# INTELIGÊNCIA ARTIFICIAL

**Vinicius Ponte Machado**

# APRESENTAÇÃO

Desde início do uso do computadores pelo homem existe uma tentativa de aproveitar a capacidade de processamento das máquinas para que esta simule o comportamento humano. Durante quase 60 anos as pesquisas neste sentido desenvolveram um ramo da ciência da computação que ficou conhecida como inteligência Artificial(IA).

Nesta apostila veremos os principais conceitos, técnicas e aplicações da Inteligência Artificial, mostrando como a ciência da computação e outras áreas do conhecimento humano influenciam a implementação dos métodos da IA. Este material é baseado em textos e referências da Internet de diversos autores. É extremamente recomendável ao aluno que, ao final da leitura de cada capítulo, acesse os materiais indicados na Weblografia e nas Referências Bibliográficas principalmente, para aprofundar a leitura desse material e complementar o que foi lido aqui.

A apostila é divida em seis unidades. Na primeira mostraremos os principais conceitos e definições de termos usados em Inteligência Artificial além de mostrar um breve histórico da IA desde seu início até a atualidade. Na unidade dois falaremos resolver problemas através de buscas. Mostraremos ainda como um problema deve ser formulado e como a usar a informação contida no próprio problema para resolvê-lo. Na unidade três veremos as principais técnicas de como o computador aprende através da chamada aprendizagem de máquina. Já na unidade quatro, falaremos sobre os principais modelos de representar e manipular o conhecimento. Na unidade cinco, o principal tópico a ser discutido será o paradigma de desenvolvimento de software chamado agentes e sua aplicação distribuída, os sistemas multiagentes. Por fim, mostraremos os princípios da robótica e suas aplicações.

**Boa Leitura !!**

Vinícius Machado

# SUMÁRIO

Corte Alpha-Beta

## UNIDADE 3. Aprendizagem de Máquina

Aprendizagem de Máquina.

Aprenizado Supervisionado

Exemplo: Abordagem com Base em Árvores de Decisão

Algoritmo ID3

Aprenizado Não-Supervisionado

Exemplo: Clusterização Conceitual

COBWEB

Aprendizagem por Reforço.

Elementos Básicos da Aprendizagem por Reforço

Exemplo: Jogo da Velha

Aplicando AR ao jogo da velha

Redes Neurais

O Neurônio Artificial

Treinamento de um Perceptron

Backpropagation

Treinamento do MLP

Por Que Utilizar Redes Neurais?

Aplicações de Redes Neurais Artificiais

Algortmos Genéticos.

Conceitos Básicos

Funcionamento do Algoritmo

Seleção dos mais Aptos

Parâmetros Genéticos

Aplicações

## UNIDADE 4. Representação do Conhecimento

Aquisição do Conhecimento

Métodos de Aquisição do Conhecimento

Entrevistas não Estruturadas

Entrevistas Estruturadas

Estudo de Caso
Descoberta de Conhecimento em Base de Dados
Métodos de Representação de Conhecimento
Lógica
Redes
Frames
Árvores de Decisão
Ontologias

## UNIDADE 5. Agentes Inteligentes

Definições
Ambientes de Agentes
Tipos de Agentes
Agente Reativo Simples
Agente Reativo com Estado Interno
Agentes Baseados em Objetivos
Agente Baseado em Utilidade
Sistemas Multiagente
Inteligência Artificial Distribuída (IAD)
Sistemas Multiagente
Comunicação e Coordenação em Sistemas Multi-Agentes

## UNIDADE 6. Introdução a Robótica

Definições
Tipos de Robôs
Organização Funcional de um Robô
Dispositivos de Robótica
Atuadores
Linguagem de Programação de Robôs
Robótica e Inteligência Artificial
Aplicações

# UNIDADE 1

## HISTÓRICO & CONCEITOS EM IA

## 1. Conceitos Gerais

A inteligência artificial surgiu na década de 50 com o objetivo de desenvolver sistemas para realizar tarefas que, no momento são melhor realizadas por seres humanos que por máquinas, ou não possuem solução algorítmica viável pela computação convencional.

Geralmente os sistemas de IA podem ser divididos em quatro categorias (

Figura 1):

- Sistemas que raciocinam de forma semelhante a dos seres humanos
- Sistemas que pensam de forma racional
- Sistemas que agem como os seres humanos
- Sistemas que agem de forma racional

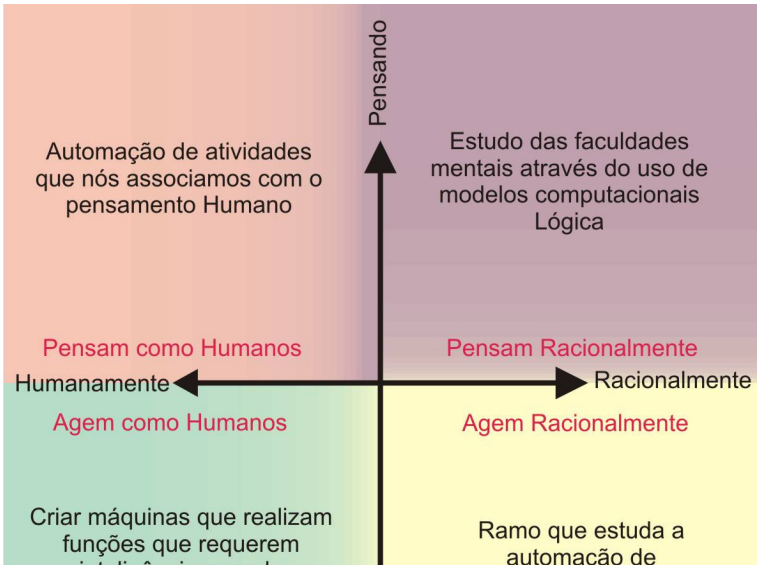


Figura 1. Categorias dos sistemas de IA

## 1.1. Sistemas que Agem como Seres Humanos

### 1.1.1. Teste de Turing

O teste de Turing, proposto por Alan Turing, foi desenhado de forma a produzir uma definição satisfatória de inteligência; Turing definiu um comportamento inteligente como sendo a habilidade de um sistema alcançar um desempenho ao nível de um ser humano em todas as tarefas cognitivas, de forma a conseguir enganar uma pessoa que o estivesse a interrogar..

O teste consistia num computador ser interrogado por uma pessoa, sem que esta estivesse vendo que estava "conversando" com um computador ou não (Figura 2).

O computador passaria no teste se a pessoa não conseguisse identificar se estava falando com um computador ou com outro ser humano.

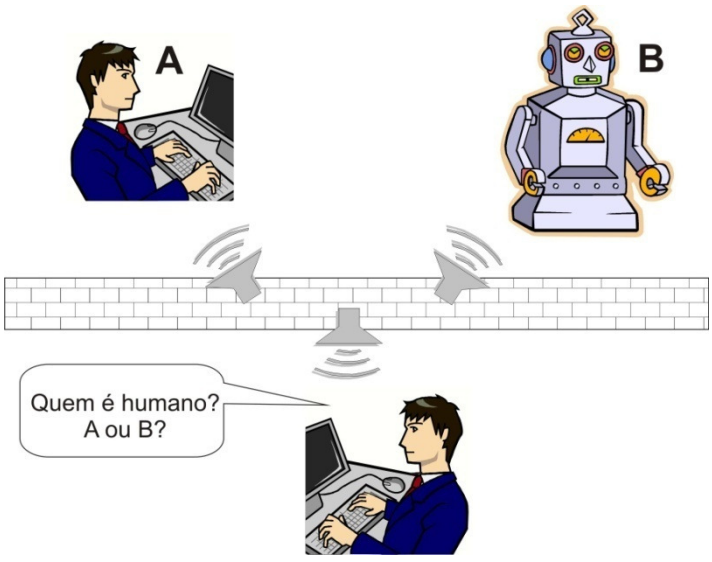



Figura 2. Teste de Turing

Requisitos de um sistema para executar o teste de Turing:

- Capacidade de processar uma linguagem natural;
- Capacidade de representar o conhecimento (o sistema deverá ser capaz de guardar toda a informação fornecida antes e durante o interrogatório);
- Dispor de uma forma de automatizar o raciocínio de forma a usar a informação guardada para responder às questões e inferir novas conclusões;
- Capacidade de se adaptar a novas circunstâncias e de detectar padrões (aprendizagem - machine learning).

Vale ressaltar que até hoje, nenhuma máquina conseguiu passar no teste de Turing, premissa para que uma sistema agisse como ser humano.

## 1.2. Sistemas que Raciocinam de Forma Semelhante à dos Seres Humanos

A principal questão dos sistemas que raciocinam com seres humanos é justamente entender como raciocinam os seres humanos. Pesquisadores nesta área usam a introspecção (tentativa de "pegar" os próprios pensamentos à medida que estes vão fluindo) e as experiências psicológicas como teorias para desenvolver tais sistemas.

Dispondo de teorias suficientemente precisas acerca do funcionamento da mente humana, torna-se possível expressar essas teorias num programa de computador.

Se as entradas e as saídas de um programa corresponder ao comportamento humano, dispomos de uma evidência de que alguns dos mecanismos do programa podem estar funcionando como nos seres humanos.

Newell e Simon que desenvolveram o GPS (General Problem Solving) não se contentavam com que o seu programa resolvesse os problemas de forma correcta. Para eles era mais importante comparar os passos de raciocínio seguidos pelo programa com os passos seguidos por várias pessoas na resolução dos mesmos problemas [Newell & Simon, 1961].

## 1.3. Sistemas que Pensam de Forma Racional

Em 1965, já existiam programas que podiam, dado tempo e memória suficientes, buscar na descrição de um problema, em notação lógica e encontrar uma solução para esse mesmo problema, caso esta existisse. Se não houvesse solução o programa poderia nunca parar de procurar. No entanto, não é fácil traduzir conhecimento informal em lógica formal, particularmente quando esse conhecimento não é 100% certo;

Por outro lado, apenas alguns fatos, podem extinguir todos recursos computacionais, a não ser que o programa seja guiado, de forma a selecionar quais os passos de raciocínio que deve efetuar primeiro.

## 1.4. Agentes Racionais

Agir racionalmente, significa agir de forma a atingir um dado conjunto de objetivos, dados um conjunto de crenças. Um agente é uma entidade que percebe o ambiente no qual está inserido através de sensores e afeta essa ambiente por meio de atuadores. [Russel & Norvig, 2003].

Para agir de forma racional, um agente tem algumas formas de pensar racionalmente, de forma a identificar (inferir) a ação correta para atingir os objetivos propostos.

Por outro lado, existem situações onde, provavelmente, não existe uma ação correta a ser tomada, mas, no entanto, alguma decisão deve ser tomada.

Em alguns casos agir racionalmente, não significa inferir a ação através de um processo de pensamento racional.

Exemplo: Se colocarmos a mão debaixo de uma torneira com água quente, temos o reflexo imediato de retira-lá. Este reflexo (ato racional) é preferível a tomar a decisão após deliberar cuidadosamente qual seria a melhor ação a tomar.

# 2. Histórico e Evolução da IA

As correntes de pensamento que se cristalizaram em torno da IA já estavam em gestação desde os anos 30 [Barr & Feigenbaum, 1981]. No entanto, oficialmente, a IA nasceu em 1956 com uma conferência de verão em Dartmouth College, NH, USA. Na proposta dessa conferência, escrita por John McCarthy (Dartmouth), Marvin Minsky (Hardward), Nathaniel Rochester (IBM) e Claude Shannon (Bell Laboratories) e submetida à fundação Rockfeller, consta a intenção dos autores de realizar “um estudo durante dois meses, por dez homens, sobre o tópico inteligência artificial". Ao que tudo indica, esta parece ser a primeira menção oficial à expressão “Inteligência Artificial" [McCorduck, 1979]. Desde seus primórdios, a IA gerou polêmica, a começar pelo seu próprio nome, considerado presunçoso por alguns, até a definição de seus objetivos e metodologias. O desconhecimento dos princípios que fundamentam a inteligência, por um lado, e dos limites práticos da capacidade de processamento dos computadores, por outro, levou periodicamente a promessas exageradas e às correspondentes decepções [Bittencourt, 2010].

Dada a impossibilidade de uma definição formal precisa para IA, visto que para tanto seria necessário definir, primeiramente, a própria inteligência, foram propostas algumas definições operacionais: “uma máquina é inteligente se ela é capaz de solucionar uma classe de problemas que requerem inteligência para serem solucionados por seres humanos" [Michie & Meltzer, 1969]; “Inteligência Artificial é a parte da ciência da computação que compreende o projeto de sistemas computacionais que exibam características associadas, quando presentes no comportamento humano, à inteligência" [Barr & Feigenbaum, 1981].

## 2.1. 1943 - 1956 : A Gestação

Existem duas linhas principais de pesquisa para a construção de sistemas inteligentes: a linha *conexionista* e a linha *simbólica* . A linha conexionista visa à modelagem da inteligência humana através da simulação dos componentes do cérebro, isto é, de seus neurônios, e de suas interligações. Esta proposta foi formalizada inicialmente em 1943, quando o neuropsicólogo McCulloch e o lógico Pitts propuseram um primeiro modelo matemático para um neurônio.

Um primeiro modelo de *rede neuronal* , isto é, um conjunto de neurônios interligados, foi proposto por Rosenblatt. Este modelo, chamado *Perceptron* , teve

suas limitações demonstradas por Minsky e Papert [Minsky & Papert 1969] em livro onde as propriedades matemáticas de redes artificiais de neurônios são analisadas. Durante um longo período essa linha de pesquisa não foi muito ativa, mas o advento dos microprocessadores, pequenos e baratos, tornou praticável a implementação de máquinas de conexão compostas de milhares de microprocessadores, o que, aliado à solução de alguns problemas teóricos importantes, deu um novo impulso às pesquisas na área. O modelo conexionista deu origem à área de redes neuronais artificiais.

- 1949: Algoritmo para modificar os pesos das ligações entre os neurônios (Donald Hebb)
- Início dos anos 50: Programas de xadrez para computador (Claude Shannon 1950 e Alan Turing 1953)
- 1951: Primeira rede neural (Marvin Minsky e Dean Edmonds)
- 1956 : Conferência Dartmouth (10 participantes)
- Logic Theorist (LT) - programa que era capaz de provar teoremas. (Allen Newell e Herbert Simon).
- **Surge o nome Inteligência Artificial (John McCarthy)**
- 1952 - 1969 : Período de muito entusiasmo e grandes expectativas (muitos avanços com sucesso)

Inicialmente, a pesquisa em manipulação de símbolos se concentrou no desenvolvimento de formalismos gerais capazes de resolver qualquer tipo de problemas. O sistema GPS, *General Problem Solver*, projetado por Ernst e Newell (1969), é um exemplo deste tipo de pesquisa. O GSP Imitava o homem na forma de resolver problemas. Chegou-se à conclusão de que a forma em como dividia um objetivo em sub objetivos e possíveis ações era similar à forma em como o homem o fazia. Estes esforços iniciais ajudaram a estabelecer os fundamentos teóricos dos sistemas de símbolos e forneceram à área da IA uma série de técnicas de programação voltadas à manipulação simbólica, por exemplo, as técnicas de busca heurística. Os sistemas gerais desenvolvidos nesta época obtiveram resultados interessantes, por vezes até impressionantes, mas apenas em domínios simplificados, onde o objetivo era principalmente a demonstração da técnica utilizada, e não a solução de um problema real. O problema com os sistemas gerais

é que a sua extensão a domínios de problemas reais se mostrou inviável. Isto se deveu a duas razões, uma relacionada com características teóricas dos métodos utilizados, e outra associada à natureza do conhecimento do mundo real.

## 2.2. 1952 - 1969 : Período de Muito Entusiasmo

Neste período temos vários marcos históricos na IA:

- IBM produz alguns dos primeiros programas de IA, entre os quais, em 1959 o Geometry Theorem Prover.
- Arthur Samuel desenvolveu um programa capaz de jogar damas ao nível de um jogador de torneio. O programa jogava melhor do que o seu autor.
- 1958 : John McCarthy no Lab Memo n.1 do MIT define a linguagem de programação Lisp (List Processing) que se transformou na linguagem dominante da IA. O Lisp é a segunda linguagem de programação mais antiga ainda em uso. A linguagem Fortran é um ano mais antiga.
- Em 1958 McCarty publicou um artigo intitulado “Programs with common sense”, onde descrevia um programa hipotético designado por “Advice taker”, o qual pode ser visto como o primeiro sistema completo da IA. Este artigo não perdeu a sua relevância ao fim de mais de 40 anos.

## 2.3. 1966 - 1974 : Uma Dose de Realidade

Durante a década de setenta, a IA estava praticamente restrita ao ambiente acadêmico. Os objetivos da pesquisa eram, principalmente, a construção de teorias e o desenvolvimento de programas que verificassem estas teorias para alguns poucos exemplos. É interessante notar que o fato de que não havia interesse em construir programas de IA “de verdade", isto é, com aplicações práticas, não se deve a uma eventual incompetência em programação dos pesquisadores em IA. Pelo contrário, foi a inspiração desses ``hackers" que levou a conceitos hoje integrados à ciência da computação, como: tempo compartilhado, processamento simbólico de listas, ambientes de desenvolvimento de ``software", orientação objeto, etc., além da mudança da relação usuário-computador ao eliminar a intermediação de um operador e colocar cada usuário diante de sua estação de trabalho.

Uma mudança importante ocorreu ao longo da década de setenta em relação aos critérios acadêmicos de julgamento de trabalhos em IA: houve uma crescente exigência de formalização matemática. Se no início dos anos setenta, um programa, mesmo tratando de alguns poucos exemplos de um problema até então não tratado, já era considerado IA, isto não acontecia mais em 1980 [Bittencourt, 2010]. O programa em si passou a ser a parte menos importante; a análise formal da metodologia, incluindo o poder de decisão, completude e complexidade, além de uma semântica bem fundada, passou a ser o ponto fundamental. A década de setenta marcou também a passagem da IA para a "vida adulta": com o aparecimento dos primeiros Sistemas Especialistas, a tecnologia de IA passou a permitir o desenvolvimento de sistemas com desempenho intelectual equivalente ao de um ser humano adulto, abrindo perspectivas de aplicações comerciais e industriais.

Ao contrário dos métodos fracos (usam pouca informação acerca do domínio do problema e mecanismos gerais de procura) os sistemas que dispõem de uma base de conhecimento podem resolver problemas mais complexos.

- DENDRAL - Análise de compostos orgânicos para determinar a sua estrutura molecular.
- MYCIN – Sistema pericial (expert system) capaz de diagnosticar infecções no sangue (dispunha de mais de 450 regras). Este sistema tinha um desempenho tão bom quanto de alguns médicos especialistas e melhor do que de médicos ainda com pouca experiência.
- 1972 : Linguagem Prolog (programação em lógica).

## 2.4. Década De 80: IA Transforma-se numa Indústria

1981 : O Japão lança o projeto "Quinta geração", um plano para construir em 10 anos computadores inteligentes. As instruções dos processadores eram instruções em PROLOG. Estes sistemas deveriam ser capazes de fazer milhões de inferências por segundo. Uma das ambições do projeto era a compreensão da linguagem natural (projeto que veio revitalizar a IA em todo o mundo).

1982 : Surge o primeiro sistema pericial a ser comercializado, o R1. O programa ajudava a configurar encomendas de computadores. Em 1986 estimou-se que a Digital tinha poupado cerca de 40 milhões de dólares graças ao R1.

1986 : Retorno das redes neurais artificiais.

## 2.5. 90 - 20xx: IA Moderna

- 1991: Sistemas de IA utilizados com sucesso na guerra do Golfo.
- 1991: Um sistema pericial analisa um caso médico, chega a um diagnóstico e é capaz de explicar porque chegou a esse diagnóstico, expondo os fatores que mais o influenciaram.
- 1993: Sistema capaz de conduzir um carro numa auto-estrada a cerca de 90 Km/h. O sistema usa câmaras de vídeo, radar e laser.
- 1993: Um sistema detecta colisões na rua, chamando automaticamente o 911.
- 1994: Um sistema de reserva de viagens é capaz de entender frases como “quero ir de Boston para São Francisco”. O sistema percebe mal uma em cada 10 palavras, mas é capaz de recuperar, porque compreende a forma em como as frases são compostas.
- 2000: Começam a surgir brinquedos inteligentes.
- 2001: Computador se comunica ao nível de uma criança com 15 meses.

Atualmente, os SE's são considerados como parte de uma tecnologia de desenvolvimento de “software" estabelecida, sendo objeto de diversas conferências internacionais e submetida a avaliações rigorosas de desempenho. Entre os diversos benefícios associados ao desenvolvimento de SE's podem-se citar: distribuição de conhecimento especializado, memória institucional, flexibilidade no fornecimento de serviços (consultas médicas, jurídicas, técnicas, etc.), facilidade na operação de equipamentos, maior confiabilidade de operação, possibilidade de tratar situações a partir de conhecimentos incompletos ou incertos, treinamento, entre outros. Atualmente, existem milhares de SE's em operação nos mais variados domínios, e a influência da IA em outros campos da computação, como engenharia de “software”, bancos de dados e processamento de imagens vem crescendo constantemente.

As principais áreas de pesquisa em IA simbólica são atualmente: sistemas especialistas, aprendizagem, representação de conhecimento, aquisição de conhecimento, tratamento de informação imperfeita, visão computacional, robótica, controle inteligente, inteligência artificial distribuída, modelagem cognitiva,

arquiteturas para sistemas inteligentes, linguagem natural e interfaces inteligentes. Além das linhas conexionista e simbólica, observa-se hoje o crescimento de uma nova linha de pesquisa em IA, baseada na observação de mecanismos evolutivos encontrados na natureza, tais como a auto-organização e o comportamento adaptativo. Nesta linha, os modelos mais conhecidos são os algoritmos genéticos e os autômatos celulares.

A gradativa mudança de metas da IA, desde o sonho de construir uma inteligência artificial de caráter geral comparável à do ser humano até os bem mais modestos objetivos atuais de tornar os computadores mais úteis através de ferramentas que auxiliam as atividades intelectuais de seres humanos, coloca a IA na perspectiva de uma atividade que praticamente caracteriza a espécie humana: a capacidade de utilizar representações externas, seja na forma de linguagem, seja através de outros meios [Hill 1989]. Deste ponto de vista, a computação em geral e a IA em particular são o ponto culminante de um longo processo de criação de representações de conhecimento, iniciado com as primeiras pinturas rupestres. Esta nova perspectiva coloca os programas de IA como produtos intelectuais no mesmo nível dos demais, ressaltando questões cuja importância é central para os interesses atuais da IA, por exemplo, como expressar as características individuais e sociais da inteligência utilizando computadores de maneira a permitir uma maior produtividade, e como as propriedades das representações utilizadas auxiliam e moldam o desenvolvimento de produtos intelectuais.

## Exercícios

1. Defina Inteligência. O que é um comportamento inteligente de uma máquina?
2. Em que consiste é o “Teste de Turing?”
3. O que você entende por Inteligência Artificial? Contraste sua resposta com as quatro categorias “Sistemas que pensam como os humanos”, “Sistemas que agem como os humanos”, “Sistemas que pensam racionalmente” e “Sistemas que agem racionalmente”.
4. Descreva o paradigma simbolista.
5. Descreva o paradigma conexionista.
6. O que são sistemas especialistas?
7. Quais foram as principais dificuldades para a continuação das pesquisas em Redes Neurais no início de seu desenvolvimento?

8. Quais os principais desafios de IA hoje?

## Weblografia

Universidade Aberta do Piauí – UAPI
http://www.ufpi.br/uapi
Universidade Aberta do Brasil- UAB
http://www.uab.gov.br
Secretaria de Educação a Distância do MEC – SEED
http://www.seed.mec.gov.br
Associação Brasileira de Educação a Distância – ABED
http://www.abed.org.br
Instituto Sem Fronteiras (ISF)
http://www.isf.org.br/
Teste de Turing
http://www.din.uem.br/~ia/maquinas/turing.htm
[Bittencourt, 2010] - Breve história da Inteligência Artificial
http://www2.dem.inpe.br/ijar/AIBreveHist.pdf
Inteligência Artificial
http://www.papociencia.ufsc.br/IA1.htm

## Resumo

A Inteligência Artificial (IA) é uma área de pesquisa da ciência da computação que busca métodos ou dispositivos computacionais que simulam a capacidade humana de resolver problemas. Desde da sua criação, as pesquisas em Inteligência Artificial tentam implementar computacionalmente sistemas que pensam e agem de forma racional e que pensam e agem de forma semelhante aos seres humanos. Essas quatro vertentes mostram que há uma teoria ou paradigma unificado estabelecido que guie a pesquisa em IA..

Nesta unidade veremos em que diferem essas abordagens da IA ao mesmo tempo em que faremos um resgate histórico de todas as fases das pesquisa em Inteligência Artificial, desde da sua concepção passando pela fase de encantamento até a realidade das aplicações atuais.

## Referências Biblográficas

A. Newell e H. A. Simon (1961). GPS, a program that simulates human thought, em E. Feigenbaum e J. Feldmann, Hrsg. (1995) Computers and Thought, ISBN 0262560925.

Russel S. J., Norvig P. 2003. Artificial Intelligence, A Modern Approach, Second Edition. Prentice Hall.

A. Barr and E.A. Feigenbaum, editors. The Handbook of Artificial Intelligence, volume I-II. William Kaufmann Inc., Los Altos, California, 1981.

P. McCorduck. Machines Who Think. Freeman, San Francisco, 1979.

J. McCarthy and P.J. Hayes. Some philosophical problems from the standpoint of artificial intelligence. In D. Michie and B. Meltzer, editors, Machine Intelligence 4, pages 463-502. Edinburgh University Press, Edinburgh, GB, 1969.

Minsky, M. and Papert, S. (1969). Perceptrons. MIT Press, Cambridge.

# UNIDADE 2

## RESOLUÇÃO DE PROBLEMAS

## ATRAVÉS DE BUSCAS

## 1. Problemas e Espaço de Problemas

A resolução de problemas é fundamental para a maioria das aplicações de IA. De fato, a habilidade para resolver problemas é freqüentemente usada como medida de inteligência para máquinas e homens. Existem basicamente dois tipos de problemas. O primeiro tipo pode ser resolvido por algum procedimento determinístico. São os problemas computáveis. Contudo, existem poucos problemas do mundo real que possuem soluções computacionais. Esses problemas são colocados na segunda categoria, ou seja, dos problemas não computáveis. Esses problemas são resolvidos através de uma busca por uma solução. Esse é o método de solução de interesse da Inteligência Artificial.

Antes de formular um problema e utilizar métodos de busca para solucioná-lo é importante **formular o objetivo** a ser alcançado. Os objetivos auxiliam a organizar o comportamento do programa através da limitação dos objetivos que se pretende alcançar. Dessa forma, a formulação do objetivo é o primeiro passo na resolução de problemas. Um objetivo é um conjunto de *estados do mundo* – exatamente os estados em que o objetivo satisfeito. Ações podem ser vistas como aquelas que causam transições entre estados do mundo.

Ao formular o problema deve-se definir o objetivo, ou seja, definindo o estado final que satisfaz o agente; detalhar seu estado inicial; a função que leva aos próximos estados; o teste de objetivo, que determina se o estado final foi alcançado; e o custo (path cost). Essa formulação define o espaço de estados possíveis, através do estado inicial juntamente com todos os estados alcançáveis através da função, formando um grafo com nós representando cada estado, e arcos entre nós representando ações. Mas esta formulação deve ser bem feita, pois o espaço de estados definido a partir da formulação impacta na eficiência de busca da solução.

Portanto um problema em IA é definido em termos de:

- um espaço de estados possíveis, incluindo;
- um estado inicial, que descreve a situação inicial do problema;
- um (ou mais) estado final (objetivo);
- exemplo 1: dirigir de Teresina a Parnaíba;
- espaço de estados: todas as cidades da região
- exemplo 2: jogo de 8-números

| | | |
|---|---|---|
| 4 | 5 | 8 |
| | 1 | 6 |
| 7 | 2 | 3 |

| | | |
|---|---|---|
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 7 | 8 | |

- um conjunto de ações (ou operadores) que permitem passar de um estado a outro
- ex1. dirigir de uma cidade a outra
- ex2. mover uma peça do jogo de n-números (n-puzzle)

Podemos perceber que a formulação do problema é o processo de decidir que ações e que estados devem ser considerados para alcançar um objetivo. Ou seja, deve-se encontrar um **caminho** (seqüência de ações ou operadores) que leva do estado inicial a um estado final (objetivo). Portanto deve-se percorrer um caminho do estado inicial até o estado final passando pelo **espaço de estados** que é o conjunto de todos os estados alcançáveis a partir do estado inicial por qualquer seqüência de ações.

A passagem de um estado para outro é dada através da aplicação de operadores que caracterizam uma determinada ação. Em cada estado visitado é realizado um teste para verificar se este é o estado final (objetivo). Esse estado é chamado de teste de término ou **teste de objetivo**. O caminho (estados visitados do início até o final) pode ser expresso como uma função. A função de **custo de caminho** é geralmente a soma dos custos das ações individuais ao longo do caminho.

No caso de haver mais de uma solução para o problema (vários estados finais), o custo de caminho é utilizada para indicar qual tem o menor custo, indicando assim a melhor solução.

Exemplo de Formulação de Problemas (Jogo de 8 números):

- estados = cada possível configuração do tabuleiro
- estado inicial = qualquer um dos estados possíveis
- teste de término = ordenado, com branco na posição [3,3]
- operadores = mover branco (esquerda, direita, para cima e para baixo)
- custo do caminho = número de passos da solução

Ida para Parnaíba:

- estados = cada possível cidade do mapa;
- estado inicial = Teresina ;
- teste de término = estar em Parnaíba;
- operadores = dirigir de uma cidade para outra;
- custo do caminho = número de cidades visitadas, distância percorrida, tempo de viagem, grau de divertimento, etc .

## 1.1. Como Encontrar uma Solução

Uma vez o problema bem formulado, o estado meta (objetivo) deve ser "buscado" no espaço de estados. A busca é representada em uma "árvore[1] de busca":

1. Raiz: corresponde ao estado inicial
2. Expande-se o estado corrente: aplica-se a função sucessor ao estado corrente, gerando um novo conjunto de sucessores
3. Escolhe-se o próximo estado a expandir seguindo uma estratégia de busca
4. Prossegue-se até sucesso (atingir estado meta – retorna solução) ou falha

## 1.2. Medida de Desempenho na Busca

Um dos obstáculos mais difíceis de superar quando são aplicadas técnicas de IA aos problemas do mundo real é a magnitude e complexidade da maioria das situações. Nos primeiros anos da pesquisa em IA, desenvolver bons métodos de busca era o principal objetivo. Os pesquisadores acreditam que a busca é a base da resolução de problemas, que é um ingrediente crucial da inteligência.
O desempenho de um algoritmo de busca depende de alguns fatores:

- Completeza: sempre encontra uma solução se esta existir;
- Otimização: encontra a solução ótima
- Complexidade de tempo: tempo gasto para encontrar uma solução;
- Complexidade de espaço: memória necessária para encontrar uma solução.

[1] Árvore, no contexto da programação e ciência da computação, é uma estrutura de dados que herda as características das topologias em árvore. Conceitualmente diferente das listas encadeadas, em que os dados se encontram numa seqüência, nas árvores os dados estão dispostos de forma hierárquica.

Além disso, podemos dizer que a busca possui um custo total que é a soma do custo da solução (esforço para expandir os nós e o custo da busca (a quantidade de nós visitados até o nó solução).

## 2. Métodos de Busca.

### 2.1. Métodos de Busca Cega

Busca cega ou busca exaustiva (Exhaustive Search) é um meio de resolução de problemas onde a idéia é transformar um problema de raciocínio diretamente em um problema de navegação num espaço de estados, no qual, a partir de um estado inicial, pode-se buscar uma seqüência de ações que conduzem ao estado final desejado. A solução ótima é o conjunto de seqüência de ações que leva ao estado final desejado com o menor custo.

Na busca cega, não se sabe qual o melhor nó da fronteira (últimos nós alcançados pela função) a ser expandido (aquele que acarreta um menor custo até o objetivo). Contudo, pode-se utilizar de estratégias de busca, definidas pela ordem de expansão dos nós. A avaliação do desempenho destas estratégias se dá verificando a completude (sempre encontra uma solução, se ela existe), a complexidade do tempo (em função do número de nós gerados/expandidos), a complexidade do espaço (em função do número máximo de nós armazenados em memória) e a otimização (sempre encontra a solução de menor custo).

As principais estratégias de busca cega são: Busca em Largura, Busca em Profundidade, Busca em Profundidade Limitada e Busca de Aprofundamento Iterativo.

#### 2.1.1. Medida de Desempenho na Busca

A complexidade do tempo e do espaço são medidas em termos de um fator máximo de ramificação da árvore de busca (*b* – branching), da profundidade da solução de menor custo (*d* – depth) e da profundidade máxima do espaço de estados (*m*).

- Fator de ramificação: *b* (número máximo de sucessores de qualquer nó);
- Profundidade do nó objetivo menos profundo: *d*
- tempo: medido em termos do número de nós gerados durante a busca

- espaço: número máximo de nós armazenados.

### 2.1.2. Busca em Largura

A Busca em Largura expande sempre o nó menos profundo ainda não expandido; é completa desde que *d* seja finito, seu tempo é da ordem de $b^{d+1}$, tal qual o espaço, e em geral não é ótima, exceto quando o custo é constante a cada passo.

Na busca em largura temos (Figura 3):

1. Expande-se o nó raiz;
2. Expande-se todos os nós gerados pelo raiz;
3. Todos os nós na profundidade d são expandidos antes dos nós na profundidade d+1.

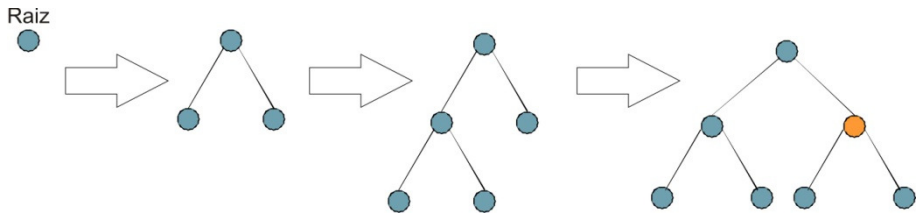



**Figura 3: Ordem de expansão dos nós na busca em largura**

### 2.1.3. BUSCA DE CUSTO UNIFORME

Modifica a busca em largura expandindo o nó da fronteira com menor custo de caminho na fronteira do espaço de estados cada operador pode ter um custo associado diferente, medido pela função g(n), para o nó n, onde g(n) dá o custo do caminho da origem ao nó n (Figura 4).

A Busca de Custo Uniforme expande sempre o próximo nó ainda não expandido que possui caminho de menor custo; é completa, desde que o custo de cada nó seja diferente de 0, o tempo e o espaço são iguais ao número de nós com custo(nó) menor que o custo(solução ótima), e é ótima, já que os nódulos expandem em ordem crescente de custo(nó).

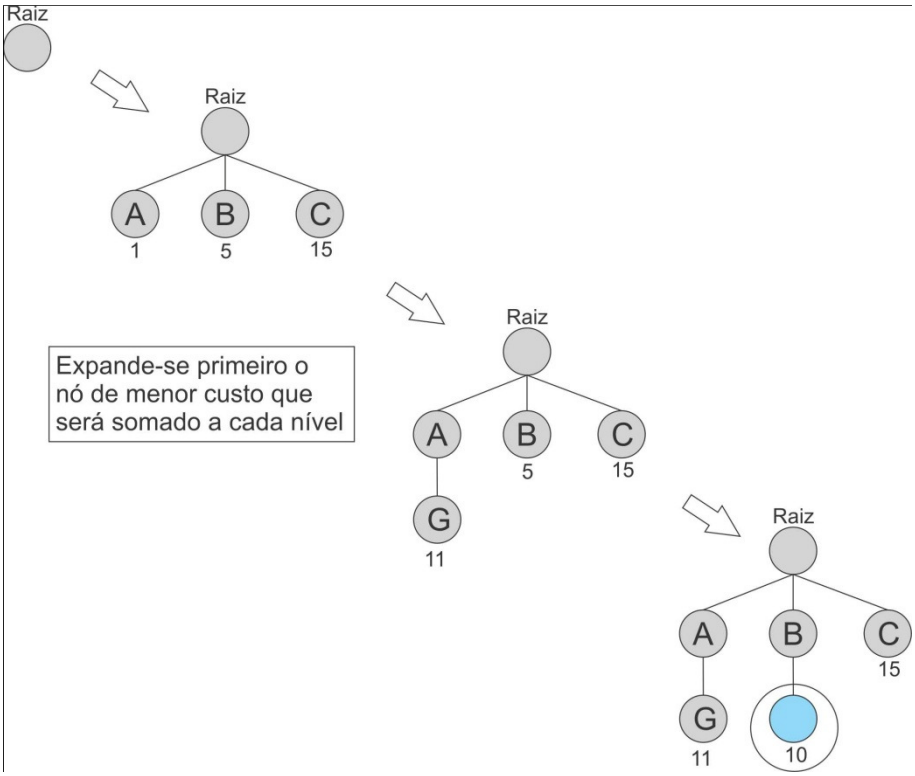



**Figura 4: Ordem de expansão dos nós na busca de custo uniforme**

### 2.1.4. BUSCA EM PROFUNDIDADE

A Busca em Profundidade expande sempre o nó mais profundo ainda não expandido; não é ótima, pois falha em árvores de profundidade infinita (nesse caso pode-se arbitrar um limite de profundidade – Busca em Profundidade Limitada), seu tempo é O(bm), seu espaço é O(b*m), ou seja, crescimento linear, e não é ótima, devendo ser evitada em árvores muito profundas ou profundidade infinita.

A busca em profundidade progride através da expansão do primeiro nó filho da árvore de busca, e se aprofunda cada vez mais, até que o alvo da busca seja encontrado ou até que ele se depare com um nó que não possui filhos - nó folha (). Então a busca retrocede (backtrack) e começa no próximo nó. Numa implementação não-recursiva, todos os nós expandidos recentemente são adicionados a uma pilha, para realizar a exploração.

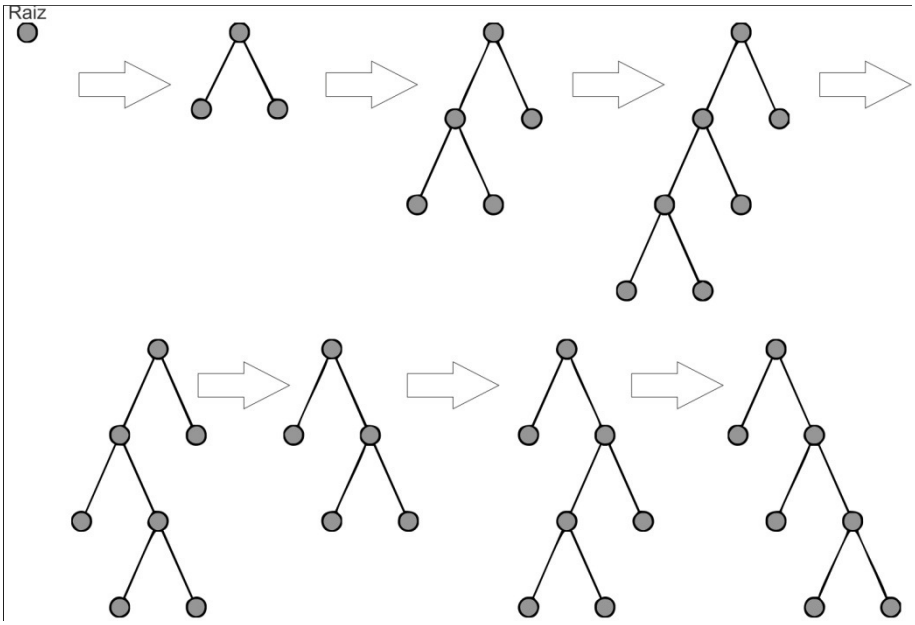



**Figura 5: Ordem de expansão dos nós na busca em profundidade**

### 2.1.5. BUSCA EM PROFUNDIDADE ITERATIVA

A Busca de Aprofundamento Iterativo é uma variação da Busca em Profundidade, que utiliza um limite de profundidade L, que inicia em 0 e vai sendo incrementado de 1 a cada iteração. É completa desde que d seja finito, seu tempo é da ordem de bd, o espaço é O(b*d), e em geral não é ótima, exceto quando o custo é constante a cada passo; é o método de busca cega preferido para grandes espaços de busca e quando a profundidade da solução é desconhecida.

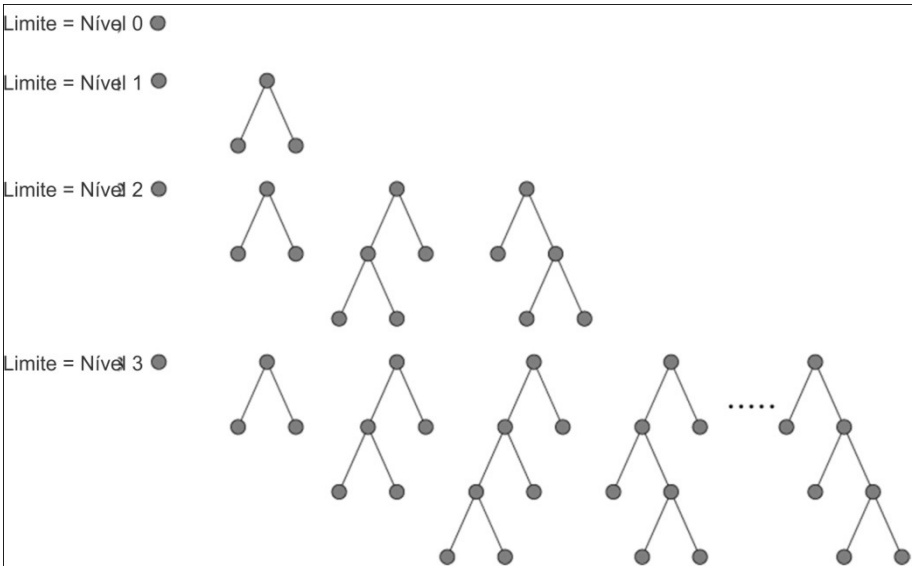



**Figura 6: Ordem de expansão dos nós na busca em profundidade iterativa**

## 2.2. BUSCA COM INFORMAÇÃO

Diferentemente da Busca Exaustiva, onde não se sabe qual o melhor nó de fronteira a ser expandido, a Busca Heurística estima qual o melhor nó da fronteira a ser expandido com base em funções heurísticas, tornando-se assim uma busca não exaustiva do espaço de estados, utilizando atalhos para sua exploração que descartam subespaços sem explorá-los. Em contrapartida, muitas vezes não se tem a garantia de sempre encontrar a melhor solução ou até uma solução em todas as instâncias do problema.

Heurísticas tratam de métodos em que, embora a exploração seja feita de forma algorítmica, o progresso é obtido pela avaliação do resultado usando alguma informação contida no problema. Alguns algoritmos de busca heurística global são: Busca gulosa, A*, RBFS, e SMA*.

### 2.2.1. BUSCA GULOSA

O algoritmo da busca gulosa pela melhor escolha tem como estratégia expandir o nó supostamente mais próximo a origem (heurística), através de uma estimativa. A distância do nó a expandir ao objetivo é estimada através de uma função heurística. Mas esta estratégia sofre dos mesmos problemas da Busca Exaustiva (cega) em Profundidade, fazendo com que ela não seja completa (pois pode ocorrer loops, por exemplo), não seja ótima (não encontra o menor caminho até o objetivo), e um pior caso de tempo e espaço da ordem de bm (b = fator de ramificação da árvore, e m = profundidade máxima do espaço de estados).

Para melhor entendimento, vejamos o tradicional problema de cálculo da distância entre as cidades Romenas de Arad – Bucharest (mapa da

Figura 7) através dessa busca.

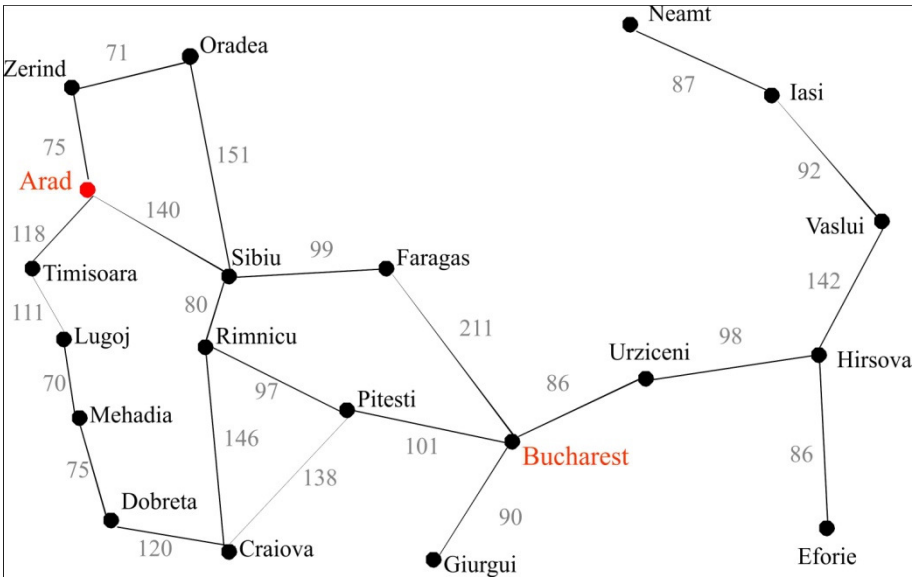



**Figura 7: Mapa com as distâncias entre as cidades da Romênia**

Considere também o quadro a seguir que exibe a menor distância, em linha reta, até a cidade de Bucarest (Tabela 1).

Escolhe-se a rota com a menor distância, (quadro anterior):

- Primeiro passo: Saindo da cidade de Arad (nó raiz), o algorítmo pode conduzir às cidades de Zerind, Timisoara e Sibiu. Em linha reta, Timisoara está a 329 km de Buscarest; Zerind a 374 e Sibiu a 253. Aparentemente, esse último caminho é a melhor escolha.

Segundo passo: Partindo de Sibiu, Pode-se chegar às cidades de Fagaras e Rimnicu Vilcea. Fagaras está a 178 km de Bucarest e a Rimnicu Vilcea a 193. Aparentemente o melhor caminho passa por Fagaras.

**Tabela 1: Distância em linha reta a até Bucarest**

| Cidade | Distância |
|---|---|
| Arad | 366 |
| Bucharest | 0 |
| Craiova | 160 |
| Dobreta | 242 |
| Eforie | 161 |
| Fagaras | 178 |
| Giurgiu | 77 |
| Hirsova | 151 |
| Iasi | 226 |
| Lugoj | 244 |

| Cidade | Distância |
|---|---|
| Mehadai | 241 |
| Neamt | 234 |
| Oradea | 380 |
| Pitesti | 98 |
| Rimnicu | 193 |
| Sibiu | 253 |
| Timisoara | 329 |
| Urziceni | 80 |
| Vaslui | 199 |
| Zerind | 374 |

Resultado do caminho escolhido pela busca gulosa: Arad->Sibiu->Fagaras->Bucarest (Figura 8).

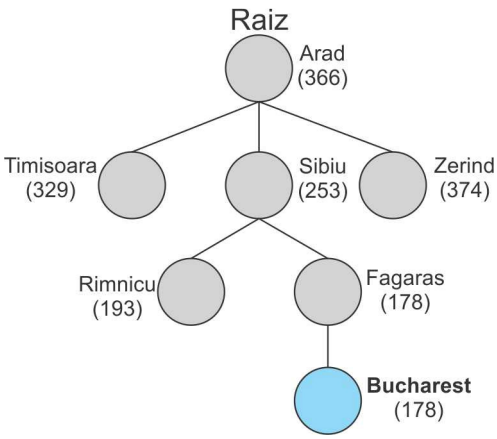



**Figura 8: Resultado do caminho escolhido pela busca gulosa**

### 2.2.2. ALGORITMO A*

O algoritmo A* busca expandir o nó com menor valor Custo(Caminho) na fronteira do espaço de estados, utilizando a heurística Custo(Caminho) = Custo(Caminho Percorrido) + Custo(Caminho Restante) (f(n) = g(n) + h(n)), o que permite uma simplificação, eliminando caminhos redundantes. Como a exploração é em uma árvore de busca de largura primeira, este algoritmo tem vantagens e limitações parecidas à busca cega de custo uniforme. É uma estratégia completa, quando a heurística é admissível, isto é, quando ela sempre subestima o custo real (Custo(Caminho Percorrido)), para todo nó.

Da mesma forma, é ótima para heurísticas admissíveis, garantindo mais eficiência do que qualquer outro algoritmo completo e ótimo usando a mesma heurística, e seu tempo e espaço são no pior caso O(bd), onde d = profundidade da solução de menor custo.

Retornando ao exemplo da busca gulosa (

Figura 7), poderemos, realmente, encontrar a rota mais curta entre Arad e Bucarest, tendo em vista que a redução ao mínimo de f(n) = g(n) + h(n) combina as vantagens da busca com custo uniforme e a busca gulosa. Quando se repetem os estados que são analisados e se assegura que h(n) nunca faça uma superestimativa, encontramos o melhor caminho, segundo a busca A*.

- g(n) = Custo da rota de partida ao nó n

- h(n) = Custo da rota mais barata que vai de n à meta
- f(n) = Custo estimado da solução mais barata passando por n
- f(n)= g(n) + h(n)

Vejamos como o diagrama da viagem de Arad a Bucarest por meio da busca A*:

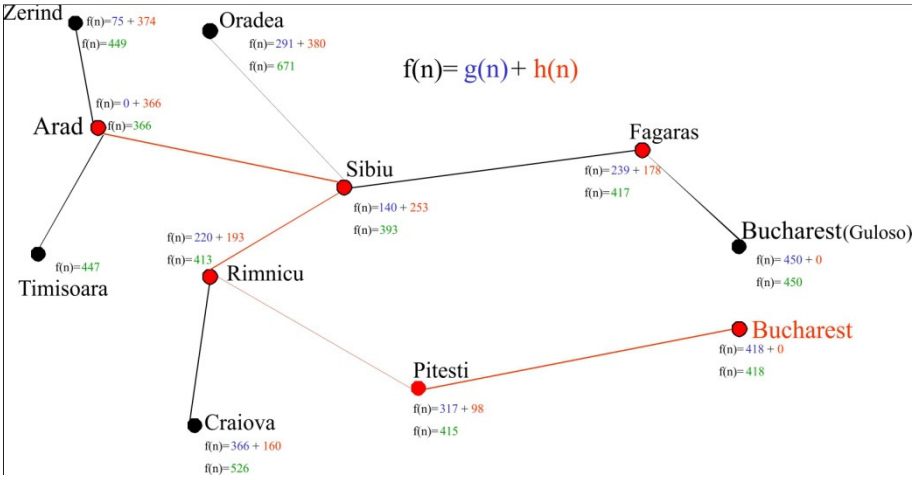



**Figura 9: Diagrama da viagem de Arad a Bucarest por meio da busca A***

Onde g(n) = rota inicial até n e h(n) = unidade estabelecida no quadro de distancias em línha reta a Bucarest. Explicando: Primeiro passo: Saindo da cidade de Arad (nó raiz), o algorítmo pode conduzir às cidades de Timisoara, Zerind ou Sibiu.

- Timisoara está a 118 km de Arad e a 329, Em linha reta, de Bucarest: total = 447;
- Zerind está a 75 km de Arad e a 374, Em linha reta, de Bucarest: total = 449;
- Sibiu está a 140 km de Arad e a 253, Em linha reta, de Bucarest: total = 393;

Sibiu é a primeira escolha do algorítmo A*. Partindo de Sibiu, o algorítmo pode conduzir às cidades de Fagaras ou Rimnicu Vilcea. Fagaras está a 239 km de Arad e a 178, Em linha reta, de Bucarest: total = 417. Rimnicu Vilcea está a 220 km de Arad e a 193 (via Pitesti), em linha reta de Bucarest: total = 413. Rimnicu Vilcea está a 220 km de Arad e a 146 de Craiova que, em linha reta de Bucarest, dista 160 km fazendo um total de 526km.

Resultado do caminho escolhido pelo algorítmo A*: Arad->Sibiu->Rimnicu Vilcea->(via Pitesti)->Bucarest

### 2.2.3. RECURSIVE BEST-FIRST-SEARCH (RBFS)

Este algoritmo A* tem algumas extensões, que tem o objetivo de limitar número de nós guardados na memória, já que o A* guarda todos os nós expandidos durante a busca. São eles: IDA*, que é semelhante à busca em aprofundamento iterativo, mas com a diferença de que o limite de expansão é dado por f(n) e não por d; Recursive Best-First-Search (RBFS), semelhante à busca recursiva em profundidade, com a diferença de que não desce indefinidamente na exploração, havendo um controle da recursão pelo valor de f(n), e se existir algum nó em um dos ancestrais que ofereça melhor estimativa que o nó atual, a recursão retrocede e a busca continua no caminho alternativo, sendo assim completa e ótima quando a heurística é admissível, com espaço O(bd) no pior caso e tempo difícil de se estimar.

### 2.2.4. SMA* (Simple Limited Memory A*)

O SMA utiliza toda a memória disponível, e quando a memória está cheia, ele elimina o pior nó folha (maior custo de f(n)) para continuar a busca, e é completa e ótima quando a memória alocada é suficiente.
O desempenho da busca deve está condicionado à qualidade da função heurística, que deve ser admissível e ter alta precisão (qualidade da distância do nó corrente ao nó objetivo).

## 2.3. Busca Competitiva

Em comparação com os problemas e algoritmos apresentados até aqui, a presença de oponentes torna o problema de decisão em Inteligência Artificial mais complicado e interessante.
Os problemas apresentados nos algoritmos anteriores apenas tinham em conta contingências como o espaço em memória e o tempo disponível. O oponente introduz incerteza e não toma decisões á sorte, tentará sim, jogar de forma minimizar o número de jogadas ruins para si.

Mas o que torna os jogos realmente mais interessantes e diferentes dos restantes problemas na Inteligência Artificial é o fato serem muito mais difíceis de resolver, o xadrez, por exemplo, tem um fator de ramificação em cada jogada de 35,

e nos jogos o número de jogadas médias por jogador chegas às 50. Logo a árvore de pesquisa terá cerca de 35100 nós.
Assim facilmente se conclui que a incerteza dos jogos não se deve ao fato de haver falta de informação mas sim com o fato de não haver tempo de calcular todas as hipóteses ou ramificações possíveis. Levando a que cada jogador decidida, tentando adivinhar a melhor jogada baseando-se em conhecimento anterior.
Nesta sessão serão apresentadas técnicas para escolher uma boa jogada (busca competitiva) em jogos através de cortes nas árvores de estados (corte ou poda alfa-beta) e da avaliação heurística de estados, permitindo assim evitar a pesquisa em ramos que não são relevantes para a escolha final, sem realizar uma pesquisa completa na árvore de todas as jogadas possíveis.

### 2.3.1. Decisões Perfeitas em Jogos com Dois Adversários (Algortimo MIN-MAX)

O algoritmo MIN-MAX aplica-se quando existem dois agentes, MAX e MIN, jogando um de cada vez. O jogador Max joga primeiro, e no fim será atribuída uma pontuação a cada um dos jogadores, mediante as decisões tomadas por ambos.
A soma é nula , ou seja o que um ganha o outro perde e a informação completa pois os agentes conhecem as acções um do outro. Os componentes de procura são os seguintes:

- Estado Inicial - que inclui as posições no tabuleiro e a identificação de quem irá jogar.
- Função Sucessor ou Operadores - retorna os nós resultantes dos movimentos legais.
- Teste Terminal - que determina se o jogo acabou.
- Função de Utilidade - atribui valores aos estados terminais, avaliando numericamente o resultado final do jogo para cada um dos jogadores.

No caso do Jogo da Velha, a Função de Utilidade pode atribuir três valores diferentes aos nós terminais: 1 – se o jogador ganha; 0 – se o jogador empata; -1 – se o jogador perde.
O método de busca consiste na ideia de maximizar a utilidade (ganho) supondo que o adversário vai tentar minimizá-la. O Minimax faz busca cega em profundidade, o agente é Max e o adversário é Min (

Figura 10).

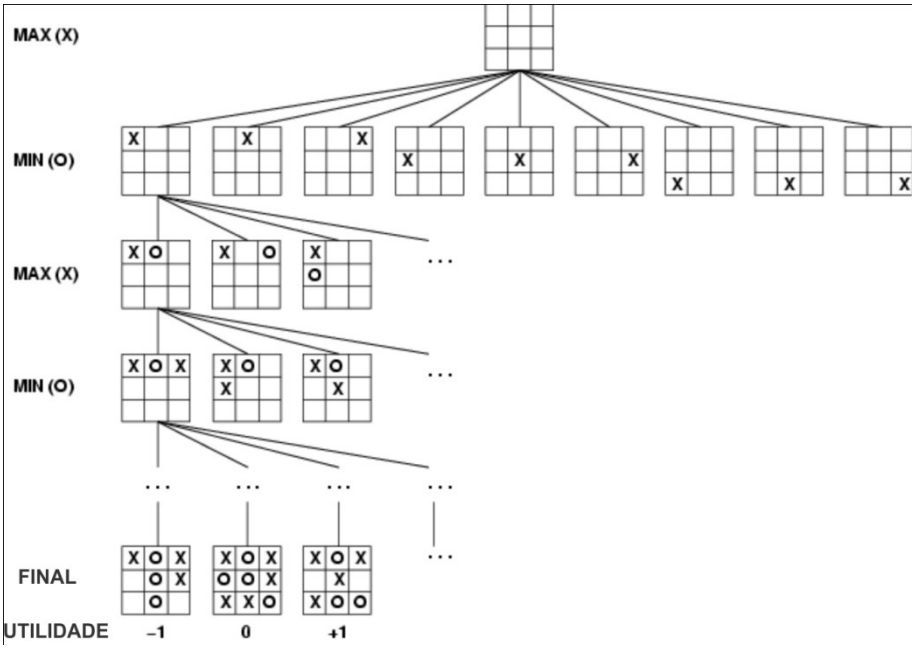



**Figura 10: Algortimo Min-Max no jogo da velha**

Se fosse o caso de se tratar de uma pesquisa normal, bastava percorrer-se a árvore até aos nodos terminais e escolher o caminho que levasse ao nodo com maior valor de utilidade. Mas não é assim, visto existir outro jogador.
Assim, é necessário, escolher a partir de cada nodo filho, o menor valor de utilidade, e copiá-lo para o nodo pai, recursivamente até ao nodo inicial. Isto deve-se ao fato, de o jogador MIN tentar minimizar o ganho do jogador MAX, pelo que ele tentará escolher a jogada, dentro das possíveis, que dê menos pontos ao jogador adversário.

Observando a Figura 11 podemos observar que apesar de o jogador MAX querer obter o maior valor possível (14 pontos), ele não poderá seguir pelo caminho A3, pois o jogador MIN, como resposta, poderá seguir o caminho A33, o que fará com que o jogador MAX obtenha uma pontuação de 2 pontos. Assim sendo, após terem sido copiados o menor dos valores dos nós filhos para os nós pais (que representam as melhores jogadas possíveis do jogador MIN), o jogador MAX, irá escolher o nó com o maior valor dentro destes.

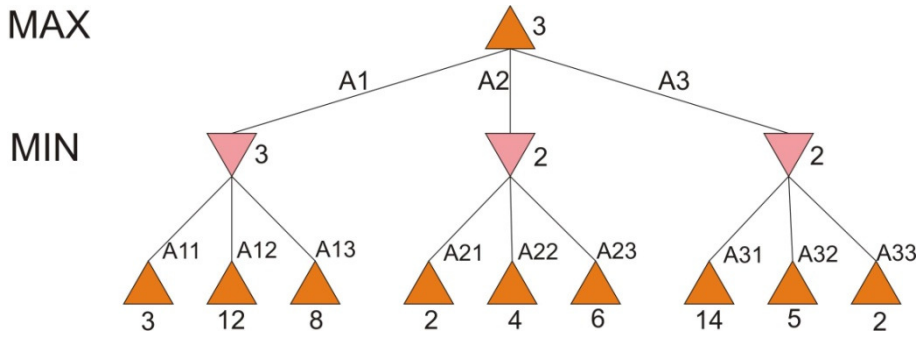



**Figura 11: Evolução do Algortitmo MIN-MAX**

É gerada inicialmente uma árvore de todas as jogadas e as respectivas respostas de cada jogador, depois é aplicado um valor de utilidade aos nós finais, e através destes valores é decidido que valores irão ser atribuídos aos nós pais, de acordo com o interesse do jogador MIN, isto é, selecionar sempre o menor valor.
No topo da árvore, irá ser selecionado o caminho a partir do nó filho que possua o maior valor, de acordo com o interesse do jogador MAX.
O Algoritmo Mini-Max é completo apenas no caso de a árvore for finita, é ótimo se jogarmos contra um adversário ótimo e se não for esse o caso, o MAX jogará ainda melhor.

Quanto à complexidade no tempo, O (bm), onde, b é o fator de ramificação e m é a profundidade e complexidade no espaço O (bm) exploração primeiro em profundidade, a complexidade vai de encontro com o tipo de jogo em questão, ou seja, irá variar consoante o nível de dificuldade do jogo. Exemplo o jogo de xadrez tem um nível de dificuldade muitíssimo superior ao do jogo da velha o que leva ao fato do nível de complexidade no caso do jogo de xadrez ser muito mais elevado. Logo é perceptível que o problema é inviável para qualquer jogo minimamente complexo.

### 2.3.2. Corte Alpha-Beta

O algoritmo Corte Alpha-Beta, tem como função, limitar a expansão de nós desnecessários, pelo algoritmo MiniMax. Este inicia-se aplicando o algoritmo MiniMax, que faz uma pesquisa Depth-First. Quando são encontrados todos os nós terminais de uma ramificação, é escolhido o menor valor destes que irá ser chamado de β.
β = melhor valor no caminho para MIN

Este valor ira ser copiado para o nó pai, sucessivamente até chegarmos á primeira ramificação de MAX (primeiro caminho possível de MAX). E α, que se encontra no primeiro nó da árvore, tomará então o valor de β, no nó inicial de MAX.

α = melhor valor no caminho de MAX

De seguida, dá-se continuidade á expansão da árvore, até se encontrar outro nó terminal, atribuindo-se um novo valor a β. É então realizada uma comparação do valor deste nó, β, com o de α. Caso seja inferior, não é necessário continuar a expansão do respectivo nó pai, visto que, este irá sempre ter um valor inferior ao caminho anteriormente encontrado para MAX.

Este valor é comparado com o valor de α. Caso α seja menor que β, não são expandidas mais ramificações desse nó (corte dos restantes nós), visto o caminho não interessar a MAX, e o valor é copiado para o nó pai. Caso o valor de β seja igual ou superior a α, dá-se continuidade á pesquisa do próximo nó terminal.

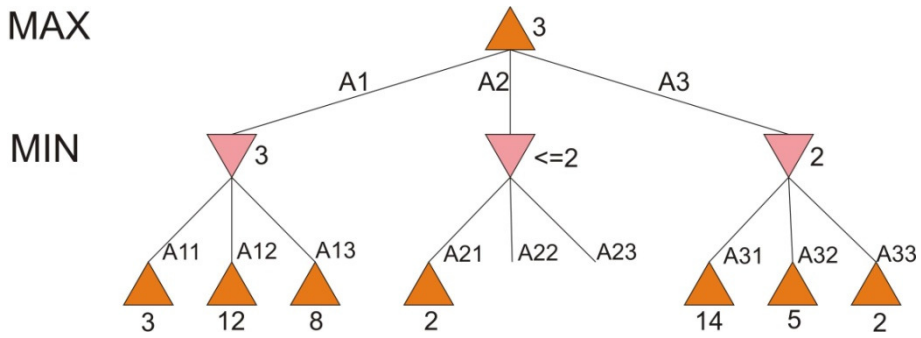
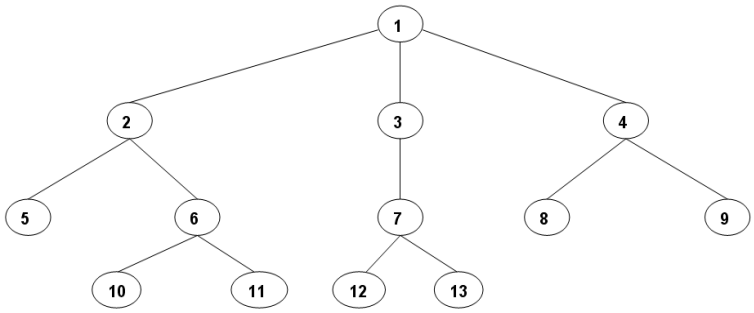



**Figura 12: Corte alpha-beta**

## Exercícios

1. Defina o que vem a ser um problema para IA com explicando os termos, **espaço de estados, estado inicial, estado final, ações, custo de caminho e solução ótima**.
2. Quais são as diferenças entre busca cega e busca com informação?
3. Considere a seguinte árvore de busca:



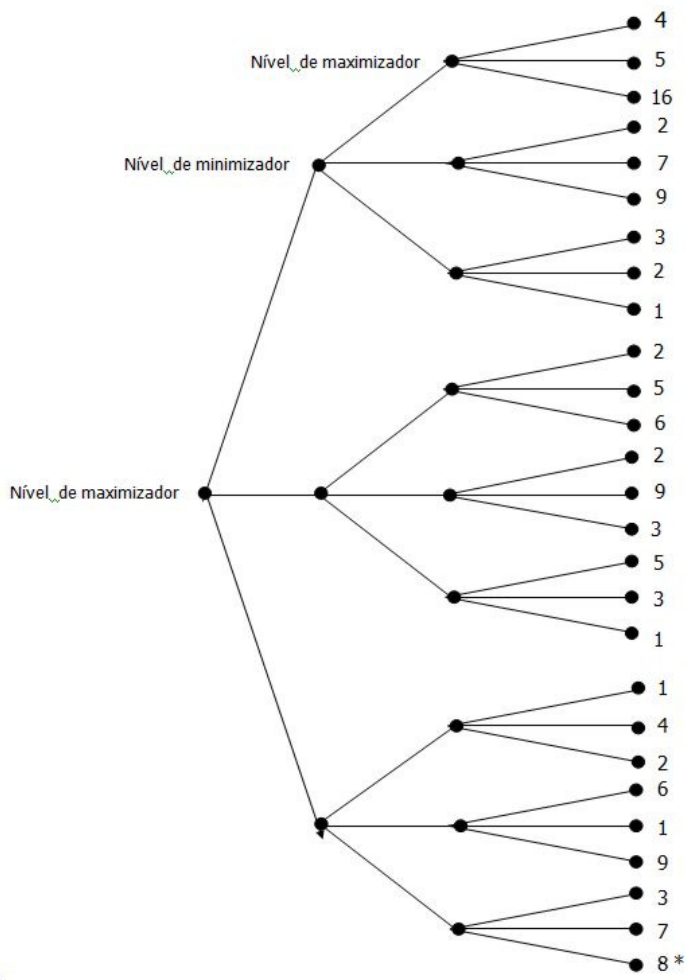

Faça a simulação dos seguintes algoritmos de busca mostrando os estados sucessivos da lista de nós a expandir (considere que os nós são visitados da esquerda para a direita).

a. Profundidade.

b. Extensão.

c. Profundidade Iterativa níveis 2 e 3

4. Quais são os principais critérios de avaliação de uma estratégia de busca? Explique cada um deles.
5. Assinale V (verdadeiro) ou F (falso) para as seguintes afirmativas:
    a. ( ) O uso de informações amenizam o problema da exploração cega dos algoritmos de busca.
    b. ( ) O algoritmo de busca em profundidade utiliza mais memória que o de busca em largura.
    c. ( ) Busca é uma técnica da IA que serve para solucionar qualquer problema que possa ser visto como um espaço de busca, isto é, como um conjunto de estados e trasições. Porém, em alguns problemas esse espaço é difícil de ser definido.
    d. ( ) O teste de Turing serve para verificar se algo foi construído com técnicas de IA.
    e. ( ) No caso da busca heurística, se o cálculo do valor de h for sempre 0, a busca heurística se comporta exatamente como a busca em largura com custo.
6. Considere a árvore de jogo de profundidade 3 e fator de ramificação 3 abaixo.




Para responder a questão objetiva, execute obrigatoriamente os passos indicados abaixo. A questão objetiva somente terá valor se os seguintes passos forem executados.

a. Encontre os valores minimax do nó raiz (estado inicial), utilizando a poda alfa-beta. Comece a exploração pela ramificação mais abaixo, indicada por um *.
b. Circule os nós que não são necessários para a avaliação do valor-minimax do nó raiz.
c. Quantas avaliações de folhas são necessárias para encontrar o valor-minimax do nó raiz?

7. Considere as seguintes colocações:
   a. A busca minimax é do tipo em profundidade; então, em qualquer instante temos de considerar os nós ao longo dos vários caminhos na árvore, independente do nó ser de MIN ou de MAX.

b. A efetividade (eficiência) do algoritmo minimax é altamente dependente da ordem em que os sucessores são examinados.
c. O algoritmo alfa-beta permite podar grandes partes do espaço de busca do jogo inteiro, conseqüentemente ele não tem de fazer a busca em toda a distância até os estados terminais.
d. O algoritmo minimax não é ótimo, pois ele gera o espaço de busca do jogo inteiro.

Marque a alternativa correta:

i. Somente III e IV estão corretas
ii. Somente II e III estão corretas
iii. Somente I, III e IV estão corretas
iv. Somente IV estão corretas
v. Todas estão erradas.

8. Descreva, com suas palavras, o algoritmo de poda alfa-beta.
9. Explique cada uma das estratégias abaixo, destacando qual nó da fronteira é escolhido, como a fronteira se comporta e o desempenho para cada uma delas:
    a. Busca por custo uniforme (menor primeiro)
    b. Busca heurítica pelo melhor primeiro (gulosa)
    c. Busca heurística em profundidade
    d. Busca A*
10. Crie uma função heurística h para o quebra-cabeça de 8 peças que as vezes realize estimativas exageradas (dê um exemplo pelo menos), e mostre como ela pode levar o algoritmo A* a uma solução não ótima em um problema específico. Prove que, se a função h nunca superestimar por um valor maior que c, A* usando h retornará uma função cujo custo excede o da solução ótima por não mais que c.

## Resumo

Quando nós, humanos, nos deparamos com um problema intuitivamente pensamos em todas as possíveis soluções para escolher dentre estas a que tenha a melhor solução. Essa mesma filosofia é utilizada pela a IA para resolver problemas que seguem uma determinada formulação.

Esta unidade é dedicada a estudar os mecanismos de busca que permitem a resolução de problemas. Veremos como formular tais problemas e como resolvê-los sem informação (busca cega) e com informação.

Veremos ainda como avaliar o desempenho desses mecanismos de busca e como lidar em situações que existe competição (busca competitiva). Ao final da unidades os exercícios permitirão, além de fixar os conceitos aprendidos, permitir que se resolvam problemas usando as técnicas vistas nesta unidade.

## Weblografia

Universidade Aberta do Piauí – UAPI
http://www.ufpi.br/uapi
Universidade Aberta do Brasil- UAB
http://www.uab.gov.br
Secretaria de Educação a Distância do MEC – SEED
http://www.seed.mec.gov.br
Associação Brasileira de Educação a Distância – ABED
http://www.abed.org.br
Instituto Sem Fronteiras (ISF)
http://www.isf.org.br/
Busca em Espaço de Estados
http://www.scribd.com/Busca-Espaco-Estados-Inteligencia-Artificial/d/26828107
Definição de problema
http://www.cin.ufpe.br/~vtc/Res04.doc
Agentes Baseados em Conhecimento
http://www.cin.ufpe.br/~lms4/SI/resumosAulas.doc
Resoluções de Problemas
http://www.professeurs.polymtl.ca/michel.gagnon/Disciplinas/Bac/IA/ResolProb/resproblema.html

## Referências Bibliográficas

RUSSEL, Stuart; NORVIG, Peter: Inteligência Artificial. Campus, São Paulo, 2004. 1040p

BITTENCOURT, Guilherme: Inteligência Artificial – Ferramentas e Teorias. Editora da UFSC. 2ª. Edição. Florianópolis, 2001. 362p

RICH, Elaine; KNIGHT, Kevin: Inteligência Artificial. Makron Books. 2ª. Edição. São Paulo, 1994. 722p.

# UNIDADE 3

## APRENDIZAGEM DE MÁQUINA

# 1. Aprendizagem de Máquina

O aprendizado de máquina (AM) é um segmento da Inteligência Artificial que possui um elemento essencial para um comportamento inteligente, a capacidade de aprendizado.

A área de AM é responsável por pesquisar métodos computacionais adequados para a aquisição de novos conhecimentos, novas habilidades e novas formas de organização do conhecimento já existente. O aprendizado possibilita que o sistema faça a mesma tarefa ou tarefas sobre uma mesma população de uma maneira mais eficiente a cada execução. O campo do aprendizado de máquina é concebido pela questão de como construir programas, que automaticamente melhoram com a sua experiência [Michell, 1997].

Segundo Batista:

> "Aprendizado de Máquina – AM – é uma subárea de pesquisa muito importante em inteligência Artificial – IA – pois a capacidade de aprender é essencial para um comportamento inteligente. AM estudo métodos computacionais para adquirir novos conhecimentos, novas habilidade e novos meios de organizar o conhecimento já existente." (2003, p. 11).

## 1.1. Aprenizado Supervisionado

O aprendizado supervisionado ocorre quando os conjuntos de exemplos são fornecidos ao sistema com suas respectivas classes, com isto objetiva-se classificar os novos conjuntos ainda não rotulados. O problema deste modo é encontrar um conjunto de exemplos satisfatório que permita esta condição.

Aprendizado supervisionado (Atividades de Predição):

- Classificação: previsão de classes discretas pré-definidas
- Regressão: previsão de um valor numérico contínuo

O algoritmo de aprendizado (indutor) recebe um conjunto de exemplos de treinamento para os quais os rótulos da classe associada são conhecidos. Cada exemplo (instância ou padrão) é descrito por um vetor de valores (atributos) e pelo rótulo da classe associada. O objetivo do indutor é construir um classificador que possa determinar corretamente a classe de novos exemplos ainda não rotulados.

### 1.1.1. Exemplo: Abordagem com Base em Árvores de Decisão

É um método de aprendizagem supervisionado que constrói árvores de classificação a partir de exemplos (ID3, C4.5, [Quinlan, 1986], CART [Breiman, 1986]). Os métodos baseados em árvores para classificação, dividem o espaço de entrada em regiões disjuntas para construir uma fronteira de decisão.
As regiões são escolhidas baseadas em uma otimização heurística onde a cada passo os algoritmos selecionam a variável que provê a melhor separação de classes de acordo alguma função custo.

**Algoritmo ID3**

ID3 é um algoritmo simples que construí uma árvore de decisão sob as seguintes premissas:
Cada vértice (nó) corresponde a um atributo, e cada aresta da árvore a um valor possível do atributo. Uma folha da árvore corresponde ao valor esperado da decisão segundo os dados de treino utilizados. A explicação de uma determinada decisão está na trajetória da raiz a folha representativa desta decisão.
Cada vértice é associado ao atributo mais informativo que ainda não tenha sido considerado. Para medir o nível de informação de um atributo se utiliza o conceito de entropia da Teoria da Informação. Menor o valor da entropia, menor a incerteza e mais utilidade tem o atributo para a classificação.
Passos para construção de uma árvore de decisão:

1. Seleciona um atributo como sendo o nodo raiz ;
2. Arcos são criados para todos os diferentes valores do atributo selecionado no passo 1;
3. Se todos os exemplos de treinamento sobre uma folha pertencerem a uma mesma classe, esta folha recebe o nome da classe. Se todas as folhas possuem uma classe, o algoritmo termina;
4. Senão, o nodo é determinado com um atributo que não ocorra no trajeto da raiz, e arcos são criados para todos os valores. O algoritmo retorna ao passo 3.

Tomemos como exemplo os dados da Tabela 2. A classe selecionada para esse caso é *empréstimo (sim/não).*

A Figura 13 apresenta um exemplo de Árvore de Decisão, na qual constam os dados da Tabela 2 que relatam as condições para uma pessoa receber um empréstimo. Nesse caso existem duas possíveis classes: Sim (receber empréstimo) e Não (não receber empréstimo). O atributo Montante foi selecionado primeiro pois maior ganho de informação, ou seja quando selecionado o baixo o algortimo já consegue classificar como empréstimo a ser concedido, independente de qualquer para atributo/valor da tabela.

A base de dados é dividida entre os dados que possuem *montante=médio* e *montante=alto.* Para cada uma dessas sub-divisões é escolhido outro atributo, aquele que tiver maior ganho de informação. E assim sucessivamente. No exemplo da Figura 13 para o ramo *montante=médio* foi escolhido o atributo *salário* e para o ramo *montante=alto* foi escolhido o atributo *conta.*

**Tabela 2: Tabela de dados sobre concessão de empréstimo**

| caso | montante | idade | salário | conta | empréstimo |
|---|---|---|---|---|---|
| 1 | médio | sênior | baixo | sim | *não* |
| 2 | médio | sênior | baixo | não | *não* |
| 3 | baixo | sênior | baixo | sim | ***sim*** |
| 4 | alto | média | baixo | sim | ***sim*** |
| 5 | alto | jovem | alto | sim | ***sim*** |
| 6 | alto | jovem | alto | não | *não* |
| 7 | baixo | jovem | alto | não | ***sim*** |
| 8 | médio | média | baixo | sim | *não* |
| 9 | médio | jovem | alto | sim | ***sim*** |
| 10 | alto | média | alto | sim | ***sim*** |
| 11 | médio | média | alto | não | ***sim*** |
| 12 | baixo | jovem | baixo | não | ***sim*** |
| 13 | baixo | sênior | alto | sim | ***sim*** |
| 14 | alto | média | baixo | não | *não* |

A classificação, nesse caso, resulta numa estrutura de árvore, que pode ser usada para todos os objetos do conjunto [BRADZIL, 1999].

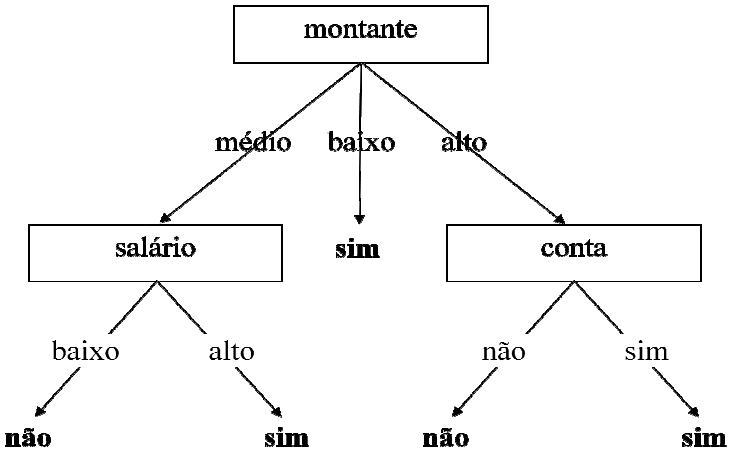



**Figura 13: Exemplo de uma Árvore de Decisão para concessão de empréstimo**

## 1.2. APRENIZADO NÃO-SUPERVISIONADO

O modo de aprendizagem não-supervisionado supõe que o conjunto de exemplos não está rotulado, com isto o sistema tenta classificar estes conjuntos agrupando os semelhantes em determinadas classes.

Aprendizado não-supervisionado (Atividades de Descrição):

- Agrupamentos: agrupar instâncias similares em aglomerados;
- Associação: Detecção de associações entre atributos;

O indutor analisa os exemplos fornecidos e tenta determinar se alguns deles podem ser agrupados de alguma maneira, formando agrupamentos ou clusters. Após a determinação dos agrupamentos, em geral, é necessário uma análise para determinar o que cada agrupamento significa no contexto problema sendo analisado.

### 1.2.1. EXEMPLO: CLUSTERIZAÇÃO CONCEITUAL

A partir de um conjunto de descrições de objetos (eventos, observações, fatos) produz um esquema de classificação sobre esses dados. O chamado aprendizado por observação usa uma função para descobrir classes com boa descrição conceitual. Também é conhecido como Taxonomia Numérica, pois envolve a produção de uma hierarquia de classe, usando medida matemática de similaridade entre as instâncias.

O agrupamento conceitual foi Introduzido inicialmente por [Michalski,1980] como um processo de construção de uma rede de conceitos. Caracteriza uma coleção de objetos com nós associados a conceitos, descrevendo classes de objetos e links associados às relações entre as classes. Podemos definir agrupamento conceitual como:

- Dado:
  - ✓ Um conjunto de objetos.
  - ✓ Um conjunto de atributos usados para caracterizar os objetos.
  - ✓ Um corpo de conhecimento adquirido - incluindo problemas de restrições, propriedades dos atributos, critérios para avaliação de qualidade da classificação construída.
- Encontrar:
  - ✓ Uma hierarquia de classes de objetos
  - ✓ Cada nó deve formar um conceito coerente
  - ✓ Compacto
  - ✓ Facilmente representado em termos de uma definição ou regra que tenha uma interpretação natural para humanos

Tomemos como exemplo descrição de animais da Tabela 3.

**Tabela 3: Base de Dados sobre características do corpo dos animais**

| Nome | Cobertura do Corpo | Cavidades do Coracão | Temperatura do Corpo | Fertilização |
|---|---|---|---|---|
| mamífero | pelos | 4 | regulada | interna |
| pássaro | penas | 4 | regulada | interna |
| réptil | pele seca | 4 imperfeitas | não regulada | interna |
| anfíbio | pele úmida | 3 | não regulada | externa |
| peixe | escamas | 2 | não regulada | externa |

Como não há uma classe indicada como no algoritmo supervisionado é feito uma agrupamento por semelhança dos registros. Um exemplo dessa classificação pode ser vista na

Figura 14.

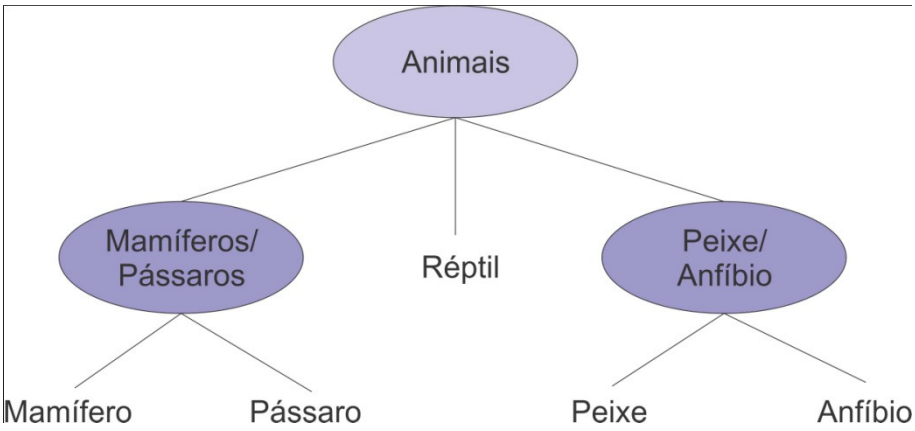



**Figura 14: Agrupamento - Base de dados de características dos animais**

### Cobweb

O Cobweb [Fischer, 1987] é um método de agrupamento incremental do tipo árvore de classificação, cujos nós são conceitos probabilísticos, onde os objetos são inseridos um a um. Em cada nó se guardam as probabilidades (no. de instâncias e números de exemplos contidos em cada classe).

Baseado no princípio de que um bom agrupamento deve minimizar a distância entre objetos em um grupo (**Similaridade inter-grupo**) e maximizar a distância entre objetos de grupos diferentes (**Similaridade intra-grupo**). Sendo assim o objetivo do COBWEB é encontrar um bom equilíbrio entre essas distâncias, sendo uma troca entre a similaridade intra-grupo e a dissimilaridade inter-grupo dos objetos.

Ao tentar classificar um objeto o algoritmo faz quatro operações:

- Inserção: Determinar que categoria 'melhor' hospeda um objeto novo.
- Criar nova classe: Cria um novo agrupamento para o objeto
- Combinar: Combina duas classes e coloca o novo objeto nela.
- Divisão: Divide uma classe e coloca o objeto em ambas as classes.

Para cada uma das operações é calculado o Category Utility (C.U.) a operação que tiver o maior valor de C.U. será executada. Essa métrica favorece a criação de conceitos que maximizam o potencial de inferência sobre informações. [Rebolsas & Furtado, 2004] Category Utility para uma partição de conceitos ($CU(\{C_1,C_2,..., C_n\})$) é:

$$CU(C) = \frac{1}{n}\sum_{k=1}^{n} P(C_k)\left[\sum_i \sum_j P(A_i = V_{ij}|C_k)^2 - \sum_i \sum_j P(A_i = V_{ij})^2\right]$$

## 2. Aprendizagem por Reforço.

Aprendizagem por Reforço (AR) permite ao agente adquirir uma capacidade de conhecimento do ambiente que não estava disponível em tempo de projeto [Sutton & Barto, 2002]. AR é baseada na existência de um crítico externo ao ambiente, que avalia a ação tomada, mas sem indicar explicitamente a ação correta. Formalmente, AR utiliza uma estrutura composta de estados, ações e recompensas conforme mostra a Figura 15.

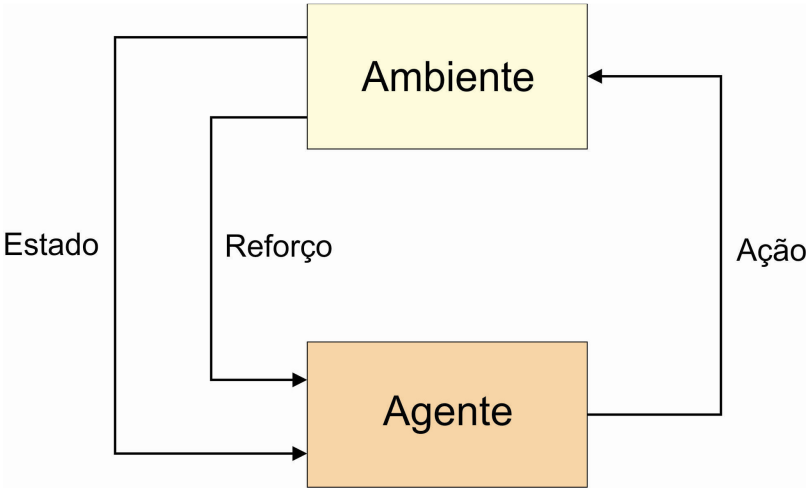



**Figura 15: Diagrama da Aprendizagem por Reforço**

O agente atua em um ambiente descrito por um conjunto de possíveis estados e pode executar, para cada estado, uma ação dentro de um conjunto de ações possíveis, recebendo um valor de reforços a cada vez que executa uma ação. Este reforço indica o valor imediato da transição estado-ação-novo estado. Ao longo do tempo, este processo produz uma seqüência de pares estado-ação, e seus respectivos valores de reforços. O objetivo do agente é aprender uma política que maximize uma soma esperada destes reforços a longo prazo.

Uma consideração importante a ser feita quando se lida com AR é conflito exploração X explotação. O agente deve lidar com o problema de haver um compromisso entre escolher a exploração de estados e ações desconhecidos, de

modo a coletar nova informação, ou a explotação de estados e ações que já foram aprendidos e que irão gerar altas recompensas, de modo a maximizar seus reforços. Sendo assim, por um lado o agente deve aprender quais ações maximizam os valores das recompensas obtidas no tempo, e, por outro, deve agir de forma a atingir esta maximização, explorando ações ainda não executadas ou regiões pouco visitadas do espaço de estados.

É importante, portanto, estabelecer uma política mista de explotação e exploração (política ε-greedy), que inclua a escolha intencional (com probabilidade ε) de se executar uma ação que não é considerada a melhor no estado atual, visando a aquisição de conhecimentos a respeito de estados ainda desconhecido ou pouco visitados. A escolha desta ação ocorre de forma aleatória. Em uma política de explotação pura (greedy) escolhem-se as ações que se julguem (talvez erroneamente, caso o algoritmo de AR ainda esteja longe da convergência) serem as melhores para resolver o problema.

## 2.1. Elementos Básicos da Aprendizagem por Reforço

Além de agente e ambiente, pode-se identificar quatro elementos fundamentais:

- Política (de controle ou de tomada-de-decisão)
- Função ganho (reward function)
- Função valor (value function)
- Modelo do ambiente (opcional)

A Política de Controle define o comportamento do agente sendo uma função de estados para ações *Ação = p(estado).* A política pode ser simplesmente uma tabela ou, em outros casos, pode envolver computações complexas.

Em geral políticas podem ser estocásticas[2]. No estado s, a probabilidade de se tomar a ação a é dada por p(s,a).

A função ganho define a meta em um problema AR. Ela associa um estado (ou par estado-ação) do ambiente a um número, denominado ganho. O objetivo do aprendiz é maximizar o ganho acumulado em longo prazo.

[2] Padrões estocásticos são aqueles que surgem por meio de eventos aleatórios. Por exemplo, o lançar de dados resulta em numéricos estocásticos, pois qualquer uma das 6 faces do dado tem iguais probabilidades de ficar para cima quando de seu arremesso.

Enquanto a função ganho indica os movimentos promissores imediatos, a função valor estado indica o ganho total que pode ser acumulado no futuro se iniciarmos no estado em consideração

A função valor indica o ganho acumulado em longo termo a partir de um estado, levando em conta os estados que sucedem o estado em consideração:

$$V(s) = E[r_{t+1} + r_{t+2} + r_{t+3} + \ldots : s_t = s]$$

Já o Modelo do Ambiente imita o comportamento do próprio ambiente. Dados o estado e a ação, o modelo antecipa o próximo estado e o ganho

- Estado corrente no instante t: s
- Ação a ser tomada: a
- Modelo antecipa o próximo estado:

$$P(s_{t+1}=s' \mid s_t= s, a_t = a)$$

Esses modelos são usados para planejamento (qualquer método para decidir um curso de ação ao considerarmos situações futuras antes de encontrá-las).

Existem duas classes de modelos: Especificado através de equações do Processo de Decisão Markoviano e o definido implicitamente, que no caso usa-se um simulador.

O modelo de equações é representado como segue:

- Espaço de estados
- Conjunto de ações que podem ser tomadas em cada estado:
  - A(s) para s Î S
- Probabilidades de transição:
  - $P(s_{t+1}= s' \mid s_t = s, a_t = a)$
- Ganhos:
  - $P(r_{t+1} = r \mid s_{t+1} = s', s_t = s, a_t = a)$

Em outro modelo desenvolve-se um simulador do ambiente. Dado o estado corrente $s_t$ e a ação $a_t$, o simulador responde com o próximo estado $s_{t+1}$ e o ganho $r_{t+1}$

## 2.2. Exemplo: Jogo da Velha

Considere como cenário para o exemplo do jogo da velha um jogador experiente que jamais perde e considere um adversário imperfeito, que às vezes toma decisões incorretas. A questão é: Como poderíamos construir um jogador

artificial que “identifica” as imperfeições do oponente e aprende a maximizar as chances de vitória?

Para responder essa questão temos que projetar um aprendiza que “identifica” as imperfeições do oponente e aprende a maximizar as chances de vitória. Uma solução satisfatória não pode ser obtida através de técnicas clássicas. A técnica MiniMax, por exemplo, baseada na teoria dos jogos, assume um comportamento particular do oponente. Os métodos de otimização para problemas de decisão sequencial (programação dinâmica) são capazes de calcular uma solução ótima. Contudo, eles exigem uma especificação completa do oponente, incluindo distribuições de probabilidade de seus movimentos.

O comportamento do adversário pode não ser conhecido a priori, mas podemos identificá-lo a partir das seguintes interações:

- Jogar com o oponente e identificar o modelo, até um certo nível de confiança
- Aplicar técnicas de programação dinâmica para calcular a política de decisão ótima

Como podemos observar a aprendizagem de máquina não é muito diferente dos passos acima.

**Aplicando AR ao jogo da velha**

- Criamos uma tabela de números, V(s), com uma entrada para cada estado s do jogo
- V(s) é a estimativa mais recente de vencermos o jogo a partir do estado s
- V(s) é o valor do estado s
- A tabela V representa a função valor

Um estado $s_1$ é considerado melhor do que um estado $s_2$, se $V(s_1) > V(s_2)$. Uma linha com três X’s tem probabilidade 1 de vitória – já ganhamos o jogo. Uma linha com três O’s tem probabilidade 0 de vitória – o jogo já está perdido. Para os demais estados, chutamos probabilidade ½ de vitória.

Na maioria das vezes selecionamos o movimento que nos leva ao estado com maior valor, ou seja, o estado com maior probabilidade de vitória (explotar). Ocasionalmente, selecionamos aleatoriamente dentre os demais movimentos possíveis (vasculhar / explorar).

Enquanto jogamos, atualizamos a tabela V e tentamos obter estimativas mais precisas das probabilidades de vencer. Para tanto, atualizamos o valor do estado após cada movimento guloso (explotação). O valor corrente é ajustado para se tornar mais próximo do último valor.

Como regra de atualização utilizamos:

$V(s) \leftarrow V(s) + \alpha[V(s'') - V(s)]$ onde $\alpha$ é um número positivo pequeno, chamado de passo (taxa de aprendizagem).

A regra de atualização acima é chamada de método de aprendizagem por diferença temporal (temporal-difference learning method).

A partir de um modelo simples como apresentado acima, um agente com aprendizagem por reforço explora as fraquezas do oponente, a partir de sua própria experiência. Com o passar do tempo, a experiência é convertida em valores da função valor-estado V(s). Esta experiência induz a política de controle/decisão.

## 3. Redes Neurais.

Redes Neurais Artificiais são técnicas computacionais que apresentam um modelo matemático inspirado na estrutura neural de organismos inteligentes e que adquirem conhecimento através da experiência. Uma grande rede neural artificial pode ter centenas ou milhares de unidades de processamento; já o cérebro de um mamífero pode ter muitos bilhões de neurônios.

O sistema nervoso é formado por um conjunto extremamente complexo de células, os neurônios (Figura 16). Eles têm um papel essencial na determinação do funcionamento e comportamento do corpo humano e do raciocínio. Os neurônios são formados pelos dendritos, que são um conjunto de terminais de entrada, pelo corpo central, e pelos axônios que são longos terminais de saída.

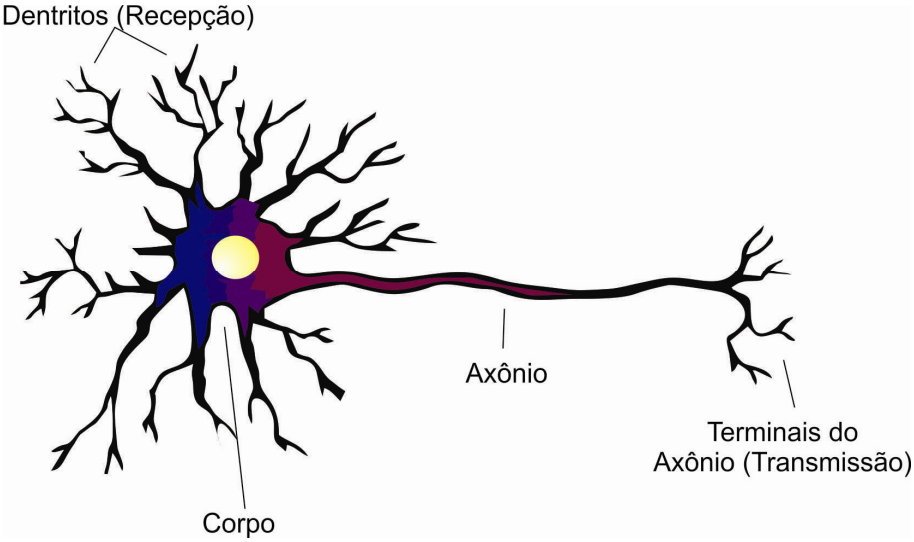



**Figura 16: Modelo de neurônio humano**

Os neurônios estão conectados uns aos outros através de sinapses, e juntos formam uma grande rede, chamada REDE NEURAL. As sinapses transmitem estímulos através de diferentes concentrações de Na+ (Sódio) e K+ (Potássio), e o resultado disto pode ser estendido por todo o corpo humano. Esta grande rede proporciona uma fabulosa capacidade de processamento e armazenamento de informação.

As redes neurais artificiais consistem em um método de solucionar problemas de inteligência artificial, construindo um sistema que tenha circuitos que simulem o cérebro humano, inclusive seu comportamento, ou seja, aprendendo, errando e fazendo descobertas. São mais que isso, são técnicas computacionais que apresentam um modelo inspirado na estrutura neural de organismos inteligentes e que adquirem conhecimento através da experiência. Uma grande rede neural artificial pode ter centenas ou milhares de unidades de processamento, enquanto que o cérebro de um mamífero pode ter muitos bilhões de neurônios.

## 3.1. O Neurônio Artificial

Uma rede neural artificial é composta por várias unidades de processamento, cujo funcionamento é bastante simples. Essas unidades, geralmente são conectadas por canais de comunicação que estão associados a determinado peso. As unidades fazem operações apenas sobre seus dados locais, que são entradas recebidas

pelas suas conexões. O comportamento inteligente de uma Rede Neural Artificial vem das interações entre as unidades de processamento da rede.

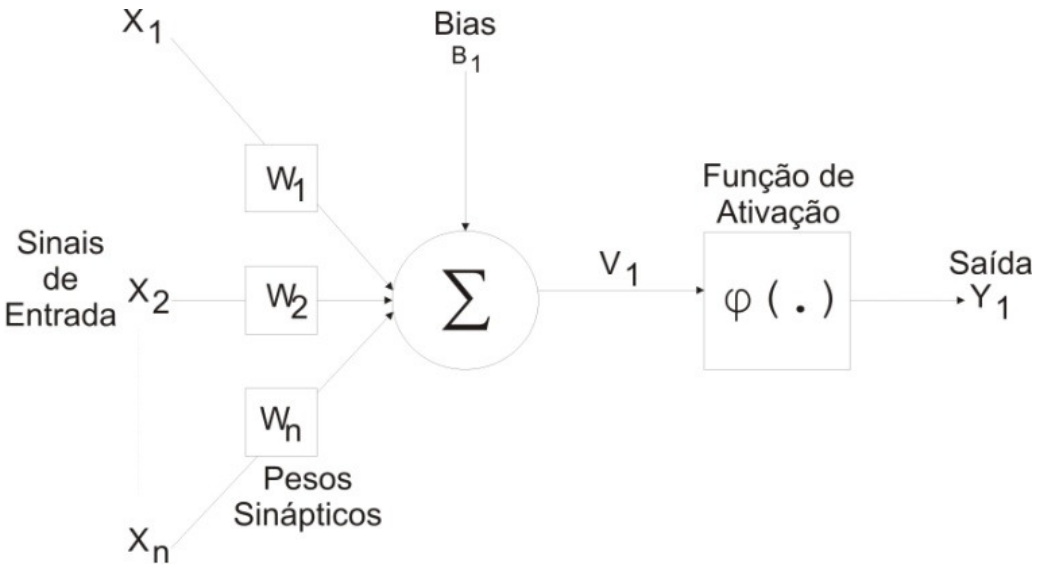



**Figura 17: Modelo de um neurônio artificial**

A operação de uma unidade de processamento, proposta por McCullock e Pitts em 1943 [McCullock & Pitts, 1943], pode ser resumida da seguinte maneira:

- Sinais são apresentados à entrada;
- Cada sinal é multiplicado por um número, ou peso, que indica a sua influência na saída da unidade;
- É feita a soma ponderada dos sinais que produz um nível de atividade;
- Se este nível de atividade exceder um certo limite (threshold) a unidade produz uma determinada resposta de saída.

Suponha que tenhamos p sinais de entrada $X_1$, $X_2$, ..., $X_p$ e pesos $w_1$, $w_2$, ..., $w_p$ e limitador t; com sinais assumindo valores booleanos (0 ou 1) e pesos valores reais. Neste modelo, o nível de atividade a é dado por:

$a = w_1X_1 + w_2X_2 + ... + w_pX_p$

A saída y é dada por

y = 1, se a >= t ou

y = 0, se a < t.

A maioria dos modelos de redes neurais possui alguma regra de treinamento, onde os pesos de suas conexões são ajustados de acordo com os padrões apresentados. Em outras palavras, elas aprendem através de exemplos.

### 3.1.1. TREINAMENTO DE UM PERCEPTRON

O treinamento supervisionado do modelo de rede Perceptron, consiste em ajustar os pesos e os *thresholds* de suas unidades para que a classificação desejada seja obtida. Para a adaptação do *threshold* juntamente com os pesos podemos considerá-lo como sendo o peso associado a uma conexão, cuja entrada é sempre igual à -1 e adaptar o peso relativo a essa entrada.

Quando um padrão é inicialmente apresentado à rede, ela produz uma saída. Após medir a distância entre a resposta atual e a desejada, são realizados os ajustes apropriados nos pesos das conexões de modo a reduzir esta distância.Este procedimento é conhecido como Regra Delta. Esse principio se aplica a maioria dos treinamentos das redes neurais.

Deste modo, temos o seguinte esquema de treinamento (Figura 18):

- Iniciar todas as conexões com pesos aleatórios;
- Repita até que o erro E seja satisfatoriamente pequeno (E = e)

Para cada par de treinamento (X,d), faça:

- Calcular a resposta obtida O;
- Se o erro não for satisfatoriamente pequeno E > e, então:
- Atualizar pesos: $W_{novo} := W_{anterior} + \eta EX$

Onde:

- O par de treinamento (X, d) corresponde ao padrão de entrada e a sua respectiva resposta desejada;
- O erro E é definido como: Resposta Desejada - Resposta Obtida (d - O);
- A taxa de aprendizado neta é uma constante positiva, que corresponde à velocidade do aprendizado.

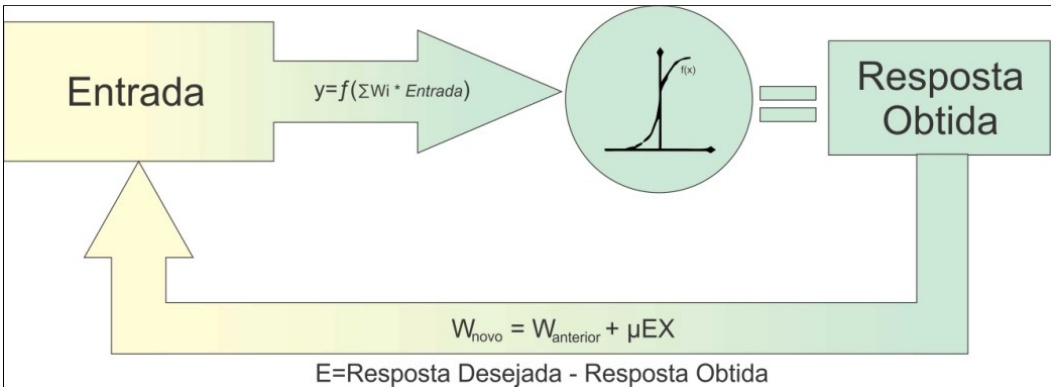



**Figura 18: Esquema de Treinamento de um Perceptron**

As respostas geradas pelas unidades são calculadas através de uma função de ativação. Existem vários tipos de funções de ativação, as mais comuns (

Figura 19) são: *Hard Limiter*, *Threshold Logic* e *Sigmoid*.

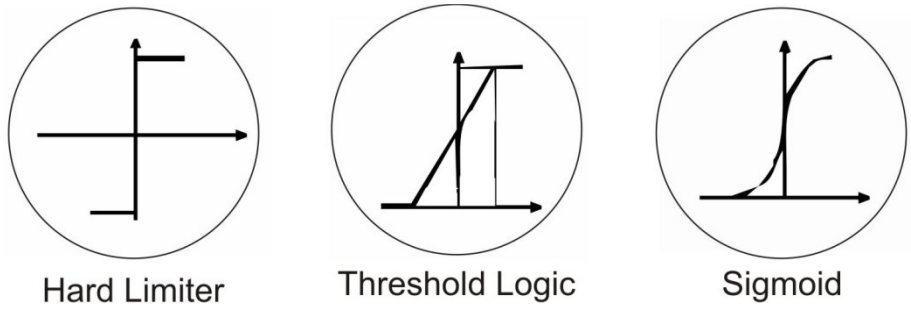



**Figura 19: Funções de Ativações mais comuns**

## 3.2. A REDE NEURAL ARTIFICIAL (MULTILAYER PERCEPTRON)

Arquiteturas neurais são tipicamente organizadas em camadas, com unidades que podem estar conectadas às unidades da camada posterior. Quando Redes Neurais Artificiais de uma só camada são utilizadas os padrões de treinamento apresentados à entrada são mapeados diretamente em um conjunto de padrões de saída da rede, ou seja não é possível a formação de uma representação interna. Neste caso, a codificação proveniente do mundo exterior deve ser suficiente para implementar esse mapeamento.

Tal restrição implica que padrões de entrada similares resultem em padrões de saída similares, o que leva o sistema à incapacidade de aprender importantes mapeamentos. Como resultado, padrões de entrada com estruturas similares,

fornecidos do mundo externo, que levem a saídas diferentes não são possíveis de serem mapeados por redes sem representações internas, isto é, sem camadas intermediárias. Um exemplo clássico deste caso é a função ou-exclusivo (XOR).

Minsky e Papert [Minsky & Papert, 1969] analisaram matematicamente o Perceptron e demostraram que redes de uma camada não são capazes de solucionar problemas que não sejam linearmente separáveis. Como não acreditavam na possibilidade de se construir um método de treinamento para redes com mais de uma camada, eles concluíram que as redes neurais seriam sempre suscetíveis a essa limitação.

Contudo, o desenvolvimento do algoritmo de treinamento backpropagation, por Rumelhart, Hinton e Williams em 1986 [Rumelhart et. al., 1986], precedido por propostas semelhantes ocorridas nos anos 70 e 80, mostrou que é possível treinar eficientemente redes com camadas intermediárias, resultando no modelo de Redes Neurais Artificiais mais utilizado atualmente, as redes Perceptron Multi-Camadas (MLP), treinadas com o algoritmo backpropagation.

Nessas redes, cada camada tem uma função específica. A camada de saída recebe os estímulos da camada intermediária e constrói o padrão que será a resposta. As camadas intermediárias funcionam como extratoras de características, seus pesos são uma codificação de características apresentadas nos padrões de entrada e permitem que a rede crie sua própria representação, mais rica e complexa, do problema.

### 3.2.1. Backpropagation

Durante o treinamento com o algoritmo backpropagation, a rede opera em uma sequência de dois passos. Primeiro, um padrão é apresentado à camada de entrada da rede. A atividade resultante flui através da rede, camada por camada, até que a resposta seja produzida pela camada de saída. No segundo passo, a saída obtida é comparada à saída desejada para esse padrão particular. Se esta não estiver correta, o erro é calculado. O erro é propagado a partir da camada de saída até a camada de entrada, e os pesos das conexões das unidades das camadas internas vão sendo modificados conforme o erro é retropropagado.

As redes que utilizam backpropagation trabalham com uma variação da regra delta, apropriada para redes multi-camadas: a regra delta generalizada. A regra

delta padrão essencialmente implementa um gradiente descendente no quadrado da soma do erro para funções de ativação lineares. Redes sem camadas intermediárias, podem resolver problemas onde a superfície de erro tem a forma de um parabolóide com apenas um mínimo. Entretanto, a superfície do erro pode não ser tão simples, como a ilustrada na figura abaixo, e suas derivadas mais difíceis de serem calculadas. Nestes casos devem ser utilizadas redes com camadas intermediárias. Ainda assim, as redes ficam sujeitas aos problemas de procedimentos "hill-climbing", ou seja, ao problema de mínimos locais.

A regra delta generalizada funciona quando são utilizadas na rede unidades com uma função de ativação semi-linear, que é uma função diferenciável e não decrescente. Note que a função threshold não se enquadra nesse requisito. Uma função de ativação amplamente utilizada, nestes casos, é a função sigmoid.

A taxa de aprendizado é uma constante de proporcionalidade no intervalo [0,1], pois este procedimento de aprendizado requer apenas que a mudança no peso seja proporcional à neta.

Entretanto, o verdadeiro gradiente descendente requer que sejam tomados passos infinitesimais. Assim quanto maior for essa constante, maior será a mudança nos pesos, aumentando a velocidade do aprendizado, o que pode levar à uma oscilação do modelo na superfície de erro. O ideal seria utilizar a maior taxa de aprendizado possível que não levasse à uma oscilação, resultando em um aprendizado mais rápido.

O treinamento das redes MLP com backpropagation pode demandar muitos passos no conjunto de treinamento, resultando um tempo de treinamento considerávelmente longo. Se for encontrado um mínimo local, o erro para o conjunto de treinamento pára de diminuir e estaciona em um valor maior que o aceitável. Uma maneira de aumentar a taxa de aprendizado sem levar à oscilação é modificar a regra delta generalizada para incluir o termo momentum, uma constante que determina o efeito das mudanças passadas dos pesos na direção atual do movimento no espaço de pesos.

Desta forma, o termo momentum leva em consideração o efeito de mudanças anteriores de pesos na direção do movimento atual no espaço de pesos. O termo momentum torna-se útil em espaços de erro que contenham longas gargantas, com curvas acentuadas ou vales com descidas suaves, como o apresentado na figura acima.

### 3.2.2. Treinamento do MLP

O treinamento supervisionado da rede MLP utilizando backpropagation consiste em dois passos. No primeiro, um padrão é apresentado às unidades da camada de entrada e, a partir desta camada as unidades calculam sua resposta que é produzida na camada de saída, o erro é calculado e o no segundo passo, este é propagado a partir da camada de saída até a camada de entrada, e os pesos das conexões das unidades das camadas internas vão sendo modificados utilizando a regra delta generalizada.

**Resumo do Algoritmo:**

1. Inicialização: Inicialize os pesos sinápticos e os bias aleatóriamente, com valores no intervalo [-1;1];
2. Apresentação dos Exemplos de Treinamento:
   Treinamento "on-line": Para cada exemplo do conjunto de treinamento, efetue os passos 3 e 4.
   Treinamento "em lote": Para cada "época" do conjunto de treinamento, efetue os passos 3 e 4.
3. Computação para Frente (Propagação): Depois de apresentado o exemplo do conjunto de treinamento *T = {(x(n),d(n)}*, sendo *x(n)* a entrada apresentada à rede e *d(n)* a saída desejada, calcule o valor da ativação $v_j$ e a saída para cada unidade da rede, da seguinte forma:
   $v_j = \sum_{i=1}^{m} w_{ji} x_i + b$ , para o cálculo do valor da ativação e $f(v) = \frac{1}{1+e^{(-\alpha v)}}$ ,para o cálculo da saída y da unidade k, utilizando a função sigmóide, como no exemplo, ou uma outra função se necessário.
   Utilize a saída das unidades de uma camada como entradas para a seguinte, até a última camada. A saída das unidades da última camada será a resposta da rede.
4. Calcule o Sinal de Erro: Fazendo a saída $y_j = O_j(n)$, será $O_j(n)$ a resposta da rede, calcule o sinal de erro através da seguinte formula:
   $e_j(n)=d_j(n)-O_j(n)$ , onde $d_j(n)$ é a saída desejada com resposta para cada unidade na interação (n).

Este sinal de erro será utilizado para computar os valores dos erros das camadas anteriores e fazer as correções necessárias nos pesos sinápticos.

**5.** Computação para Trás (Retropropagação): Calcule os erros locais, d, para cada unidade, desde a camada de saída até a de entrada. O gradiente local é definido por:

$\delta_j(n) = e_j(n)O_j(n)(1 - O_j(n))$ para a unidade da camada de saída ou $\delta_j(n) = O_j(n)(1 - O_j(n))\sum \delta_k w_{jk}$ , para as unidades das demais camadas. Onde:

$O_j(1-O_j)$ - é a função de ativação diferenciada em função do argumento, i.e., valor de ativação;

$d_k$ - é o erro das unidades da camada anterior conectadas a unidade j;

$w_{jk}$ - são os pesos das conecções com a camada anterior.

Após o cálculo dos erros de cada unidade, calcule o ajuste dos pesos de cada conexão segundo a regra delta generalizada e atualize os pesos:

$\Delta w_{kj}(n+1) = \alpha w_{kj}(n) + \eta \delta_j y_j$ , para o cálculo dos ajustes dos pesos

Faça: $w(n+1) = w(n) + \Delta w_{kj}(n)$ , para atualizar os pesos sinápticos, onde:

α - é a constante de momentun, quando a= 0, esta função funciona como a regra delta comum;

$\eta$ - é a taxa de aprendizado;

$d_j$ - é o erro da unidade;

$y_j$ - é a saída produzida pela unidade j;

6. Interação: Refaça os itens 3, 4 e 5 referentes à propagação, cálculo do erro e retropropagação, apresentando outros estímulos de entrada, até que sejam satisfeitas as condições de treinamento; as quais podem ser:
    - O erro da rede está baixo, sendo pouco alterado durante o treinamento;
    - O número máximo de ciclos de treinamento foi alcançado.

## 3.3. Por Que Utilizar Redes Neurais?

De acordo com diversas estruturas neurais e algoritmos de aprendizagem propostos por vários pesquisadores, redes neurais possuem certas características exclusivas de sistemas biológicos. Tais características entram em conflito com os tradicionais métodos computacionais. Sistemas de computação baseados em redes neurais tem a capacidade de receber ao mesmo tempo várias entradas e distribuí-las de maneira organizada. Geralmente, as informações armazenadas por uma rede neural é compartilhada por todas as suas unidades de processamento. Característica que contrasta com os atuais esquemas de memória, onde a informação fica confinada em um determinado endereço.

Em um sistema de rede neural, a informação pode parecer ter representação redundante, porém, o fato de que ela se encontre distribuída por todos os elementos da rede significa que mesmo que parte da rede seja destruída, a informação contida nesta parte ainda estará presente na rede, e poderá ser recuperada. Portanto, a redundância na representação de informações em uma rede neural, diferente de outros sistemas, transforma-se em uma vantagem, que torna o sistema tolerante a falhas. Os atributos de uma rede neural, tais como aprender através de exemplos, generalizações redundantes, e tolerância a falhas, proporcionam fortes incentivos para a escolha de redes neurais como uma escolha apropriada para aproximação para a modelagem de sistemas biológicos. Todo o potencial de uma rede neural pode ser enumerado nos parágrafos seguintes.

O modelo de rede neural tem muitos neurônios conectados por pesos com capacidade de adaptação que que podem ser arranjados em uma estrutura paralela. Por causa deste paralelismo, a falha de alguns nêurons não causam efeitos significantes para a performance de todo o sistema, o que é chamado de tolerância a falhas.

A principal força na estrutura de redes neurais reside em sua habilidades de adaptação e aprendizagem. A habilidade de adaptação e aprendizagem pelo ambiente significa que modelos de redes neurais podem lidar com dados imprecisos e situações não totalmente definidas. Uma rede treinada de maneira razoável tem a habilidade de generalizar quando é apresentada à entradas que não estão presentes em dados já conhecidos por ela.

A característica mais significante de redes neurais está em sua habilidade de aproximar qualquer função continua não linear de um grau de correção desejado. Esta habilidade das redes neurais as tem tornado útil para modelar sistemas não lineares na combinação de controladores não lineares.

Redes Neurais podem ter várias entradas e várias saídas, eles são facilmente aplicáveis à sistemas com muitas variáveis. Com o avanço em tecnologias de hardware, existem componentes com funções voltadas á sistemas com implementações voltadas para redes neurais, o que traz uma velocidade adicional à computação neural.

## 3.4. APLICAÇÕES DE REDES NEURAIS ARTIFICIAIS

Aplicações de redes neurais são inúmeras. Muitos recebem sua primeira introdução lendo a respeito das técnicas no prognóstico de mercados financeiros. Grupos de investimento conhecidos utilizam redes neurais para analisar pelo menos uma parte do mercado financeiro e fazerem suas seleções.

O reconhecimento ótico de caracteres (OCR) é outro tipo de aplicação que já existe e está crescendo, e em breve estaremos em constante contato com esse tipo de aplicação. Outras aplicações bem sucedidas das técnicas de redes neurais artificiais são: análise de pesquisa de mercado, como acima citado, controle de processos industriais, aplicações climáticas, e identificação de fraude de cartão de crédito. Um banco americano chamado Mellon Bank instalou um sistema de detecção de fraudes de cartão de crédito implementado com técnicas de redes neurais e os prejuízos evitados pelo novo sistema conseguiram cobrir os gastos de instalação em seis meses. Vários outros bancos começam a utilizar sistemas baseados em redes neurais para controlar fraudes de cartão de crédito. Estes sistemas têm a capacidade de reconhecer uso fraudulento com base nos padrões criados no passado com uma precisão melhor que em outros sistemas.

Outro exemplo da utilização de redes neurais para melhoria na tomada de decisões é no diagnóstico médico. Em seu aprendizado, são submetidos uma série de diagnósticos de pacientes, de várias características, com vários sintomas e os resultados de seus testes. Também serão fornecidos os diagnósticos médicos para cada doença. Então quando forem apresentados os dados de um novo paciente, com seus sintomas, a rede fornecerá um diagnóstico para os novos casos. Isto

essencialmente criará um sistema com o conhecimento de vários médicos, e fornecerá um diagnóstico inicial em tempo real à um médico. É importante mencionar que com isso o que se pretende é implementar uma ferramenta de auxílio ao médico, e não um programa que o substitua.

Outras aplicações:

- análise e processamento de sinais;
- controle de processos;
- robótica;
- classificação de dados;
- reconhecimento de padrões em linhas de montagem;
- filtros contra ruídos eletrônicos;
- análise de imagens;
- análise de voz;
- avaliação de crédito;
- análise de aroma e odor- um projeto que está em desenvolvimento, buscando a análise de odor via nariz eletrônico;
- análise e diagnóstico de descargas parciais pelo reconhecimento do padrão acústico- trata-se de uma tese de mestrado cujo objetivo é criar um sistema com capacidades de classificar o padrão acústico de uma descarga parcial ;

## 4. Algortmos Genéticos

Os problemas de otimização são baseados em três pontos principais: a codificação do problema, a função objetivo que se deseja maximizar ou minimizar e o espaço de soluções associado. Pode-se imaginar um problema de otimização como uma caixa preta com n botões, onde cada botão é um parâmetro do problema, e uma saída que é o valor da função objetivo, indicando se um determinado conjunto de parâmetros é bom ou não para resolver este problema.

Os algoritmos genéticos são uma família de modelos computacionais inspirados na evolução, que incorporam uma solução potencial para um problema específico numa estrutura semelhante a de um cromossomo e aplicam operadores de seleção e "cross-over" a essas estruturas de forma a preservar informações críticas relativas à solução do problema. Normalmentes os AG's são vistos como otimizadores de

funções, embora a quantidade de problemas para o qual os AG's se aplicam seja bastante abrangente.

As técnicas de busca e otimização tradicionais iniciam-se com um único candidato que, iterativamente, é manipulado utilizando algumas heurísticas (estáticas) diretamente associadas ao problema a ser solucionado. Geralmente, estes processos heurísticos não são algorítmicos e sua simulação em computadores pode ser muito complexa. Apesar destes métodos não serem suficientemente robustos, isto não implica que eles sejam inúteis. Na prática, eles são amplamente utilizados, com sucesso, em inúmeras aplicações.

Por outro lado, as técnicas de computação evolucionária operam sobre uma população de candidatos em paralelo. Assim, elas podem fazer a busca em diferentes áreas do espaço de solução, alocando um número de membros apropriado para a busca em várias regiões. Os Algoritmos Genéticos (AGs) diferem dos métodos tradicionais de busca e otimização, principalmente em quatro aspectos:

1. AGs trabalham com uma codificação do conjunto de parâmetros e não com os próprios parâmetros.
2. AGs trabalham com uma população e não com um único ponto.
3. AGs utilizam informações de custo ou recompensa e não derivadas ou outro conhecimento auxiliar.
4. AGs utilizam regras de transição probabilísticas e não determinísticas.

Algoritmos Genéticos são muito eficientes para busca de soluções ótimas, ou aproximadamente ótimas em uma grande variedade de problemas, pois não impõem muitas das limitações encontradas nos métodos de busca tradicionais.

Além de ser uma estratégia de gerar-e-testar muito elegante, por serem baseados na evolução biológica, são capazes de identificar e explorar fatores ambientais e convergir para soluções ótimas, ou aproximadamente ótimas em níveis globais.

"Quanto melhor um indivíduo se adaptar ao seu meio ambiente, maior será sua chance de sobreviver e gerar descendentes": este é o conceito básico da evolução genética biológica.A área biológica mais proximamente ligada aos Algoritmos Genéticos é a Genética Populacional.

Os pesquisadores referem-se a "algoritmos genéticos" ou a "um algoritmo genético" e não "ao algoritmo genético", pois AGs são uma classe de procedimentos com muitos passos separados, e cada uma destes passos possui muitas variações possíveis.

## 4.1. Conceitos Básicos

Antes de prosseguir com a análise das características dos algoritmos genéticos, alguns conceitos básicos são necessários; estes conceitos podem ser naturalmente expostos explicando o funcionamento destes algoritmos.

- cromossomo (genótipo) - cadeia de bits que representa uma solução possível para o problema.
- gene - representação de cada parâmetro de acordo com o alfabeto utilizado (binário, inteiro ou real).
- fenótipo - cromossomo codificado.
- população - conjunto de pontos (indivíduos) no Espaço de Busca.
- geração - iteração completa do AG que gera uma nova população.
- aptidão bruta - saída gerada pela função objetivo para um indivíduo da população.
- aptidão normalizada - aptidão bruta normalizada, entrada para o algoritmo de seleção.
- aptidão máxima - melhor indivíduo da população corrente.
- aptidão média - aptidão média da população corrente.

Deve ser observado que cada cromossomo, chamado de indivíduo no AG, corresponde a um ponto no espaço de soluções do problema de otimização. O processo de solução adotado nos algoritmos genéticos consiste em gerar, através de regras específicas, um grande número de indivíduos, população, de forma a promover uma varredura tão extensa quanto necessária do espaço de soluções.

## 4.2. Funcionamento do Algoritmo

Inicialmente, é gerada uma população formada por um conjunto aleatório de indivíduos que podem ser vistos como possíveis soluções do problema. Durante o processo evolutivo, esta população é avaliada: para cada indivíduo é dada uma nota, ou índice, refletindo sua habilidade de adaptação a determinado ambiente.

### 4.2.1. Seleção dos mais Aptos

Uma porcentagem dos mais adaptados são mantidos, enquanto os outros são descartados (darwinismo). O princípio básico do funcionamento dos AGs é que um critério de seleção vai fazer com que, depois de muitas gerações, o conjunto inicial de indivíduos gere indivíduos mais aptos. A maioria dos métodos de seleção são projetados para escolher preferencialmente indivíduos com maiores notas de aptidão, embora não exclusivamente, a fim de manter a diversidade da população. Um método de seleção muito utilizado é o Método da Roleta, onde indivíduos de uma geração são escolhidos para fazer parte da próxima geração, através de um sorteio de roleta.

Neste método, cada indivíduo da população é representado na roleta proporcionalmente ao seu índice de aptidão (Figura 20). Assim, aos indivíduos com alta aptidão é dada uma porção maior da roleta, enquanto aos de aptidão mais baixa é dada uma porção relativamente menor da roleta. Finalmente, a roleta é girada um determinado número de vezes, dependendo do tamanho da população, e são escolhidos, como indivíduos que participarão da próxima geração, aqueles sorteados na roleta.

| Indivíduo $S_i$ | Aptidão $f(S_i)$ | Aptidão Relativa |
|---|---|---|
| $S_1$ 10110 | 2.23 | 0.14 |
| $S_2$ 11000 | 7.27 | 0.47 |
| $S_3$ 11110 | 1.05 | 0.07 |
| $S_4$ 01001 | 3.35 | 0.21 |
| $S_5$ 00110 | 1.69 | 0.11 |

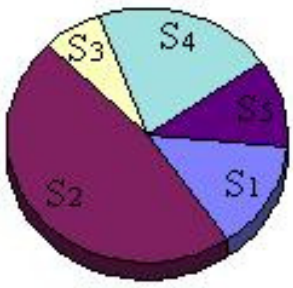



**Figura 20: Aptidões dos indivíduos dispostas na Roleta**

Operações Genéticas

Os membros mantidos pela seleção podem sofrer modificações em suas características fundamentais através de mutações e cruzamento (crossover) ou recombinação genética gerando descendentes para a próxima geração. Este

processo, chamado de reprodução, é repetido até que uma solução satisfatória seja encontrada.

O cruzamento é o operador responsável pela recombinação de características dos pais durante a reprodução, permitindo que as próximas gerações herdem essas características. Ele é considerado o operador genético predominante, por isso é aplicado com probabilidade dada pela taxa de crossover Pc, que deve ser maior que a taxa de mutação. Este operador pode, ainda, ser utilizado de várias maneiras; as mais utilizadas são:

- um-ponto: um ponto de cruzamento é escolhido e a partir deste ponto as informações genéticas dos pais serão trocadas. As informações anteriores a este ponto em um dos pais são ligadas às informações posteriores à este ponto no outro pai, como é mostrado no exemplo da Figura 21:
-

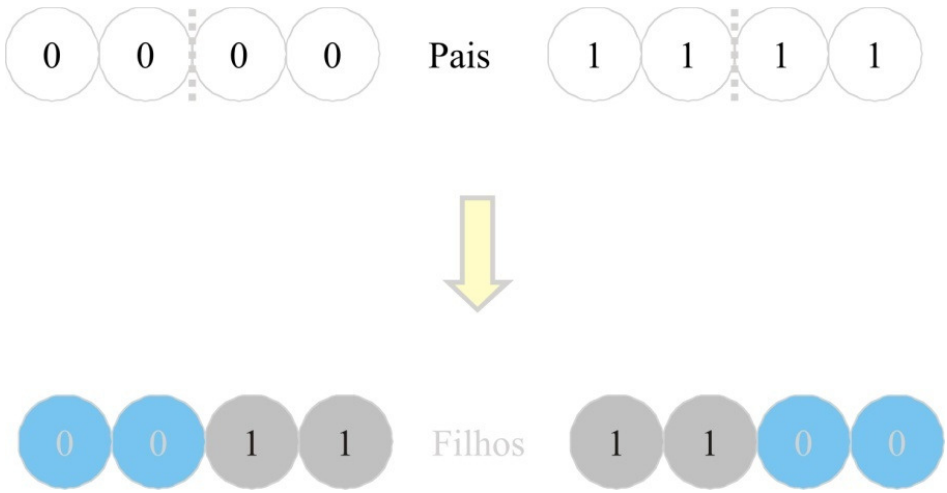



**Figura 21: Cruzamento em um ponto**

- multi-pontos: é uma generalização desta idéia de troca de material genético através de pontos, onde muitos pontos de cruzamento podem ser utilizados.
- uniforme: não utiliza pontos de cruzamento, mas determina, através de um parâmetro global, qual a probabilidade de cada variável ser trocada entre os pais.

O operador de mutação é necessário para a introdução e manutenção da diversidade genética da população, alterando arbitrariamente um ou mais componentes de uma estrutura escolhida, como é ilustrado na figura abaixo, fornecendo assim, meios para introdução de novos elementos na população. Desta forma, a mutação assegura que a probabilidade de se chegar a qualquer ponto do

espaço de busca nunca será zero, além de contornar o problema de mínimos locais, pois com este mecanismo, altera-se levemente a direção da busca.

O operador de mutação é aplicado aos indivíduos com uma probabilidade dada pela taxa de mutação Pm; geralmente se utiliza uma taxa de mutação pequena, pois é um operador genético secundário.

Embora possam parecer simplistas do ponto de vista biológico, estes algoritmos são suficientemente complexos para fornecer mecanismos de busca adaptativo poderosos e robustos.

Esse ciclo é repetido um determinado número de vezes. Abaixo é mostrado um exemplo de algoritmo genético. Durante esse processo, os melhores indivíduos, assim como alguns dados estatísticos, podem ser coletados e armazenados para avaliação.

```
Procedimento AG
        {t = 0;
         inicia_população (P, t)
         avaliação (P, t);
         repita até (t = d)
                 {t = t +1;
                  seleção_dos_pais (P,t);
                  cruzamento (P, t);
                  mutação (P, t);
                  avaliação (P, t);
                  sobrevivem (P, t)
                 }}
```

onde:

t - tempo atual;

d - tempo determinado para finalizar o algoritmo;

P – população

### 4.3. Parâmetros Genéticos

É importante também, analisar de que maneira alguns parâmetros influem no comportamento dos Algoritmos Genéticos, para que se possa estabelecê-los conforme as necessidades do problema e dos recursos disponíveis.

Tamanho da População. O tamanho da população afeta o desempenho global e a eficiência dos AGs. Com uma população pequena o desempenho pode cair, pois

deste modo a população fornece uma pequena cobertura do espaço de busca do problema. Uma grande população geralmente fornece uma cobertura representativa do domínio do problema, além de prevenir convergências prematuras para soluções locais ao invés de globais. No entanto, para se trabalhar com grandes populações, são necessários maiores recursos computacionais, ou que o algoritmo trabalhe por um período de tempo muito maior.

Taxa de Cruzamento. Quanto maior for esta taxa, mais rapidamente novas estruturas serão introduzidas na população. Mas se esta for muito alta, estruturas com boas aptidões poderão ser retiradas mais rapidaum valor alto, a maior parte da população será substituída, mas com valores muito altos pode ocorrer perda de estruturas de alta aptidão. Com um valor baixo, o algoritmo pode tornar-se muito lento.

Taxa de Mutação. Uma baixa taxa de mutação previne que uma dada posição fique estagnada em um valor, além de possibilitar que se chegue em qualquer ponto do espaço de busca. Com uma taxa muito alta a busca se torna essencialmente aleatória. Intervalo de Geração. Controla a porcentagem da população que será substituída durante a próxima geração. Com um valor alto, a maior parte da população será substituída, mas com valores muito altos pode ocorrer perda de estruturas de alta aptidão. Com um valor baixo, o algoritmo pode tornar-se muito lento.

### 4.4. Aplicações

Um sistema com bom desempenho em um ambiente dinâmico, geralmente exige soluções adaptativas. Sistemas adaptativos tentam resolver problemas acumulando conhecimento sobre o problema e utilizando estas informações para gerar soluções aceitáveis. Estes problemas, tipicamente, se encontram nas áreas de configuração de sistemas complexos, alocação de tarefas, seleção de rotas, e outros problemas de otimização e aprendizado de máquina.

Seguem-se alguns exemplos de sistemas adaptativos:

- Controle de Sistemas Dinâmicos;
- Indução e Otimização de Bases de Regras;
- Encontrar Novas Topologias Conexionistas:
    - Engenharia de Sistemas Neurais Artificiais;

    - Modelagem de Estruturas Neurais Biológicas;
- Simulação de Modelos Biológicos:
    - Comportamento;
    - Evolução;
- Evolução Interativa de Imagens;
- Composição Musical.

## 5. Exercícios

1. Defina o que vem a ser aprendizado de máquina (machine learning), caracterizando os dois tipos de aprendizado: supervisionado e não-supervisionado.
2. Explique o funcionamento básico dos algoritmos que trabalham com árvores de decisão (ex. ID3).
3. Construa uma árvore de decisão, utilizando o "Decision-Tree-Learning" para a seguinte tabela de dados:

| ATRIBUTOS | | | **Condução** |
|---|---|---|---|
| **Zona** | **Trafego** | **Cansaço** | |
| Cidade | Intenso | Elevado | Calma |
| Estrada | Baixo | Baixo | Desportiva |
| Estrada | Baixo | Elevado | Desportiva |
| Estrada | Intenso | Baixo | Desportiva |
| Estrada | Intenso | Elevado | Calma |
| Cidade | Baixo | Elevado | Calma |
| Cidade | Baixo | Baixo | Desportiva |

4. Explique o conceito e a importância da dissimilaridade inter-grupo (inter-class dissimilarity) e da similaridade intra-grupo (intra-class similarity) no algoritmo COBWEB.
5. Explique a aprendizagem por reforço com base nos conceitos de **ação, estado, recompensa, política, exploração** e **explotação.**
6. Qual das alternativas abaixo melhor preenche os espaços na seguinte frase:

Uma rede Perceptron simples é uma rede de ..... com n entradas e ..... saídas, que aprende classificações linearmente separáveis, como, por exemplo, os operadores and e or, mas não o operador ..... Além disso, o Perceptron tem como função de transferência a função ..... Quanto a tipo de aprendizagem, o Perceptron ..... os exemplos.

a. ( ) 1 camada, m, xor, limiar, generaliza.
b. ( ) 1 camada, m, not, step0, generaliza.
c. ( ) x camadas, m, xor, limiar, generaliza.
d. ( ) x camadas, m(m < n), xor, sign, especializa.
e. ( ) x camadas, m(m > n), not, sign, especializa.
f. ( ) 1 camada, m(m > n), xor, limiar, especializa.

7. Defina o que é uma Rede Neural e quais os elementos fundamentais de uma Rede Neural? Ilustre e exemplifique.
8. O Perceptron é um classificador linear? Justifique.
9. Explique o funcionamento dos algoritmos genéticos com base nos conceitos: população, cromossomos, descendentes, adaptação, crossover, mutação, seleção natural, taxa de mutação.
10. Nos algoritmos genéticos como funciona o método da roleta para a seleção dos mais aptos?

## Resumo

Uma das formas de fazer o ser humano melhorar continuamente a realização de suas tarefas é o processo de aprendizado. Quanto mais se aprende melhor capacidade de se realizar uma determinada ação. A aprendizagem de máquina, estudada nesta unidade, proporciona diversos mecanismos para que sistema computacional possa aprender e assim, melhorar seu desempenho a cada execução.

Aqui estudaremos algumas das diferentes abordagens da chamada aprendizagem de máquina que possibilitarão tal aprendizado.

Em um primeiro momento veremos as diferenças entre aprendizagem supervisionada e não-supervisionada. Em seguida é mostrado os conceitos que norteiam os estudos da aprendizagem por reforço. Falaremos ainda dos fundamentos teórico das redes neurais umas das técnicas mais pequisadas

atualmente e suas principais aplicações. Encerraremos esta unidade mostrando como funcionam os algoritmos genéticos.

Os exercícios complementam o que foi estudado através de situações práticas e reforço na teoria apresentada.

## Weblografia


Universidade Aberta do Piauí – UAPI
http://www.ufpi.br/uapi
Universidade Aberta do Brasil- UAB
http://www.uab.gov.br
Secretaria de Educação a Distância do MEC – SEED
http://www.seed.mec.gov.br
Associação Brasileira de Educação a Distância – ABED
http://www.abed.org.br
Padrões estocáticos
http://pt.wikipedia.org/wiki/Estoc%C3%A1stico
Sistemas Inteligentes
http://www.espie.cinted.ufrgs.br/~dsbit/trimestre1/rosa/sistemas_inteligentes.doc
Aprendizagem de Máquina
http://tecsofia.wordpress.com/2007/11/13/aprendizado-de-maquina/
[Bradzil, 1999]. Construção de modelos de decisão a partir de dados.
http://www.nacc.up.pt/~pbrazdil/ Ensino/ML/ModDecis.html
Utilização de Aprendizagem por Reforço para Modelagem Autônoma do Aprendiz em um Tutor Inteligente
http://nlx.di.fc.ul.pt/~guelpeli/Arquivos/Artigo17.pdf
Uma Introdução a Redes Neurais
http://www.din.uem.br/ia/neurais/
Redes Neurais
http://www.gildario.com.br/index.php?option=com_content&view=article&id=63&Itemid=65
Redes Neurais – Treinamento
http://www.solv.com.br/prof/redeneural/pag9.htm
Processos de Aprendizado e Treinamento da Rede Perceptron
http://elson_mendes.sites.uol.com.br/rn/rn_aprend.html

Redes Neurais Artificiais

http://www.ppgia.pucpr.br/~euclidesfjr/Metodos_Inteligentes/0206/RNA-XIIERI.pdf

Classificação de Dados de Variabilidade de Freqüência Cardíaca Utilizando Técnicas de Redes Neurais

http://www.sbeb.org.br/cbeb2008/Intelig%EAncia%20Artificial%20e%20Redes%20Neurais/p_1664.pdf

Algoritmos Genéticos: Fundamentos e Aplicações

http://www.gta.ufrj.br/~marcio/genetic.html

Algoritmos Genéticos

http://www2.dem.inpe.br/ijar/Algrgene1.doc

**Referências Biblográficas**


RUSSEL, Stuart; NORVIG, Peter: Inteligência Artificial. Campus, São Paulo, 2004. 1040p

BITTENCOURT, Guilherme: Inteligência Artificial – Ferramentas e Teorias. Editora da UFSC. 2ª. Edição. Florianópolis, 2001. 362p

RICH, Elaine; KNIGHT, Kevin: Inteligência Artificial. Makron Books. 2ª. Edição. São Paulo, 1994. 722p.

T. M. Mitchell. Machine Learning. McGraw–Hill Science/Engineering/Math, 432 páginas, ISBN 0070428077, 1997.

BATISTA, G.E.A.P.A, Pré-processamento de dados em aprendizado de máquina Supervisionado. 2003. Tese (Doutorado) – Instituto de Ciências Matemáticas e de Computação. Universidade de São Paulo, São Carlos.

Quinlan, J.R. (1986). Induction of Decision Trees.Machine Learning 1:1, 81–106

Breiman,L., Friedman,J.H., Olshen,R.A., and Stone,C.J. (1984), Classification and Regression Trees, Pacific Grove, CA: Wadsworth.

Michalski, R. S. (1980). "Knowledge acquisition through conceptual clustering: A theoretical framework and an algorithm for partitioning data into conjunctive concepts". International Journal of Policy Analysis and Information Systems 4: 219–244.

Fisher, D. H. Knowledge acquisition via incremental conceptual clustering. Mach. Learn., 2(2):139–172, 1987. ISSN 0885-6125.

Rebouças, Ricardo ; Furtado, V. . Formation of Probabilistic Concepts through observations containing discrete and continuos attributes. In: Florida Artificial Intelligence Research Symposium - FLAIRS, 2004, Miami. FLAIRS 2004

Proceedings. Palo Alto, CA : American Association for Artificial Intelligence, 2004. v. 1.

R. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. 2nd Edition. MIT Press, Cambridge, 2002.

McCulloch, W. S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. Bull. Math. Biophysics, 5:115-133.

Minsky, M.L., Papert, S.A., Perceptrons, Cambridge, MA : MIT Press, 1969.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986) Learning representations by back-propagating errors. Nature, 323, 533--536.

# UNIDADE 4

## REPRESENTAÇÃO DO CONHECIMENTO

# 1. Aquisição do Conhecimento

Para utilizar um corpo de conhecimento em uma máquina, é necessário escolher uma maneira de representá-lo. Todo programa de computador contém o conhecimento sobre um determinado problema a ser resolvido. O conhecimento está nos algoritmos que o programa emprega e nos procedimentos de decisão que determinam qual destes algoritmos empregar em determinada circunstância. Quando carrega-se um programa em um computador, pode-se dizer que o computador " adquiriu o respectivo conhecimento; entretanto, na maioria dos programas, estas informações não são representadas explicitamente e não podem ser facilmente atualizadas ou manipuladas.

A aquisição de conhecimento não pode limitar-se à adição de novos elementos de conhecimento à base de conhecimentos; é necessário integrar o novo conhecimento ao conhecimento já disponível, através da definição de relações entre os elementos que constituem o novo conhecimento e os elementos já armazenados na base. Dois tipos de mecanismos para a definição de tais relações foram propostos: ligar os elementos de conhecimento diretamente através de ponteiros, ou reunir diversos elementos relacionados em grupos (em inglês "clustering").

Outro ponto importante na aquisição de conhecimento é o tratamento de incoerências. Dependendo da forma como o novo conhecimento é adquirido, pode haver erros de aquisição. Estes erros podem resultar da própria natureza do conhecimento, como em dados obtidos através de sensores sujeitos a ruído, ou podem ser gerados pela interface humana existente entre o mundo real e o sistema de representação. Técnicas foram desenvolvidas para evitar erros de aquisição, como, por exemplo, a especificação de regras de aquisição em que o tipo de conhecimento esperado é definido. Estas técnicas são comuns aos sistemas de representação de conhecimento e aos sistemas de gerenciamento de bancos de dados. Por outro lado, uma base de conhecimento pode ser examinada periodicamente com a finalidade de detectar incoerências eventualmente introduzidas no processo de aquisição. Este método é limitado pelo fato de que linguagens de representação razoavelmente expressivas não contam com procedimentos completos de verificação conhecidos. Finalmente, deve-se observar que a adequação do formalismo de representação ao tipo de conhecimento do

mundo real a ser representado é fundamental para a eficiência do processo de aquisição.

## 1.1. Métodos de Aquisição do Conhecimento

### 1.1.1. Entrevistas não Estruturadas

As entrevistas não estruturadas (unstructured interwiews) são projetadas para permitir ao especialista que discuta o problema a ser resolvido de modo natural (Durkin, 1994). Assim é possível obter a compreensão dos conceitos mais importantes a respeito do domínio e conhecer a estratégias que o especialista utiliza para resolver o problema. As principais vantagens desse método são: fornecer uma compreensão geral do problema, auxiliar na identificação de conceitos e objetivos; fornecer condições para compreender os métodos para a resolução de problemas. A desvantagem das entrevistas não estruturadas é uma grande quantidade de informações fragmentadas ou superficiais, pobres em detalhes.

### 1.1.2. Entrevistas Estruturadas

A entrevista estruturada mantém o foco do problema a ser resolvido de uma forma dirigida. Este tipo de entrevista adquire detalhes específicos a respeito de determinado aspecto do problema antes de passar para outros pontos (Durkin, 1994). A entrevista estruturada pode ser utilizada quando é necessária uma informação específica a respeito de um tópico estabelecido. As vantagens deste método são: mantém o foco em um determinado assunto, fornece informações detalhadas e relações estruturadas entre os conceitos. As desvantagens encontradas são: alguns conceitos não relatados na entrevista podem não ser abordados; fornece uma compreensão fraca do conhecimento procedural, como regras ou estratégias para solucionar problemas.

As técnicas de entrevistas discutidas nesta seção são de natureza introspectiva. O especialista tenta responder às questões recorrendo a seus conceitos e compreensão sobre o problema. Estudos em psicologia demonstraram que a introspeção não é uma maneira eficaz de obter um conhecimento completo e confiável para a resolução de problemas.

### 1.1.3. Estudo de Caso

Devido as limitações dos métodos de entrevista, os engenheiros de conhecimento tiveram que recorrer a outras técnicas que tornassem possível a aquisição de conhecimento. Uma alternativa às entrevistas são os estudos de caso. Um caso é um problema que foi solucionado no passado e que contém a solução e a descrição dos passos para obtê-la. Existem duas abordagens para sua utilização durante uma seção de aquisição de conhecimento: retrospectiva e observacional. Na abordagem retrospectiva o especialista é levado a rever um caso e a explicar como o problema foi nele resolvido. A técnica observacional pede aos especialistas para discutirem a solução de um caso enquanto o engenheiro observa o seu processo de resolução.

### 1.1.4. Descoberta de Conhecimento em Base de Dados

O conhecimento ainda pode estar na própria base de dados. De acordo com [Fayyad, 1996], o conceito de descoberta de conhecimento em bases de dados pode ser resumido como o processo não-trivial de identificar padrões novos, válidos, potencialmente úteis e, principalmente, compreensíveis em meio às observações presentes em uma base de dados.

O objetivo último da descoberta do conhecimento em bases de dados não é o de simplesmente encontrar padrões e relações em meio à imensa quantidade de informação disponível em bases de dados, e sim a extração de conhecimento inteligível e imediatamente utilizável para o apoio às decisões.

O processo de descoberta do conhecimento [Berry, 2000] [Pyle, 1999] é composto por várias etapas. A

Figura 22 ilustra o ciclo de descoberta do conhecimento em bases de dados e suas etapas. A origem diversa dos dados que serão utilizados, coletados em diferentes instantes de tempo em lugares distintos, cria um esforço inicial de consolidação e agrupamento de toda a informação que irá servir de base para o processo. A compreensão do negócio e do ambiente no qual os dados estão inseridos é crítica para o entendimento dos mesmos.

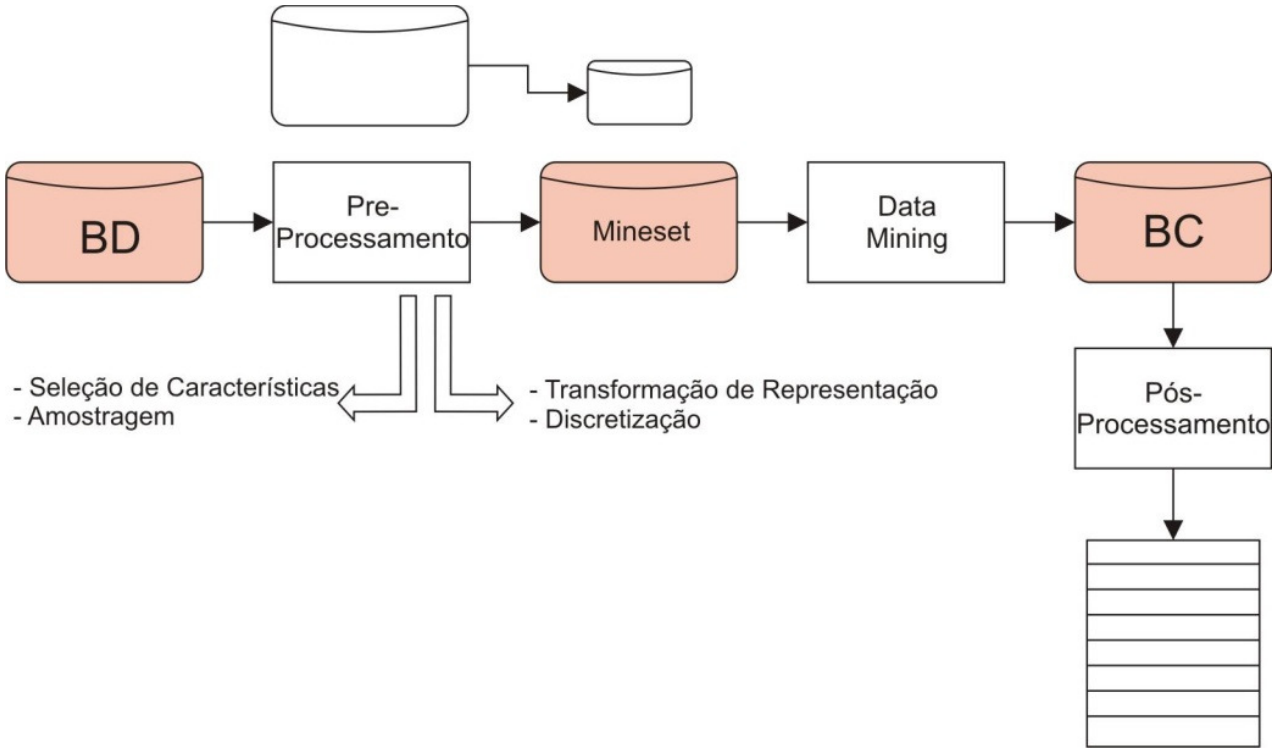



**Figura 22: Ciclo de descoberta do conhecimento em bases de dados e suas etapas**

Dada essa diversidade e heterogeneidade dos dados, esforços de pré-processamento e limpeza dos mesmos são cruciais na geração de dados que possam vir a ser trabalhados em busca de conhecimento útil. É essencial que seja realizada a investigação de inconsistências e problemas devido a diferenças de escalas, assim como o tratamento de valores fora da normalidade (outliers) e observações errôneas.

Realizadas essas tarefas iniciais, que tornam os dados tratáveis e homogêneos, a mineração dos dados pode ser iniciada, na busca por padrões e relações que façam sentido e sejam úteis para o problema a ser resolvido ou objetivo a ser alcançado, podendo utilizar alguns (ou uma combinação deles) dos algoritmos vistos na Unidade III. Finalmente, a interpretação, compreensão e aplicação dos resultados encontrados é o passo que torna o conhecimento adquirido através de bases de dados um real insumo para o apoio às decisões [Klosgen, 2002].

## 2. Métodos de Representação de Conhecimento

O conhecimento é o destaque dos sistemas de IA. A representação do conhecimento é o componente fundamental em sistemas inteligentes. Pelos mecanismos de representação, os formalismos de IA, o conhecimento é codificado através de objetos, atributos, objetivos, ações e é processado através de estruturas e procedimentos [Bench-Capon, 1990]. Para representar o desempenho de especialistas humanos, o sistema deve possuir não só um conjunto de informações mas, também, a habilidade de utilizá-las na resolução de problemas de forma criativa, correta e eficaz. Esta habilidade representa uma série de palpites e regras intuitivas que o especialista utiliza para resolver os problemas; sua aplicação possibilita, de uma maneira mais econômica, a chegada a soluções aceitáveis, embora nem sempre ótimas. Um sistema inteligente precisa conhecer o contexto do fato em estudo e reconhecer os processos que causariam mudanças nos fatos. Para resolver problemas, em alguns casos, é recomendado conhecer tudo sobre o problema e quais as possíveis soluções que se pretende encontrar, juntamente com algumas estratégias para solucionar cada problema. O estudo da representação do conhecimento consiste nos caminhos que podem ser trilhados para codificá-lo em um programa computacional.

Existem muitas formas de representação do conhecimento através de formalismos computacionais, scripts, frames, redes semânticas, regras, grafos conceituais, representações formulário e os conceitos, objetos e fatos. Esses formalismos podem ser empregados para representar casos em sistemas de RBC e representar domínios em SEs (Weber, 1998). Nas próximas seções estes formalismos e conceitos são explicados brevemente.

### 2.1. Lógica

A lógica é a base para a maioria dos formalismos de representação de conhecimento, seja de forma explícita, como nos sistemas baseados na linguagem Prolog, seja disfarçada na forma de representações específicas que podem facilmente ser interpretadas como proposições ou predicados lógicos.

É um modo de declaração que representa o conhecimento uma das mais primitivas formas de representação do raciocínio ou conhecimento humano. A lógica

preposicional é considerada a forma mais comum da lógica. Baseia-se em que proposição só pode ter um dos seguintes valores: verdadeira ou falsa. A lógica de predicados é considerada como uma extensão da lógica preposicional. Na lógica de predicados os elementos fundamentais são, além do objeto, também os seus predicados.

Uma das principais formas de representar conhecimento em lógica e também manipulá-lo é através da linguagem PROLOG. Prolog é uma linguagem de programação lógica e declarativa. Isso significa que em vez de o programa estipular a maneira de chegar à solução, passo a passo, (como nas linguagens procedimentais ou imperativas), limita-se a fornecer uma descrição do problema que se pretende computar. Usa uma coleção base de dados de *fatos* e de relações lógicas (*regras*) que exprimem o domínio relacional do problema a resolver.

## 2.2. Redes

As redes semânticas são grafos direcionados ligados por nós para representar objetos e conexões e a relação entre objetos [Dean et. al., 1995]. A rede semântica é usada para representar elementos de uma representação tal como uma classe, suas instâncias e suas características. Seus arcos são direcionados e representam as relações entre os atributos. Quando um atributo não deve ser herdado, as redes semânticas necessitam de tratamento de exceção.

Portanto, uma rede semântica é uma estrutura para a representação do conhecimento definida como um padrão de nodos interligados por arcos rotulados. As redes deste tipo não só captam as definições dos conceitos mas também, inerentemente, proporcionam ligações com outros conceitos. Uma variedade de redes semânticas tem sido desenvolvidas como variações deste simples padrão. Algumas são propostas como modelos da memória humana e significado de representação, enquanto outras são usadas como componentes de compreensão de linguagem e sistemas de raciocínio.

Embora seja útil imaginar as redes semânticas utilizando esta notação gráfica característica, é lógico que elas não podem ser representadas desta maneira em um programa de computador. Assim por exemplo, na linguagem LISP, cada nodo seria um átomo, as ligações seriam as propriedades, e os nodos da outra extremidade seriam os valores.

## 2.3. Frames

Um frame é uma estrutura de dados que representa uma entidade através de suas características e potenciais habilidades. Suas características estão representadas por pares atributo-valor e as potencialidades são representadas por métodos. Um frame abstrato (ou frame de classe) não tem instâncias, por esta razão seus atributos não são valorados, suas subclasses são ligadas a instâncias da entidade representada por essa classe.

Frame é uma representação de um objeto complexo. Ele é identificado por um nome e consiste em conjunto de slots. Cada frame possui ao menos um frame hierarquicamente superior e, portanto, constitui uma base com mecanismo de herança. Um frame especial é a raiz desta hierarquia de herança.

Sistemas baseados em cadeias semânticas e sistemas baseados em frames podem ser considerados semelhantes com respeito às suas estruturas, mas diferem no que representam. Quer dizer, enquanto cadeias semânticas representam objetos simples, um sistema de frames pode representar objetos complexos.

## 2.4. Árvores de Decisão

Trata-se de um modelo prático de uma função recursiva que determina o valor de uma variável e, baseando-se neste valor, executa-se uma ação. Esta ação pode ser a escolha de outra variável ou a saída. As árvores de decisão são treinadas de acordo com um conjunto de “exemplos previamente classificados” e, posteriormente, outros exemplos são classificados de acordo com essa mesma árvore.

A construção de uma árvore de decisão parte da descrição de um problema do qual deve ser especificado as variáveis, ações e a seqüência lógica para a tomada de decisão. Depois de construída teremos uma visão gráfica da tomada de decisão (

Figura 23).

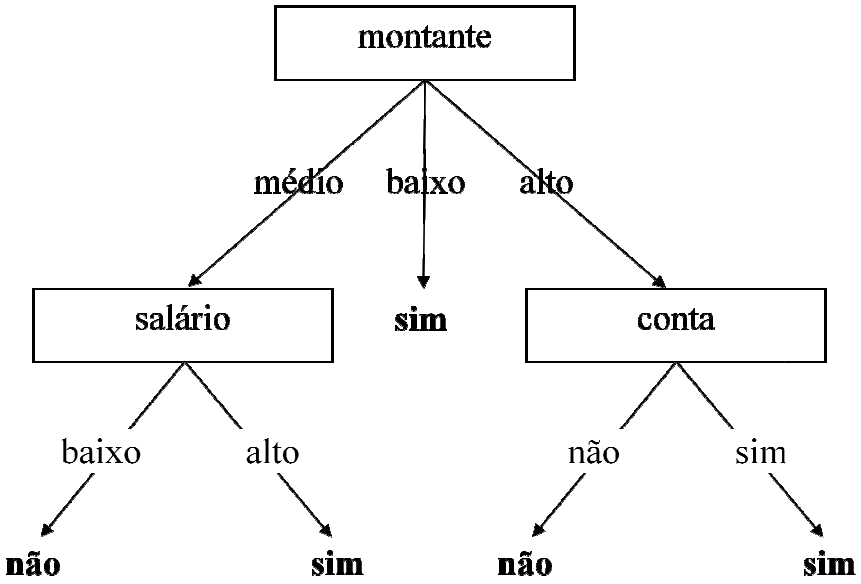



**Figura 23: Visão gráfica de uma árvore de decisão**

As variáveis são representadas pelas condições mostradas na estrutura acima, as saídas são as ações a serem tomadas. A seqüência lógica se trata da ordem que serão dispostas as condições devido ao seu grau de relevância pré-definido no início da criação do modelo.

Os ramos da árvore correspondem a cada possibilidade lógica que levam à próxima possibilidade ou à ação a ser tomada. Nem sempre a combinação das condições descritas leva a uma ação definitiva, quando isto ocorre o decisor tem o papel de optar pela ação a ser tomada.

## 2.5. Ontologias

Ontologia é um modelo de dados que representa um conjunto de conceitos dentro de um domínio e os relacionamentos entre estes. Uma ontologia é utilizada para realizar inferência sobre os objetos do domínio.

Atualmente, as ontologias compartilham muitas semelhanças estruturais, independente da linguagem em que são expressas. Como mencionado acima, a maioria das ontologias descrevem indivíduos (exemplares), classes (conceitos), atributos e relacionamentos. Esta seção descreve cada um desses componentes.

**Indivíduos** (exemplares) são os componentes básicos de uma ontologia. Os indivíduos em uma ontologia podem incluir objetos concretos como pessoas, animais, mesas, automóveis, moléculas, planetas, assim como indivíduos abstratos como números e palavras. Para ser exato, uma ontologia não precisa necessariamente incluir indivíduos, porém um dos propósitos gerais de uma ontologia é apresentar um meio de classificação de indivíduos, mesmo que estes não sejam explicitamente parte da ontologia.

**Classes** (conceitos) são grupos abstratos, conjuntos ou coleções de objetos. Eles podem conter indivíduos, outras classes, ou uma combinação de ambos. Alguns exemplos de classes[2]:

Ontologias se diferenciam nos seguintes aspectos: se classes podem conter outras classes, se uma classe pode pertencer a si mesma, se existe uma classe universal (isto é, uma classe contendo tudo), etc. Algumas vezes estas restrições são feitas para evitar alguns paradoxos conhecidos.

Uma classe pode incluir ou estar incluída em outras classes. Por exemplo, Veículo inclui Carro, já que (necessariamente) qualquer coisa que é membro de Carro é também membro de Veículo. A relação de inclusão é utilizada para criar uma hierarquia de classes, geralmente com uma classe geral como Coisa no topo, e classes específicas como Ford Fiesta 2002 na base.

Objetos em uma ontologia podem ser descritos através de **atributos**. Cada atributo tem pelo menos um nome e um valor, e é utilizado para armazenar informação que é específica para o objeto ligado a ele.

O valor de um atributo pode ser um tipo de dados completo; neste exemplo, o valor do atributo chamado Motor é uma lista de valores, não um valor simples.

Um uso importante dos atributos é a descrição de **relacionamentos** (também conhecidos como relações) entre objetos na ontologia. Geralmente, uma relação é um atributo cujo valor é outro objeto na ontologia. Muito do poder das ontologias vem da habilidade de descrever estas relações. O conjunto de todas as relações descreve a semântica do domínio.

O tipo mais importante de relação é a relação de inclusão (é-superclasse-de, é-um, é-subtipo-de ou é-subclasse-de), que define quais objetos são membros de quais classes de objetos. A adição de relacionamentos é-um cria uma taxonomia hierárquica, uma estrutura de árvore que descreve que objetos se relacionam com

quais outros. Neste estrutura, cada objeto é um "filho" de uma "classe pai". Outro tipo comum de relação é a do tipo parte-de que representa como objetos se combinam para formar objetos compostos.

Além das relações comuns como é-um e parte-de, as ontologias geralmente incluem outros tipos de relações que refinam ainda mais a semântica do modelo. Estas relações geralmente são específicas do domínio e são utilizadas para responder tipos particulares de questões.

## Exercícios

1. Explique a importância de se representar conhecimento em computador.
2. Qual o papel o engenheiro do conhecimento?
3. Quais são os métodos de Aquisição do Conhecimento? Explique cada um deles.
4. O que vem a ser a descoberta de conhecimento em base de dados? Explique as etapas deste processo.
5. Explique por que devemos utilizar diferentes formas de representar o conhecimento.
6. O que são Frames?
7. O que é a forma de representação do conhecimento na forma de clusters?
8. Explique a importância das árvores de decisão para representar conhecimento.
9. O que são ontologias?

## Resumo

O conhecimento ao ser inserido em um sistema computacional passa por uma série de processos tais como aquisição (descoberta de conhecimento), mineração e finalmente a representação do conhecimento propriamente dita.

Estes processos tem como finalidade formalizar o conhecimento, ou seja, disponibilizá-lo de uma forma que possa ser facilmente usado e interpretado pelos próprios sistemas computacionais e pelos usuários desses sistemas.

Nesta unidade apresentaremos os principais modelos para representar o conhecimento, dando destaque ao processo de descoberta de conhecimento em base de dados e as principapais formas de se representar o conhecimento como frames, árvores de decisão, frames e ontologias.

## Weblografia

Universidade Aberta do Piauí – UAPI
http://www.ufpi.br/uapi
Universidade Aberta do Brasil- UAB
http://www.uab.gov.br
Secretaria de Educação a Distância do MEC – SEED
http://www.seed.mec.gov.br
Associação Brasileira de Educação a Distância – ABED
http://www.abed.org.br
Sistemas Tutorais Inteligentes
http://www.educacaoliteratura.com.br/index%20101.htm
Inteligência Artificial Aplicada a Nutrição na Prescrição de Planos Alimentares
http://www.eps.ufsc.br/disserta99/camargo/
Introdução à abordagem Capitalização de conhecimento
http://www.batebyte.pr.gov.br/modules/conteudo/conteudo.php?conteudo=1807
Linguagem PROLOG
http://fit.faccat.br/~alexandre/prolog.php
Introdução a Linguagem PROLOG
http://puig.pro.br/Logica/palazzo.pdf
Proposta de uma Ontologia para Avaliação de Desempenho Empresarial
http://www.san.uri.br/~portalcomp/TCCs/Oliveira.doc

## Referências Biblográficas

Russel, Stuart; Norvig, Peter: Inteligência Artificial. Campus, São Paulo, 2004. 1040p

Bittencourt, Guilherme: Inteligência Artificial – Ferramentas e Teorias. Editora da UFSC. 2ª. Edição. Florianópolis, 2001. 362p

Rich, Elaine; Knight, Kevin: Inteligência Artificial. Makron Books. 2ª. Edição. São Paulo, 1994. 722p.

Durkin, J.. Expert systems design and development. Prentice Hall, USA, 1994.
Fayyad, U.; Shapiro, G. P.; Smith, P. The KDD process for extracting useful knowledge from volumes of Data. Communications of the ACM, New York, v. 39, n. 11, p. 27-35 , Nov. 1996

M. Berry; G. Linoff. Mastering Data Mining: The Art and Science of Customer Relationship Management. John Wiley & Sons, 2000.

D. Pyle. Data Preparation for Data Mining. Morgan Kaufmann, 1999.
W. Klosgen; J. M. Zytkow; J. Zyt. Handbook of Data Mining and Knowledge Discovery. Oxford University Press, 2002.

Bench - Capon, T, J. M.. Introduction to kowledge representation, in: Knowledge representation na approach to artificial intelligence. Academic press, The A. P. I. C. Series, nº32, 1990.

Dean, T., Allen, J., Aloimonos, Y.. Artificial Intelligence Theory and Practice. Addison-Wesley Publishing Company, Menlo-Park, (1995)

# UNIDADE 5

## AGENTES INTELIGENTES

## 1. Definições

Quanto comparados com os modelos de Vida Artificial, os modelos e teorias de Agentes Inteligentes, discutidos a seguir, apresentam um maior nível de abstração e simbolismo, em maior consonância com a IA Simbólica. Enquanto que nos modelos de Vida Artificial, a complexidade emergente, a interatividade e evolutibilidade são características fundamentais, nos modelos de agentes inteligentes parte-se do princípio de que a inteligência, de alguma forma, já deve estar presente nos elementos autônomos.

O termo agente é aplicado de forma coerente (embora com significados distintos) em praticamente todas as área do conhecimento na qual existem as noções de indivíduo e comunidade, como em:

* Sociologia [Giddens, 1979];
* Economia [Holmstrom & Tirole, 1989];
* Comportamento Animal [Dawkins, 1979];
* Robótica [Meyer et al, 1993];
* Software [Finin et al, 1992], etc.

A característica que unifica os conceitos de agente presentes em todas as áreas acima é a existência de uma espaço no qual elementos autônomos representam, manipulam e trocam informações e conhecimento, demonstrando uma capacidade cognitiva relativamente elevada, quando comparados, por exemplo, a moléculas, vegetais e vermes.

Na área de computação uma das definições mais aceitas é a de [Russel & Norvig, 2004]: Qualquer coisa que percebe seu ambiente através de sensores agindo neste ambiente por meio de seus atuadores, visando atingir seu objetivo (Figura 24), ou seja, um agente é um sistema computacional que está situado em um ambiente e que é capaz de ações autônomas neste ambiente em ordem de cumprir seus objetivos.

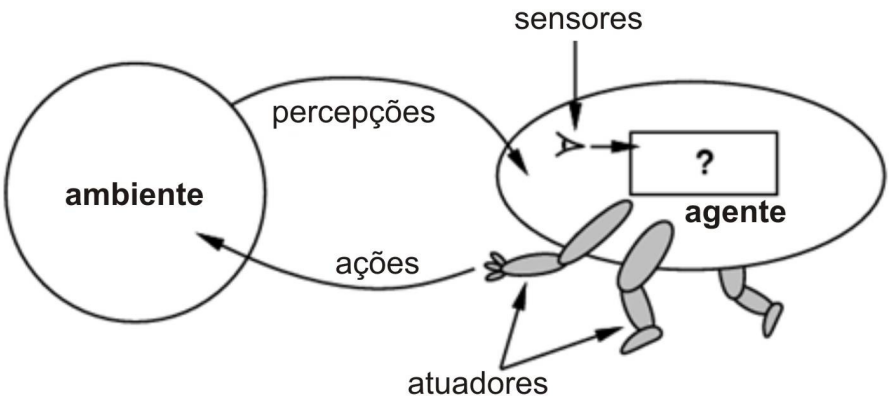



**Figura 24: Agente segundo Russel & Norvig, 2003**

Definir a inteligência de um agente é dizer que este faz “a coisa certa”. Esta “Coisa Certa” é aquela que faz com que o agente obtenha mais sucesso. Isso leva ao problema de medir como e quando o gente obteve sucesso. Isso leva ao seguinte dilema:

- Os agentes têm que realizar ações corretas
- São corretos os que são bem-sucedidos
- Como definir quando e como eles obtêm sucesso?

É neste ponto que surgem alternativas para definir o sucesso de um agente baseado em 5 fatores:

- Percepção

Entradas perceptivas do agente em qualquer momento

- Sua seqüência de percepção

História completa de tudo o que o agente já percebeu

- As ações que são capazes de executar
- Sua medida de desempenho, que define o grau de sucesso
- Seu conhecimento sobre o ambiente

Logo podemos definir o agente racional ideal como sendo a entidade que para cada seqüência de percepção possível, deve saber se sua ação maximizará sua medida de desempenho, baseado na evidência de sua seqüência de percepção e no conhecimento que ele traz consigo.

Para ilustrar esses fatores tomemos como exemplo um Agente de limpeza (Figura 25).

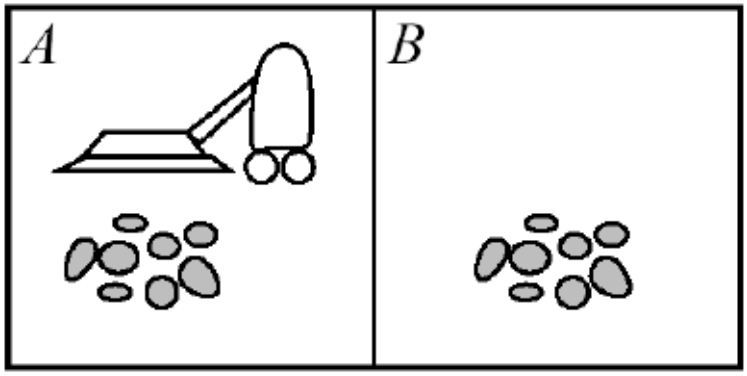



**Figura 25: Ambiente do Agente de Limpeza [Russel & Norvig, 2003]**

- Percepção: Quantidade de sujeira, local e conteúdo, por exemplo, [A, sujo].
- Ações: Direita, Esquerda, Sugar, NoOp
- Sequencia de Percepções:

| Seqüência de Percepções | Ação |
|---|---|
| [A, limpo] | Direita |
| [A, sujo] | Aspirar |
| [B, limpo] | Esquerda |
| [B, sujo] | Aspirar |
| [A, limpo] [A, limpo] | Direita |
| [A, limpo] [A, sujo] | Aspirar |
| ... | ... |
| [A, limpo] [A, limpo] [A, limpo] | Direita |
| [A, limpo] [A, limpo] [A, sujo] | Aspirar |

- Medidas de desempenho:
  - ✓ quantidade de lixo retiradas X horas de trabalho
  - ✓ quantidade de lixo retiradas X horas de trabalho descontadas a

Com base nos conceitos vistos, para um agente racional para o mundo do aspirador de pó (agente de limpeza) temos:

- Medida de Desempenho: 1 ponto de bonificação para cada quadrado limpo em cada unidade de tempo, ao longo de 1.000 períodos
- Conhece a "geografia" do ambiente, mas não sabe onde tem sujeira e nem a posição inicial.
- Quadrados limpos permanecem limpos e aspirar limpa o quadrado atual

- As ações Esquerda e Direita movem o agente nesta direções; exceto quando isto leva o agente para fora do ambiente (fica parado)
- Ações disponíveis: Esquerda, Direita, Aspirar, NoOP (não faz nada)
- O agente percebe: a sua posição e se nessa posição existe sujeira

Outras questões também deve ser levadas em consideração quando se fala no conceito de agentes como por exemplo a racionalidade e autonomia. Um agente racional, na visão da Inteligência Artificial, é uma entidade capaz de agir de forma a alcançar resultados otimizados. Dessa forma, um Agente só possui sentido baseado nos seus objetivos e na sua capacidade de atingi-los. Na maioria das vezes, devido às incertezas do ambiente de tarefas, o resultado procurado pode ser apenas o melhor resultado esperado, um máximo local do problema.

Um agente racional deve ser autônomo. Isso quer dizer que, dentre outras coisas, ele deve aprender baseado na sua percepção e nos efeitos para cada ação que realiza. Quando um agente se baseia somente no conhecimento anterior de seu projetista ele não tem autonomia. Contudo, se o agente consegue aprender por experiência e alterar seu comportamento ele pode ter autonomia funcional.

Agente autônomo deve aprender o que puder para compensar o conhecimento prévio parcial ou incorreto. Na prática, raramente os agentes têm autonomia completa desde o início.

Agente autônomos são mais flexíveis, podem se adaptar a novas condições de ambiente como por exemplo um agente de reconhecimento de fala que possui um conjunto pré-definido de padrões, mas pode aprender o sotaque de um novo usuário. Um agente inteligente verdadeiramente autônomo deveria ser capaz de operar com sucesso em um grande variedade de ambientes, dado um tempo suficiente para se adaptar.

Um agente é definido em um programa de agente, que é caracterizado por uma função de agente. A função de agente é responsável por determinar uma ação, baseada na percepção atual, ou em uma seqüência de percepções. A função de agente é dependente das características do ambiente de tarefas. Ela é responsável por maximizar a medida de desempenho do agente. É importante ressaltar que o conceito de agente racional é apenas uma ferramenta de modelagem para facilitar o entendimento de problemas, e não um conceito que determina o que é e o que não é racional.

## 2. Ambientes de Agentes

Um dos passos principais do projeto de um agente é especificar o ambiente de tarefa de forma tão completa quanto possível. Existem várias características que devem ser levadas em consideração para definir um ambiente:

- Acessibilidade: Um ambiente é acessível quando o agente tem acesso, através de seus sensores, à todas informações relativas ao ambiente e que influem em seu processo de decisão.
- Determinístico ou Estocástico: Se o próximo estado do ambiente pode ser determinado pelo próximo estado do agente, então o ambiente é determinístico.
- Episódico ou seqüencial: Um ambiente é dito episódico quando a experiência do agente é dividida em episódios que não dependam de ações passadas. Nesta caso, cada episódio = percepção + ação.
- Dinâmico ou estático: Se o estado do ambiente pode mudar enquanto o agente está processando informações então o ambiente é dinâmico, caso contrário, será estático.
- Discreto ou contínuo: Se o número de ações e percepções é limitado então o ambiente é discreto, caso contrário, será contínuo.
- Agente único ou agentes simples: Refere a quantidade de agentes no ambiente. Por exemplo, em um jogos de palavras cruzadas o ambiente agente único. Já no Xadrez temos agentes múltiplos em ambiente competitivo. Na tarefa de dirigir táxi o ambiente possui agentes múltiplos cooperativos. Ambiente com agentes múltiplos precisam de uma forma de comunicação.

## 3. Tipos de Agentes

Pode de acordo com Russel & Norvig (2003) existem diversas classificações de agentes que mostraremos a seguir.

### 3.1. Agente Reativo Simples

Os agentes reativos (Figura 26) são simples e limitados funcionando somente se a decisão correta puder ser tomada com base apenas na percepção atual. O seu ambiente é completamente observável. Em sua execução gera uma descrição abstrata do estado a partir do que foi percebido e tenta associar sua pecepção com um regra previamente programada, retornando a primeira regra que "casou" com a descrição do estado.

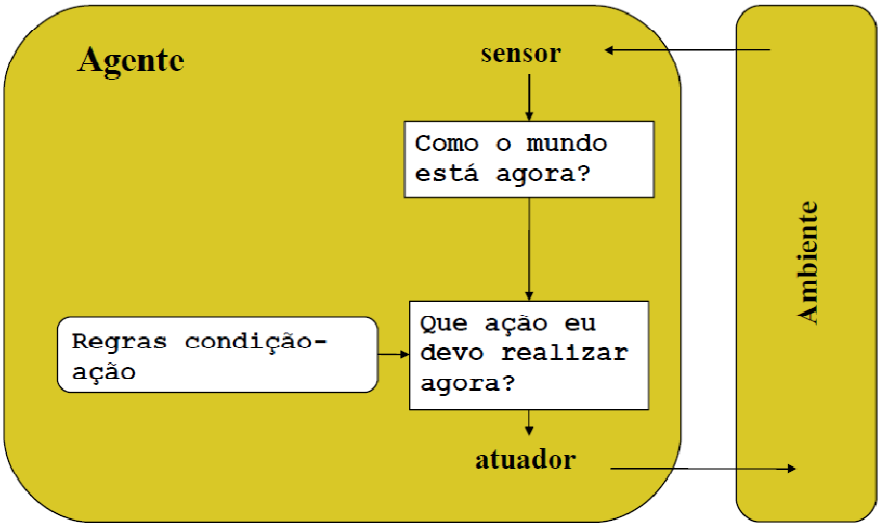



**Figura 26: Modelo de agente reativo [Russel & Norvig, 2003]**

### 3.2. Agente Reativo com Estado Interno

O agente com estado interno (Figura 27) deve controlar as partes do mundo que ele não pode perceber. São reflexivos que se baseiam também no estado passado do ambiente. Neste caso ele deve manter um estado interno que dependa do histórico de percepções e reflita os aspectos não observados no estado atual. Dois tipos de conhecimento são necessários para atualizar o estado interno do agente (modelo do mundo):

- Como o ambiente evoluí independente do agente e
- Como as ações do próprio agente afetam o mundo.

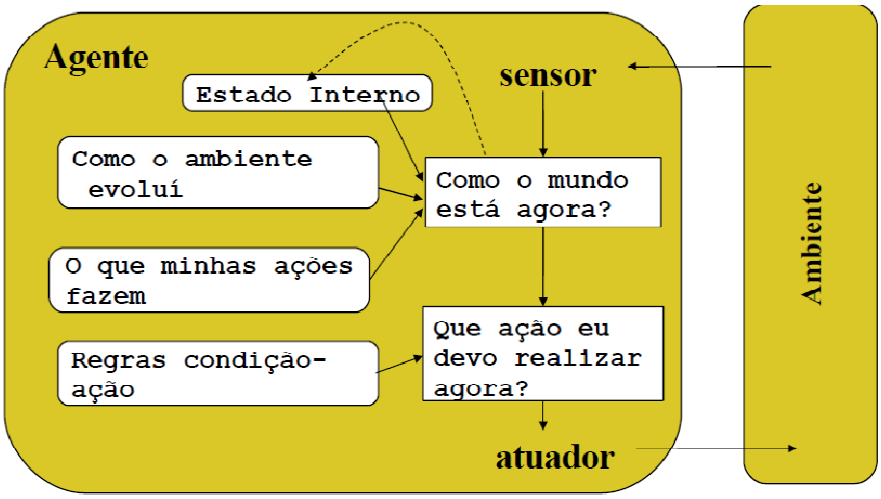



**Figura 27: Modelo de agente reativo com estado interno [Russel & Norvig, 2003]**

### 3.3. Agentes Baseados em Objetivos

Neste tipo de arquitetura, o agente precisa de algum tipo de informação sobre o seu objetivo a qual decreve situações desejáveis. Combinando informações sobre o objetivo do agente e os resultados de suas ações, o agente pode escolher aquelas que alcancem o objetivo. Para encontrar seqüências de ações que alcançam os objetivos este tipo de agente deve implementar algoritmos de Busca e Planejamento.

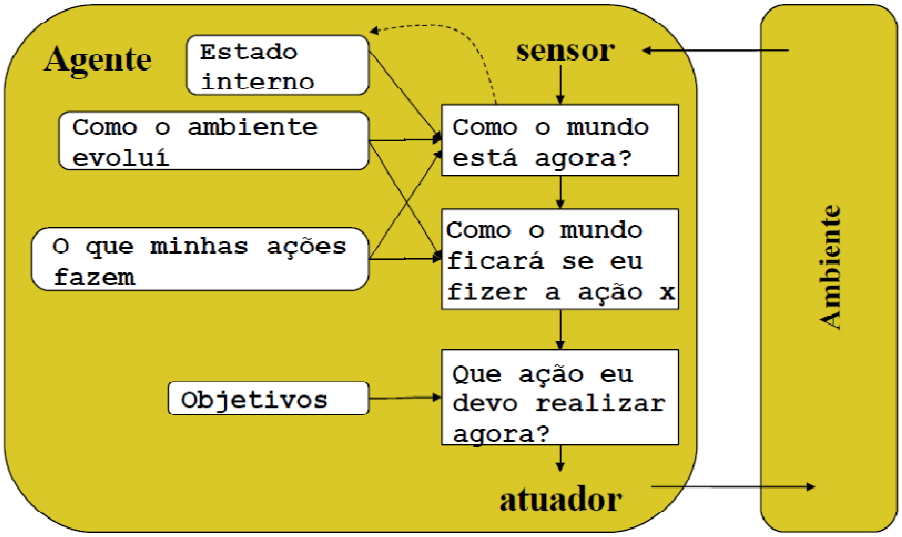



**Figura 28: Modelo de agente baseado em objetivos [Russel & Norvig, 2003]**

A seleção da ação baseada em objetivo pode ser:

- Direta: quando o resultado de uma única ação atinge o objetivo ou
- Complexa: quando será necessário longas seqüências de ações para atingir o objetivo.
-

### 3.4. Agente Baseado em Utilidade

Se um estado do mundo é mais desejável que outro, então ele terá maior utilidade para o agente. A utilidade é uma função que mapeia um estado para um número real que representa o grau de satisfação com este estado. A especificação completa da função de utilidade – decisões racionais em dois tipos de casos:

- Quando existem objetivos conflitantes (velocidade x segurança) a função de utilidade especifica o compromisso apropriado e
- Quando existem vários objetivos que se deseja alcançar e nenhum deles pode ser atingido com certeza – ponderar a importância dos objetivos
-

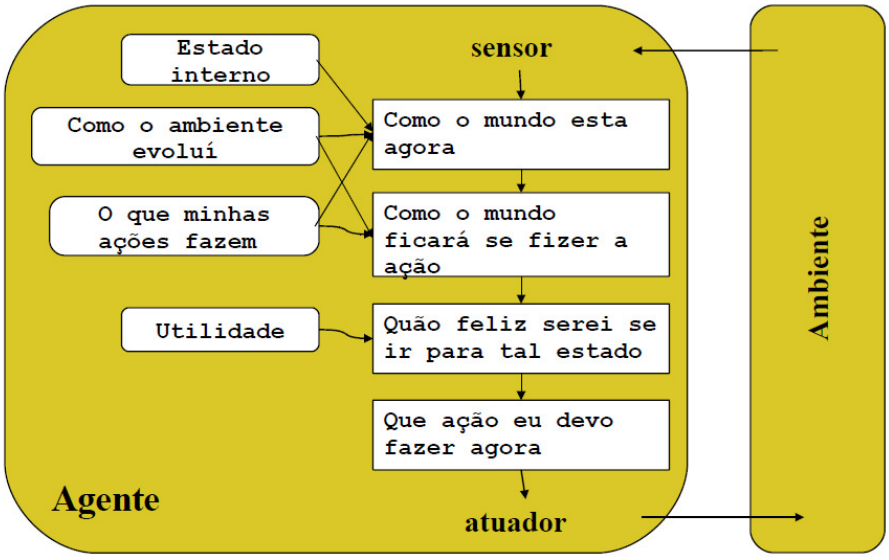



**Figura 29:Modelo de agente baseado em utilidade [Russel & Norvig, 2003]**

### 3.5. Agentes com Aprendizagem

Em agentes sem aprendizagem tudo o que o agente sabe foi colocado nele pelo projetista. A aprendizagem também permite ao agente atuar em ambientes

totalmente desconhecidos e se tornar mais competente do que o seu conhecimento inicial poderia permitir

- Elemento de aprendizado:
  - ✓ Responsável pela execução dos aperfeiçoamentos;
  - ✓ Utiliza realimentação do crítico sobre como o agente está funcionando;
  - ✓ Determina de que maneira o elemento de desempenho deve ser modificado para funcionar melhor no futuro
- Crítico:
  - ✓ Informa ao elemento de aprendizado como o agente está se comportando em relação a um padrão fixo de desempenho;
  - ✓ É necessário porque as percepções não fornecem nenhuma indicação de sucesso;

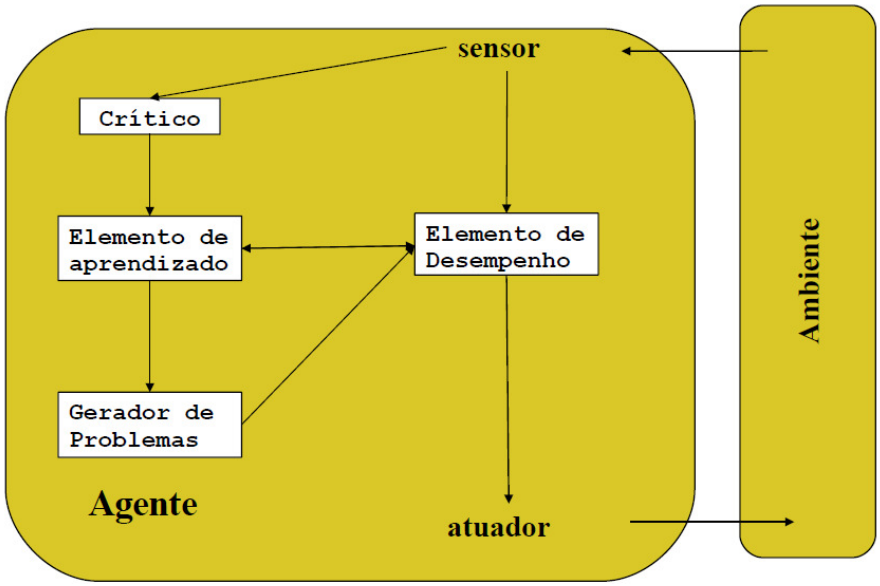



**Figura 30: Modelo de agente com aprendizagem [Russel & Norvig, 2003]**

# 4. Sistemas Multiagente

## 4.1. Inteligência Artificial Distribuída (IAD)

Existem diversos motivos para distribuir sistemas inteligentes. O principal deles é que alguns domínios de aplicação, por exemplo, controle de tráfego aéreo e distribuição de energia elétrica são inerentemente distribuídos no espaço. Outras razões incluem:

- Melhorar a adaptabilidade, a confiabilidade e a autonomia do sistema
- Reduzir os custos de desenvolvimento e manutenção

- Aumentar a eficiência e a velocidade
- Permitir a integração de sistemas inteligentes existentes de maneira a aumentar a capacidade de processamento e principalmente a eficiência na solução de problemas
- Permitir a integração dos computadores nas redes de atividades humanas

Além dessas razões deve-se ainda salientar que para problemas realmente grandes a única possibilidade de solução é a solução distribuída o que coloca a IAD como a única técnica indicada quando o problema ultrapassa um grau razoável de complexidade.

O surgimento dessa área se deve a necessidade de resolver alguns tipos de problemas onde a solução é inerentemente distribuída, geograficamente ou funcionalmente. A IAD pode ser considerada como uma união da IA tradicional (origem psicológica) coma a metáfora do comportamento social (origem sociológica) e irá abordar os sistemas de agentes como sociedades de agentes inteligentes.

Nesse ramo de pesquisa ainda são estudadas as técnicas de raciocínio e conhecimento que serão úteis para determinados grupos de agentes computacionais integrarem uma sociedade de agentes. IAD está centrada na resolução de cinco tipos de problemas:

1. como descrever, decompor e alocar tarefas para um conjunto de agentes;
2. como promover a interação e comunicação entre agentes (linguagens de comunicação, protocolos, o quê e quando comunicar);
3. como coordenar, controlar e assegurar o comportamento coerente (assegurar um comportamento global coerente em um conjunto de agentes);
4. como administrar conflitos e incertezas (resolver conflitos e coordenar as ações dos agentes);
5. como definir que linguagens e ambientes de programação devem ser utilizados para a implementação dos agentes.

IAD se dividiu em dois enfoques: a solução distribuída de problemas (SDP - *distributed problem solving*) e sistemas multiagentes (SMA - multiagent systems).

A SDP tem como foco principal o problema conforme a tradição na IA simbólica. Seus objetivos são utilizar a capacidade de processamento e a robustez oferecidas pela tecnologia de redes para atacar problemas naturalmente distribuídos ou excessivamente complexos. Para a SDP os agentes são pré-programados para

cooperar e seus métodos visam garantir que esta cooperação ocorra de maneira coerente, robusta e eficiente. A qualidade de um sistema de SDP é medida pelo desempenho global do sistema e os agentes envolvidos em SDP são programados dividir tarefas e comunicar-se de maneira confiável.

## 4.2. Sistemas Multiagente

Os Sistemas Multiagentes (SMA) são projetados para resolverem diversos tipos de problemas que requerem coordenação do comportamento inteligente entre uma coleção de agentes inteligentes e autônomos, envolvendo aspectos como seus conhecimentos, objetivos, habilidades e planos de forma conjunta para realizar ações ou solucionar problemas.

São agentes colaborativos e buscam a solução conjunta para alcançar os objetivos dentro de um ambiente. Isso ocorre pois, na maioria das situações, eles têm conhecimento da sua própria existência e da existência de outros agentes. Portanto o objetivo dos SMA é estudo das pressuposições básicas sobre agentes que garantam a possibilidade de ação cooperativa em sociedade.

Conforme suas funções, cada agente possui diferentes papéis na sua sociedade, para alcançar seus objetivos. Nesta sociedade, um problema complexo pode ser resolvido de forma eficiente por um grupo de agentes competentes que poderão dividir este problema em partes menores que serão solucionadas entre os agentes, conforme suas capacidades e objetivos que evoluem de forma dinâmica e autônoma, a fim de elaborar a resolução do problema.

Uma das razões para se utilizar a abordagem de SMA é que em algumas aplicações o conhecimento é essencialmente distribuído e um problema pode ser dividido em partes menores, facilitando a sua resolução. Baseando-se no grau de cooperação existente entre os agentes, pode se distinguir dois tipos de sistemas multiagentes:

- CMAS – Cooperative Multiagent Systems (Sistemas Cooperativos Multiagentes)
  - ✓ abrange sistemas onde os agentes desenvolvidos pelo mesmo projetista, buscando o melhor aproveitamento do sistema.
  - ✓ Para estes sistemas, o desempenho de cada agente em particular não é um fator decisivo;
- SMAS – Self-Interested Multiagents System (Sistemas Multiagentes com Interesses Próprios)

- ✓ sistemas onde os agentes foram programados por diferentes projetistas, visando não o benefício na cooperação, mas os benefícios procedentes de sua atuação autônoma.

Os sistemas multiagentes dividem-se em duas classes principais: Agentes Reativos e Agentes Cognitivos [Russell & Norvig, 2003]. A abordagem reativa baseia-se na idéia de que agentes com ações elementares podem realizar trabalhos complexos.

Segundo [Alvares, 1997], nos sistemas multiagentes reativos não há representação explícita do conhecimento; não há representação do ambiente; não há memória das ações; a organização é etológica, ou seja, similar a dos animais; e existe grande número de membros.

Os agentes reativos são muito simples e não possuem representação do seu ambiente. Suas reações dependem unicamente de sua percepção deste ambiente. A Inteligência Artificial clássica define um problema de uma maneira global, criando métodos de resolução que se aplicam diretamente e exclusivamente sobre esta definição. Os modelos de SMA Reativos, por outro lado, concebem o problema como sendo um conjunto de agentes interagindo entre si, onde cada um destes possui seus próprios objetivos individuais. Uma forma usual de representar os comportamentos dos agentes é através de um conjunto de regras.

Os sistemas multiagente cognitivos são baseados em modelos organizacionais humanos, como grupos, hierarquias e mercados. Nestes sistemas os agentes mantêm uma representação explícita de seu ambiente e dos outros agentes da sociedade; podem manter um histórico das interações e ações passadas; a comunicação entre os agentes é direta, através de mensagens; seu mecanismo de controle é deliberativo, ou seja, os agentes raciocinam e decidem seus objetivos, planos e ações; seu modelo de organização é sociológico; uma sociedade contém poucos agentes.

Apesar da classificação (Figura 31), os sistemas multiagente podem não ser totalmente cognitivos ou reativos. Um sistema pode ser uma mistura dos dois para atender a solução de um determinado problema.

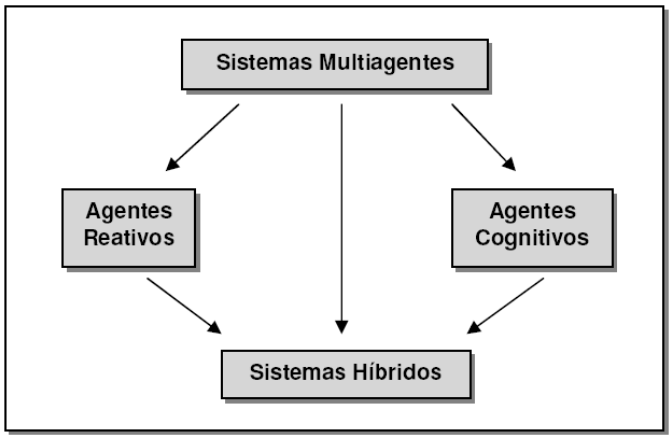



**Figura 31: Classificação dos Sistemas Multiagente**

### 4.2.1. Comunicação e Coordenação em Sistemas Multi-Agentes

As duas principais características de um agente, por definição, é perceber e agir. A habilidade de comunicar-se faz parte da percepção (quando recebe mensagens) e da ação (quando envia mensagens). A construção de sistemas multiagentes requer a escolha da arquitetura correta. Isto inclui a questão de como distribuir as responsabilidades do sistema entre os seus agentes (componentes) e como estes agentes devem interagir para cumprir estas responsabilidades.
Um sistema multiagente sempre deve considerar também as características da infraestrutura de comunicação a ser usada e as estratégias de coordenação das atividades dos agentes.

De maneira similar à sociedade humana, há uma série de situações nas quais é necessária a comunicação entre agentes. Tais situações incluem:

- Trocar informações entre os agentes a respeito da parte do sistema que cada um deles está,
- Consultar outros agentes sobre determinados aspectos do ambiente,
- Responder as solicitações de outros agentes,
- Aceitar requerimento e propostas,
- Compartilhar experiências e conhecimento.

A comunicação pode ocorrer de forma indireta, utilizando uma arquitetura do tipo quadro-negro e de forma direta, utilizando arquiteturas baseadas em trocas de mensagens, através das linguagens de comunicação de agentes (ACL).

A coordenação de agentes é o processo de gerenciar dependências entre atividades [Malone, 1994] em que agentes se engajam para garantir que um grupo de agentes tenha um comportamento coerente [Nwana & Jennings, 1996]. Exemplos:

- controle de tráfego aéreo
- time de futebol
- desfile de carnaval
- trânsito de automóveis em uma cidade
- operação militar
- vôo de bando de pássaros
- sistema imunológico animal
- construção de avião, estrada, etc.
- operação cirúrgica
- orquestra

A coordenação é um pressuposto fundamental para o trabalho conjunto. Um SMA pressupõe coordenação entre seus agentes! Porque geralmente há dependências entre as ações uma vez que nenhum agente pode resolver o problema sozinho. Contudo deve-se respeitar as restrições globais e garantir a harmonia na execução conjunta de tarefas.

Portanto, em outras palavras coordenação em um SMA é o processo pelo qual um agente raciocina sobre suas ações locais e as de outros agentes para garantir que a comunidade funcione coerentemente, visando garantir que todas as partes necessárias existam na sociedade e a interação que possibilite a execução das atividades. Além disso a coordenação deve garantir que todos atuem consistentemente e que tudo seja feito com dos recursos disponíveis.

Para haver coordenação com sucesso é preciso:

- Uma estrutura...
- Flexibilidade nas interações
  - ✓ Comunicação
  - ✓ Negociação
- Conhecimento e raciocínio

- ✓ Para reconhecer interações potenciais entre planos de ação.

## Exercícios

1. Defina o que vem a ser um agente na IA.
2. O que é a medida de desempenho de uma agente?
3. Quais são as características de um ambiente de agente? Descreva cada um deles
4. Faça um estudo comparativo entre os diferentes tipos de agentes.
5. Defina as principais diferenças entre a IAD distribuída e convencional.
6. Na IAD, qual a diferença entre a Solução de Problemas Distribuídos e Sistemas MultiAgentes?
7. Baseando-se no grau de cooperação existente entre os agentes, explique os dois tipos de sistemas multiagentes (CMAS – Cooperative Multiagent Systems e SMAS – Self-Interested Multiagents System).
8. Qual a importância da comunicação em um SMA
9. Em que conceitos se baseia a coordenação de agentes em um SMA?

## Resumo

Os agentes inteligentes tem sido amplamente utilizados em várias áreas de ciência da computação. Esse paradigma de implementação é o que mais se assemelha com o comportamento humano, trazendo consigo conceitos como percepção, autonomia e aprendizado.

Além disso o uso de vários agentes em conjunto, caracterizando um sistema multiagente, vêm se tornando uma alternativa quando se fala em inteligência artificial distribuído.

Nesta unidade apresentaremos as principais características dos agentes e seus diferentes tipos apresentando também os atributos necessários para a implementação dos sistemas multiagente.

## Weblografia

Universidade Aberta do Piauí – UAPI

http://www.ufpi.br/uapi

Universidade Aberta do Brasil- UAB

http://www.uab.gov.br

Secretaria de Educação a Distância do MEC – SEED

http://www.seed.mec.gov.br

Uma Introdução aos Agentes Inteligentes

http://www.computacao.gigamundo.com/2009/04/06/uma-introducao-aos-agentes-inteligentes/

Agentes Inteligentes

http://www.cic.unb.br/~jhcf/MyBooks/ciber/doc-ppt-html/AgentesInteligentes.html

Inteligência Artificial Distribuída

http://www.das.ufsc.br/gia/iaft-apoio/iad.pdf

Coordenação de Atividades em Ambientes de Aprendizagem Colaborativos

http://www.tecgraf.puc-rio.br/publications/artigo_2002_coordenacao_atividades_aprendizagem.pdf

## Referências Biblográficas


Russel, Stuart; Norvig, Peter: Inteligência Artificial. Campus São Paulo, 2003. 1040p
Bittencourt, Guilherme: Inteligência Artificial – Ferramentas e Teorias. Editora da UFSC. 2ª. Edição. Florianópolis, 2001. 362p

Rich, Elaine; Knight, Kevin: Inteligência Artificial. Makron Books. 2ª. Edição. São Paulo, 1994. 722p.

Giddens, A. Central problems in social theory: action, structure and contradiction in social analysis. USA: University of California, 1979.

Holmstron, B.; Tirole, J. The theory of the firm. In: Schmalensee, R.; Willig, R. D., eds. Handbook of Industrial Organization, v.I. Amsterdan: Elsevier, 1989. p.61-133.

Meyer, J.-A.; Roitblat, H.; Wilson, S. W. From animals to animats 2: proceedings of the second international conference on simulation of adaptive behavior. USA: MIT Press, 1993. 523p.
Finin, T.; Mckay, D.; Fritzson, R. An overview of KQML: a knowledge query and manipulation language, Draft. USA: University of Maryland, Mar 1992. Disponível: site UMBC Agent Web. URL: http://www.cs.umbc.edu/kqml. Consultado em 10 mar. 1999.

Dawkins, R. The selfish gene. Oxford: Oxford, 1976. 352p.

Alvares, L. and Sichman, J. Introdução aos sistemas multiagentes. Jornada de Atualização em Informática, 1997. W.C. Hill.

The mind at AI: Horseless carriage to clock. *The AI Magazine*, pages 29-41, Summer 1989.

Malone, T. W. & Crowston, K. The Interdisciplinary Study of Coordination. ACM Computing Surveys, vol. 26, no. 1, pp. 87-119.

Nwana, H. S. (1996) “Software Agents: An Overview”. Knowledge Engineering Review, vol.11, n. 3, pp. 1-40, Cambrigde University Press.

# UNIDADE 6

## INTRODUÇÃO A ROBÓTICA

## 1. Definições

Já há muito tempo o homem procura empregar dispositivos para realizar tarefas em ambiente inóspitos ou perigosos à saúde do homem, ou tarefas repetitivas ou que envolvam a análise de uma quantidade elevada de informações e que necessitam de resultados com precisão e eficiência. E em todas essas atividades executadas por mecanismos construídos pelo homem há a tomadas de decisões de acordo com as informações coletadas do ambiente.

O desenvolvimento da robótica iniciou-se na metade do século XVIII, com a construção de bonecas mecânicas de tamanho humano que tocavam música, passando pela invenção de máquinas programáveis para tecer, no início do século XIX, e máquinas operadas por controlador baseado em registros de sinais elétricos, no ano de 1946. A construção de robôs utilizados em linhas de montagens iniciou-se a partir dos anos 70 com o desenvolvimento de linguagem de programação, estudos e pesquisas em inteligência artificial que visam ao desenvolvimento de sistemas que pareçam comportar-se de forma inteligente.

Atualmente os robôs são os dispositivos que executam essas tarefas. E é a robótica o campo que se preocupa com o desenvolvimento de tais dispositivos. A diversidade de tipos de robôs que existem impedem que haja uma definição de robô que seja universalmente aceite. No entanto há um conjunto comum de componentes que essa diversidade de robôs partilha, como por exemplo, sistemas de locomoção, sensores e atuadores.

Portanto, convencionou-se que um robô é uma máquina programável que imita as ações ou aparências de uma criatura inteligente, geralmente um humano. A Federação Internacional de Robótica (IFR) define o robô como sendo “uma máquina que pode ser programada para executar tarefas que envolvam ações de manipulação, e em alguns casos ações de locomoção, sob um controle automático”. Já de acordo com a *Robotics Industries Association* (Associação das Indústrias de Robótica) temos a seguinte definição de robô:

> “*Um robô é um dispositivo mecânico articulado programável, que consegue, de forma autônoma e pode, devido à sua capacidade de processamento:*

- *obter informação do meio envolvente utilizando sensores;*

- *tomar decisões sobre o que deve fazer com base nessa informação e em informação à priori;*
- *manipular objetos do meio envolvente utilizando atuadores.*"

A robótica é uma área multidisciplinar altamente ativa, que busca o desenvolvimento e a integração de técnicas e algoritmos para a criação de robôs. Para fazer um robô funcionar os engenheiros têm que dominar técnicas de diversos ramos da ciência tais como matemática, física, mecânica, eletrônica, teoria de controlo de sistemas, automação industrial, visão computacional, comunicações, processamento de sinais, entre outras.

## 2. Tipos de Robôs

Devido a várias diferenças em função de características e propriedades, existem diversas classes de robôs que se diferenciam em suas aplicações e formas de trabalhar.

Os **Robôs Inteligentes** são manipulados por sistemas multifuncionais controlados por computador, são capazes de interagir com seu ambiente através de sensores e de tomar decisões em tempo real. Atualmente dedicam-se grandes esforços no desenvolvimento desse tipo de robô.

**Robôs com controle por computador** são semelhantes aos robôs inteligentes, porém não tem a capacidade de interagir com o ambiente. Se estes robôs forem equipados com sensores e software adequado, se transformam em robôs inteligentes.

Já os **robôs de aprendizagem** limitam-se a repetir uma seqüência de movimentos, realizados com a intervenção de um operador ou uma seqüência memorizada.

Os **robôs manipuladores** são sistemas mecânicos multifuncionais, cujo sistema de controle permite governar o movimento de seus membros das seguintes formas:

- manual, quando o operador controla diretamente os movimentos;
- de seqüência variável, quando é possível alterar algumas das características do ciclo de trabalho.

# 3. Organização Funcional de um Robô

Em termos gerais, a arquitetura de um robô se organiza como o diagrama da Figura 32.

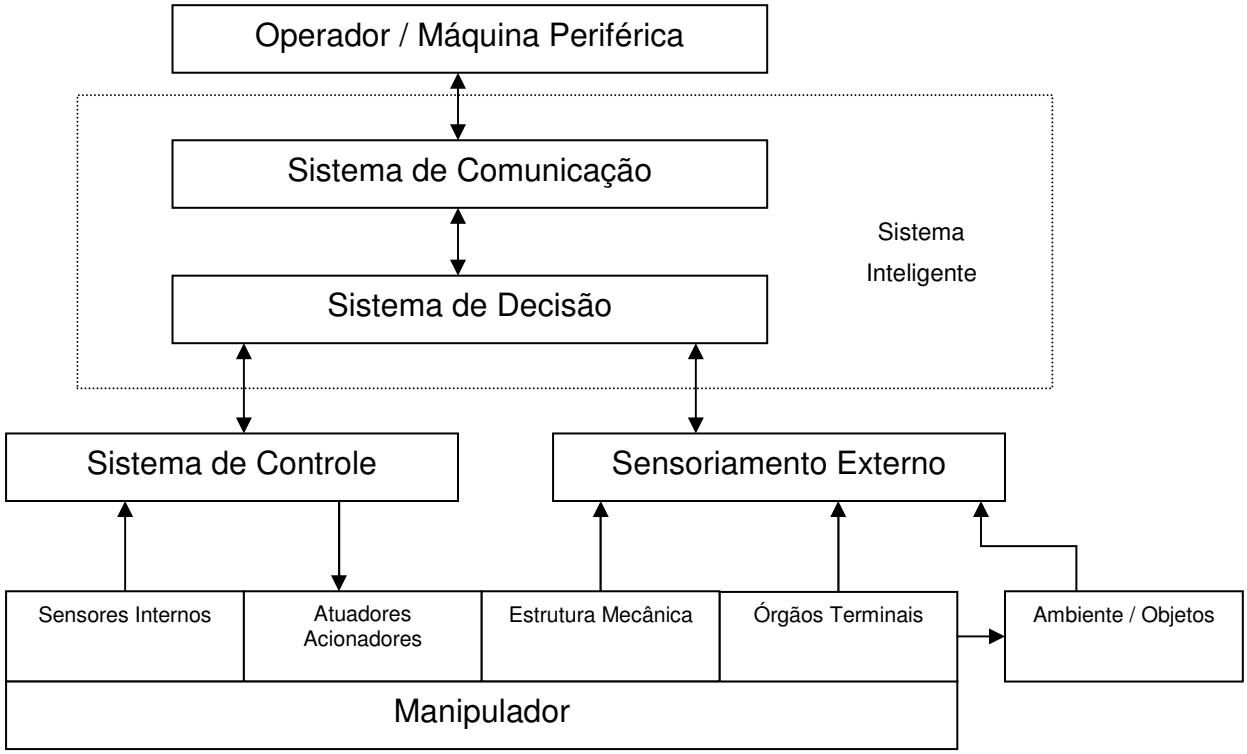



**Figura 32: Organização Funcional de um Robô [ADADE FILHO, 1999]**

Sistema de ação mecânica (**manipulador**) é constituído por uma estrutura mecânica que suporta ou conduz o órgão terminal de atuação (garra ou ferramenta) e também pelos elementos de acionamento que atuam sobre esta estrutura, proporcionando seu movimento.

O Sistema de percepção (s**ensoriamento**): visa obter informações sobre o estado do robô, dos objetos no espaço de trabalho e das máquinas com as quais ele trabalha.

Já o Sistema de controle comanda os atuadores e conseqüentemente a estrutura mecânica do robô, através de sinais adequados sintetizados a partir das informações advindas do operador, do sistema de percepção e de seu próprio sistema de inteligência.

No Sistema de Inteligência temos a comunicação e decisão a ser realizada pelo robô. É o nível hierárquico mais alto de controle, constituído por um conjunto de programas lógicos de representação, comunicação (troca de informação) e tratamento dirigido (inteligente) das informações advindas do operador, do sistema de percepção ou do sistema de controle. Esse sistema deve executar as seguintes tarefas:

- Comunicação com o operador, com os sistemas de percepção, de supervisão e de segurança e com o sistema de sincronização com outras máquinas. Eventualmente poderá haver um sistema de análise e síntese de voz;
- Interpretação adequada das mensagens;
- Resolução de problemas associados a cada mensagem;
- Geração de planos de ação e de mensagem para o operador ou outras máquinas;
- Gestão de base de dados associada ao sistema;
- Geração dos movimentos de referência a serem executados pelo sistema de controle;
- Coordenação dos movimentos do manipulador com o seu ambiente;
- Geração de informações complementares para o sistema de controle.

## 4. DISPOSITIVOS DE ROBÓTICA

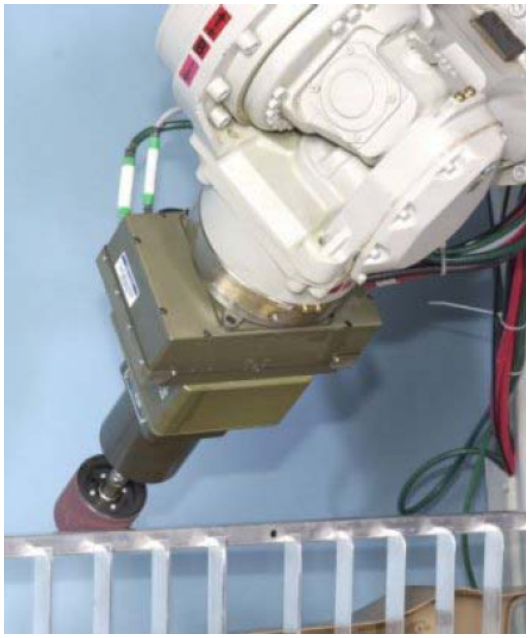


**Figura 33: Com ajuda de sensores os robôs memorizam movimentos de uma tarefa para depois repeti-la através de seus atuadores (Fellipe de Souza)**

Os robôs usam sensores para obter informações do seu mundo em volta, para desempenhar as suas tarefas, e em especial no manuseio de objetos. Há sensores para muitas grandezas tais como posição, distância, visão, acústicos e muitos outros.

E quanto a natureza destes sensores também há muitos tipos: óticos, fotoelétricos, infra-vermelhos e ultra sônicos. Em geral um sensor mede uma característica do ambiente ou espaço em que ele está e proporciona sinais elétricos. Estes dispositivos simulam os sentidos humanos, principalmente a visão. Mas os robôs têm a vantagem de poder detectar características físicas que nós humanos não conseguimos detectar com os nossos sentidos, como por exemplo: os campos magnéticos, ondas ultra-sónicas, entre outros.

As dificuldades que os sensores por vezes têm são relacionadas com a interferência nas medidas que fazem, ou em outras. Ora o sensor pode sofrer a interferência, ora ele pode interferir em algumas grandezas do sistema. Por exemplo, os medidores de esforço ou pressão podem ser sensíveis à temperatura.

Em geral um sensor dá a sua medida como um sinal elétrico. Se desejamos a medida em outra grandezas é necessário usar um transdutor. Transdutores são dispositivos que transformam um determinado tipo de medida (ou grandeza física) num outro tipo diferente.

Os **sensores óticos** podem medir quase todas as grandezas físicas. Estes sensores são chamados de óticos porque usam técnicas magnético-óticas, ou de laser, ou com fibras ópticas, ou de reflexão de luz ou outras radiações eletromagnéticas.

As vantagens dos sensores ópticos (sobre os sensores não óticos ou convencionais) são uma maior sensibilidade na medição, passividade elétrica, larga amplitude dinâmica, configuração de ponto e distribuída (isto é, podem medir localmente ou uma região grande) e capacidade multiplexadora isto é, podem receber ou enviar vários sinais em um mesmo canal de comunicação.

Os sensores fotoelétricos de luz são uma forma de visão para a robótica. Estes sensores mudam a resistência, o díodo, ou o transistor conforme detecta luz. Ou seja, quando um feixe de luz é detectado eles respondem seja criando ou trocando um sinal elétrico que será analisado permitindo que o dispositivo tome uma decisão. Com o uso de filtros um sensor de luz pode criar respostas seletivas com

as quais o robô unicamente poderá perceber determinadas cores. O uso de sensores de luz permite aos robôs se comunicarem.

Para sistemas mais complexos os sensores de luz não são suficientes. Por exemplo: eliminar um produto defeituoso da banda transportadora numa linha de produção. Esta é uma tarefa que os humanos fazem com certa facilidade mas porque não usam somente a visão, mas sobretudo o cérebro, na tomada de decisão.

Nestes casos os robôs necessitam do auxílio do computador para fazer a seleção com base em informações que os sensores de luz produzem. Para isto o computador muitas vezes tem que usar técnicas de Inteligência Artificial (reconhecimento de padrões) que simulam o funcionamento do nosso cérebro na tomada de decisões. O computador processa e envia uma informação de volta para o dispositivo robótico com uma ordem (de aceitar ou rejeitar o produto).

Os sensores de infra-vermelho são usados para comportamentos simples dos robô, como por exemplo, evitar obstáculos ou mesmo para os robôs se deslocarem. O robô emite um raio para um obstáculo e mede a distância de maneira similar a um radar (em aviões) ou sonar (em navios).

Mas mesmo a visão robótica pura e simples é ainda muito imperfeita e portanto, um dos grandes desafios para a engenharia de hoje em dia. Para poder gerar imagens tridimensionais a partir de 2 imagens muito semelhantes em um tempo curto se requer uma grande quantidade de memória e de um processador muito poderoso.

É difícil programar um robô para que ele saiba o que deve ignorar e que não deve ignorar das imagens que ele “vê”. Os robôs têm problemas para interpretar sombras, trocas de luzes e brilho. Além disso, para poder ter percepção da profundidade é necessário que tenham visão estereoscópica (3D), como nós humanos que temos dois olhos.

Os sensores de tato também ajudam aos robôs (que não têm capacidade de visão) a caminhar. Os sensores contatam e enviam um sinal para que o robô saiba que há tocado com algum objeto. Os sensores mais comuns para isto são os do tipo piezelétrico.

Com os sensores de posição tornam possível ensinar a um robô a fazer uma função repetitiva em função dos seus movimentos. Os sensores localizados em certos pontos do próprio robô guardam informações sobre as trocas de posições.

Desta forma o robô poderá então recordar a informação e repetir o trabalho na forma exata que foi realizado inicialmente.

Alguns outros tipos de sensores robóticos, ou combinação dos já mencionados acima:

- Acelerômetro: Detecta movimentos, vibrações, e ângulos com respeito à gravidade.
- Sensor de corrente: Mede o uso de corrente e potência pelo robô.
- Bússola digital: Detecta orientação com respeito ao campo magnético terrestre.
- Encoders: (Linear ou translacional, Rotary ou de rotação e Slot ou de ranhura): Usado para determinar distância translacional, velocidade rotacional e/ou ângulo das partes móveis do robô.
- Emissor e detector infra-vermelho: Emite e detecta raios infra-vermelho. Pode ser usado para sinalizar, para evitar obstáculos, e para detectar cor.
- Sensores de carga e de momento (torque): Mede momentos e outras forças do robô.
- *Rangefinder*: Detecta limites de obstáculos de poucos centímetros até vários metros. Modulado para estar imune a irradiações de infra-vermelho do ambiente.
- Sonar ou sensor ultra sônico: Detecta obstáculos e pode determinar a dureza / maciez dos objetos pela eco-locação.
- Chaves táteis de choques: Detecta contacto físico do robô quando colide com algo.
- Sensor piroelétrico: Detecta fogo e outras fontes de calor (como velas acesas, chamas, etc.). Também é usado para detectar movimento de pessoas e animais, pois irradiam calor do seu corpo.
-

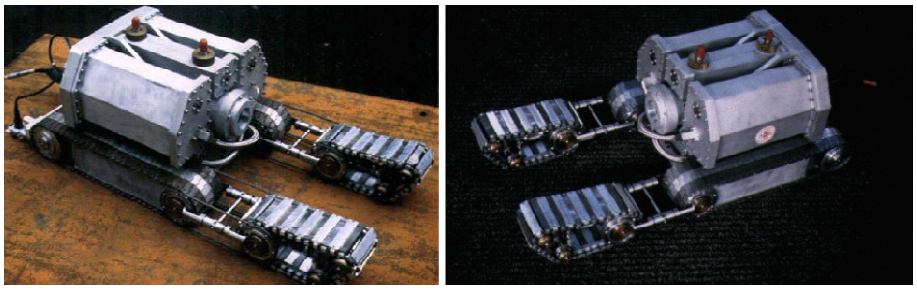


**Figura 34: O robô construído na Universidade Carnegie Mellon nos Estados Unidos para fazer inspeções do interior de tanques e reservatórios de combustíveis (Fellipe de Souza)**

## 4.1. Atuadores

Os atuadores são usados em automação para entregar ao sistema a excitação necessária para seu funcionamento, na forma do tipo de energia adequado. Se o funcionamento do sistema estiver baseado em algum movimento de uma de suas partes, serão necessários atuadores para fornecer energia mecânica para o movimento. Se o sistema for térmico, será necessário um atuador que forneça energia térmica para atingir uma dada temperatura desejada.

Os atuadores se dividem em: hidráulicos, pneumáticos e elétricos. Os atuadores hidráulicos se caracterizam por terem como fonte de energia um líquido que se desloca por um conduto com uma pressão adequada. Este líquido é geralmente “óleo” ou “água”.

O atuador pneumático tem como fonte de energia um gás pressurizado, geralmente ar comprimido. Já os atuadores elétricos usam energia elétrica.

Os atuadores hidráulicos são os mais antigos pois foram os primeiros a serem usados e são normalmente empregado em sistemas onde se requer altas velocidades nos movimentos, com pouco controle sobre o posicionamento final, em aplicações onde o momento exigido é relativamente baixo.

Os atuadores pneumáticos funcionam com energia pneumática (ar comprimido) e executam movimentos lineares, rotativos e semi-rotativos ou angulares. As três variáveis básicas para o controlo desses movimentos são o sentido do movimento, a velocidade e a força.

Atuadores elétricos são equipamentos eletromecânicos que substituem com alta confiabilidade a operação manual de válvulas em:

- locais de difícil acesso ou periculosidade elevada para o operador;
- casos que demandam conjugado de actuação elevado;
- condições onde for requerido posicionamento rápido, especialmente em válvulas cujo número total de voltas seja grande;
- regime de trabalho com alta-frequência de manobras;
- controlo automático de processos onde as válvulas operam em duas posições extremas ou com reposicionamentos intermediários (modulação).

## 5. Linguagem de Programação de Robôs

Os robôs são controlados por computador. Alguns sistemas fazem uso de vários processadores, de modo a tratar adequadamente as situações de comunicação/controle previstas, processar os comandos dados pelo usuário e sua conversão em comandos para o robô.

As categorias básicas de programação podem ser por aprendizagem e através de linguagem. A programação por aprendizagem consiste em fazer o robô movimentar-se, em fase de aprendizagem, segundo uma seqüência de movimento requisitado e registrar o movimento na memória do controlador. Na programação por linguagem textual é possível definir coordenadas e estabelecer a lógica e seqüência do ciclo de trabalho. O uso de linguagem amplia as possibilidades de cálculos, permite o detalhamento do fluxo lógico e um maior uso de sensores e comunicação.

Durante o desenvolvimento industrial, alguns equipamentos programáveis acabaram por se beneficiar com a padronização de suas linguagens de programação. Essa padronização é altamente vantajosa, permitindo a migração rápida de programas entre plataformas e fazendo com que o aprendizado da linguagem seja útil em uma grande quantidade de equipamentos. É impossível deixar de citar os Controladores Lógicos Programáveis (CLP) que usam, por exemplo, a linguagem LADDER e os Comandos Numéricos Computadorizados (CNC) que empregam em sua maioria a linguagem conhecida popularmente como Código G. Apesar da padronização, deve ficar claro que há pequenas diferenças entre equipamentos devido às características particulares de cada um, que acabam levando a adaptações da linguagem utilizada.

Pelo fato desses equipamentos possuírem grandes diferenças entre suas configurações e aliado ao fato de poderem ser aplicados a tarefas extremamente distintas, cada fabricante acabou desenvolvendo sua própria linguagem. Uma pesquisa na Internet revela a grande quantidade de linguagens existentes.

## 6. Robótica e Inteligência Artificial

Os modernos conceitos de controle de robôs, assim como de utilização de sensores de visão e táteis, têm suas raízes nas pesquisas sobre inteligência

artificial. Assim, em vez de armazenar imagens de peças e objetos dentro da memória dos sistemas de visão, certas características dos objetos são armazenadas, como, por exemplo, perímetro, área, número de furos e assim por adiante. Conforme os sistemas vão-se tornando mais complexos e começam a ter capacidade para tratar em ambientes tridimensionais abarrotados de dados, eles exigem maior inteligência por parte do sistema de visão.

Outra área promissora é a pesquisa no campo de programação ao nível de tarefa. Em vez de programar o robô para realizar um dado movimento, será possível emitir comandos e o robô será capaz de reagir e planejar a execução da tarefa. Por exemplo, ao comandar seu criado-robô para “pegar o meu jornal”, ele precisará processar esse comando, definir a trajetória até onde o jornal estará, identificar o objeto jornal, pegá-lo, caminhar por uma trajetória de retorno e entregar o jornal.

Esse aperfeiçoamento, com o intuito de dotar os robôs de uma inteligência para executar as tarefas necessárias, é conseguido com o desenvolvimento de outros campos da inteligência artificial, como as Redes Neurais e a Lógica Fuzzy.

As redes neurais:

- são adaptativas, ou seja, aprendem com a experiência;
- são capazes de generalizar a experiência, e são capazes de resolver corretamente situações com um certo grau de variabilidade;
- se forem ligadas à sensores adequados, são capazes de reconhecer padrões complexos, visuais ou outros, como forma, cor, som, textura, etc.

A lógica fuzzy facilita a construção de regras, tais como: ande mais rápido, vire um pouco para a esquerda, diminua bastante a velocidade, obstáculo perto, entre outras.

## 7. Aplicações

O desenvolvimento de sistemas “inteligentes” que interagem com o meio ambiente permite a exploração de locais de difícil acesso ou onde a presença do homem pode ser posta em risco, além das aplicações atuais, outras poderão ser desenvolvidas.

A indústria que mais usa manipuladores industriais, ou braços robóticos, como são também chamados, é principalmente a indústria automóvel. Isso inclui as indústrias

montadoras de veículos assim como também as indústrias de auto-peças. Na verdade estas e outras grandes indústrias usam não só os robôs manipuladores como também “automação” em geral. Isso quer dizer robôs que não são os braços manipuladores e outros tipos de máquinas automatizadas. Muitos outros exemplos ainda poderiam ser acrescentados. Uma lista completa de indústrias que usam automação e robótica seria infindável.

Na sociedade os robôs estão nas linhas de montagens da industrias metalúrgica e de precisão, poderão auxiliar idosos, deficientes ou enfermos nas atividades diárias, virão como companheiros na formas de pequenos animais como cachorro e gatos e continuarão a prestar seus serviços na segurança pública, na prospecção a grandes profundidades ou nos pontos distantes do espaço.

## Exercícios

1. Defina o que vem a seu um robô?
2. Qual o papel e o que estuda a robótica?
3. Enumere e defina os tipos de robôs.
4. Qual o papel dos sensores em um sistema robótico?
5. Qual a diferença e qual o ramo de atuação dos sensores óticos e os fotoelétricos.
6. Em que são utilizados os sensores infra-vermelho?
7. Qual o papel dos atuadores no sistema robótico?
8. Os atuadores se dividem em: hidráulicos, pneumáticos e elétricos. Explique a diferença entre eles.

## Resumo

Das sub-áreas da IA uma das que chamam mais atenção é a robótica. Tanto que, atualmente é considerada por muito uma área científica desassociada da inteligência artificial.

Muita dessa atenção se deve ao fato que a robótica é uma campo de pesquisa com maior penetração social. Seja pelo cinema, seja pela utilização na maioria das industrias ou seja pelo seu uso no cotidiano das pessoas.

Nesta unidade apresentaremos as principais definições de termos associados a robótica explicando através deles como se organiza funcionalmente um robô. Mostraremos ainda as principais aplicações da robótica.

## Weblografia

Universidade Aberta do Piauí – UAPI

http://www.ufpi.br/uapi

Universidade Aberta do Brasil- UAB

http://www.uab.gov.br

Secretaria de Educação a Distância do MEC – SEED

http://www.seed.mec.gov.br

Associação Brasileira de Educação a Distância – ABED

http://www.abed.org.br

Robótica

http://www.ele.ita.cta.br/~tojo/arquivos/robotica_doc.doc

Classificação Geral dos Robôs

http://www.din.uem.br/~ia/robotica/classif.htm

Robótica - *Felippe de Souza*

http://webx.ubi.pt/~felippe/texts/robotica_cap0.pdf

Linguagem de Programação de Robô

http://www.mecatronicaatual.com.br/secoes/leitura/418

Robôs na Industria

http://www.demnet.ubi.pt/~felippe/texts/robotica_cap3.pdf

ROBOTICS – sensing thinking acting.

http://www.thetech.org/exhibits_events/online/robotics/index.html

International Association for Automation and Robotics in Construction

http://www.iaarc.org/frame/features/market/background.htm

RIA - Robotic Industries Association

http://www.robotics.org

## Referências Biblográficas

Russel, Stuart; Norvig, Peter: Inteligência Artificial. Campus, São Paulo, 2004. 1040p.

GROOVER, M.P.; WEISS, M.; NAGEL, R. N.; ODREY, N. G. Robótica: Tecnologia e Programação 1989 MacGraw-Hill.

Adade Filho, A. Fundamentos de Robótica Versão 2.0 julho 1999 São José dos Campos, CTA-ITA-IEMP.

---

***Vinicius Ponte Machado***

CV. http://lattes.cnpq.br/9385561556243194

Doutor em Engenharia Elétrica de Computação pela Universidade Federal do Rio Grande do Norte (2009) e Mestre em Informática Aplicada pela Universidade de Fortaleza (2003). Atualmente é professor adjunto da Universidade Federal do Piauí. Tem experiência na área de Ciência da Computação, com ênfase em Gestão do Conhecimento e Inteligência Artificial, atuando principalmente nos seguintes temas: sistemas multiagente, redes neurais artificiais e Redes Industriais.