada e a camada que mostra o resultado é chamada de camada de saída. A camada interna, onde ocorre o processamento interno da rede é tradicionalmente chamada de camada escondida. Uma RNA pode conter uma ou várias camadas escondidas, de acordo com a complexidade do problema.

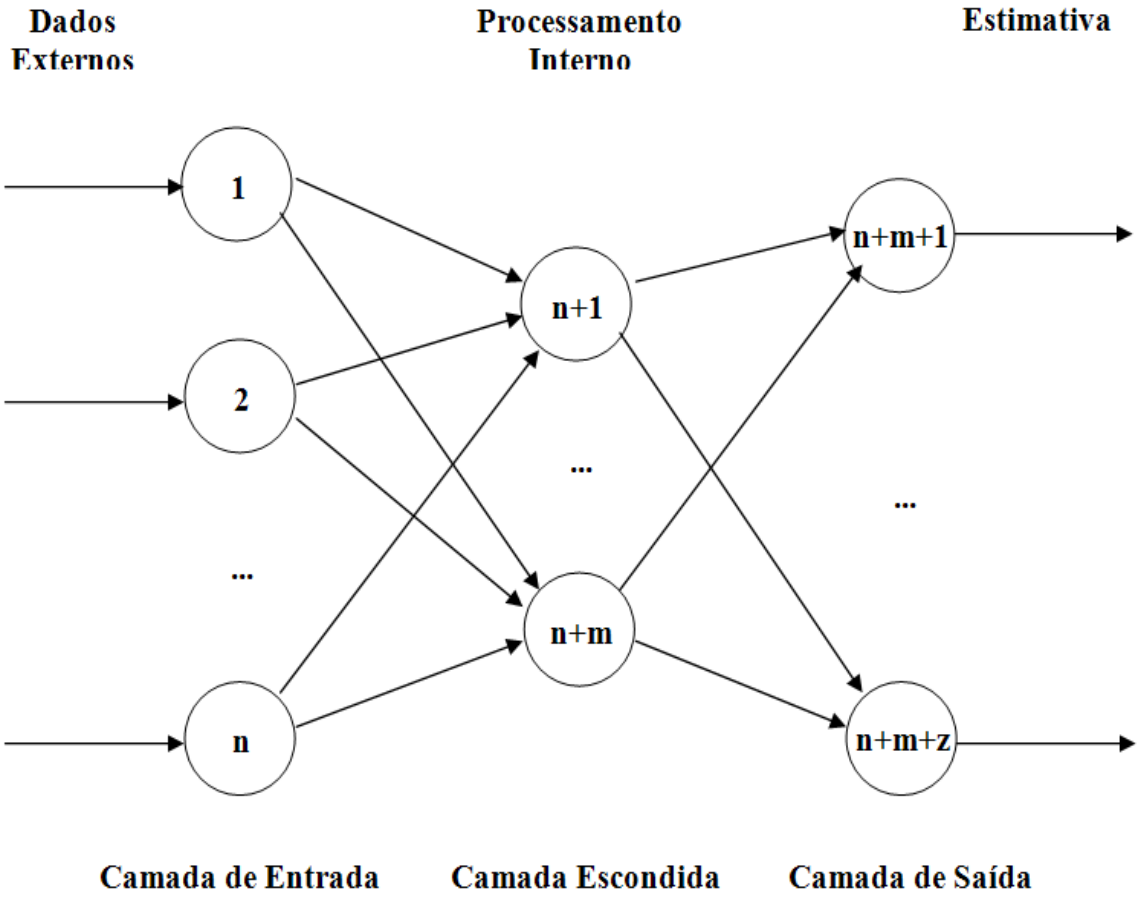


Figura 5.4 Arquitetura de uma RNA simples

Em geral o processamento das RNAs ocorre da esquerda para a direita (RNAs feed forward). Para fins computacionais os neurônios são rotulados com uma numeração seqüencial de cima para baixo, da esquerda para a direita, conforme ilustrado na figura 5.4.

A transmissão do sinal de uma célula para outra é um complexo processo químico, no qual substâncias específicas são liberadas pelo neurônio transmissor. O efeito é um aumento ou uma queda no potencial elétrico no corpo da célula receptora. Se este potencial alcançar o limite de ativação da célula, um pulso ou uma ação de potência e duração fixa é enviado para outros neurônios. Diz-se então que o neurônio está ativo.

O neurônio artificial foi projetado para imitar algumas das principais características de um neurônio biológico. Essencialmente, as entradas são aplicadas a um neurônio artificial, cada uma representando a saída de outros neurônios conectados a ele (vide figura 5.4). Cada entrada é multiplicada por um peso correspondente (W_{ij}), gerando entradas ponderadas, de forma análoga à força das sinapses. Em seguida todas estas entradas ponderadas são somadas, obtendo-se um valor NET (potencial de ativação do neurônio artificial) que será comparado com o valor limite para ativação do neurônio (F). Caso este valor alcance o valor limite de ativação do neurônio, ele será ativado.

Caso contrário ele permanecerá inativo. A figura 5.5 mostra o modelo que implementa esta idéia:

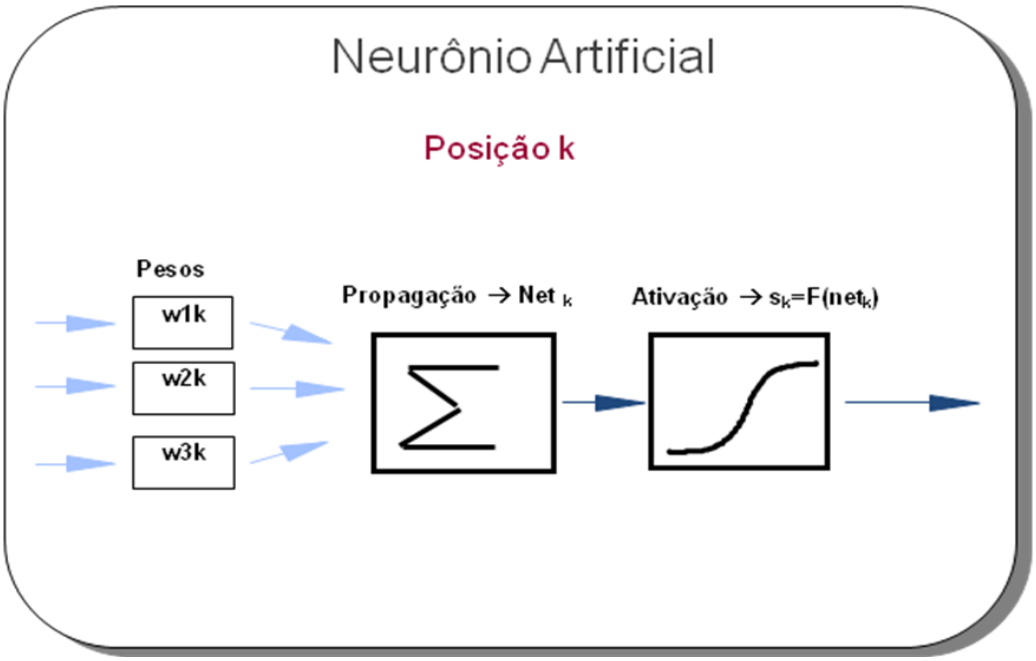


Figura 5.5: Estrutura Interna de um Neurônio Artificial

5.2. Modelagem

Conforme pode ser observado na figura acima, em essência, os elementos básicos de um neurônio artificial são:

- **Conexões entre os Processadores**

  A cada conexão de entrada de um neurônio artificial, existe um valore real, denominado peso sináptico, que determina o efeito desta entrada sobre o neurônio em questão. Quanto maior / menor este valor, maior a influência positiva / negativa do neurônio de onde sai a referida conexão. De forma análoga, a cada conexão de saída há um peso sináptico que determina a influência deste neurônio sobre o neurônio de chegada desta conexão.

  W_{ik} é uma notação para pesos sinápticos utilizada com freqüência, onde *i* é a identificação do neurônio de saída e *k*, a identificação do nerônio de chegada. A figura 5.6 ilustra esta notação.

Figura 5.6: Conexão e peso sináptico entre os neurônios *i* e *k*

Alguns autores representam os pesos sinápticos trocando na notação a ordem entre os neurônios de partida e chegada. Assim sendo, nesta notação alternativa a força da conexão entre os neurônios *i* e *k* da figura 5.6 seria representada por W_{ki}.

Na figura 5.7, são apresentados dois exemplos de pesos sinápticos, sendo um positivo e o outro negativo.

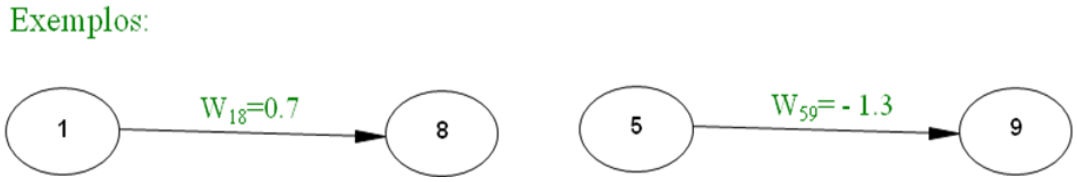


Figura 5.7: Exemplos de pesos sinápticos

- **Regra de Propagação**

  É a forma com que os estímulos (as saídas) provenientes de outros neurônios artificiais são combinados aos pesos sinápticos correspondentes para compor o potencial de ativação de um neurônio (por exemplo, o neurônio *k*). A regra de propagação estabelece, portanto, o potencial de ativação do neurônio (net_k). Uma regra de propagação muito utilizada é o produto escalar entre o vetor de entrada e o vetor de pesos, conforme ilustrado na figura 5.8.

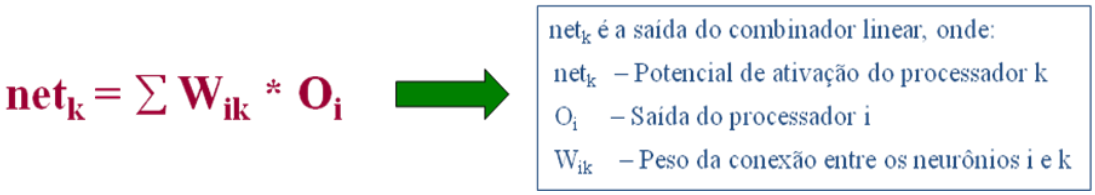


Figura 5.8: Exemplo de Regra de Propagação

Convém reparar que o somatório varia em função da quantidade de neurônios conectados ao neurônio *k*, conforme ilustra a figura 5.9.

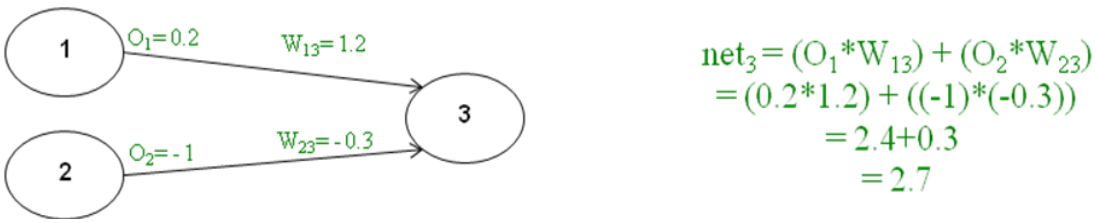


Figura 5.9: Exemplo de aplicação da regra de propagação da figura 5.8

- **Função de Ativação**

  A função de ativação de um neurônio artificial determina o novo valor do estado de ativação deste neurônio, a partir de seu potencial de ativação net_k. Determina a saída

efetiva de um neurônio artificial. Notação: $S_k = F(net_k + b_k)$. A função F é denominada função de ativação do neurônio. A saída de um neurônio *k*, representada por S_k, pode muitas vezes, ser referenciada por O_k ($S_k = O_k$). A constante b_k, denominada bias, tem o efeito de aumentar ou diminuir a entrada líquida da função de ativação, dependendo se ele é positivo ou negativo, respectivamente, conforme ilustra a figura 5.10.

$v_k = u_k + b_k$, onde $u_k = net_k$

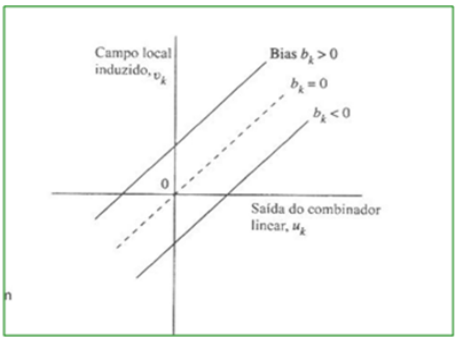


Figura 5.10: Exemplo de contribuição da constante bias

A função de ativação de um neurônio artificial pode assumir várias formas, sendo algumas delas expostas a seguir.

a) Linear – a saída do neurônio é expressa conforme indicado na figura 5.11.

$$y = a \times net$$

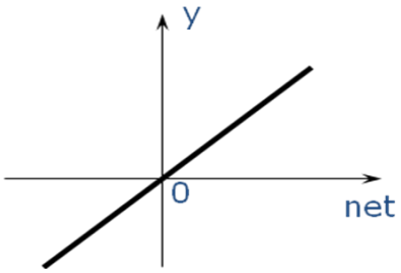


Figura 5.11: Função de ativação linear

b) Rampa (ou Linear por Partes) – Conforme o próprio nome sugere, conjuga funções lineares a partir da definição do parâmetro *a*. O gráfico da função rampa encontra-se ilustrado na figura 5.12.

$$y = \begin{cases} a \ \ se \ \ net \geq a \\ net \ \ se \ \ -a < net < a \\ -a \ \ se \ \ net \leq -a \end{cases}$$

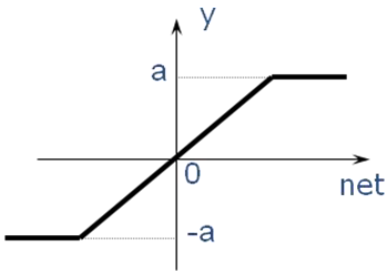


Figura 5.12: Função de ativação rampa

c) Degrau – Também chamada de onda quadrada binária, encontra-se ilustrada na figura 5.13. A figura 5.14 apresenta uma variação da função degrau denominada degrau bipolar.

$$y = \begin{cases} 1 \ \ se \ \ net > 0 \\ 0 \ \ se \ \ net \leq 0 \end{cases}$$

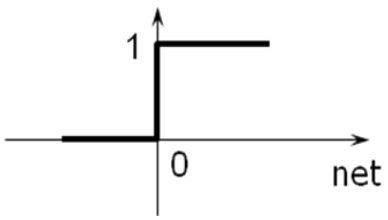


Figura 5.13: Função de ativação degrau

$$y = \begin{cases} 1 \ \ se \ \ net > 0 \\ -1 \ \ se \ \ net \leq 0 \end{cases}$$

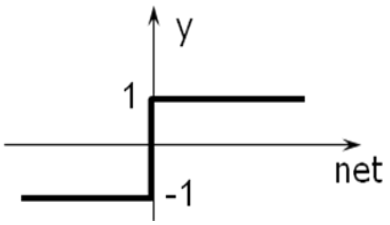


Figura 5.14: Função de ativação degrau bipolar

d) Sigmóide – Trata-se de uma das funções de ativação mais utilizadas na construção de redes neurais. É contínua e diferenciável em todos os pontos, além de assintótica em relação a 0 e 1. A figura 5.15 mostra a função e seu gráfico. Nela pode-se perceber que quanto maior o valor do parâmetro *a*, maior a inclinação na sigmóide.

$$y = \frac{1}{1 + e^{(-a \times net + b)}}$$

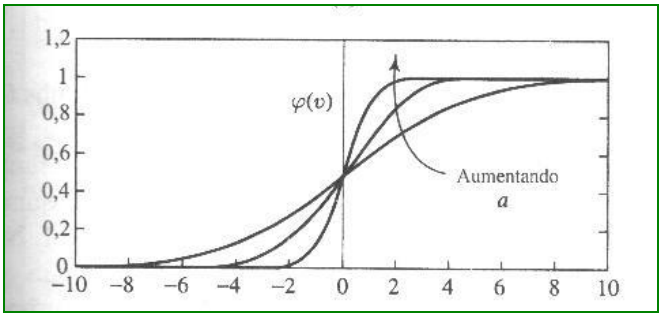


Figura 5.15: Função de ativação sigmoidal

e) Tangente Hiperbólica – Função também diferenciável, assintótica em relação aos valores -1 e 1, conforme ilustra a figura 5.16.

$$y = \frac{2}{(1 + e^{(-2a \times net + b)}) - 1}$$

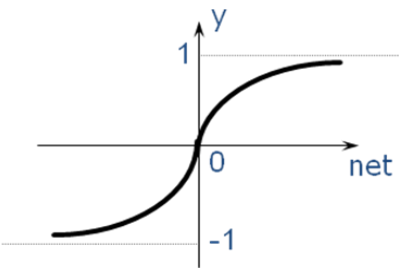


Figura 5.16: Função de ativação tangente hiperbólica

5.3. Arquiteturas de RNAs

É interessante ressaltar que o modelo simples de neurônio artificial apresentado ignora diversas características do neurônio natural, tais como a não consideração dos atrasos de tempo que afetam a dinâmica do sistema – as entradas produzem saídas imediatas – e a não inclusão dos efeitos de sincronismo ou de modulação de freqüência – característica que alguns pesquisadores acham de fundamental importância. Apesar destas limitações, as RNAs formadas por simples neurônios artificiais possuem atributos semelhantes aos do sistema biológico, como a capacidade de aprendizado e generalização, podendo-se dizer que a essência do funcionamento do neurônio natural tenha sido absorvida. Assim sendo, vale a pena destacar uma comparação entre as topologias de redes neurais recorrentes e não recorrentes.

As RNAs não-recorrentes são aquelas que não possuem realimentação de suas saídas para suas entradas e por isso são também ditas "sem memória". A estrutura das RNAs ditas não-recorrentes é em camadas, podendo estas RNAs serem formadas por uma (RNA de camada única) ou mais camadas (RNA multi-camada). Redes neurais multi-camadas contêm um conjunto de neurônios de entrada, uma camada de saída e uma ou mais camadas escondidas. A camada de entrada da rede apenas distribui os padrões. A

camada de saída apresenta o resultado final do processamento da rede. As RNAs de uma camada são também chamadas de "perceptrons", e possuem um espectro de representações limitado. As RNAs multi-camadas, por suprirem as deficiências das redes de uma única camada, possuem uma gama maior de aplicações bem sucedidas.

RNAs recorrentes são estruturas de processamento capazes de representar uma grande variedade de comportamentos dinâmicos. A presença de realimentação de informação permite a criação de representações internas e dispositivos de memória capazes de processar e armazenar informações temporais e sinais seqüenciais. A presença de conexões recorrentes ou realimentação de informação pode conduzir a comportamentos complexos, mesmo com um número reduzido de parâmetros.

Conforme exposto acima, pode-se perceber que existem algumas classificações utilizadas na caracterização da arquitetura ou toplogia das redes neurais artificiais. Cada uma delas enfoca um aspecto distinto. Os itens a seguir procuram sumarizar algumas detas classificações, apresentando alguns exemplos ilustrativos.

- Classificação quanto ao número de camadas

  - Redes de Camada Única – Conforme o próprio nome sugere, são redes que possuem apenas uma camada de neurônios artificiais.

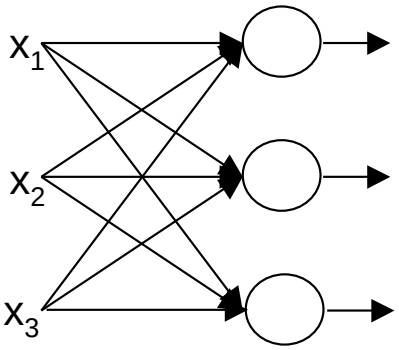


Figura 5.17: Exemplo de rede de camada única

  - Redes de Múltiplas Camadas – Enquadram-se nesta situação as redes que possuem mais de uma camada de elementos processadores.

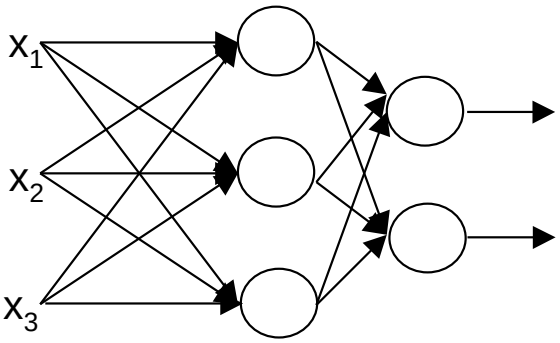


Figura 5.18: Exemplo de rede de múltiplas camadas (2 camadas neste caso)

- Classificação quanto ao tipo de conexão

  - Redes Feedforward (Acíclicas ou Não Recorrentes) – O fluxo de processamento da informação ocorre da esquerda para a direita. Neste caso, não há retorno de sinal para camadas anteriores.

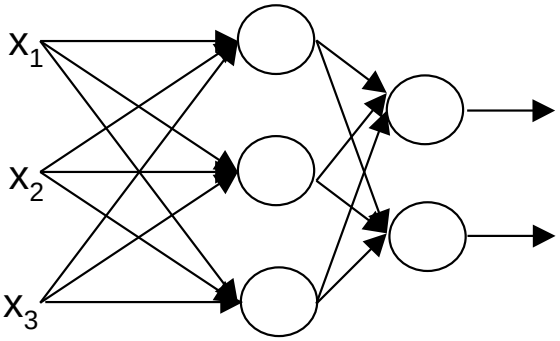


Figura 5.19: Exemplo de rede feedforward

  - Redes Feedback (Cíclicas ou Recorrentes) – Este tipo de rede se caracteriza pelo fato de haver retorno do sinal, em sentido contrário ao fluxo de processamento normal da rede.

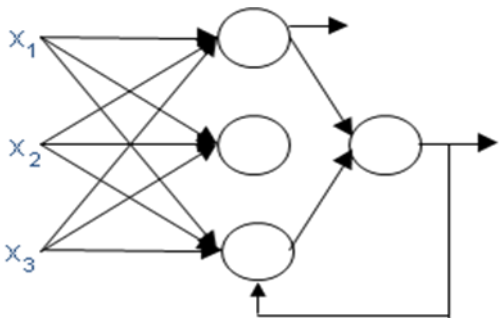


Figura 5.20: Exemplo de rede recorrente

- Redes com Recorrência Autoassociativa – Neste caso a saída de cada neurônio serve como entrada para todos os outros processadores.

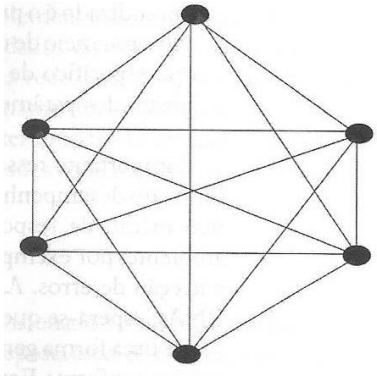


Figura 5.20: Exemplo de rede com recorrência autoassociativa

- Classificação quanto ao tipo de conectividade

  - Redes Parcialmente Conectadas – Neste caso, há neurônios em uma camada que não se conectam a todos os neurônios da camada seguinte.

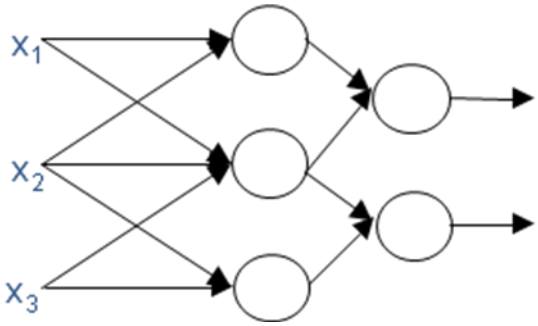


Figura 5.21: Exemplo de rede parcialmente conectada

  - Redes Completamente Conectadas – Nesta classe, encontram-se as redes neurais em que todos os neurônios de uma dada camada se conectam a todos os neurônios da camada seguinte.

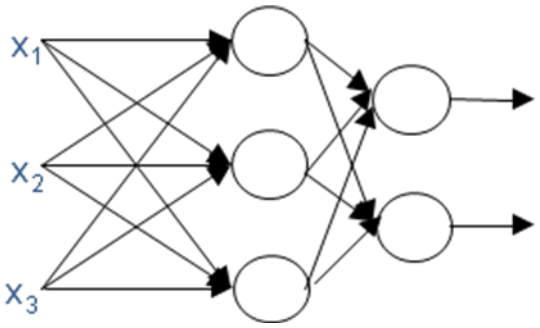


Figura 5.22: Exemplo de rede completamente conectada

5.4. Processamento Neural

O processamento em uma rede neural artificial pode ser dividido em duas fases distintas:

- Aprendizado (*Learning*) ou Fase de Treinamento – Nesta fase, ocorre o processo de atualização dos pesos sinápticos para a aquisição do conhecimento. Os pesos são atualizados conforme o algoritmo de aprendizado escolhido é aplicado sobre os dados históricos disponíveis. *"Aprendizado é o processo pelo qual os parâmetros livres de uma rede neural (pesos) são adaptados por meio de uma forma continuada de estímulo pelo ambiente externo, sendo o tipo específico de aprendizado definido pela maneira particular como ocorrem os ajustes dos parâmetros livres."*

  De forma genérica, o valor do vetor de pesos w(t+1) no instante t+1 pode ser escrito como: $\mathbf{w}(t+1) = \mathbf{w}(t) + \delta\mathbf{w}(t)$, onde $\delta\mathbf{w}(t)$ é o ajuste aplicado aos pesos.

  Convém enfatizar que os algoritmos de aprendizado diferem, basicamente, na forma como $\delta w(t)$ é calculado.

  Basicamente, o processo de aprendizado possui os seguintes passos:

  - ✓ A rede neural é estimulada pelo ambiente ao receber um padrão de entrada retirado de um conjunto histórico de padrões ou dados.
  - ✓ A rede neural sofre modificações em seus parâmetros livres (pesos sinápticos)
  - ✓ A rede neural responde de uma nova maneira ao ambiente

  Estes passos se repetem até que algum critério de parada seja alcançado. São exemplos de critérios de parada:

  - ✓ Número de iterações máximo alcançado – é o número máximo de vezes que um padrão ou conjunto de padrões é apresentado a rede neural.
  - ✓ Erro produzido pela rede atinge um patamar abaixo de limiar definido – Neste caso, a medida do erro produzido pela RNA a cada estímulo de entrada é considerada como referência para interrupção do treinamento. Tal interrupção deve ocorrer sempre que o erro produzido pela rede atingir um valor considerado suficientemente pequeno.

A participação do usuário, responsável pela utilização da RNA na solução de um problema, é necessária para escolher os parâmetros a serem utilizados em ambos os critérios de parada mencionados. Assim, tanto o número de iterações máximo quanto a margem de erro suficientemente pequena são definidos pelo usuário.

- Aplicação (Recall), Teste ou Recuperação da Informação – Esta fase depende da conclusão da fase de aprendizado. Isto significa que uma rede neural somente deve ser aplicada na solução de um problema após ter passado pela fase de aprendizado. A fase de aplicação consiste em calcular a saída da rede a partir da apresentação de um padrão de entrada. Na fase de aplicação não há atualização dos pesos sinápticos. Estes pesos são utilizados no cálculo da saída da rede produzida em função dos estímulos de entrada.

De todas as propriedades interessantes das redes neurais artificiais, nenhuma captura tão bem a habilidade humana de aprendizado. Ao invés de especificar todos os detalhes de uma computação, tem-se a possibilidade de treinar uma rede para fazer esta computação. Isto significa que as redes neurais podem tratar problemas onde regras apropriadas não são conhecidas previamente, devendo estas regras ser abstraídas a partir dos dados.

O objetivo do treinamento de uma RNA é fazer com que a aplicação de um conjunto de entradas produza um conjunto de saídas desejado ou, no mínimo, um conjunto de saídas consistentes. Simplificadamente cada conjunto de entrada ou saída pode ser chamado de vetor. O treinamento é realizado pela aplicação seqüencial dos vetores de entradas (e em alguns casos também os de saída), enquanto os pesos da rede são ajustados de acordo com um procedimento de treinamento pré-determinado (algoritmo de treinamento). Durante o treinamento de uma rede, espera-se que os pesos gradualmente convirjam para determinados valores, tal que a aplicação dos vetores de entrada produza as saídas desejadas (ou consistentes).

Os procedimentos de treinamento que levam as RNAs a aprender determinadas tarefas podem ser classificados em três classes de treinamento:

- **Supervisionado:** Neste tipo de aprendizado, a rede neural recebe um conjunto de entradas padronizadas e seus correspondentes padrões de saída esperados, onde ocorrem ajustes nos pesos sinápticos até que o erro entre os padrões de saída gerados pela rede e os padrões desejados fique abaixo de um limite máximo de tolerância de erro especificado pelo usuário. Em outras palavras, a cada par (entrada, saída desejada), um "professor" compara a saída desejada com a saída produzida pelo sistema, calculando um erro que é usado no ajuste dos pesos da rede a fim de minimizar tal erro.

  Procura minimizar a diferença entre a soma ponderada das entradas pelos pesos e a saída desejada: $e(t) = d(t) - y(t) \qquad W(t+1) = W(t) + \eta e(t)X(t)$

  Onde η é a taxa de aprendizado, $e(t)$ é uma medida de erro e $X(t)$ entrada do neurônio, ambas no instante t do processo de aprendizado.

  A aprendizagem por correção de erro envolve a minimização da soma dos erros quadráticos das saídas:

$$\varepsilon^2 = 1/2 \sum_{i=1}^{p} (y_d^i - y)^2$$

Onde:

- ✓ p é o número de exemplos
- ✓ y_d^i é a saída desejada para o vetor de entrada x_i e y é a saída corrente da rede para o vetor x_i.

Portanto, o conjunto de dados formado pelos pares de entrada e saída (x_i, y_d^i) define a superfície de erro.

Para cada valor possível de w, a soma dos erros quadráticos do conjunto de dados é calculada e um valor de erro é obtido.

A superfície formada por todos os valores de erro resulta na superfície de erro para o conjunto de dados.

O valor de w que minimiza o erro corresponde à solução de erro mínimo, ou mínimo global associado ao problema.

Conforme mencionado, o aprendizado (ou treinamento) supervisionado necessita de um par de vetores composto do vetor de entrada e do vetor alvo que se deseja como saída. Juntos, estes vetores são chamados de par de treinamento ou vetor de treinamento, sendo interessante ressaltar que geralmente a rede é treinada com vários vetores de treinamento.

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

- **Não-supervisionado:** Neste tipo de aprendizado, a rede neural trabalha os dados de forma a determinar algumas propriedades dos conjuntos de dados. A partir destas propriedades é que o aprendizado é constituído. Não há um professor para supervisionar o processo de aprendizagem, ou seja, não há para cada entrada, uma saída desejada. Durante o processo, os padrões são apresentados continuamente à rede e a existência de regularidades nesses dados faz com que o aprendizado seja possível. Assim sendo, regularidade e redundância nas entradas são características essenciais para haver aprendizado não supervisionado. Este tipo de aprendizado é aplicável em problemas que visam à descoberta de características relevantes nos dados de entrada como, por exemplo, nas tarefas de agrupamento de dados.

  O treinamento não supervisionado, conforme comentado, não requer vetor alvo para as saídas e, obviamente, não faz comparações para determinar a resposta ideal. O treinamento não supervisionado modifica os pesos da rede de forma a produzir

saídas que sejam consistentes, isto é, tanto a apresentação de um dos vetores de treinamento, como a apresentação de um vetor que é suficientemente similar a um dos vetores de treinamento, irá produzir o mesmo padrão nas saídas. O processo de treinamento não supervisionado extrai as propriedades estatísticas do conjunto de treinamento e agrupa os vetores similares em classes.

- **Por Reforço:** Muitas vezes considerado um caso particular de aprendizado supervisionado, nessa abordagem o crítico externo não informa qual a resposta desejada, mas procura maximizar o reforço das ações boas executadas pela rede. Se uma ação tomada pelo sistema de aprendizado é seguida de estados satisfatórios, então a tendência do sistema de reproduzir essa ação particular é reforçada. Se não for seguida de estados satisfatórios, a tendência do sistema produzir essa ação é enfraquecida. Este tipo de aprendizado tem aplicabilidade especialmente recomendada para tarefas de controle.

5.5. Modelos de RNAS

5.5.1. Perceptron de Camada Única

O exemplo mais antigo de modelo de rede neural é o modelo perceptron. Tem como objetivo classificar corretamente o conjunto de estímulos aplicados externamente á rede em uma de duas classes.

Redes Perceptron de Camada Única são redes baseadas no modelo perceptron e possuem as seguintes características:

- ✓ Utilizam aprendizado supervisionado,
- ✓ Possuem apenas uma camada de pesos ajustáveis,
- ✓ São feedforward,
- ✓ Utilizam o produto escalar como regra de propagação,
- ✓ Adotam função de ativação degrau.

A figura 5.23 apresenta um exemplo genérico de rede perceptron, que estabelece uma fronteira de decisão linear. Por esta razão, redes deste modelo têm sua aplicação adequada somente para problemas linearmente separáveis como os das portas lógicas “AND” e “OR”. Problemas em que as classes não sejam linearmente separáveis como o da porta lógica “XOR” não podem ser resolvidos por redes perceptron.

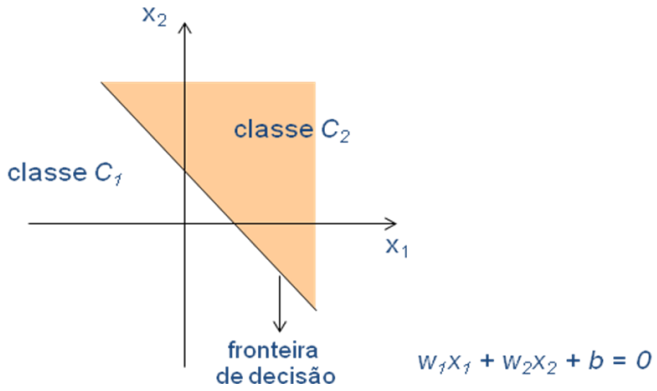


Figura 5.23: Exemplo de rede perceptron e sua fronteira de decisão linear

A seguir encontra-se enunciado em pseudocódigo o algoritmo de aprendizado em redes perceptron de camada única.

1 – Inicializar η (taxa de aprendizado) e o vetor de pesos W

2 – Repetir para cada par do conjunto de treinamento

2.1 – Atualizar o vetor de pesos para cada um dos nós da rede segundo a regra

$$W(t+1) = W(t) + \eta \cdot e \cdot X(t)$$

3 – Até $e = 0$ para todos os p elementos do conjunto de treinamento em todos os neurônios da rede

5.5.2. Adaline

O surgimento do modelo Adaline foi quase simultâneo ao Perceptron de Camada Única. Como diferenças principais em relação ao seu contemporâneo, no modelo Adaline, as redes neurais possuem função de ativação linear e a função de erro a ser minimizada é quadrática conforme fórmula abaixo. A minimização é feita pelo método do gradiente descendente.

$$E = \frac{1}{2}\sum_{i=1}^{p}\left(d^i - y^i\right)^2$$

As redes Adaline têm as mesmas limitações das redes Perceptron de Camada Única. Inclusive o treinamento das redes Adaline mostra-se mais demorado. Como vantagem, o treinamento em redes Adaline tende a ser mais suave do que em redes Perceptron de Camada Única.

5.5.3. Madaline

Este modelo foi um dos primeiros a envolver elementos adaptativos organizados em camadas treináveis. Trata-se de um modelo composto por múltiplos Adalines. Uma rede Madaline pode possuir vários Adalines sendo um deles com parâmetros fixos implementando critérios de decisão como E, OU e MAIORIA. As figuras 5.24 e 5.25

ilustram, respectivamente, uma Madaline aplicado ao problema da porta "XOR" e suas fronteiras de decisão.

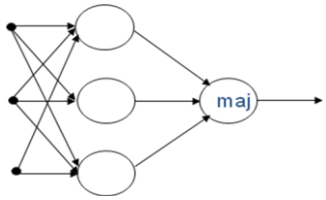


Figura 5.24: Exemplo de rede madaline para o problema da porta XOR

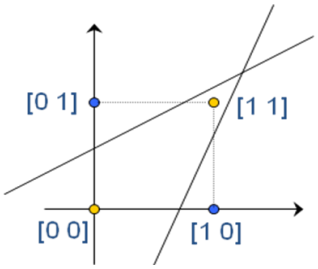


Figura 5.25: Exemplo de fronteiras de decisão para a madaline da figura 5.24

5.5.4. Perceptron de Múltiplas Camadas

O Perceptron de Múltiplas Camadas (MLP – Multi-Layer Perceptron) foi uma evolução do Madaline. Conforme o próprio nome sugere, redes deste modelo consistem de múltiplas camadas de unidades computacionais, geralmente completamente interconectadas e com processamento feedforward. Isso quer dizer que cada neurônio em uma camada tem conexões com todos os neurônios da camada seguinte. Em muitas aplicações as unidades dessas redes utilizam a função sigmóide como função de ativação.

O teorema de aproximação universal dita que toda função contínua que mapeia intervalos de números reais a algum intervalo de números reais de saída pode ser arbitrariamente aproximada com precisão por um perceptron de múltiplas camadas com somente uma camada oculta. Este resultado só é válido para classes restritas de funções de ativação (diferenciáveis em todos os pontos), por exemplo, função sigmóide ou tangente hiperbólica.

A definição do número de neurônios em cada camada de uma rede MLP é de extrema importância para o desempenho da rede, principalmente na sua capacidade de generalização. Quanto maior o número de neurônios, maior é a capacidade da rede em resolver problemas de determinada complexidade. Embora não haja regra geral para definição precisa deste número, existem diversas heurísticas passíveis de aplicação. Uma delas, proposta por Hecht-Nielsen (1988) sugere a uitlização de $(2n+1)$ neurônios na camada intermediária, onde n é o número de neurônios na camada de entrada.

No entanto, dado um problema, deve-se procurar por uma rede de estrutura mínima que atenda aos requisitos de minimização do erro quadrático do conjunto de treinamento. Ou seja, uma boa prática é iniciar com uma estrutura pequena de rede e empiricamente

introduzir neurônios gradualmente verificando seu efeito até que o objetivo seja satisfeito.

O algoritmo de treinamento de redes MLP mais popular é o back-propagation (retro-propagação do erro), que por ser supervisionado, utiliza pares de entrada e saída para, por meio de correção de erros, ajustar os pesos da rede. Nele, o treinamento utiliza uma generalização da regra delta (utilizada pelo Adaline) e ocorre em duas fases: forward e backward.

$$E(k) = \frac{e(k)^2}{2} = \frac{\left(y_d(k) - \hat{y}(k)\right)^2}{2}$$

Na fase forward, um padrão de entrada é apresentado à rede que o processa gerando a saída da rede. O erro é calculado pela fórmula acima e a fase backward é então aplicada. Nela, o back-propagation procura ajustar todos os pesos da rede (não somente das conexões que chegam à camada de saída, mas das conexões de chegam a todas as demais camadas) de forma proporcional à sua contribuição para geração do erro.

$$\Delta W_i(k+1) \approx -\lambda \times \nabla E(k)$$

5.6. Aplicações para RNAs

São inúmeras as possibilidades de aplicação das redes neurais em diversos tipos de problemas. Abaixo seguem alguns exemplos a título meramente ilustrativo.

- Reconhecimento de padrões
  - ✓ Transformação de imagens em textos (OCRs),
  - ✓ Reconhecimento de placas de veículos (em sistemas de multagem eletrônica)
  - ✓ Reconhecimento de voz
- Classificação de padrões
  - ✓ Reconhecimento de locutor (próprio ou impostor)
  - ✓ Reconhecimento biométrico (próprio ou impostor)
  - ✓ Reconhecimento de pintura (autêntica ou falsa)
  - ✓ Reconhecimento de SPAMS (SPAMS e não SPAMS)
  - ✓ Reconhecimento de Fraudes em Sinistros de Seguros
- Correção de padrões
  - ✓ Correção de imagens incompletas
  - ✓ Correção de textos incompletos
- Previsão de séries temporais
  - ✓ Planejamento da produção de bens de consumo

- ✓ Previsão de consumo energético
- ✓ Previsão de demandas telefônicas
- ✓ Cotação de ações do mercado financeiro

- Mineração de Dados
  - ✓ Análise de Crédito (na separação entre bons e maus pagadores) – Problema de Classificação
  - ✓ Análise de Limite de Crédito
  - ✓ Agruapamento de Clientes com potencial de compra similar
- Suporte à decisão – Neste item podem ser enquadradas a maioria das aplicações mencionadas já mencionadas.

Capítulo
6

Algoritmos Genéticos

6.1. Introdução

Algoritmos Genéticos são modelos computacionais de busca e otimização de soluções em problemas complexos, inspirados em princípios da teoria da evolução natural de Charles Darwin e da reprodução genética.

Segundo o princípio básico da evolução natural de Darwin, indivíduos mais aptos possuem maiores chances de sobrevivência e, conseqüentemente, mais oportunidades de gerarem descendentes e perpetuarem seus códigos genéticos pelas gerações seguintes. A identificação de cada indivíduo é expressa pelo seu código genético que fica representado nos cromossomas deste indivíduo.

Resumidamente, Algoritmos Genéticos são técnicas que procuram obter boas soluções para problemas complexos por meio da evolução de populações de soluções codificadas em cromossomas artificiais. Para tanto, Algoritmos Genéticos combinam sobrevivência dos indivíduos mais aptos (melhores soluções) e cruzamento aleatório de informação. Todo problema de difícil modelagem matemática ou com um número muito grande, possivelmente infinito, de soluções é considerado um problema complexo. A tabela 6.1 apresenta uma analogia entre Algoritmos Genéticos e Evolução Natural.

Algoritmos Genéticos empregam um processo adaptativo e paralelo de busca de soluções em problemas complexos. O processo é adaptativo, pois as soluções existentes a cada instante influenciam a busca por futuras soluções. O paralelismo do processo é decorrência natural do fato de que diversas soluções são consideradas a cada momento pelos Algoritmos Genéticos.

São exemplos de aplicações de Algoritmos Genéticos:

- Otimização de funções matemáticas
- Otimização Combinatorial
- Otimização de Planejamento
- Problema do Caixeiro Viajante
- Problema de Otimização de Rota de Veículos
- Otimização de Distribuição e Logística

- Seleção de Variáveis em Mineração de Dados
- Dentre muitos outros...

Nos Algoritmos Genéticos, um cromossoma é uma estrutura de dados que representa uma das possíveis soluções do espaço de busca de um problema. Cromossomas são então submetidos a um processo evolucionário que avalia, seleciona e combina soluções ao longo de diversos ciclos, procurando obter indivíduos mais aptos (melhores soluções).

Tabela 6.1: Analogia entre Algoritmos Genéticos e a Natureza

Evolução Natural	Algoritmos Genéticos
Meio Ambiente	Problema
Indivíduo	Solução
Cromossoma	Representação (palavra binária, vetor, etc)
Gene	Característica do Problema
Alelo	Valor da Característica
Loco	Posição na palavra, vetor
Genótipo	Estrutura
Fenótipo	Estrutura submetida ao problema
Reprodução Sexual	Operador de Cruzamento
Mutação	Operador de Mutação
População	Conjunto de Soluções
Gerações	Ciclos

Em geral, um Algoritmo Genético é caracterizado pelos seguintes componentes:

- Problema
- Representação das Soluções do Problema
- Decodificação do Cromossoma
- Avaliação
- Seleção
- Operadores Genéticos
- Técnicas e Parâmetros
- Critérios para Inicialização da População

Mais à frente, seções específicas detalham cada um destes componentes.

6.2. História dos Algoritmos Genéticos

Os algoritmos genéticos surgiram de um modelo matemático que tentava demonstrar a teoria de Darwin. Esse modelo foi publicado no livro *The Genetic Theory of Natural Selection* (Fisher, 1930).

Foi baseado nesse conceito matemático que em meados da década de 60, John Holland começou a se dedicar aos estudos de processos naturais adaptáveis e dessa forma acabou inventando os algoritmos genéticos. Os algoritmos foram desenvolvidos e aperfeiçoados durante as décadas de 60 e 70 em conjunto com seus alunos e colegas da universidade de Michigan.

O objetivo de Holland, ao criar esses algoritmos, era poder estudar formalmente o fenômeno da adaptação tão presente na natureza e a partir daí ser capaz de desenvolver modelos em que mecanismos da adaptação natural pudessem ser inseridos no contexto dos sistemas computacionais. Seu exaustivo trabalho foi recompensado em 1975 quando ele edita *Adaptation in Natural and Artificial Systems* (Holland, 1975) e, 13 anos mais tarde em 1989, David Goldberg edita *Genetic Algorithms in Search, Optimization and Machine Learning* (Goldberg, 1989), ambos considerados até hoje como os livros mais importantes sobre o assunto.

6.3. Por que usar algoritmos genéticos?

A seguir encontram-se enunciadas algumas justificativas para o uso de Algoritmos Genéticos:

- Porque algoritmos genéticos mantêm uma população de soluções que são avaliadas simultaneamente. O que torna sua execução mais dinãmica do que a avaliação sequencial de soluções geradas individualmente.
- Diferentemente de outras técnicas de busca, os algoritmos genéticos não usam somente informações locais, sendo capazes, portanto, de escapar dos chamados máximos locais durante o processo de busca por soluções.
- O processo de evolução desta técnica é guiado por um mecanismo de seleção baseado em adaptação de estruturas que avalia cada solução frente ao problema a ser solucionado.
- Estes algoritmos são indicados também para cálculo de funções que possuem algum tipo de descontinuidade ou que sua derivada não possa ser calculada, pois eles não usam informações de derivadas durante sua evolução e também não precisam de informações sobre os gráficos das funções para efetuar a busca.
- Por se tratar de uma técnica que utiliza buscas direcionadas e inteligentes, os algoritmos genéticos são indicados para lidar com problemas cujo tamanho do espaço de busca seja muito grande.

6.4. Compoenentes de um Algoritmo Genético

Nesta seção serão detalhados os principais componentes de um Algoritmo Genético, ou seja, os elementos que precisam ser especificados para uma completa definição do modelo genético que será utilizado na busca de boas soluções para o problema em questão.

Conforme comentado, algoritmos genéticos são especialmente úteis na solução de problemas complexos que envolvam necessidade de otimização. Um problema complexo, também conforme já mencionado anteriormente, é um problema de difícil formulação matemática que envolva um grande espaço de soluções.

Embora o exemplo abaixo não possa ser efetivamente considerado um problema complexo, optamos por utilize-lo, em função de sua simplicidade e facilidade de compreensão, para descrever os componentes de um algoritmo genético apresentados nas próximas subseções.

Problema: Encontrar o ponto que maximize a função $f(x)=x^2$, considerando o intervalo $[\mathbf{0...2^L - 1}]$.

Convém observer que quanto maior o valor de L, maior é o número de pontos do domínio do problema e, consequentemente, maior é o espaço de busca. Para fins de simplificação de nosso exemplo didático, vamos assumir L=6. Assim sendo, o espaço de busca será o dos números inteiros de 0 até 63, ou seja, um espaço de busca de apenas 64 pontos a serem avaliados. Neste caso, a utilização de um mecanismo de busca exaustiva seria suficiente para resolver facilmente o problema. Além disso, um conhecimento prévio básico em Análise Matemática, nos permite facilmente concluir que o ponto máximo desta função é (63; 3969). No entanto, conforme ressaltado, este exemplo será utilizado para fins didáticos na ilustração dos demais conceitos envolvidos na utilização de Algoritmos Genéticos.

6.4.1. Problema

A caracterização do problema consiste em enunciar de forma não ambígua qual será o objetivo do algoritmo genético. O enunciado do problema deve utilizar um verbo que indique uma ação cuja execução desencadeie a busca por uma solução ótima para o problema. É comum a utilização de verbos tais como maximizar e minimizar, dentre outros.

6.4.2. Representação das Soluções do Problema

A menor unidade de um Algoritmo Genético é chamada gene. Um gene representa uma unidade de informação do domínio do problema. Uma série de genes forma um cromossoma, que representa uma possível solução completa para o problema.

A representação de possíveis soluções do espaço de busca de um problema define a estrutura do cromossoma a ser manipulado pelo algoritmo. Qualquer que seja a forma de representação de um problema, esta deve ser capaz de representar todas as soluções

do espaço de busca que se deseja investigar. Algumas das principais formas de representação de soluções em AG estão comentadas a seguir.

A representação binária é a mais empregada, devido à sua simplicidade e facilidade de manipulação.

Por exemplo, para o problema enunciado anteriormente, é necessária uma representação para números inteiros. Podem ser utilizados cadeias binárias representando números inteiros. Neste caso, seriam necessários cromossomas de 6 bits para representar números entre 0 e 63. Exemplos:

C1: 001001 representa x=9

C2:000100 representa x=4

A representação diretamente por números reais é mais adequada em problemas de otimização de parâmetros com variáveis contínuas em grandes domínios (onde a representação binária requer cromossomas longos). A representação em reais é mais rápida na execução e oferece mais precisão.

Há problemas em que valores binários podem ser utilizados para representar números reais com uma precisão de *p* casas decimais. Supondo que os números reais pertençam a um intervalo $[X_{min}, X_{max}]$, são necessários *k* bits de forma que *k* atenda à seguinte relação:

$2^k \leq (X_{max} - X_{min}) * 10^p$

Como exemplo de aplicação da relação acima, considere um problema que seja minimizar a seguinte função:

$$f(x,y) = | x * y * sen(\frac{(y * \Pi)}{4}) |$$

Sendo que x,y $\in$ [-10,10]. Considerando uma precisão de duas casas decimais, tem-se:

$2^k \leq (10 - (-10)) * 10^2 = 20 * 100 = 2000$

$2^k \leq 2000$

Como:

$K = 10 \Rightarrow 2^k = 1024$

$K = 11 \Rightarrow 2^k = 2048$

Portanto, o valor de K deve ser 10, indicando a necessidade de cromossomas de 20 bits, sendo 10 para cada variável.

Nem sempre a representação binária pode ser empregada. Em muitos casos, o problema requer um alfabeto de representações com mais símbolos.

6.4.3. Decodificação do Cromossoma

O processo de decodificação de um cromossoma consiste na construção da solução real do problema que deverá ser avaliada.

Uma das vantagens da representação binária é a fácil transformação para inteiro ou real.

No exemplo de maximização da função $f(x)=x^2$ enunciado acima, a decodificação consiste em transformar a string binária em um número inteiro.

Para conversão de um binário em um valor real, utilize-se a fórmula:

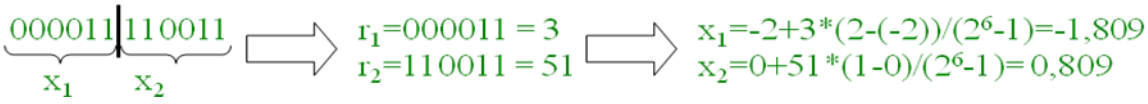
$$X_R = Xb \times \frac{C}{2^n - 1} + X_{min}$$

Onde:

$X_R \in [X_{min}, X_{máx}]$; Xb é o inteiro correspondente ao binário; n é o número de bits do cromossoma; e C é o comprimento do domínio da variável X, dado por $C=|X_{máx}-X_{min}|$

Por exemplo:

Sejam 2 variáveis reais x1 e x2 com 6 bits de representação, Dom(x1)=[-2,2] e Dom(x2)=[0,1] e o cromossoma 000011110011. Decodificando, temos:



6.4.4. Avaliação

Para que um cromossoma seja avaliado, em geral é necessário converter o cromossoma numa solução para o problema, isto é, decodificar o cromossoma.

A avaliação de cada solução em um Algoritmo Genético é um processo realizado por uma função (função de avaliação) que tem por objetivo fornecer uma medida da qualidade de tal solução (aptidão) frente ao problema.

As funções de avaliação são específicas de cada problema.

No exemplo acima, a função $f(x)=x^2$ expressa a qualidade de cada indivíduo como solução para o problema.

C1: 011100 → Representa 28 → f(28) = 784

C2: 110101 → Representa 53 → f(53) = 2809

C2 é um indivíduo melhor (mais apto) que *C1*, pois 2809 > 748.

Pode-se perceber pelo exposto que a função de avaliação tem papel de fundamental relevância na especificação de um algoritmo genético.

6.4.5. Seleção

O processo de seleção em AGs seleciona indivíduos para a reprodução. O AG tende a privilegiar indivíduos com maiores aptidões para permanecer e se multiplicar na população.

Existem vários métodos de seleção de indivíduos em AGs. Dentre eles estão o método da roleta e o método de seleção por torneio.

- Método da Roleta

Forma comum de seleção é aquela em que cada cromossoma tem a probabilidade de permanecer na próxima geração proporcional à sua aptidão. Cromossomas com maiores aptidões possuem mais espaço na roleta e conseqüentemente possuem maiores chances de serem escolhidos para o processo de reprodução. Assim, se f_i é a avaliação do *i-ésimo* indivíduo na população corrente, a probabilidade p_i (aptidão relativa) deste indivíduo ser selecionado é proporcional a:

$$p_i = \frac{f_i}{\sum_{i=1}^{n} f_i}$$

Onde n é o número de indivíduos na população.

No exemplo anterior, supondo que existam apenas dois indivíduos *C1* e *C2*, sendo:

C1: 001001 representa x=9 e C2:000100 representa x=4. Abaixo, segue o cálculo da aptidão relative de C1 e de C2.

Indivíduo	Aptidão Relativa
C1	p_1=81/(81+16)=0.84
C2	p_2=16/(81+16)=0.16

No método da roleta, é sorteado um número entre 0 e 1. O indivíduo selecionado é o primeiro indivíduo cuja aptidão acumulada seja superior ao número sorteado. No exemplo:

Indivíduo	Aptidão Relativa Acumulada
C1	0.84
C2	1.00 (0.84 + 0.16)

Para números sorteados até 0.84, o indivíduo selecionado seria *C1*. Para valores acima de 0.84, o indivíduo selecionado seria *C2*.

- Método do Torneio

Este método consiste em selecionar uma série de indivíduos da população e fazer com que eles entrem em competição direta de ser pai, usando como arma a sua avaliação. O tamanho do torneio (k) define quantos indivíduos são selecionados aleatoriamente dentro da população para competir (valor mínimo de k=2).

Neste método, não há favorecimento para os melhores indivíduos.

Por exemplo, considere os seguintes indivíduos e respectivas avaliações:

Indivíduo	Fitness
X_1	200
X_2	100
X_3	9500
X_4	100
X_5	100
X_6	10000
X_7	1
X_8	40

Suponha agora que tenham sido realizados oito torneios com três indivíduos cada, indicados abaixo. Em destaque, os indivíduos vencedores de cada torneio. São os indivíduos selecionados para prosseguir no processo evolutivo.

Torneios

X_1	X_7	X_8
X_2	X_3	X_5
X_6	X_4	X_4
X_2	X_7	X_1
X_5	X_5	X_5
X_3	X_4	X_2
X_4	X_2	X_6
X_4	X_6	X_5

6.4.6. Operadores Genéticos

Uma vez selecionados os indivíduos, o Algoritmo Genético cria novas soluções através da combinação e refinamento das informações dos cromossomas usando duas operações genéticas básicas: *crossover* e *mutação*. Essas operações produzem novas soluções que formam uma nova população. Cada nova população é chamada de geração.

Durante o *crossover*, dois cromossomas trocam algumas de suas informações contidas em determinados genes. Novos indivíduos são criados a partir da troca de material

genético entre seus pais. Os descendentes possuem características genéticas de ambos os genitores.

Existem vários exemplos de operadores de *crossover*. Muitos deles são criados especificamente para solução de determinados problemas. Optamos por relacionar alguns exemplos de operadores de *crossover* mais tradicionais.

- Crossover de 1 ponto – Representação Binária

  O *crossover* de um ponto recombina duas soluções a partir de um único ponto de corte aleatório. Sendo o *crossover* é de 1 ponto, uma posição do cromossoma é estabelecida como ponto de corte, que é a referência para troca de informações genéticas entre dois indivíduos selecionados como pais, gerando dois filhos, conforme ilustra a figura 6.1. Neste exemplo, o ponto de corte utilizado foi p=4.

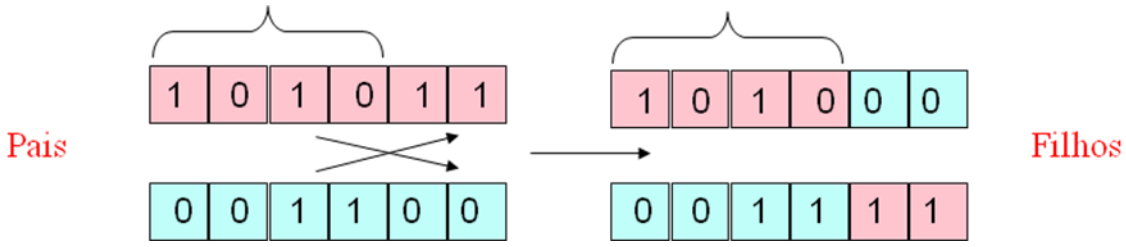


Figura 6.1: Exemplo de aplicação do crossover de 1 ponto – representação binária

A escolha do ponto de corte é aleatória, feita em cada aplicação do operador.

Pode-se perceber que um cromossoma com n genes possui n-1 pontos de corte.

Um outro exemplo: sendo *C1* e *C2*, os genitores e 3 o ponto de corte (ou de cruzamento) aleatoriamente selecionado (operador de crossover de um ponto), *D1* e *D2* seriam os descendentes nesta situação.

C1: 001001

C2: 000100

D1: 001100

D2: 000001

D1 é um cromossoma mais apto que seus genitores. No entanto, *D2* tornou-se um indivíduo bem menos apto. Ambas as situações retratam fatos que podem de fato ocorrer na natureza.

- Crossover de 1 ponto – Representação Real

  Análogo ao crossover de 1 ponto, mudando-se apenas a representação dos dados. Neste caso, a representação utilizada é a real. A figura 6.2 apresenta um exemplo de aplicação.

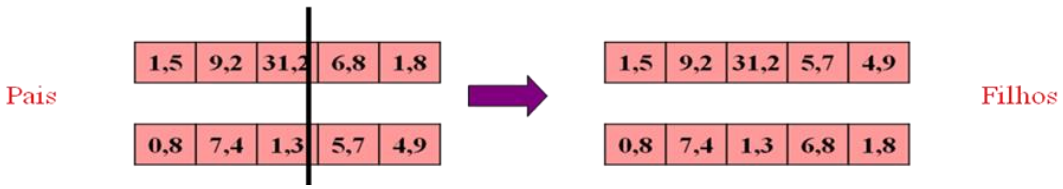


Figura 6.2: Exemplo de aplicação do crossover de 1 ponto – representação real

- Crossover de 2 pontos – Representação Binária
  No *crossover* de dois pontos são dois os pontos aleatórios escolhidos como referência para a troca de valor dos atributos (genes).

Cromossoma	**Palavra**			**Nova Palavra**
A	10	0010	10	10011110
C	11	0111	11	11001011

Figura 6.3: Operação de crossover com dois pontos de corte

- Crossover de n pontos – Representação Binária

  Demanda tantos sorteios de pontos de corte quantos forem os pontos de corte a serem considerados. Na figura 6.4 segue um exemplo de crossover de 4 pontos.

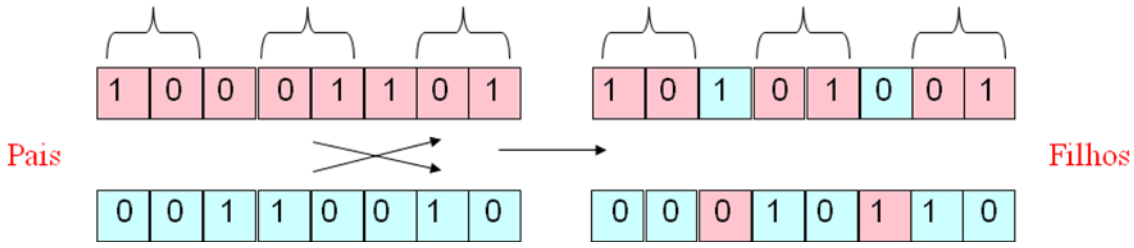


Figura 6.4: Operação de crossover com quatro pontos de corte

- Crossover Uniforme – Representação Binária
  Este tipo de *crossover*, por sua vez, é capaz de recombinar quaisquer posições entre dois genitores. Para tanto, utiliza uma palavra binária escolhida aleatoriamente para designar os bits selecionados em cada genitor na criação dos descendentes.

  Exemplo:

  G1: 1100101
  G2: 0111110
  Padrão: 0110100
  D1: 0101110
  D2: 1110101

- Crossover de Aritmético – Representação Real

Este operador consiste em aplicar uma combinação linear de dois vetores (genitores) P_1 e P_2 na geração t, segundo as fórmulas abaixo.

$F_1 = \lambda.P_1 + (1 - \lambda).P_2$

$F_2 = (1 - \lambda).P_1 + \lambda.P_2$

λ Pode ser constante ou variar em função da idade da população. A figura 6.5 ilustra uma aplicação deste operador.

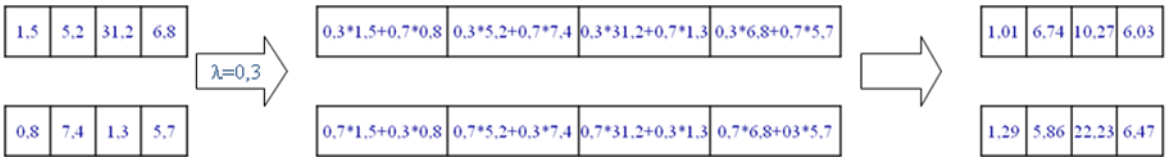


Figura 6.5: Operação de crossover aritmético

- Crossover de Média – Representação Real

  Consiste em aplicar uma média aritmética simples entre dois genitores, gerando um único filho.

  $P_1 = (x_1,y_1)$ e $P_2 = (x_2,y_2)$

  $F_1 = ((x_1+x_2)/2, (y_1+y_2)/2)$

Os cromossomas criados a partir do operador de *crossover* são submetidos à operação de mutação. A mutação permite explorar o potencial de novos indivíduos, aumentando a diversidade de indivíduos na população. O operador de mutação altera o conteúdo de uma posição do cromossoma, com uma probabilidade, em geral, bem baixa (<1%).

A taxa de mutação não pode ser nem alta e nem baixa, tendo de possuir um valor suficiente para assegurar a diversidade de cromossomos em uma população. Uma taxa de mutação alta aumenta as chances de um bom indivíduo ser destruído, causando instabilidade no modelo.

De forma análoga ao *crossover*, existem vários exemplos de operadores de *mutação*. Também muitos deles são criados especificamente para solução de determinados problemas. Optamos por relacionar alguns exemplos de operadores de *mutação* mais tradicionais.

- Mutação Clássica – Representação Binária

  Inverte o valor de um bit selecionado aleatóriamente entre todos os genes de um cromossoma. Segue um exemplo do indivíduo *C2* antes e após sofrer mutação no último bit:

  C2: 000100, antes da mutação

  C2: 000101, após a mutação

- Mutação Clássica – Representação Real

  Substitui o número real de um gene sorteado em um cromossoma por um número real aleatório, sempre que um teste aleatório for satisfeito. Por exemplo:

  $(x_1,y_1) \Rightarrow (x_1,y_{rand})$

  (73.2,41.8) => (73.2,-0.4)

- Mutação Creep – Representação Real

  Busca uma solução próxima por meio de ajustes aleatórios em ambas as direções do valor de cada gene do cromossoma: $(x_1,y_1) \Rightarrow (x_1 \pm \Delta x, y1 \pm \Delta y)$

  Método de ajuste:

$$x^{t+1} = \begin{cases} x^t + \Delta(max - x^t) \text{ se bit sorteado} = 0 \\ x^t - \Delta(x^t - min) \text{ se bit sorteado} = 1 \end{cases}$$

  Onde, max e min são os limites do domínio de x, e

  $\Delta(s)=s*rand$, onde rand é um número aleatório pertencente a [0,p], $p \leq 1$

  Cabe observar que o ajuste varia com o valor de p. Para valores pequenos de p, os ajustes são menores. Por outro lado, para valores grandes de p, os ajustes são maiores.

6.4.7. Técnicas e Parâmetros

Em geral, as técnicas, os parâmetros e os tipos de operadores genéticos afetam significativamente o desempenho dos Algoritmos Genéticos. A escolha de técnicas, parâmetros e operadores é empírica, porém em sintonia com o tipo de problema envolvido.

A inicialização da população determina o processo de criação dos indivíduos para o primeiro ciclo do algoritmo, sendo comum à formação da população inicial a partir de indivíduos aleatoriamente criados.

Em um Algoritmo Genético, vários parâmetros controlam o processo evolucionário:

- Tamanho da população – números de pontos no espaço de busca sendo considerados em paralelo a cada ciclo do algoritmo genético.
- Taxa de Crossover – probabilidade de um indivíduo ser recombinado geneticamente com outro.
- Taxa de Mutação - probabilidade do conteúdo de cada gene do cromossoma ser alterado.

- Número de Gerações – quantidade ciclos do Algoritmo Genético. Pode ser deduzido caso se tenha a informação do tamanho da população e total de indivíduos (total de indivíduos / tamanho da população).
- Total de Indivíduos – total de soluções a serem geradas e avaliadas pelo Algoritmo Genético (tamanho da população * número de gerações).

Os últimos dois parâmetros são normalmente empregados como critério de parada de um Algoritmo Genético, isto é, quando o processo evolucionário deve ser interrompido.

Um Algoritmo Genético pode ser descrito como um processo contínuo que repete ciclos de evolução controlados por um critério de parada, conforme ilustrado pela figura 6.6.

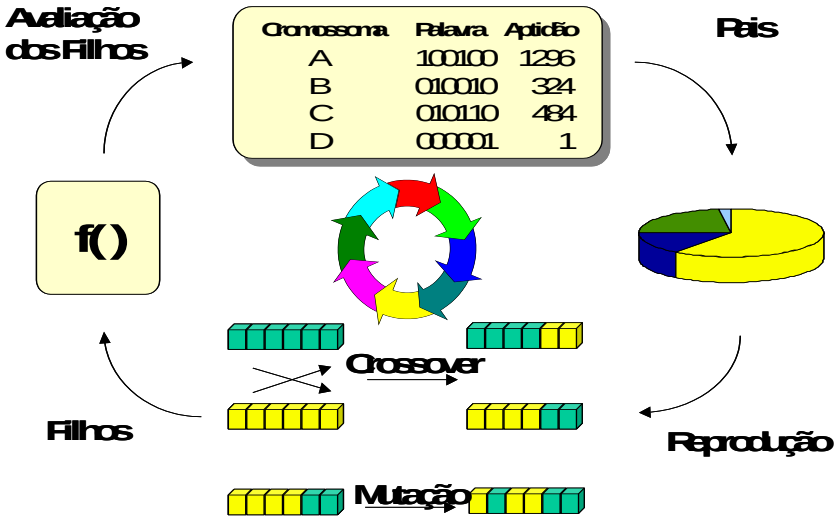


Figura 6.6: Ciclo de um Algoritmo Genético

As técnicas de reprodução determinam o critério de substituição dos indivíduos de uma população para a geração seguinte. A seguir encontram-se citados alguns exemplos:

- *Troca de toda a População*: A cada ciclo, *N* novos indivíduos são criados, substituindo a população anterior. Para tanto, *N/2* pares de indíduos são selecionados para acasalamento, gerando *N* descendentes.

- *Elitismo*: Todos os cromossomas são substituídos, sendo o cromossoma mais apto da população corrente copiado para a população seguinte.

- *Steady State*: Gera *M* indivíduos (*M*<*N*), que substituem os piores indivíduos da população corrente. Pode haver duplicação de indivíduos na população resultante.

- *Steady State sem Duplicados*: Similar à técnica de Steady State, não permitindo a presença de indivíduos duplicados na população.

As técnicas de aptidão influenciam na forma com que as aptidões dos cromossomas em uma população são numericamente atribuídas. A seguir encontram-se citados alguns exemplos:

- Método Clássico: Atribuir como aptidão de um cromossoma o valor numérico do resultado da avaliação. Este método, embora utilizado em muitos problemas, pode apresentar duas situações que precisam ser tratadas:

    a) Competição próxima – Indivíduos cujas aptidões são numericamente muito próximas dificultam a distinção entre eles quanto à qualidade da solução proporcionada;

    b) Super-indivíduos – São indivíduos com avaliação muito superior à média, que podem dominar o processo de seleção, impedindo que o Algoritmo Genético obtenha novas soluções, potencialmente melhores do que as representadas pelos super-indivíduos.

- Normalização Linear: Consiste em ordenar os cromossomas em ordem crescente de avaliação e atribuir linearmente (equação abaixo) valores de aptidões aos cromossomas entre um intervalo [valor mínimo, valor máximo], distanciados de um valor fixo entre eles.

    Onde *max* e *min* são, respectivamente, a maior e a menor avaliação identificadas em uma geração, *N* é o número de indivíduos na geração e *i* identifica o indivíduo que está tendo sua avaliação normalizada.

As técnicas de interpolação de parâmetros de um Algoritmo Genético têm como objetivo buscar o valor ideal de um determinado parâmetro para cada ciclo, durante toda a evolução. Estas técnicas podem ser lineares ou adaptativas.

Como exemplo para ilustrar a necessidade de que parâmetros do Algoritmo Genético sejam interpolados ao longo do processo evolutivo, pode-se intuitivamente perceber que a taxa de aplicação do operador de *crossover* deve ser maior nas primeiras gerações, quando a população se apresenta dispersa pelo espaço de busca. Após várias gerações, os indivíduos tendem a apresentar, na sua maioria, características muito similares. Um incremento da taxa de mutação nesta fase da evolução propicia uma pequena dispersão da população, trazendo novo material genético para a formação de melhores indivíduos.

Na interpolação linear, uma taxa ou um parâmetro é variado entre um valor inicial e um valor final, por meio de incrementos/decrementos fixos a cada *k* gerações. A interpolação adaptativa, normalmente utilizada para ajuste da taxa de aplicação de operadores genéticos, considera o desempenho destes operadores nos ciclos anteriores. Este desempenho é medido em função do sucesso dos referidos operadores na criação de melhores indivíduos. A adoção da intepolação adaptativa envolve um "overhead"

computacional por conta da necessidade de avaliação dos operadores a cada nova geração.

6.5. Estrutura Geral de um Algoritmo Genético

A seguir se encontra a estrutura geral básica de um Algoritmo Genético. Esta estrutura está descrita em pseudo-código, pois tem como objetivo apresentar em nível lógico o processo evolutivo em Algoritmos Genéticos. Algumas alterações nesta estrutura são admissíveis em função do problema analisado. Por exemplo: podem ser incluídas funções para implementar técnicas de reprodução ou ainda técnicas de aptidão. Considerando a importância da função de avaliação, optou-se por apresentar em seguida a sua descrição básica. As demais funções não foram descritas, mas podem ser obtidas na maioria das aplicações de Algoritmos Genéticos disponíveis na Internet. Em particular, os autores sugerem uma consulta à biblioteca de funções de Algoritmos Genéticos, escrita em MATLAB, disponível no site: www.ica.ele.puc-rio.br

Conforme já mencionado, cabe ressaltar que diferentes critérios de parada podem ser utilizados para terminar a execução de um Algoritmo Genético. Exemplos:

a) Após um determinado número de iterações (ciclos ou gerações);
b) Quando a aptidão média ou do melhor indivíduo não melhorar mais;
c) Quando as aptidões dos indivíduos de uma população se tornarem muito parecidas.
d) Combinações das opções acima.

Estrutura Geral de um Algoritmo Genético

T=0
Gerar População Inicial P(0)
Avaliar P(0)
Enquanto Critério de Parada não for satisfeito **faça**:
    T=T+1
    Selecionar População P(T) a partir de P(T-1)
    Aplicar Operadores de Cruzamento sobre P(T)
    Aplicar Operadores de Mutação sobre P(T)
    Avaliar P(T)
Fim Enquanto

Estrutura Geral de uma Função de Avaliação (Avaliar P(T))

Para todo indivíduo *i* da População Atual P(T) **faça**
    Avaliar a aptidão do indivíduo i
Fim Para

6.5. Desempenho dos AGs

Pelo exposto nas seções anteriores, pode-se perceber que Algoritmos Genéticos combinam mudanças aleatórias com processos probabilísticos. Assim, AGs são,

portanto, estocásticos: dificilmente repetem um resultado de um experimento para outro.

O desempenho de um AG é medido pelo grau de evolução alcançado durante o processo evolucionário. Para tanto, devido à natureza estocástica dos AGs é necessário avaliar o resultado médio de vários experimentos para se ter uma idéia do desempenho do algoritmo genético desenvolvido.

As principais medidas de desempenho são expressas por:

- Curva da média dos melhores cromossomas a cada ciclo em vários experimentos: apresenta o desempenho médio de um AG e serve para ajuste de parâmetros.
- Curva on-line da média da avaliação de todos os indivíduos até um determinado instante t em um experimento: mede a velocidade com que um AG consegue produzir boas soluções.
- Curva off-line da média da avaliação dos melhores indivíduos até um instante t em um experimento: mede o grau de convergência do AG na criação de soluções mais aptas.

6.6. Um Exemplo de Execução Manual de um Algoritmo Genético

Esta seção tem por objetivo ilustrar alguns ciclos do processamento de um algoritmo genético, mostrando, desta forma, o encadeamento dos conceitos apresentados ao longo deste capítulo. Para tanto, utilizamos um exemplo de problema extraído de (Linden, 2006).

Considere o problema: Maximizar a função

$f(x,y) = 1 + | x * y * \text{sen} (y\ \pi/4) |$ onde x e y $\in$ [0,15] e são valores inteiros.

Considerando o intervalo**,** são necessários 4 bits para cada variável, o que implica em cromossomas com 8 bits.

Seja a população inicial, sorteada aleatoriamente:

Cromossomo	x	y	g(x,y)
01000011	**4**	**3**	**9,5**
00101001	**2**	**9**	**13,7**
10011011	**9**	**11**	**71,0**
00001111	**0**	**15**	**1,0**
10011001	**5**	**5**	**18,7**
11100011	**14**	**3**	**30,7**
Somatório das avaliações			**144,6**

Considerando a utilização do método de seleção da roleta, temos a seguinte situação:

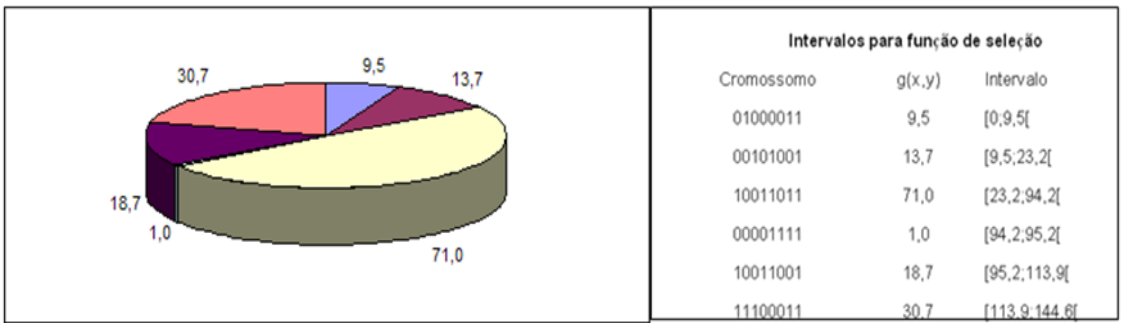


Figura 6.7: Exemplo de configuração da roleta

Sorteamos então os indivíduos para produzir a geração seguinte. (seis números entre 0 e a soma das avaliações (144,6)).

Número Sorteado	Cromossomo Escolhido
12,8	00101001
65,3	10011011
108,3	10011001
85,3	10011011
1,8	01000011
119,5	11100011

Fazendo o crossover de um ponto entre os pares de pais selecionados, e aplicando a mutação em um dos filhos gerados, tem-se, por exemplo, o resultado da figura 6.8.

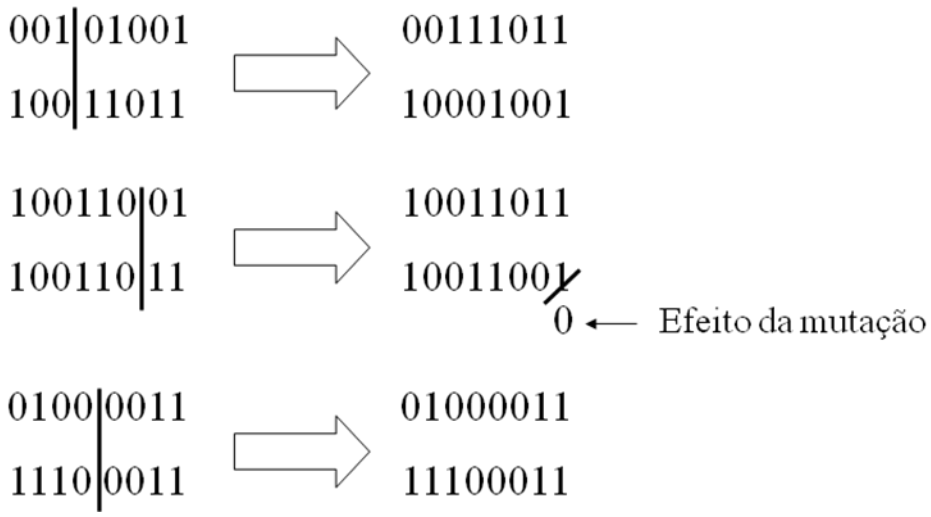


Figura 6.8: Resultado da aplicação dos operadores de crossover e mutação.

Convém ressaltar que:

- O ponto de corte variou em cada aplicação do crossover, uma vez que este número é sorteado aleatoriamente.
- Há filhos exatamente iguais aos pais.
- Para cada bit foi sorteado um número entre 0 e 99 (taxa de mutação = 1%). Igual a zero inverte bit, mantendo caso contrário.

A nova geração e a avaliação dos novos indivíduos foi a seguinte:

Cromossomo	x	y	g(x,y)
00111011	3	3	7,4
10001001	8	9	51,9
10011011	9	11	71,0
10011000	9	8	1,0
01000011	4	3	9,5
11100011	14	3	30,7
Somatório das avaliações			171,5

Convém observar neste exemplo que houve melhora na avaliação média da geração em relação a anterior: $A_{media}(G_1) = 144,6/6 = 24,1$ $A_{media}(G_2) = 171,5/6 = 28,6$

O processo se repete até completar o número máximo de gerações ou até atingir outro critério de parada.

Capítulo
7

Considerações Finais

7.1. Introdução

A Inteligência Computacional (IC) é uma área da Ciência da Computação que tem como objetivos desenvolver e aplicar recursos que permitam aos computadores terem um comportamento similar ao humano em tarefas específicas. As atividades na área de IC podem ser organizadas em três grandes grupos: atividades voltadas ao desenvolvimento tecnológico em IC, atividades de construção de sistemas inteligentes com recursos de IC e atividades envolvendo a aplicação de sistemas inteligentes. Esta taxonomia, apresentada graficamente na figura 7.1, possui seus itens comentados abaixo.

- **Desenvolvimento Tecnológico –** Este item abrange todas as iniciativas de concepção, aprimoramento e desenvolvimento de algoritmos, ferramentas e tecnologias de IC.

- **Aplicação de Recursos de IC na Construção de SI –** Este item refere-se às atividades voltadas ao desenvolvimento de sistemas inteligentes que utilizem recursos da IC. As ferramentas produzidas pelas atividades de desenvolvimento tecnológico são utilizadas na execução dessas atividades.

- **Aplicação dos SI -** Finalmente, uma vez construídos os sistemas inteligentes, esse grupo de atividades compreende a aplicação, o acompanhamento, a avaliação e o refinamento de tais sistemas.

São inúmeros os segmentos do mundo contemporâneo em que a IC pode ser aplicada. Entre eles podem ser citados: comércio, indústria, finanças, medicina, educação, energia, telecomunicações, meio ambiente, etc...

Este livro procurou apresentar uma introdução a alguns dos principais conceitos e técnicas da IC, indicando várias possibilidades de aplicação.

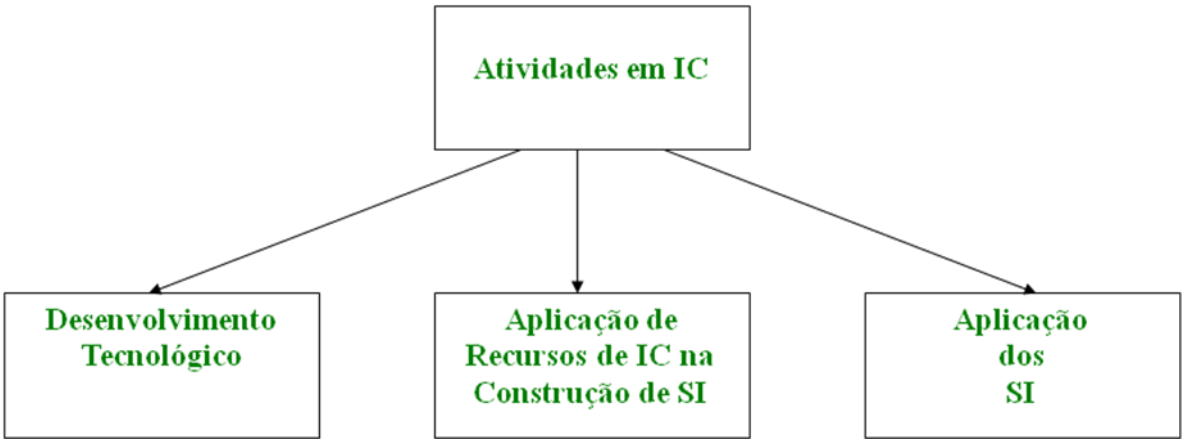


Figura 7.1: Taxonomia das Atividades em IC

Várias aplicações da IC situam-se em cenários de extrema relevância no contexto do mundo atual. A seguir encontram-se mencionadas algumas delas:

- Mineração de Dados e KDD (Knowledge Discovery in Databases – Descoberta de Conhecimento em Bases de Dados): A IC fornece uma série de recursos voltados ao auxílio do homem na análise de grandes quantidades de dados, visando o desenvolvimento e a seleção de estratégias de ação em cada contexto de aplicação.
- Mineração de Textos e KDT (Knowledge Discovery in Texts – Descoberta de Conhecimento em Dados Textuais): De forma análoga ao item anterior, a IC fornece uma série de recursos voltados ao auxílio do homem na análise de grandes quantidades de documentos textuais em busca de conhecimento que possa ser útil em cada contexto de aplicação.
- Processamento de Linguagem Natural: A IC tem contribuído para a construção de recursos computacionais que auxiliem no entendimento de linguagem natural pelo próprio computador.
- Modelagem de Sistemas Complexos: Ferramentas da IC tais como Sistemas Multi-agentes Inteligentes vêm sendo utilizadas na modelagem, simulação e entendimento de ambientes naturais envolvendo vários elementos com características e comportamentos diferenciados. Compreende, por exemplo, aplicações em Educação a Distância, Redes Sociais, Ambientes Colaborativos, dentre outras.
- Reconhecimento de Imagens: São várias as aplicações de recursos de IC no reconhecimento de imagens. Compreende desde cenários militares até aplicações de natureza cultural, tais como autenticação de pinturas e desenhos.

Os limites de aplicação da IC na atualidade são estabelecidos basicamente em função de duas características: (a) a primeira delas envolve o conhecimento que se tenha sobre a IC em si; (b) a segunda diz respeito à capacidade criativa de cada especialista em IC. Diante deste cenário, o presente texto vem procurar contribuir para disseminar e

desmistificar um pouco do conhecimento mencionado no item (a). E que o desconhecimento sobre as tecnologias da área não seja um limitador para a potencialidade de aplicação da IC junto ao mundo moderno.

Referências:

BAIÃO, E. Sistema Especialista na Área Médica, Trabalho de Conclusão de Curso, UGF, 2002.

BELLMAN, R. E. Dynamic Programming, Princeton: Princeton University Press, 1956.

BOJADZIEV, G.; BOJADZIEV, M. Fuzzy Logic for Business, Finance, and Management. World Scientific, 1997.

BRANCHMAN, R. J. What is-a is and isn´t: An analysis of taxonomic links in semantic networks. IEEE Computer, 16(10), 30-36, 1983.

BUCHANAN, B. G. et al. Constructing an Expert System. In Building Expert Systems. pp. 127-169, MA: Addison-Wesley, 1983.

CARVALHO, L. A. V. Data Mining: A Mineração de Dados no Marketing, Medicina, Economia, Engenharia e Administração. 2 ª ed., São Paulo: Érica, 2001.

CAZAROTTI, R.; MENEZES, R. Sistema Especialista para Planejamento de Atividades Físicas, Trabalho de Conclusão de Curso, UGF, 2004.

CHARNIAK, E.; McDERMOTT, D. Introduction to Artificial Intelligence, MA: Addison-Wesley, 1985.

CHAPPETTA, P. Sistema Especialista de Análise de Risco para Concessão de Limite de Cartão, Trabalho de Conclusão de Curso, UGF, 2004.

DAVIS, L. Handbook of Genetic Algorithms. VNR Comp. Library, 1990.

DAVIS, R. H. et al. What is knowledge representation? AI Magazine, 14(1), pp. 17-33, 1993.

FRANÇA, E. SEAM – Sistema Especialista para Área Médica. Trabalho de conclusão de curso – UGF, 2002.

FISHER, R. A. The Genetical Theory of Natural Selection. Clarendon, 1930.

GOLDBERG, D. E. Genetic Algorithms in Search, Optimization and Machine Learning. MA: Addison-Wesley, 1989.

GOLDSCHMIDT, R. Assistência Inteligente à Orientação do Processo de Descoberta de Conhecimento em Bases de Dados. Rio de Janeiro, Tese de Doutorado – Departamento de Engenharia Elétrica, Pontifícia Universidade Católica do Rio de Janeiro, 2004.

GOLDSCHMIDT, R., PASSOS, E. Data Mining: Um Guia Prático. Rio de Janeiro: Campus, 2005.

Gruber, T. R. A Translation Approach to Portable Ontologies. Knowledge Acquisition, 5(2):199-220, 1993.

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

HAUGELAND, J. Artificial Intelligence: The Very Idea. Cambridge: MIT Press, 1985.

HECHT-NIELSEN, R. Applications of Counterpropagation Networks, Neural Networks, v. 1, pp. 131-139, 1988.

_____ Neurocomputing, New York, Addison-Wesley, 1990.

HOLLAND., J. H. Adaptation in Natural and Artificial Systems. University of Michigan Press, 1975.

KOZA, J. Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press, 1992.

Laboratório de Inteligência Artificial. SINTA: Uma ferramenta visual para criação de sistemas especialistas, versão 1.1. Manual do usuário, UFC, 1997.

LINDEN, R. Algoritmos Genéticos. São Paulo: Brasport, 2006.

LUCENA, C. J. P. Inteligência Artificial e Engenharia do Software. Rio de Janeiro: Publicações Acadêmico-Científicas, 1987.

LUGER, G. F. Inteligência Artificial. Porto Alegre: Bookmann, 2004.

MAIDA, A. S. Frame Theory. John Wiley & Sons, 1987.

MENDES, L. A. T. Sistema Especialista na Geração de Sistemas Especialistas. Trabalho de Conclusão de Curso de Graduação. Curso de Bacharelado em Ciência da Computação do Centro Universitário da Cidade do Rio de Janeiro. 2005.

MINSKY, M. A. A Framework for Representing Knowledge. New York: McGraw-Hill, 1975.

NUNES, T. Sistema Especialista para Diagnóstico de Falhas em Aeronaves. Trabalho de Conclusão de Curso, UGF, 2003.

PASSOS, C. A. Ambiente para Desenvolvimento de Sistemas Especialistas: Edição e Prototipação. Dissertação de Mestrado - IME, 1997.

PEDRYCZ, W. & GOMIDE, F. An Introduction to Fuzzy Sets Analysis and Design. MIT, 1998.

QUINLAN, J. R. Programs for Machine Learning. San Mateo: Morgan Kaufmann Publishers Inc., 1993.

REZENDE, S. Sistemas Inteligentes: fundamentos e Aplicações. Barueri: Manole, 2003.

RIBEIRO, Horácio da C. S. Introdução aos Sistemas Especialistas, Editora LTC, 1987.

RICH, E.; KNIGHT, K. Inteligência Artificial. São Paulo: Makron, 1992.

RUSSELL, S. J.; NORVIG, P. Inteligência Artificial. 2ª. ed. Rio de Janeiro: Campus, 2004.

WINSTON, P. H. Artificial Intelligence, 3rd ed. MA: Addison-Wesley, 1992.

Apêndice A

Exemplos de Ferramentas de Inteligência Computacional – Dicas de Utilização

A.1. IcaDemo

A.1.1. Introdução

O Icademo é um software demonstrativo de técnicas computacionais inteligentes (Redes Neurais e Algoritmos Genéticos) desenvolvido no ICA - Laboratório de Pesquisa em Inteligência Computacional Aplicada da PUC-Rio para facilitar o ensino e o aprendizado de técnicas computacionais inteligentes. O Sistema foi desenvolvido para operar no sistema operacional Windows ®.

A.1.2. Aplicações do Icademo

O Icademo possui demonstrações de aplicações de redes neurais e de algoritmos genéticos, as quais serão apresentadas nas próximas seções.

A.1.2.1. Algoritmos Genéticos

O principal aplicativo relacionado aos Algoritmos Genéticos refere-se à maximização das funções F6 e F6 elevada. O Icademo permite configurar parâmetros do AG tais como as taxas de crossover e de mutação, o tamanho da população assim como a escolha de técnicas (elitismo, normalização linear, steady state) para melhoria do desempenho do algoritmo. O tipo de crossover (um ponto, dois pontos ou uniforme) também pode ser escolhido pelo usuário. Abaixo a figura A.1.1 apresenta a tela de avaliação da função F6 e a figura A.1.2 apresenta a fórmula da função F6 a ser maximizada. Ao final do processamento do AG, o IcaDemo apresenta um gráfico mostrando os desempenhos dos melhores indivíduos gerados pelo algoritmo. A avaliação de um indivíduo leva em consideração a quantidade de números “9” após o ponto decimal. Como o maior valor que a função pode alcançar é “1”, o AG procura encontrar os pares ordenados (x,y) que maximizem a função, ou seja, levem a uma imagem próxima de 1. Assim, quanto mais números “9” após o ponto decimal a imagem de um par ordenado apresentar, mais próximo está do máximo da função e, portanto, mais apto deve ser considerado.

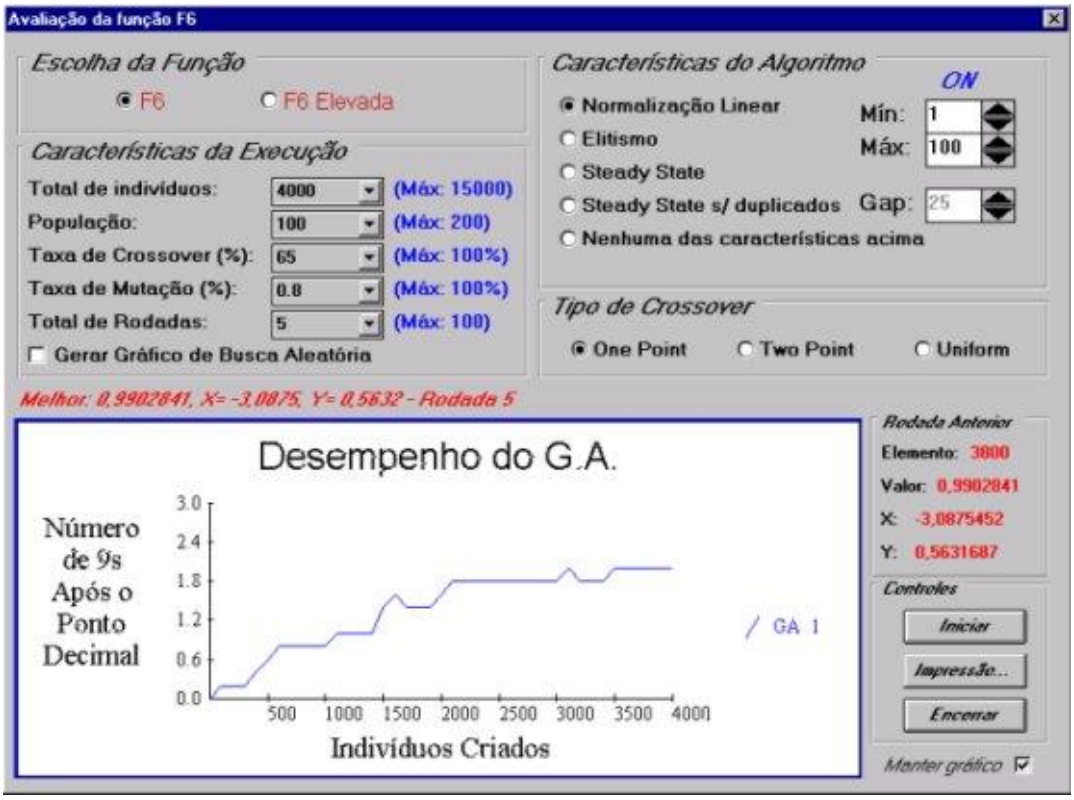


Figura A.1.1 Tela do AG para maximização da função F6

$$0.5 - \frac{(\sin\sqrt{x^2 + y^2})^2 - 0.5}{(1.0 + 0.001(x^2 + y^2))^2}$$

Figura A.1.2. Função F6

A.1.2.2. Redes Neurais

O Icademo permite demonstrar as técnicas de redes neurais através de dois algoritmos chamados Hopfield e Back Propagation. A figura A.1.3 apresenta o menu de opções para seleção entre os dois algoritmos disponibilizados.

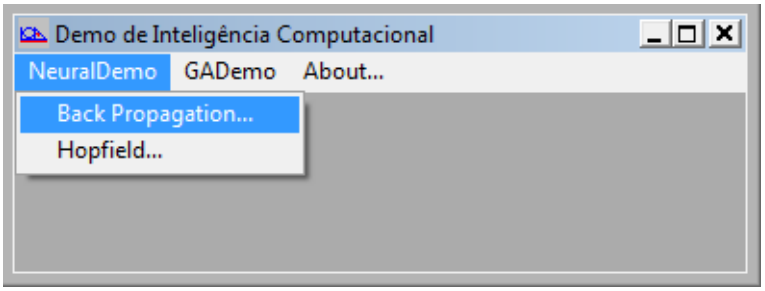


Figura A.1.3. Seleção do algoritmo para demonstração de Redes Neurais

A.1.2.2.1. O algoritmo de Hopfield

As Redes de Hopfield armazenam padrões que são recuperados a partir de estímulos de entrada. O armazenamento de padrões é realizado via Aprendizado Hebbiano. Uma característica importante deste modelo é a chamada recorrência: as saídas ligam-se as entradas por um atraso de tempo; com efeito, a resposta da rede sempre depende de seu estado anterior. O Icademo utiliza o algoritmo de Hopfield para demonstrar o reconhecimento de dígitos de 0 a 9. A entrada se dá por uma matriz de 12 linhas por 10 colunas (Figura A.1.4).

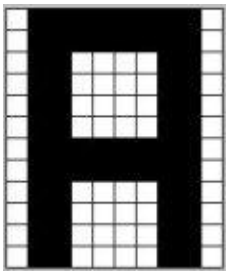


Figura A.1.4. Exemplo de matriz de entrada do algoritmo de Hopfield

O Icademo permite configurar vários aspectos do algoritmo, entre eles podemos destacar a taxa de ruído e o tipo de atualização, a figura A.1.5 apresenta a tela principal do algoritmo de Hopfield.

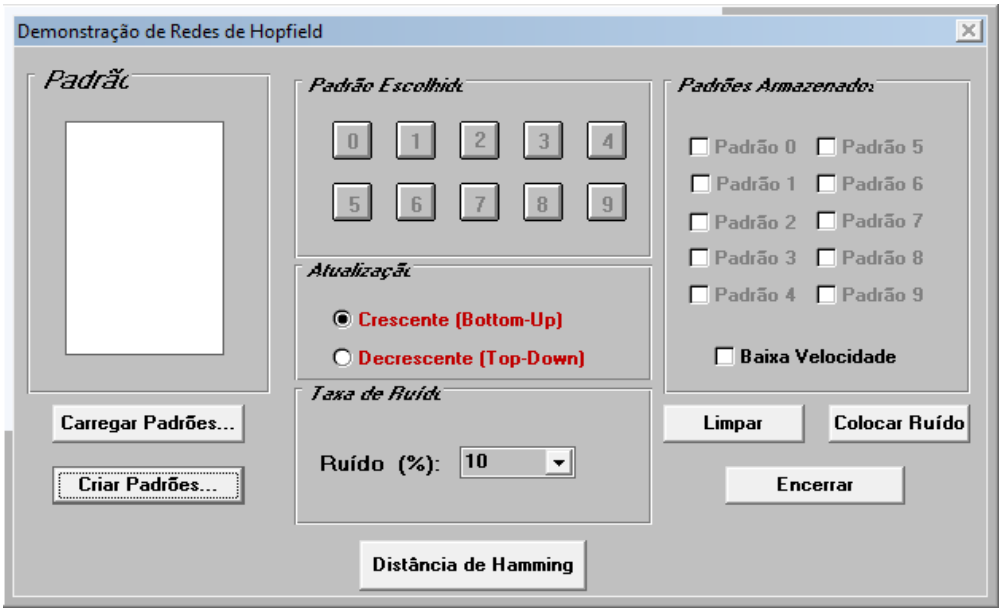


Figura A.1.5. Tela principal do Algoritmo de Hopfield

Entre as opções disponíveis ao usuário estão a possibilidade de criar padrões (Figura A.1.6) e salvá-los para uso posterior.

Figura A.1.6. Tela para entrada de padrões

B.1.2.2.2. O algoritmo Back Propagation

O Backpropagation é um algoritmo para treinamento de redes neurais do tipo perceptron com múltiplas camadas e tem como característica principal o fato de ser baseado no paradigma de aprendizado supervisionado. Conforme já comentado no capítulo 6, outra característica importante deste algoritmo é que, durante o aprendizado, ele processa em duas fases: a propagação e a retro-propagação. Na propagação o sinal

flui da esquerda para a direita e na retro-propagação, com o erro calculado, o algoritmo ajusta os pesos das conexões da direita para a esquerda. O Icademo utiliza o algoritmo Back Propagation para demonstrar o reconhecimento de dígitos de 0 a 9. A entrada da rede se dá por uma matriz de 5 linhas por 4 colunas (Figura A.1.7).

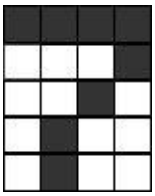


Figura A.1.7. Exemplo de matriz de entrada do algoritmo Back Propagation do IcaDemo

Assim como no algoritmo Hopfield, o Icademo permite a configuração de vários parâmetros do algoritmo, tais como o número de processadores por camada, o número de padrões e vários outros permitindo ao usuário realizar diversos experimentos no treinamento da rede neural.

A tela da figura A.1.8 mostra a possibilidade de escolha da quantidade de camadas escondidas da rede (máximo 4), assim como o número de neurônios em cada camada (máximo de 50). Convém enfatizar que o número de neurônios na camada de entrada é sempre fixo e igual a 20, uma vez que a aplicação, considera, como comentado acima, uma matriz de entrada de dimensão 5 por 4. A camada de saída também possui um número fixo de neurônios, igual a 10, por se tratarem de 10 dígitos (0 até 9).

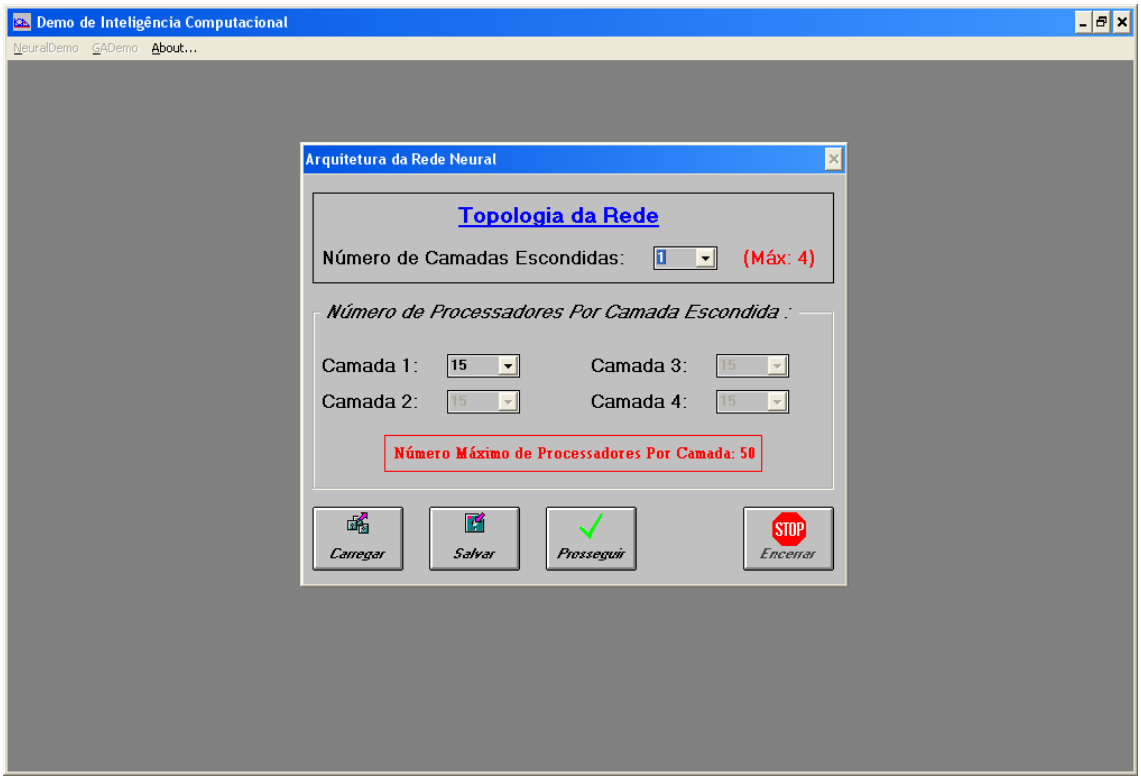


Figura A.1.8. Configuração da(s) camada(s) intermediárias da rede neural

Ao pressionar o botão "Prosseguir", o IcaDemo apresenta a tela da figura A.1.9, na qual o usuário pode carregar padrões previamente armazenados, ou digitar novos padrões. Cada padrão corresponde a um dígito informado à rede, para treinamento ou para avaliação. Ao pressionar o botão "Prosseguir", a tela da figura A.1.10 é apresentada.

Figura A.1.9. Indicação do número de padrões a serem utilizados no treinamento da rede

Na tela da figura A.1.10, o usuário pode configurar os padrões a serem utilizados no treinamento da rede neural. Para preencher um padrão, o usuário deve clicar sobre cada célula da matriz que desejar. Caso alguma célula seja erroneamente selecionada, o usuário deve clicar novamente sobre a célula indesejada, causando a sua desmarcação.

Uma vez que cada padrão tenha sido desenhado, o usuário deve selecionar qual a saída a ser ativada para o padrão correspondente. Em seguida, deve clicar sobre o botão "associar". Uma vez que tenham sido informados e associados todos os padrões a serem utilizados no treinamento da rede, o usuário deve escolher os parâmetros de aprendizado (taxa de erros, número de rodadas, taxa de aprendizado e termo de momento). Caso a rede neural já tenha sido treinada, o usuário deve clicar no botão "carregar peso" para escolher o arquivo que contenha os pesos de uma rede para o qual já tenha sido iniciado e salvo o seu treinamento. Ao final da configuração, o usuário deve pressionar a tecla "prosseguir", o que levará a abertura da tela mostrada na figura A.1.11.

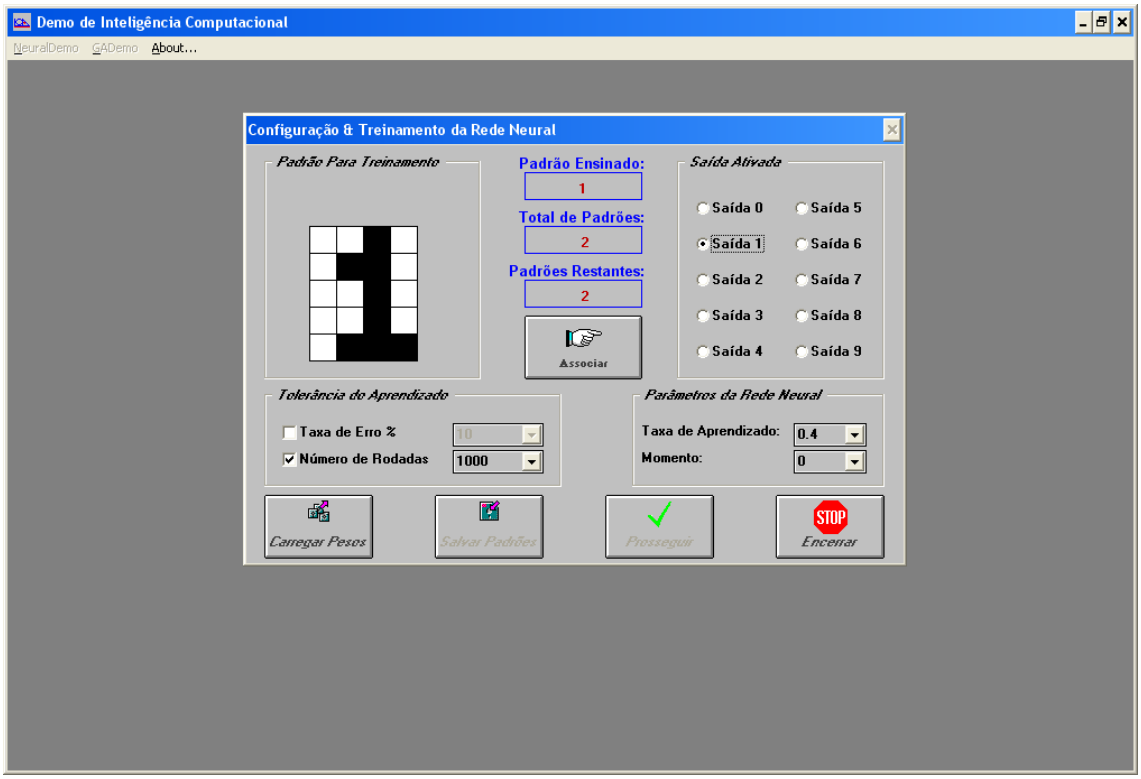


Figura A.1.10. Configuração dos padrões e parâmetros de treinamento da rede

Na tela da figura A.1.11, o usuário tem a opção de iniciar o treinamento da rede, interrompê-lo, durante seu processamento, salvar a rede neural após o treinamento e prosseguir para a etapa de avaliação da rede.

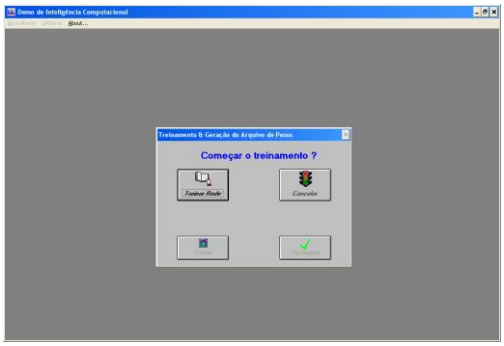


Figura A.1.11. Controle do processamento do treinamento da rede

A figura A.1.12 mostra a tela voltada à avaliação da rede neural treinada. Nesta tela, o usuário pode informar um padrão de entrada e pressionar o botão "avaliar". Neste momento, o IcaDemo apresenta o grau de ativação de cada um dos neurônios da camada de saída. No exemplo da figura, o padrão mais se assemelha ao dígito "0". A rede neural reconhece este fato ao mostrar o neurônio associado à saída "0" com maior valor de ativação. Na parte inferior da tela são apresentados dados informativos sobre o treinamento da rede.

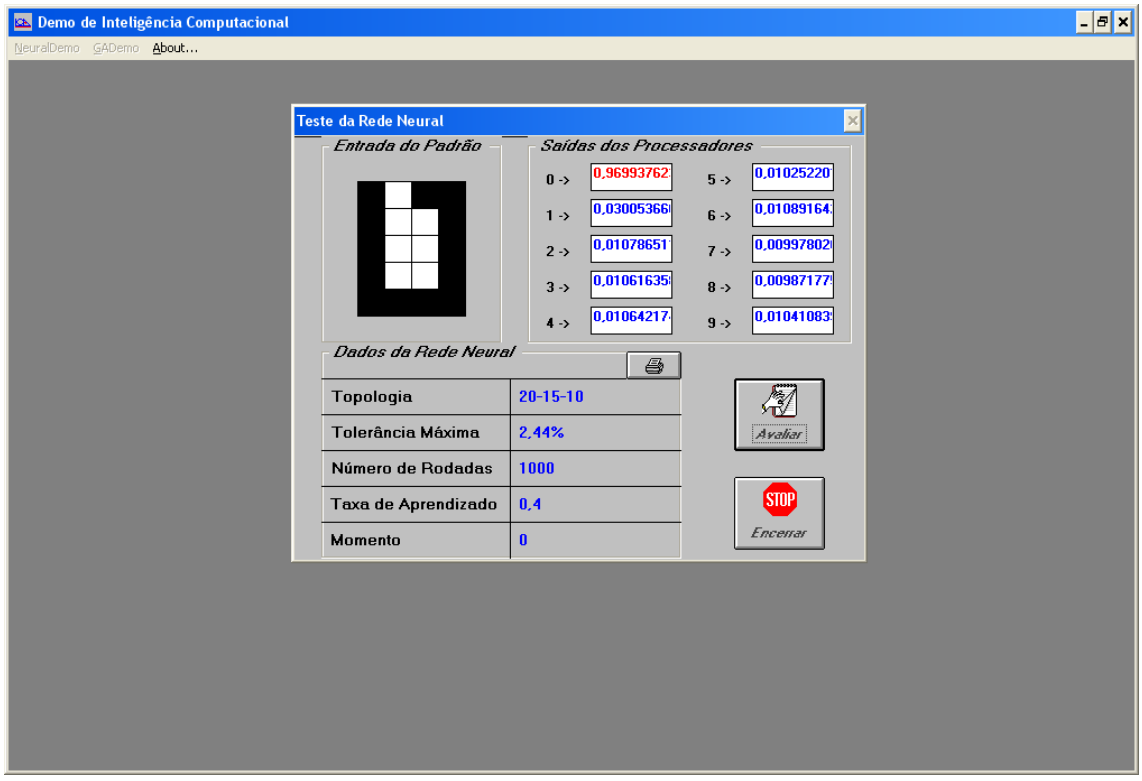


Figura A.1.12. Avaliação da rede neural treinada

A.2. SINTA

O ambiente SINTA é um software desenvolvido pela Universidade Federal do Ceará e que tem como objetivos, conforme comentado no capítulo 3, permitir a criação, a edição e o processamento de bases de conhecimento de Sistemas Especialistas.

A figura A.2.1 apresenta a tela inicial do SINTA. Neste momento, o usuário pode criar uma nova base de conhecimento ou abrir uma existente, como mostra a figura da tela A.2.2.

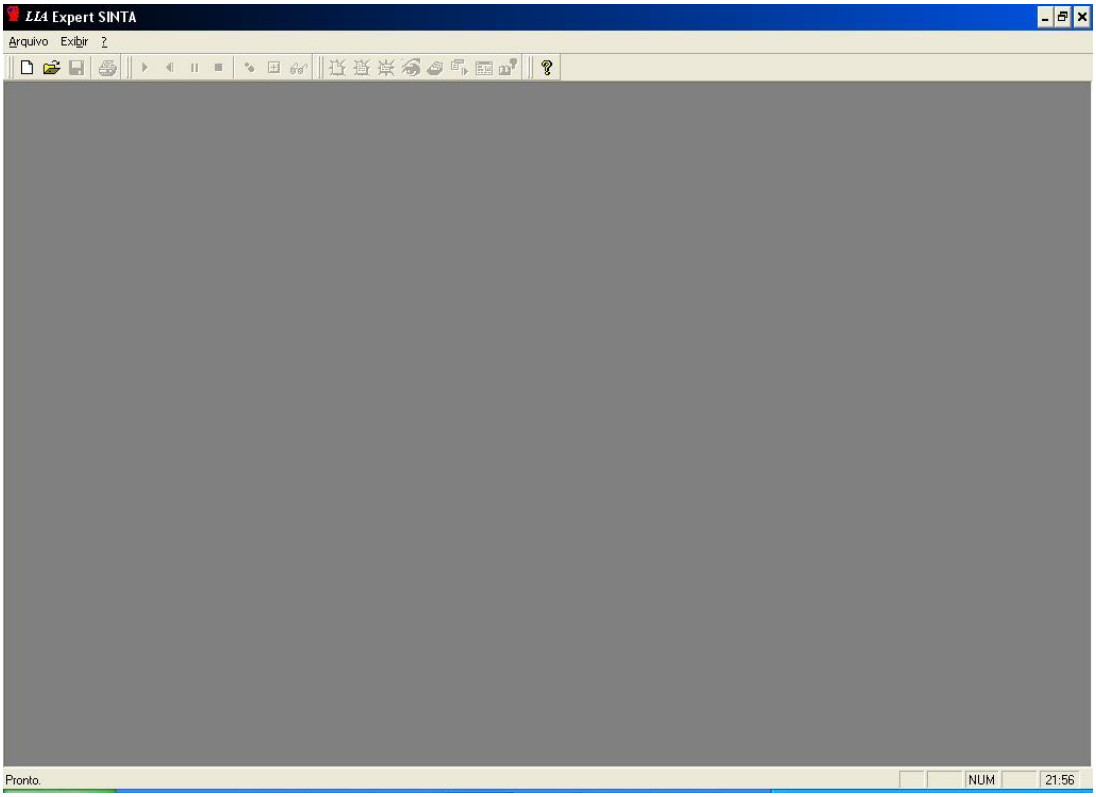


Figura A.2.1 – Tela inicial do SINTA

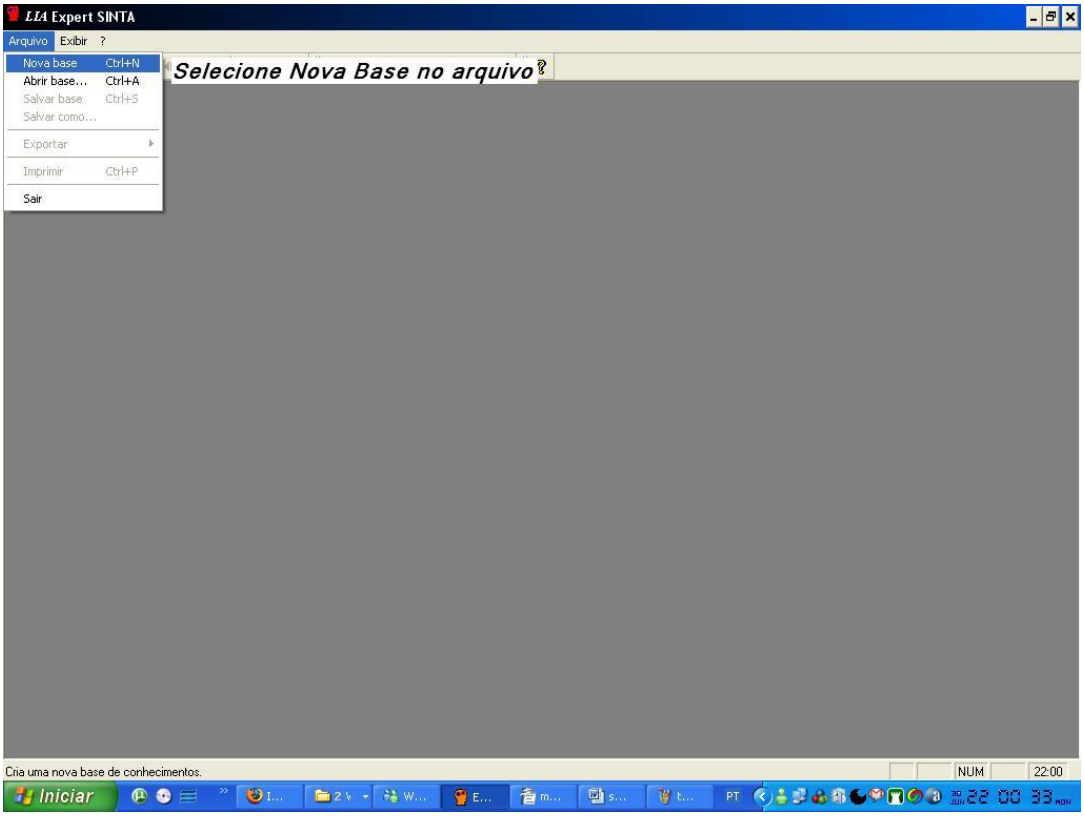


Figura A.2.2 – Selecionando nova base de conhecimento

Após criar uma nova base, o usuário pode digitar uma nova regra por meio das telas das figuras A.2.3 e A.2.4. Caso já exista alguma regra parecida com a nova regra a ser informada, o usuário pode escolhê-la como modelo. Neste caso, o SINTA faz uma cópia da regra existente (modelo) na nova regra, a ser editada.

A edição de uma regra pressupõe que as variáveis do problema tenham sido informadas. Na tela da figura A.2.5, o usuário pode criar variáveis e seus valores. Esta tela é acionada a partir da opção "Variáveis" da tela da figura A.2.3.

A tela exibida na figura A.2.6 mostra um exemplo de regra oriundo de uma base de conhecimento que já havia sido editada e que foi carregada a partir da opção "abrir base de conhecimento" do menu arquivo. Na figura A.2.7 são apresentadas as opções para edição de uma regra.

A figura A.2.8 mostra uma tela em que é possível editar uma cláusula (condição ou predicado) de uma regra.

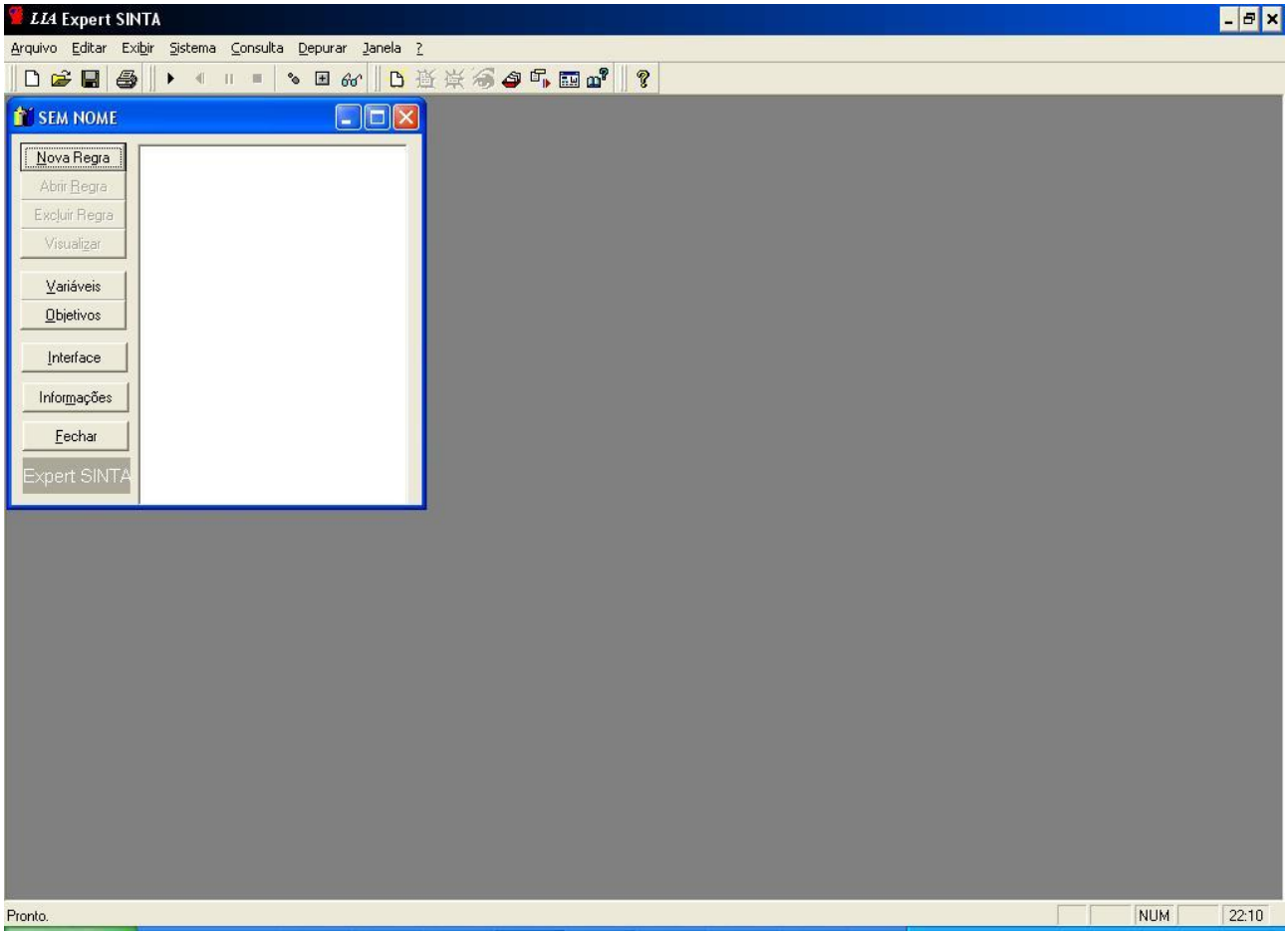


Figura A.2.3 – Criando nova regra

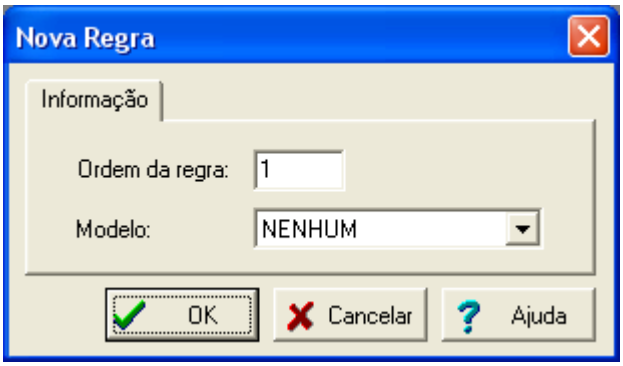


Figura A.2.4 – Criando nova regra

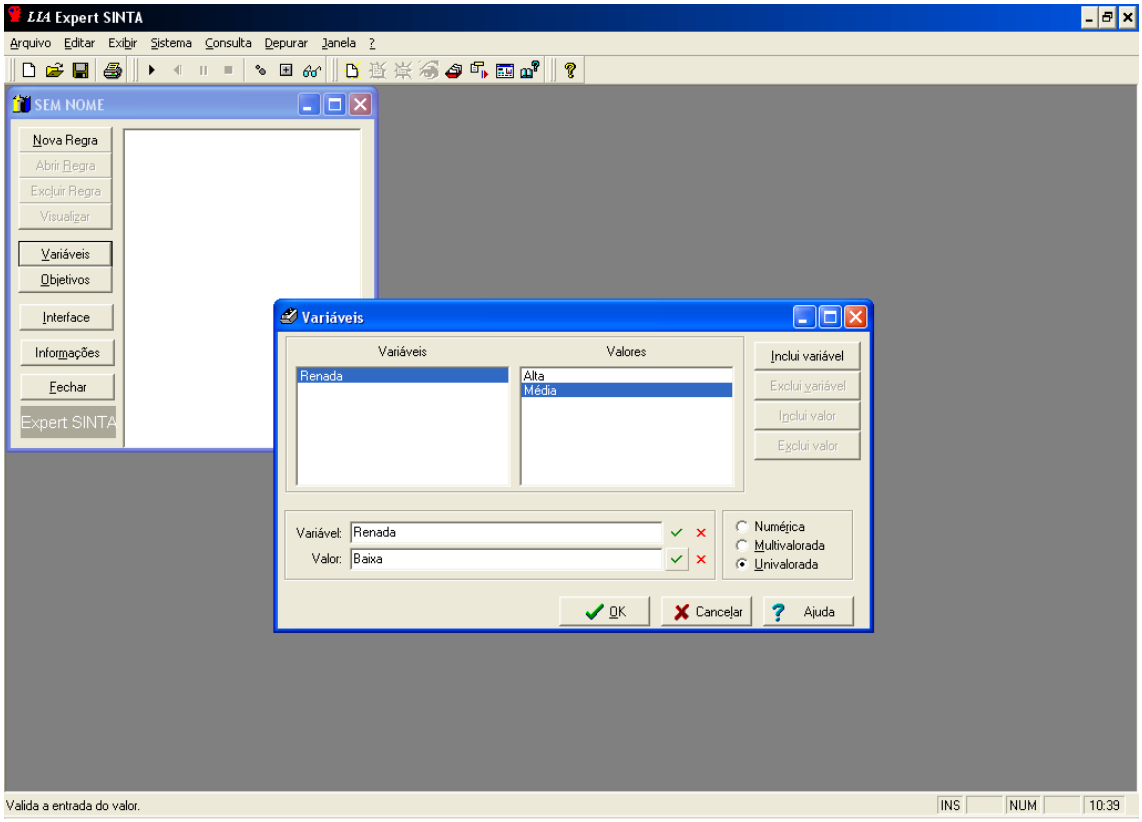


Figura A.2.5 – Exemplo de Criação de Variável e seus Valores

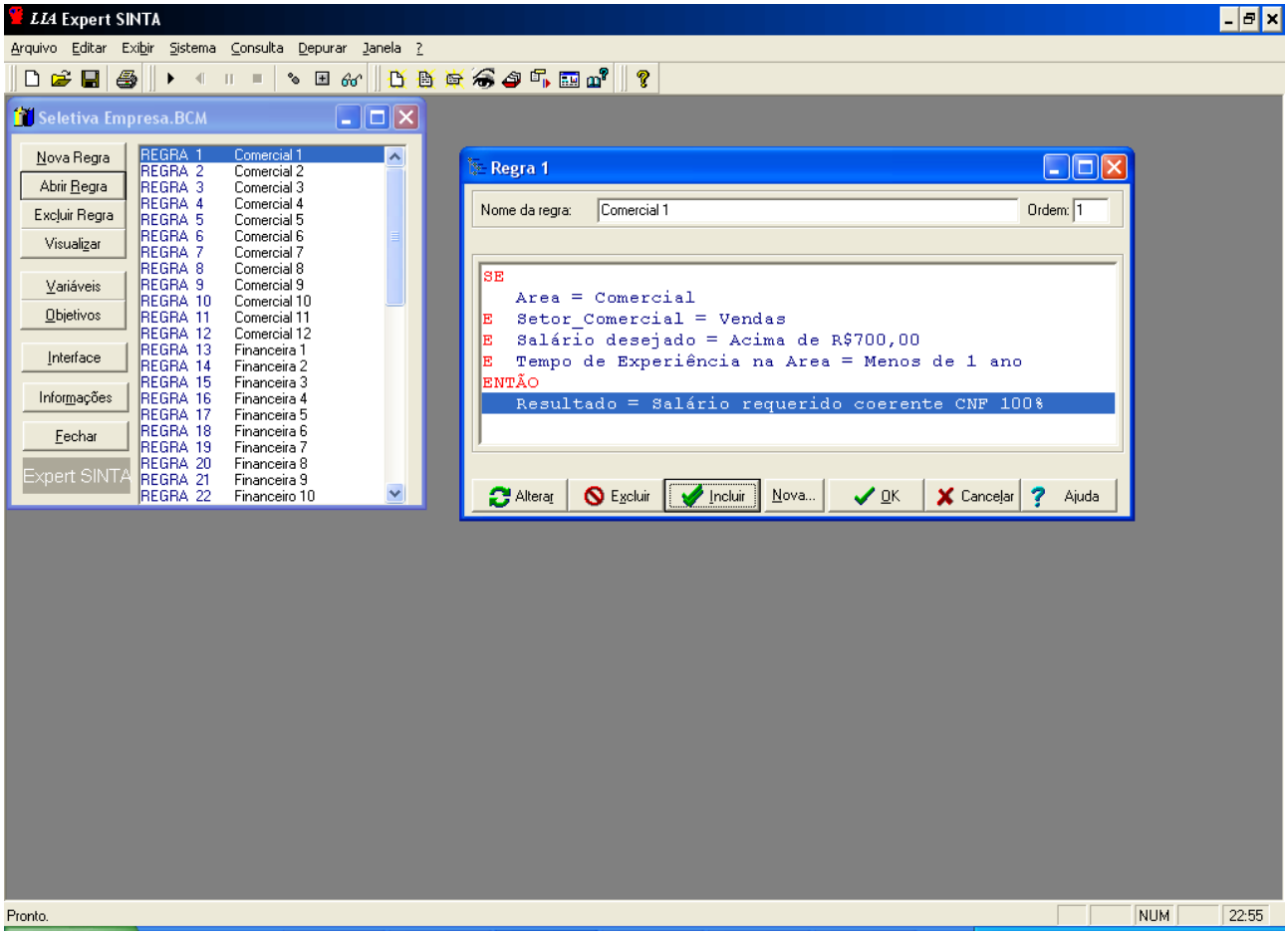


Figura A.2.6 – Exemplo de Regra de Produção já editada

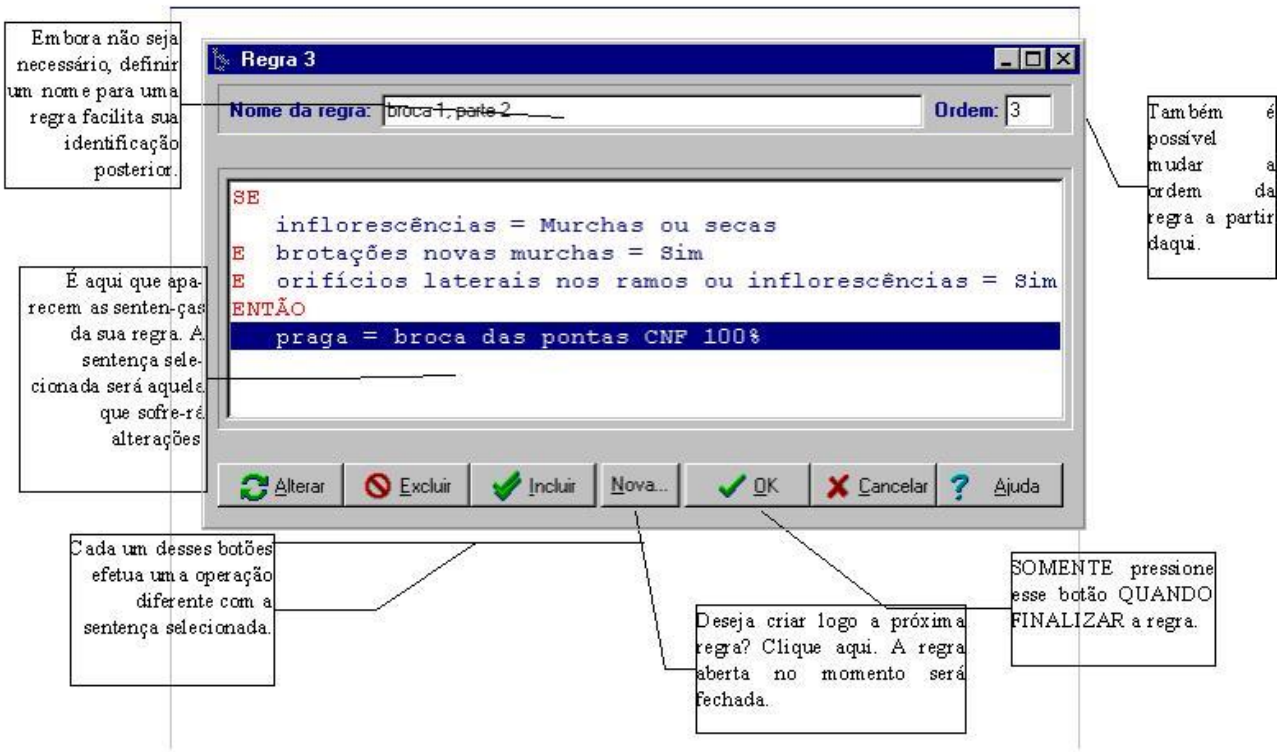


Figura A.2.7 – Opções para edição de regra do SINTA

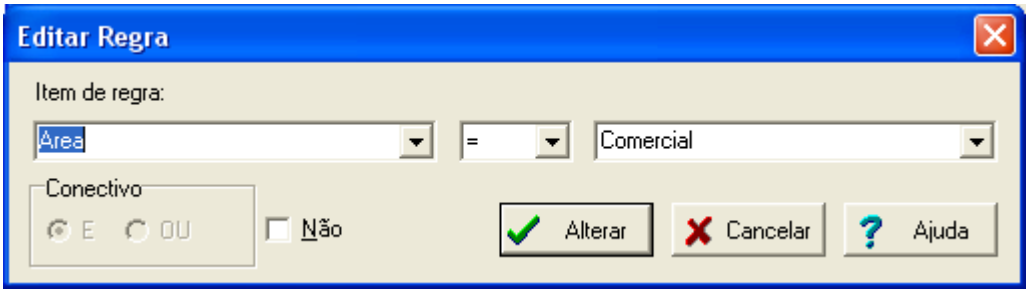


Figura A.2.8 – Edição de uma cláusula de uma regra.

O usuário poderá, a qualquer momento, excluir regras. Para tanto, é só selecionar a regra desejada e, em seguida, a opção "Excluir regra" mostrada na tela da figura A.2.9.

O usuário pode, também, visualizar todas as regras que envolvem determinadas variáveis, além de poder imprimi-las, se assim desejar. Para fazer isso, o usuário deverá escolher a função "Visualizar" na tela da figura A.2.9, sendo mostradas as telas das figuras A.2.10 e A.2.11.

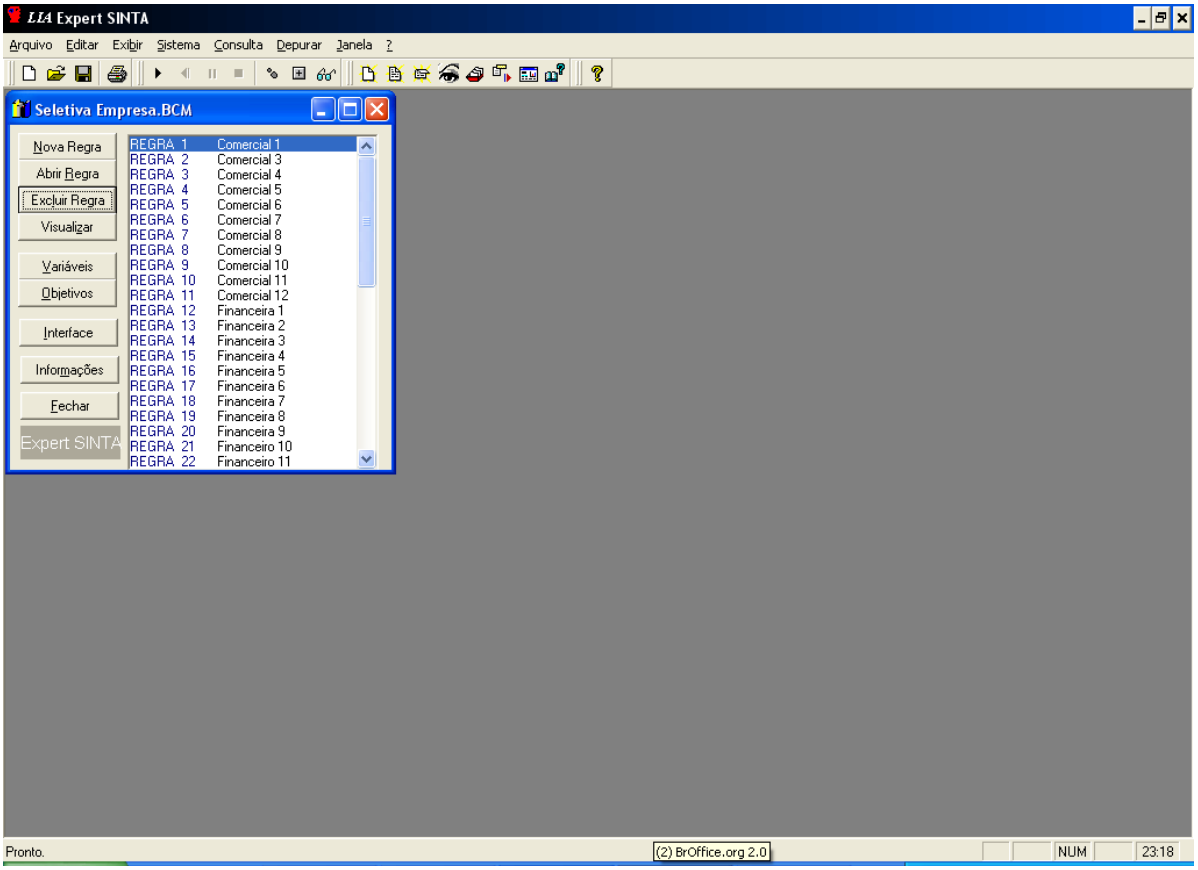


Figura A.2.9 – Possibilidade de exclusão de regra

Figura A.2.10 – Escolha de variáveis para visualizar regras

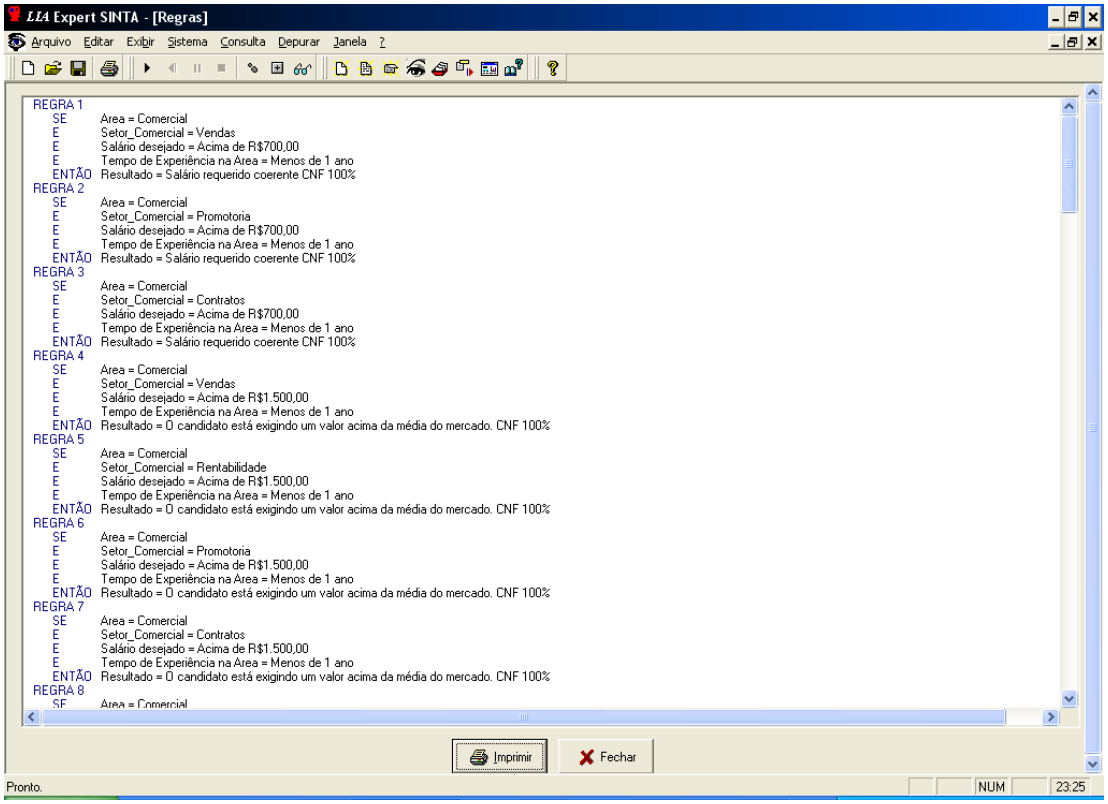


Figura A.2.11 – Visualização de regras.

O usuário pode selecionar os objetivos da base de conhecimento, mas para isso, é preciso que definamos quais das variáveis serão consideradas como objetivos. Para tanto, o usuário deve selecionar na tela da figura A.2.12 a variável desejada e clicar no botão correspondente (aquele que aponta para a lista de destino). Esta tela pode ser acessada ao pressionar "objetivos" da tela da figura A.2.3. A lista com uma seta dupla (seja para a esquerda ou para direita) move todos os itens de uma lista para outra. Para mudar a ordem dos elementos da lista de objetivos, o usuário deve clicar em um item e arrastá-lo até a posição desejada.

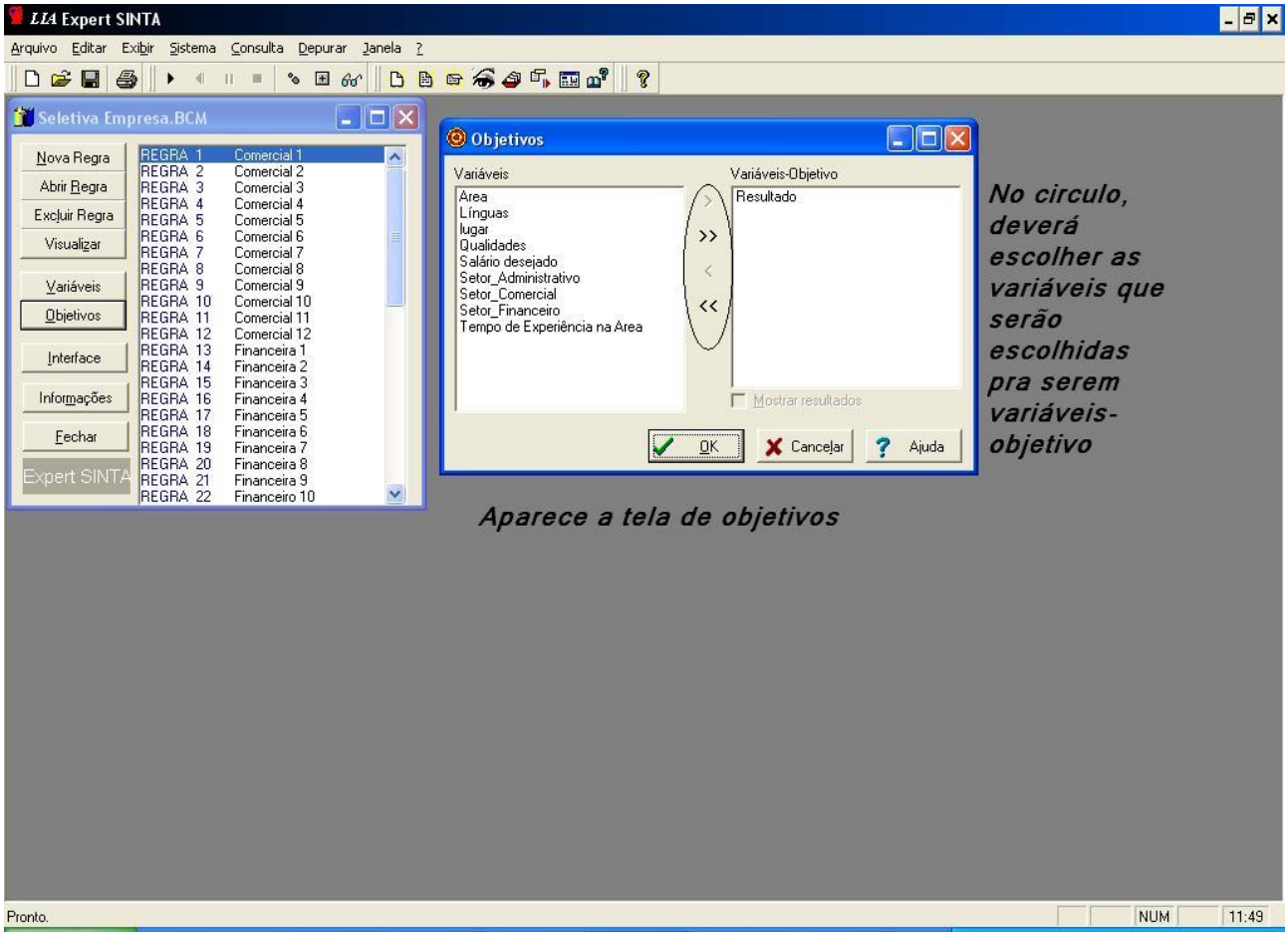


Figura A.2.12 – Escolha dos Objetivos de uma Base de Conhecimento

A fim de tornar o processamento da base de conhecimento mais fácil de ser realizado, o usuário poderá configurar as perguntas a serem realizadas para cada variável do problema. Para isso, deverá selecionar a opção interface, disponível na tela da figura A.2.3, o que acionará uma nova tela (figura A.2.13) contendo as perguntas associadas às variáveis da base de conhecimento. Uma vez configuradas as perguntas, o processamento da base de conhecimento as apresentará sempre que for necessário descobrir os valores das variáveis correspondentes.

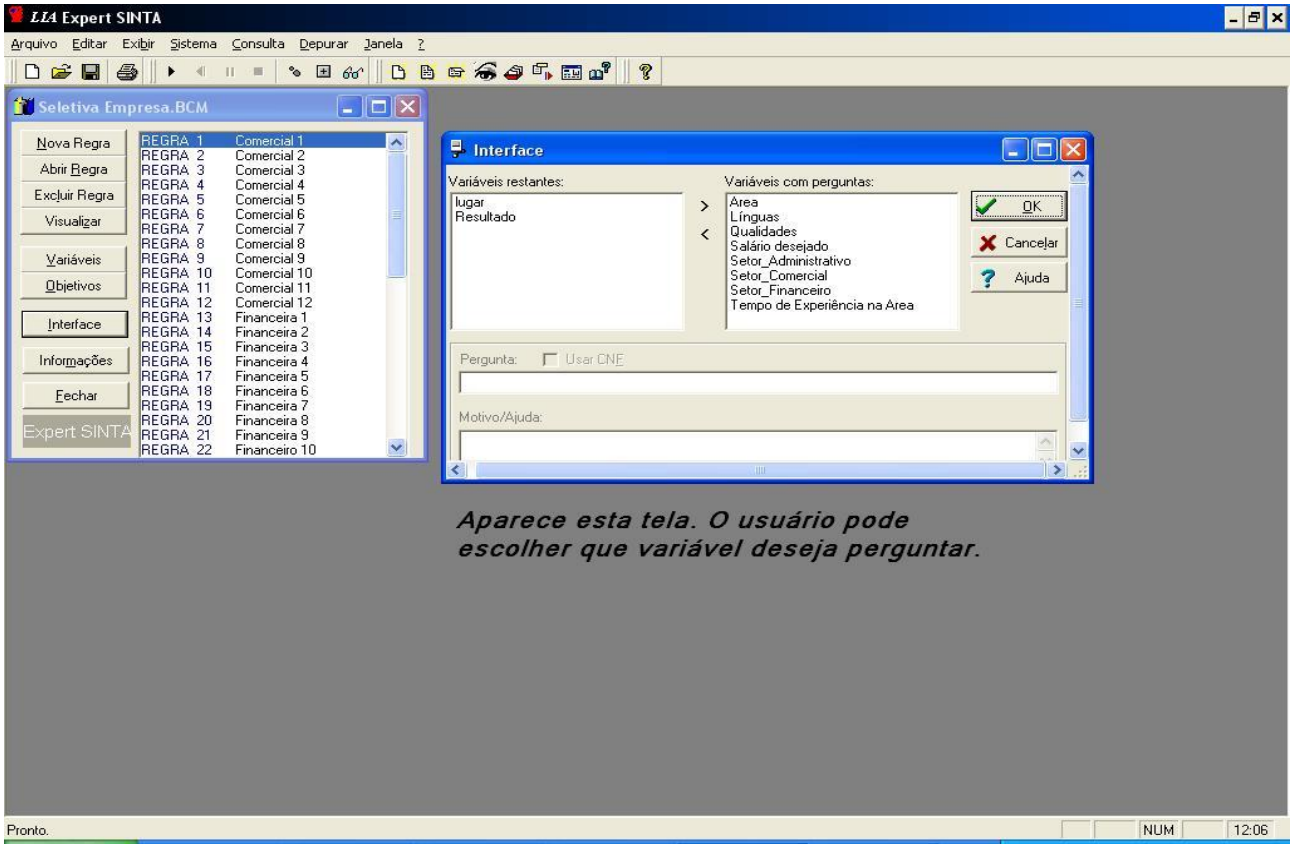


Figura A.2.13 – Tela para Configuração das Perguntas Associadas às Variáveis

Uma vez concluída a edição de uma base de conhecimento, o ambiente estará pronto para realização de consultas. Cada consulta envolve o processamento da base de conhecimento utilizando o encadeamento para trás. Isso significa que o SINTA perguntará ao usuário a respeito de todas as variáveis necessárias para concluir algo sobre as variáveis definidas como objetivos. Para iniciar uma nova consulta, o usuário deve pressionar a opção "iniciar" que está na barra de ferramentas do SINTA, conforme mostra a tela da figura A.2.13.

Ao final do processamento de uma consulta, o usuário tem a opção de verificar como o sistema chegou ao resultado final. A tela da figura A.2.14 ilustra essa possibilidade (aba "histórico").

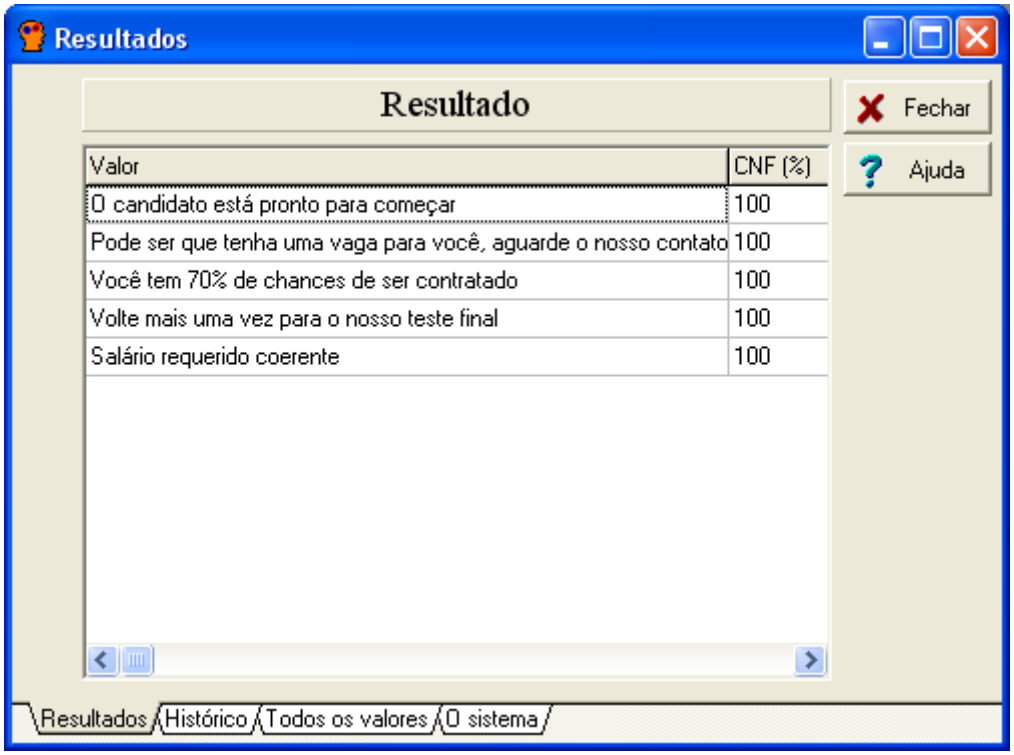


Figura A.2.14 – Tela apresentada ao final do processamento de cada consulta

A.3. SEGSE – Sistema Especialista Gerador de Sistemas Especialistas

O SEGSE, conforme o próprio nome indica, é um sistema especialista para auxiliar na criação de sistemas especialistas. Maiores detalhes sobre sua estrutura interna podem ser obtidos no capítulo 3. A seguir, encontra-se um passo a passo do seu funcionamento.

Para criar um novo projeto de Sistema Especialista, o usuário deverá fazer o seguinte procedimento a partir da tela da figura A.3.1: arrastar o mouse para o menu "Projeto" e selecionar "Novo".

Figura A.3.1 – Gerando um novo projeto de Sistema Especialista.

Conforme um novo projeto é criado, uma série de perguntas será feita ao usuário para a configuração do novo sistema especialista a ser gerado. Tais perguntas são sempre apresentadas em um mesmo formato de tela, conforme ilustra a figura A.3.2. O conteúdo das perguntas é o resultado de um processo de aquisição de conhecimento sobre como construir sistemas especailistas. Tal processo é transparente para o usuário final do SEGSE, assim como para os usuários finais dos Sistemas Especialistas gerados.

Figura A.3.2 – Um exemplo de pergunta quando há um novo projeto.

Se o usuário desejar abrir um projeto existente, deverá fazer o seguinte: Na tela da figura A.3.3, arrastar o mouse na opção "Arquivo", e escolher a opção "Abrir". Neste momento, aparecerá uma tela (figura A.3.4) onde o usuário deverá escolher o arquivo que deseja abrir. Os arquivos que serão abertos (ou salvos) estão na extensão "SEG" ou em "XML". Para salvar o arquivo, o usuário deve escolher a pasta de destino, o nome do arquivo e o tipo, "SEG" ou "XML".

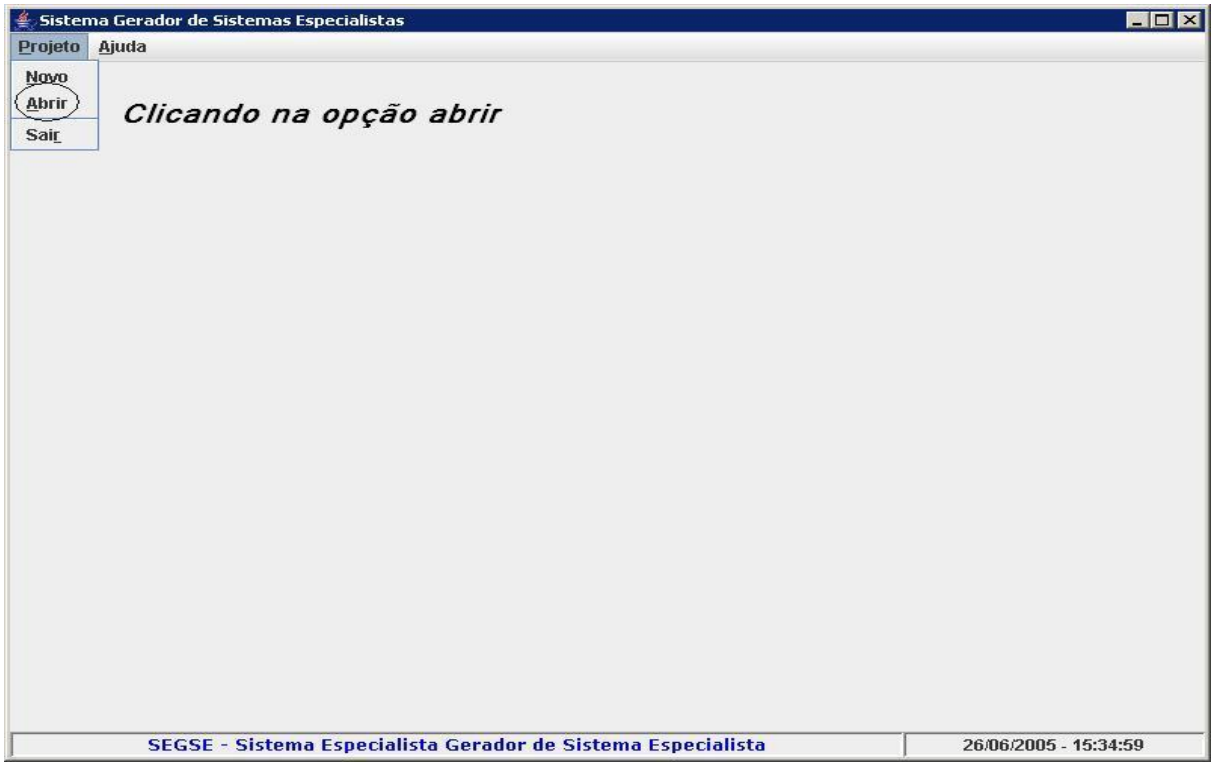


Figura A.3.3 – Escolhendo a opção "Abrir" (Projeto Existente).

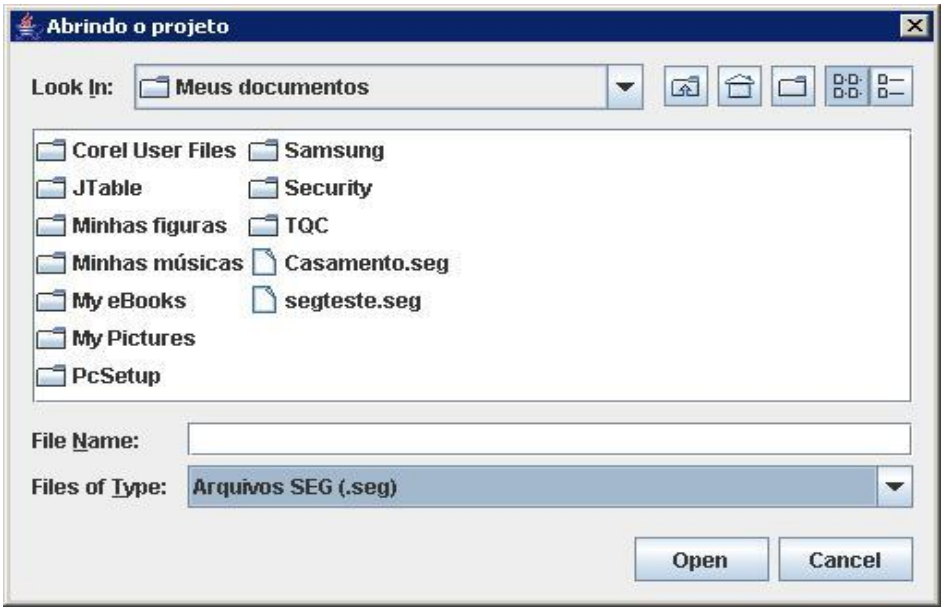


Figura A.3.4 – Escolhendo o arquivo a ser aberto.

Na sequência, o usuário deve preencher os seguintes dados informativos sobre o projeto: nome, descrição e autor, conforme mostra a tela da figura A.3.5.

Figura A.3.5 – Dados do projeto de sistema especialista.

Após, o usuário deverá escolher as opções que estão na barra de ferramentas abaixo do preenchimento dos dados na tela da figura A.3.5. Ao selecionar a opção “atributos”, o SEGSE exibe a tela da figura A.3.6.

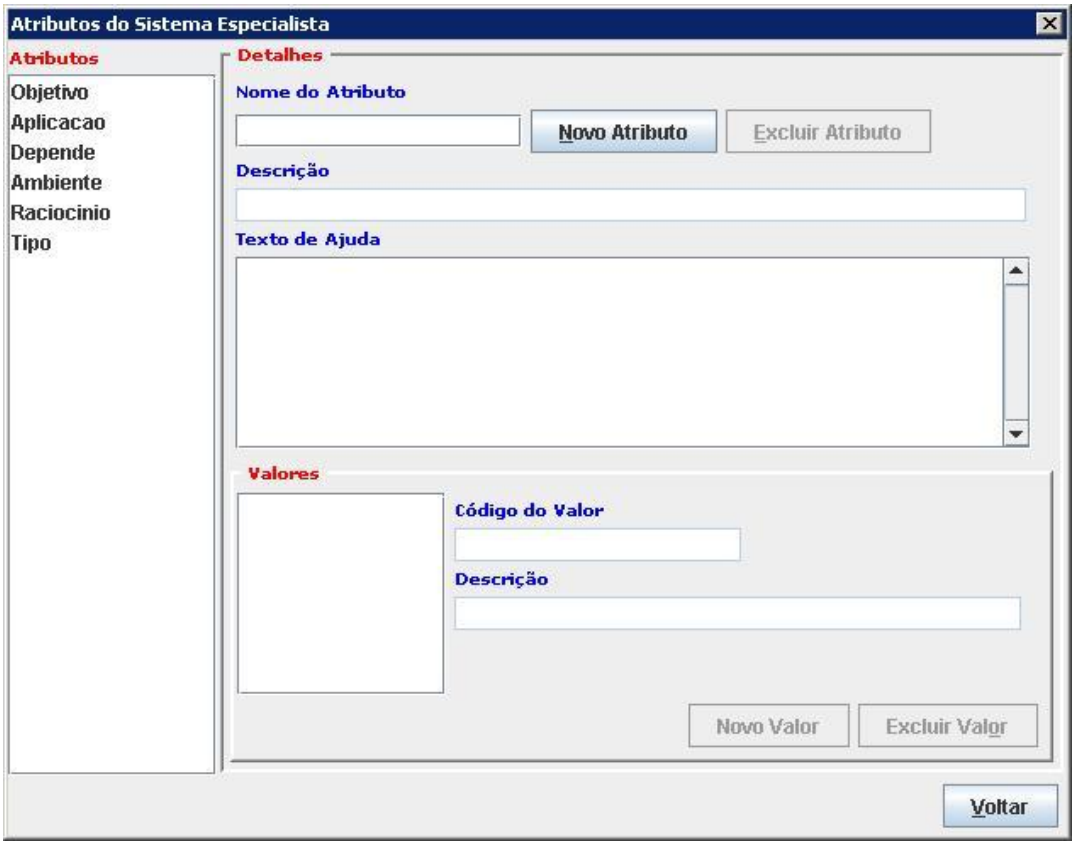


Figura A.3.6 – Edição dos Atributos do SE a ser gerado.

Na tela A.3.6, o usuário deverá digitar o nome do atributo e pressionar o botão "novo atributo". Este verifica se o nome do atributo já existe na listagem de atributos. Caso não exista, será efetuada a validação de atributo por meio de uma série de perguntas, conforme a ilustrada na figura A.3.2. Caso tal atributo não seja validado, o SEGSE solicitará ao usuário que seja informado outro atributo. Uma vez validado, o atributo será liberado para edição. Neste momento, o usuário deverá preencher os campos descrição do atributo do SE e o texto de ajuda.

Ainda na tela A.3.6, no campo "Valores", o usuário digitará cada valor associado ao atributo em questão. Após a digitação de cada valor, o usuário deverá selecionar o botão "novo valor". Neste momento, novamente o SEGSE realizará uma série de perguntas para considerar como válido o valor informado. Caso deseje excluir um valor informado erroneamente, basta o usuário selecionar a opção "Excluir valor", escolher um valor que desejará excluir e confirmar a solicitação de exclusão.

Para o usuário acessar os objetivos do SE, ele deverá selecionar a opção "objetivos" na tela da figura A.3.5. Em seguida aparecerá a tela da figura A.3.7 para escolha dos objetivos do SE.

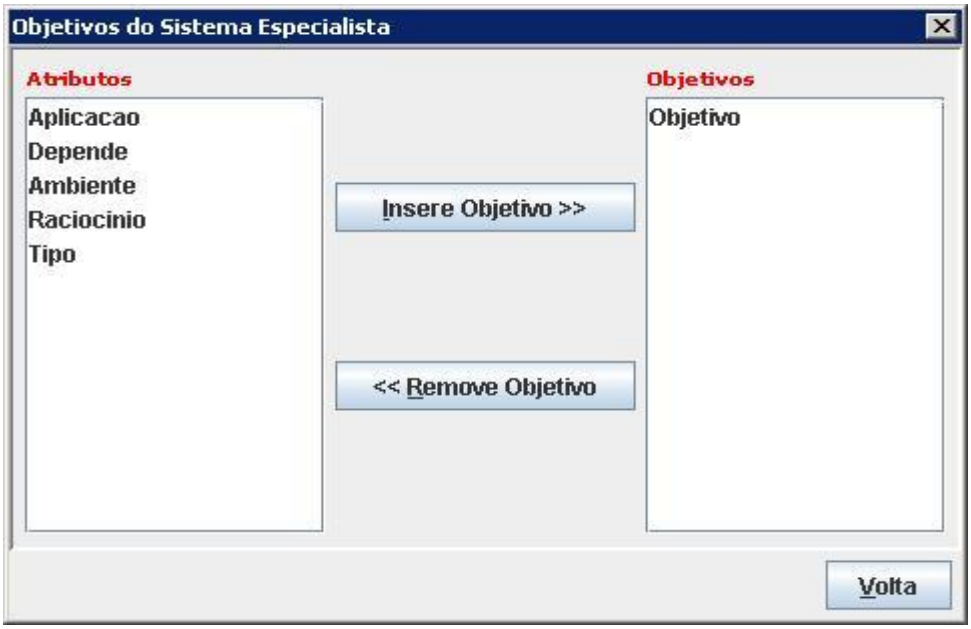


Figura A.3.7 – Seleção dos Objetivos do SE a ser gerado.

A tela da figura A.3.7 mostra o conjunto de atributos que foram previamente criados no formulário de atributos do projeto. Apenas um atributo pode ser escolhido para ser definido como objetivo. Para inserir um objetivo, o usuário deverá selecionar um dos atributos na seção "Atributos" e, em seguida, pressionar o botão "insere objetivo". Neste momento, o SEGSE apresenta uma série de perguntas para validação do objetivo. Uma vez validado o atributo como objetivo, o sistema o transfere para a lista de "objetivos". Em caso de erro, o usuário pode, de maneira análoga, remover o atributo da lista de objetivos.

Para o usuário configurar o conhecimento a ser incorporado no SE a ser gerado, ele deverá selecionar a opção "conhecimento" na tela da figura A.3.5. Em seguida aparecerá a tela da figura A.3.8 para que o usuário informe, para cada combinação de valores dos atributos, qual deverá ser a resposta do SE. Convém mencionar que a quantidade linhas varia de acordo com o total de atributos e valores informados. Deve ser associado um valor de objetivo para cada linha. Esta tabela deve ser preenchida pelo especialista, que tem conhecimento para reconhecer qual o valor de objetivo é mais adequado para cada linha.

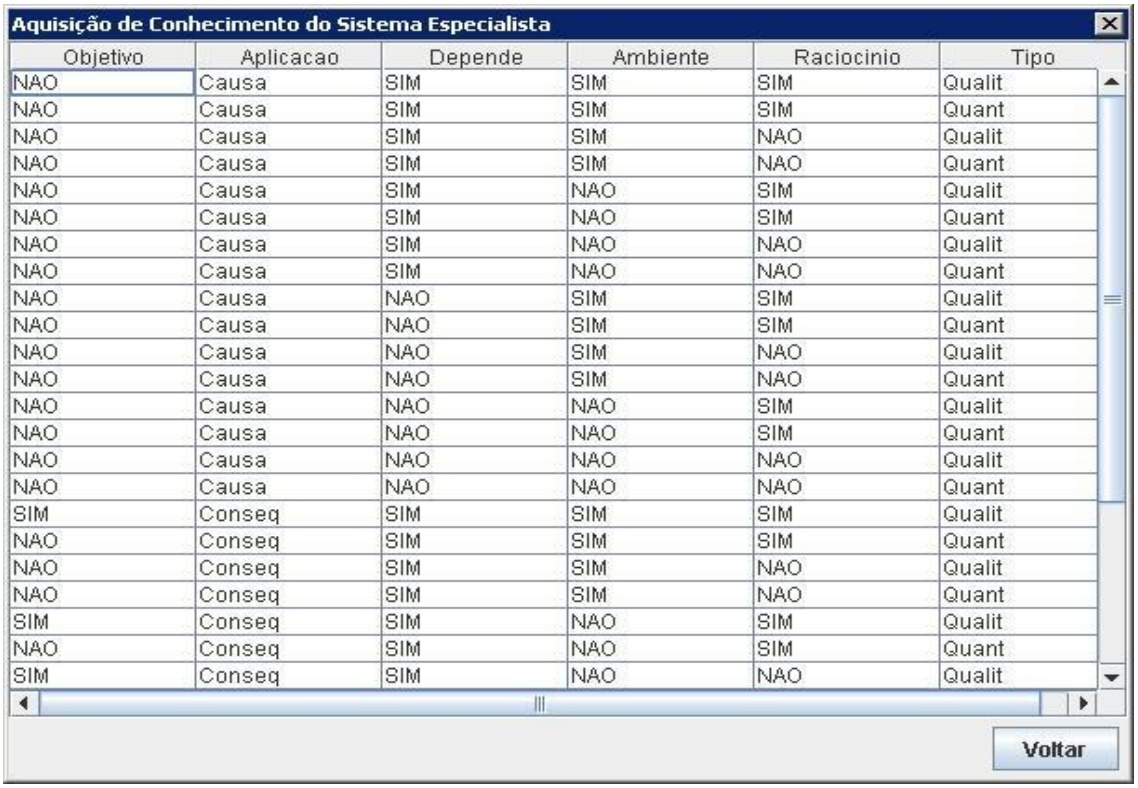


Figura A.3.8 – Tela de aquisição do conhecimento

Após concluídas todas as etapas indicadas, o usuário poderá gerar o Sistema especialista. Para tanto, o usuário deverá pressionar o botão "Gerar SE" da tela da figura A.3.5. Em seguida, será apresentada a tela da figura A.3.9.

Figura A.3.9 – Tela de geração do SE

O usuário poderá escolher a opção “visualizar regras” da tela da figura A.3.9. Tal opção mostrará a tela de visualização das regras geradas (figura A.3.10).

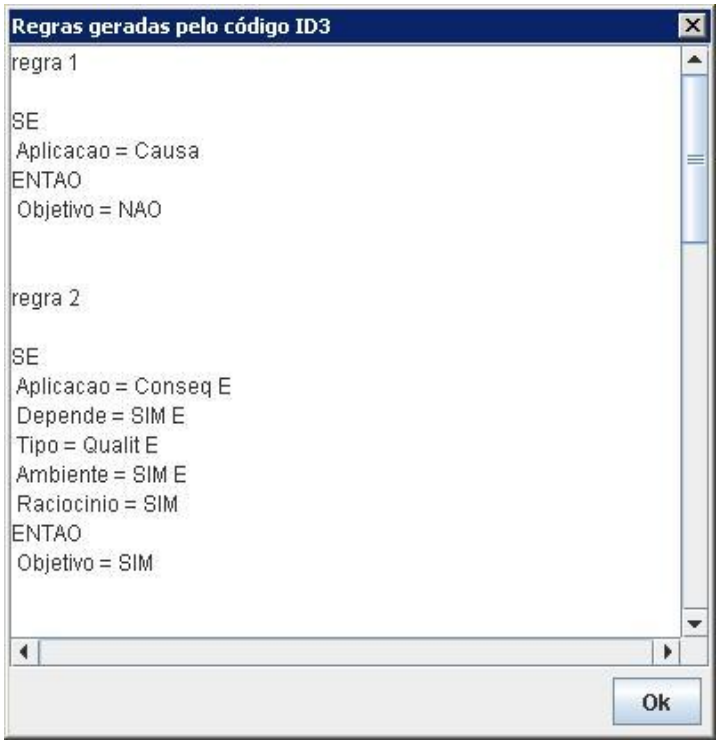


Figura B.3.10 – Visualização de Regras

Se o usuário desejar salvar o trabalho, deverá acionar a opção correspondente na tela A.3.5. É importante mencionar que os arquivos do sistema especialista gerado são armazenados na mesma pasta em que o projeto foi salvo.