Representação e Processamento do Conhecimento

Conteudista: Prof. Me. Manuel Fernández Paradela Ledón
Revisão Textual: Prof. Me. Luciano Vieira Francisco
Revisão Técnica: Prof. Me. Douglas Almendro

Objetivos da Unidade:

- Conhecer diferentes formas de representação e processamento do conhecimento;
- Conhecer as características fundamentais da lógica proposicional e da lógica de predicados;
- Conhecer as características principais das redes neurais artificiais;
- Conhecer as características fundamentais da lógica nebulosa ou *fuzzy*.

📄 **Material Teórico**

📄 **Material Complementar**

📄 Referências

📄 Material Teórico

Representação de Conhecimento

Já tínhamos comentado anteriormente que as técnicas de Inteligência Artificial (IA) precisam de conhecimento: representar ou armazenar o conhecimento, processá-lo e modificá-lo. Por este motivo, a bibliografia e as linhas de pesquisada IA estudam as distintas formas de representar e organizar o conhecimento, realizar buscas e implementar ações relacionadas ao conhecimento e à inteligência humana. Portanto, o conhecimento é fundamental para a Inteligência Artificial!

Assim, estudaremos algumas possibilidades que existem para armazenar conhecimento, algumas mais simples e outras mais complexas.

Lógica Proposicional

A lógica proposicional também é conhecida como lógica das proposições, ou ainda como lógica simbólica, uma vez que utiliza proposições ou símbolos lógicos.

Veja algumas características da lógica proposicional:

- Utiliza proposições interligadas por conectivos lógicos – operadores. Os conectivos lógicos válidos em lógica proposicional serão mostrados a seguir;
- Uma proposição lógica é uma afirmativa adequadamente formada e que terá um valor verdadeiro ou falso, por exemplo:
  p: Montevidéu é a capital do Uruguai – verdadeiro
  q: Assunção é a capital do Chile – falso

- Os símbolos proposicionais são representados por letras, geralmente minúsculas, tais como p, q, r, s, t.

Ademais, os principais conectivos lógicos utilizados em lógica proposicional são:

- $\wedge$ Conjunção, operação E, por exemplo, $p \wedge q$;
- $\vee$ Disjunção, operação OU, por exemplo, $p \vee q$;
- ~ Negação, operação NÃO, em alguns textos usa-se $\neg$, por exemplo: $\sim q$ (não q);
- → Condicional, por exemplo, p → q, "se p então q";
- $\leftrightarrow$ Bicondicional, por exemplo, $p \leftrightarrow q$, "p se e somente se q";
- ( ), Para a agrupação de termos – prioridades – e separação;
- ≤ < > ≥ = ≠ + – Operadores de relação – comparação – e aritméticos.

Quadro 1 – Verdade das principais operações lógicas

p	q	$p \wedge q$	$p \vee q$	$\sim p$	$p \leftrightarrow q$	$p \rightarrow q$
0	**0**	0	0	1	1	1
0	**1**	0	1	1	0	1
1	**0**	0	1	0	0	0
1	**1**	1	1	0	1	1

Observe neste Quadro os resultados das principais operações lógicas, onde cada coluna de resultado mostra os valores de determinado conectivo lógico para valores de p e q. Assim, um valor 0 significa falso e um valor 1 corresponde a verdadeiro.

Uma proposição lógica poderia ser modificada conhecendo algumas leis e certos teoremas fundamentais. Assim, veremos alguns dos quais, mas é sugerido aprofundar tais assuntos em livros de lógica matemática, tais como o de Alencar Filho (2006). Vejamos:

Lei da comutatividade:

- $p \wedge q$ é equivalente a $q \wedge p$;
- $p \vee q$ é equivalente a $q \vee p$.

Lei da associatividade:

- $(p \wedge q) \wedge r$ é equivalente a $p \wedge (q \wedge r)$;
- $(p \vee q) \vee r$ é equivalente a $p \vee (q \vee r)$.

Lei da distributividade:

- $p \wedge (q \vee r)$ é equivalente a $(p \wedge q) \vee (p \wedge r)$;
- $p \vee (q \wedge r)$ é equivalente a $(p \vee q) \wedge (p \vee r)$.

Lei ou teorema de Morgan:

- $\sim (p \vee q)$ é equivalente a $\sim p \wedge \sim q$;
- $\sim (p \wedge q)$ é equivalente a $\sim p \vee \sim q$.

Propriedade idempotente:

- $p \wedge p$ é equivalente a p;
- $p \vee p$ é equivalente a p.

E de que forma a lógica proposicional serviria para armazenar conhecimento? Ou seja, como as leis, teoremas ou transformações ajudariam a modificar tal saber? Vejamos dois casos:

Exemplo 1:
Sejam os símbolos proposicionais:

- ***p*:** Luis estava presente na segunda-feira;
- ***q*:** Luis estava presente na terça-feira.

A seguinte frase: "É falso afirmar que Luis estava presente na segunda-feira ou na terça-feira" poderia ser expressa em lógica proposicional como:

$$\sim (p \vee q)$$

Logo, conhecendo o teorema de Morgan, a proposição lógica anterior seria equivalente a:

$\sim p \wedge \sim q$ que, em português, poderia ser lida da seguinte maneira: "Luis não estava presente na segunda-feira e não estava presente na terça-feira", ou: "Luis não estava presente nem na segunda-feira, nem na terça-feira".

Possivelmente a frase ficou mais clara e nos permite raciocinar melhor!

Exemplo 2:

Suponhamos os símbolos proposicionais a seguir, com os seus respectivos significados:

- ***a***: o candidato conhece inglês;
- ***b***: o candidato conhece a linguagem C#;
- ***c***: o candidato conhece a linguagem Java;
- ***d***: o candidato será pré-selecionado.

Como expressá-los em lógica proposicional?

Se o candidato sabe inglês e conhece alguma das linguagens – C# ou Java –, então será pré-selecionado:

$$a \wedge (b \vee c) \rightarrow d$$

O candidato será pré-selecionado somente se sabe inglês:

$$d \leftrightarrow a$$

As leis e os teoremas que estudamos permitem expressar a mesma ideia de formas diferentes, veja: "Se o candidato sabe inglês e conhece alguma das linguagens – C# ou Java –, então será pré-selecionado":

$$a \wedge (b \vee c) \rightarrow d$$

Ademais, utilizando a Lei da distributividade, a proposição lógica anterior poderia ser expressada como:

$$(a \wedge b) \vee (a \wedge c) \rightarrow d$$

Que poderá ser lida em português desta forma: “Se o candidato sabe inglês e conhece C#, ou se sabe inglês e conhece Java, então será pré-selecionado”.

Para encerrar esta introdução à lógica proposicional, resumamos algumas das suas características:

- Permite armazenar conhecimento;
- Suas estruturas permitem raciocinar, inferir, ou seja, fornecem mecanismos simples de inferência;
- Podemos modificar o conhecimento adicionando ou eliminando símbolos proposicionais, inserindo, modificando ou excluindo proposições lógicas etc.;
- Não permite estabelecer relacionamentos entre símbolos proposicionais além das operações básicas, nem estabelecer níveis de certeza, entre outras limitações;
- A implementação – programas para computador – dos mecanismos da lógica proposicional seria relativamente fácil.

Lógica de Predicados

A Lógica de Primeira Ordem (LPO) – ou lógica de predicados –, igualmente conhecida como Cálculo de Predicados de Primeira Ordem (CPPO) – ou simplesmente cálculo de predicados –, é usualmente considerada como uma extensão da lógica proposicional. As possibilidades da lógica de predicados são superiores às da lógica proposicional.

A lógica de predicados é também uma forma de armazenar conhecimento – da dose “regras” para raciocínio lógico ou inferência.

O elemento fundamental da lógica de predicados é o próprio predicado. Assim, podemos considerá-lo como uma função que poderá ter uma lista de parâmetros ou argumentos e que retornará um valor lógico – verdadeiro ou falso. Se comparado a funções/métodos de Java e de

outras linguagens de programação ou funções matemáticas – que podem retornar valores numéricos, *string*s etc. –, um predicado sempre retornará um valor lógico.

Em lógica de predicados podemos utilizar todos os conectivos lógicos – operadores– da lógica proposicional, mas também são permitidos os chamados quantificadores.

A quantificação é uma construção que especifica para quais indivíduos de um domínio de discurso se aplica uma proposição lógica. Os quantificadores ∀ e ∃ são utilizados para este objetivo.

∀x significa "para todo o *x*", "para qualquer *x*", "qualquer que seja *x*", ou seja, exprime fatos sobre todos os objetos do universo.

∃x significa "existe um *x*", "existe algum *x*", "existe pelo menos um *x*", ou seja, exprime fatos sobre objetos particulares.

∀ e ∃ são conhecidos respectivamente como quantificadores universal e existencial.

A utilização de quantificadores permite uma melhor descrição da situação que necessitamos representar, isto é, são elementos que ajudam no sentido de uma melhor compreensão do problema.

Exemplo 1:

Progenitor (Ana, Pedro)	Ana é a progenitora do Pedro
Progenitor (Pedro, Rosa)	Pedro é o progenitor da Rosa
...	
Progenitor (Carlos, Heitor)	
Homem (Pedro)	Pedro é homem
Homem (Luis)	
Mulher (Ana)	Ana é mulher
Mulher (Rosa)	

$$\forall x: \forall y: \text{progenitor}(x,y) \wedge \text{mulher}(x) \rightarrow \text{mãe}(x,y)$$

A proposição lógica anterior teria o seguinte significado em português: "Para todo *x*, para todo *y*, se *x* for progenitor de *y* e *x* for mulher, então, *x* é mãe de *y*".

$$\forall x: \forall y: \text{progenitor}(x,y) \wedge \text{homem}(x) \rightarrow \text{pai}(x,y)$$

A proposição lógica anterior teria este sentido em português: "Para todo *x*, para todo *y*, se *x* for progenitor de *y* e *x* for homem, então *x* é pai de *y*".

Exemplo 2:

Outras proposições lógicas mais complexas poderiam ser definidas como continuações do exemplo anterior. Vejamos, então, proposições que definem avô, irmãos, primos e tio. Assim, tente fazer uma figura para cada definição – o que lhe permitirá enxergar melhor o relacionamento definido:

$$\forall x: \forall y: \forall z: \text{progenitor}(x,y) \wedge \text{progenitor}(y,z) \wedge \text{homem}(x) \rightarrow \text{avô}(x,z)$$

$$\forall x: \forall y: \forall z: \text{progenitor}(x,y) \wedge \text{progenitor}(x,z) \rightarrow \text{irmãos}(y,z)$$

$$\forall x: \forall y: \forall a: \forall b:$$

$$\text{progenitor}(x,y) \wedge \text{progenitor}(a,b) \wedge \text{irmãos}(x,a) \rightarrow \text{primos}(y,b)$$

$$\forall x: \forall a: \forall b: \text{progenitor}(a,b) \wedge \text{irmãos}(a,x) \rightarrow \text{tio}(x,b)$$

Exemplo 3:

Com fatos – axiomas – considerando a seguinte estrutura:

populacao(nome da cidade, população da cidade)

capital(nome do país, nome da cidade)

Por exemplo:

populacao(Campinas, 2000000)

Ou com uma *string*: populacao(“Campinas”, 2000000)

populacao(SaoPaulo, 15000000)

Ou com uma *string*: populacao(“São Paulo”, 15000000)

populacao(Lima,1500000)

populacao(Brasília, 3000000)

capital(Brasil,Brasília)

capital(Peru,Lima)

...

Defina, então e utilizando a lógica de predicados:

1. cidadeGrande(uma cidade que tenha mais de 2.500.000 habitantes)
   $\forall c$: $\forall h$: populacao(c,h) Λ (h > 2500000) → cidadeGrande(c);

2. capital_importante(capitais com mais de 2.500.000 habitantes)
   $\forall c$, $\forall p$: cidadeGrande(c) Λ capital(p,c) → capital_importante(c).

Ou também desta forma:

$\forall c$, $\forall p$, $\forall h$: capital(p,c) Λ populacao(c,h) Λ h > 2500000 → capital_importante(c)

Regras de Produção e Sistemas Especialistas

Comentaremos brevemente nesta seção outra forma de armazenar conhecimento e raciocinar, modo este que foi muito utilizado na elaboração dos chamados sistemas especialistas, tratam-se das regras de produção, as quais costumam ter uma estrutura semelhante à tomada de decisão, considerando um nível de certeza ou crença – um valor entre 0 e 1,0 ou percentagem.

Exemplo 1:

Se a grama está molhada hoje de manhã, então há evidência sugestiva (0,8) de que choveu ontem à noite.

Exemplo 2:

A seguir, veremos um caso de regra no sistema especialista *Mycin* que fez bastante sucesso nas décadas de 1960 e 1970, a fim de rápido diagnóstico da meningite e de outras infecções bacterianas e prescrição do devido tratamento:

---

if the infection is meningitis and the type of infection is bacterial and

the patient has undergone surgery and
the patient has undergone neurosurgery and
the neurosurgery-time was < 2 months ago and
the patient got a ventricular-urethral-shunt

then infection = e.coli(0.8) *or klebsiella*(0.75)

A base de conhecimento de um sistema especialista poderia ser construída utilizando as regras de produção, afinal, foi a técnica mais frequente nas tentativas das décadas de 1960 e 1970.

Ademais, um sistema especialista processará a base de conhecimento elaborada com o auxílio de especialistas humanos altamente qualificados, com a participação de engenheiros de conhecimento, enquanto que um motor ou uma máquina de inferência processará os dados e,

com a ajuda dos módulos de explicação e aquisição de dados, o sistema especialista se comunicará com os diferentes participantes. Veja uma descrição geral de um sistema especialista – *expert system*:

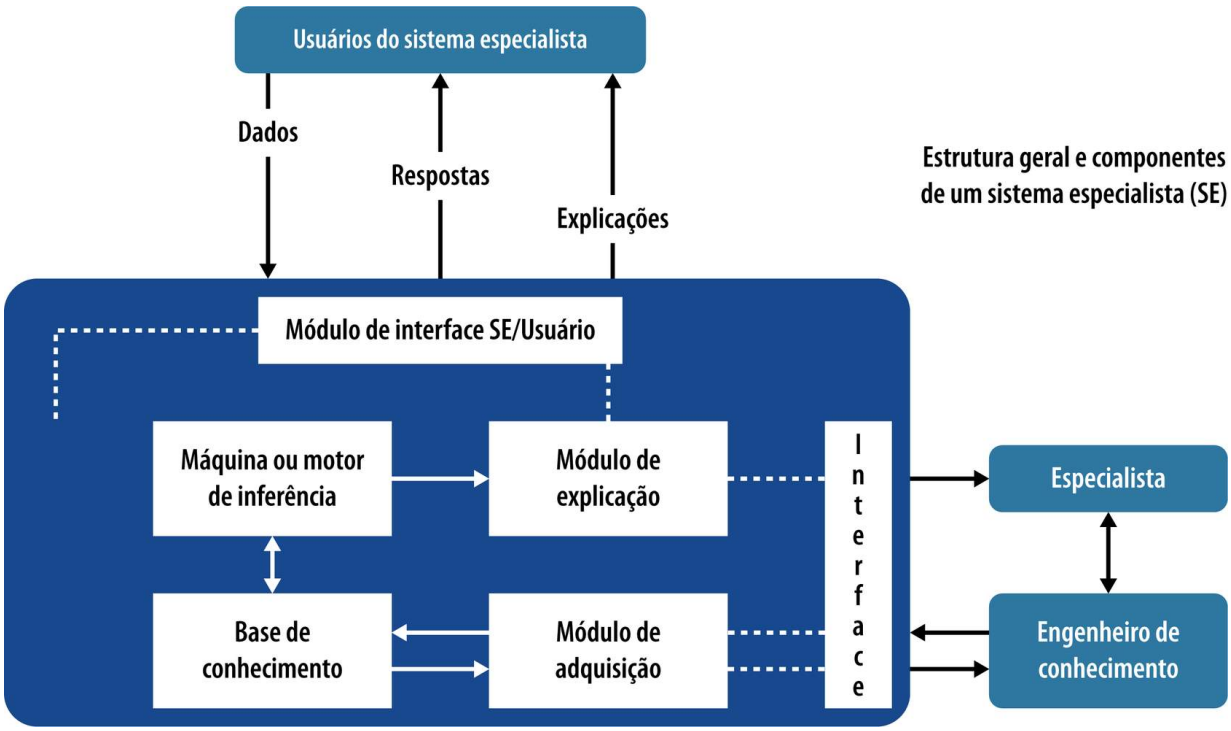

Figura 1 – Estrutura geral de um sistema especialista

Redes Neurais Artificiais

O estudo das Redes Neurais Artificiais (RNA) é importante dentro do "enfoque conexionista" da Inteligência Artificial, com ênfase no processamento paralelo, como uma forma computacional não algorítmica, sendo uma alternativa à computação algorítmica tradicional e também ao "enfoque simbólico" da IA.

As RNA se inspiram na estrutura biológica do cérebro humano, dado que este possui, aproximadamente, 1.011 neurônios, sendo que o neurônio é a célula fundamental do cérebro e suas características permitem que se comuniquem intensamente em paralelo.

Os dendritos recebem os impulsos nervosos de outros neurônios – informações de entrada –, de modo que o corpo da célula processa tais impulsos e os transmite para outros neurônios através do axônio – saída. Ademais, a sinapse é o espaço entre o axônio de um neurônio e os dendritos de outros neurônios.

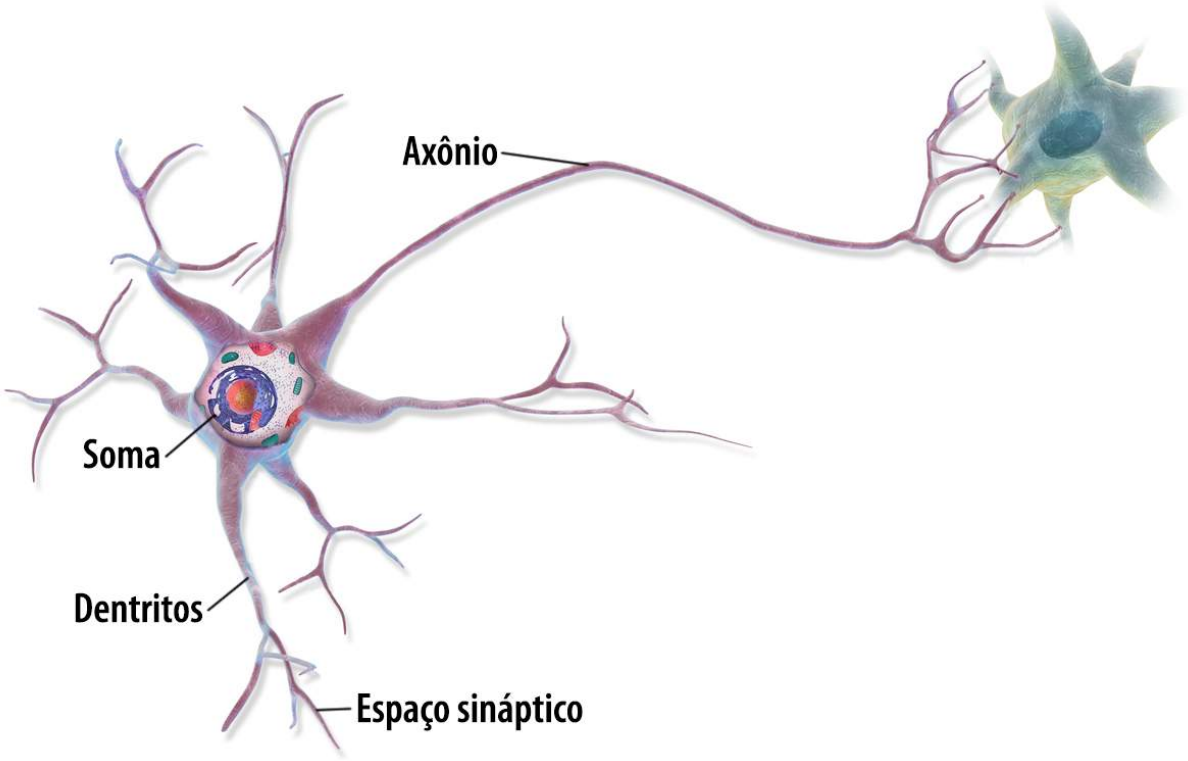

Figura 2 – O neurônio biológico
Fonte: Adaptada de Wikimedia Commons

O atrativo principal das RNA é a capacidade de aprender por meio de exemplos. Dito de outra forma, as RNA são treinadas – aprendizado – e assim poderão resolver problemas posteriormente, ou seja, apreendem.

O primeiro modelo artificial de um neurônio biológico foi proposto em 1943, por Warren McCulloch – psiquiatra e neuroanatomista – e Walter Pitts – matemático.

Nessa definição, os terminais de entrada X1, X2... Xn representam os dendritos do neurônio, cada um com um peso – positivo ou negativo – acoplado a cada entrada: pesos W1, W2... Wn.

A saída Y – axônio – será 0 ou 1, ou seja, um nodo dispara (1) ou não dispara (0). Quando a soma ponderada das entradas ultrapassa o limiar de excitação θ – threshold, letra grega theta –, o neurônio disparará. Podemos representar o neurônio biológico proposto por McCulloch e Pitts da seguinte forma:

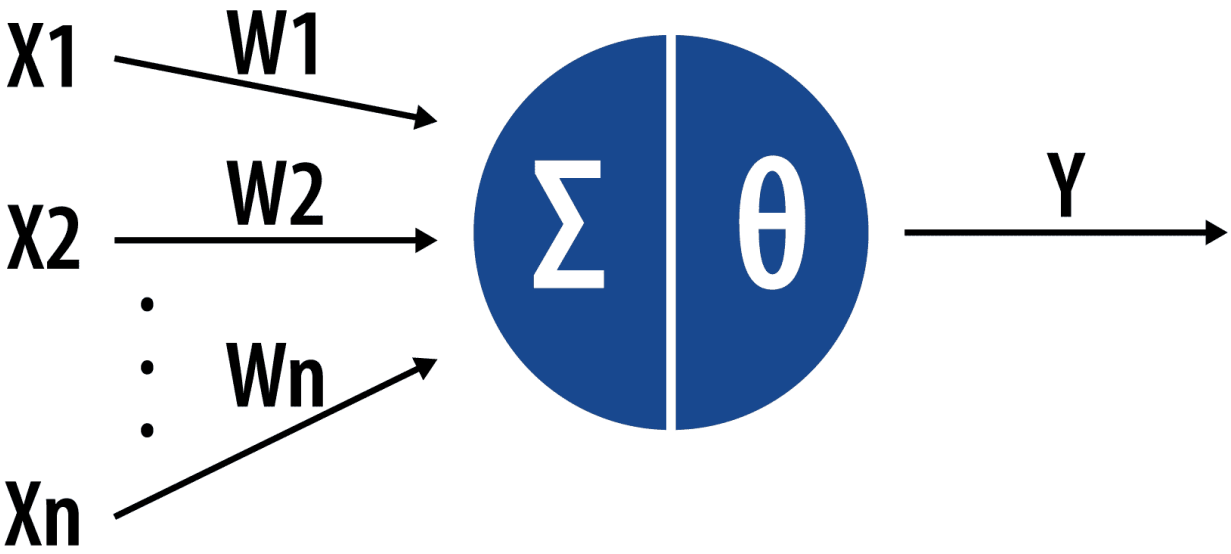

Figura 3 – Neurônio segundo McCulloch e Pitts – nodo MCP
Fonte: Adaptada de MCCULLOCH; PITTS

Os neurônios artificiais podem implementar três tipos de portas "básicas": de limiar linear – como mostrado na Figura 3, no nodo MCP –, de limiar quadrática e de limiar polinomial.

Na Figura 4, a soma ponderada se comporta como uma função quadrática, aumentando a capacidade computacional do nodo:

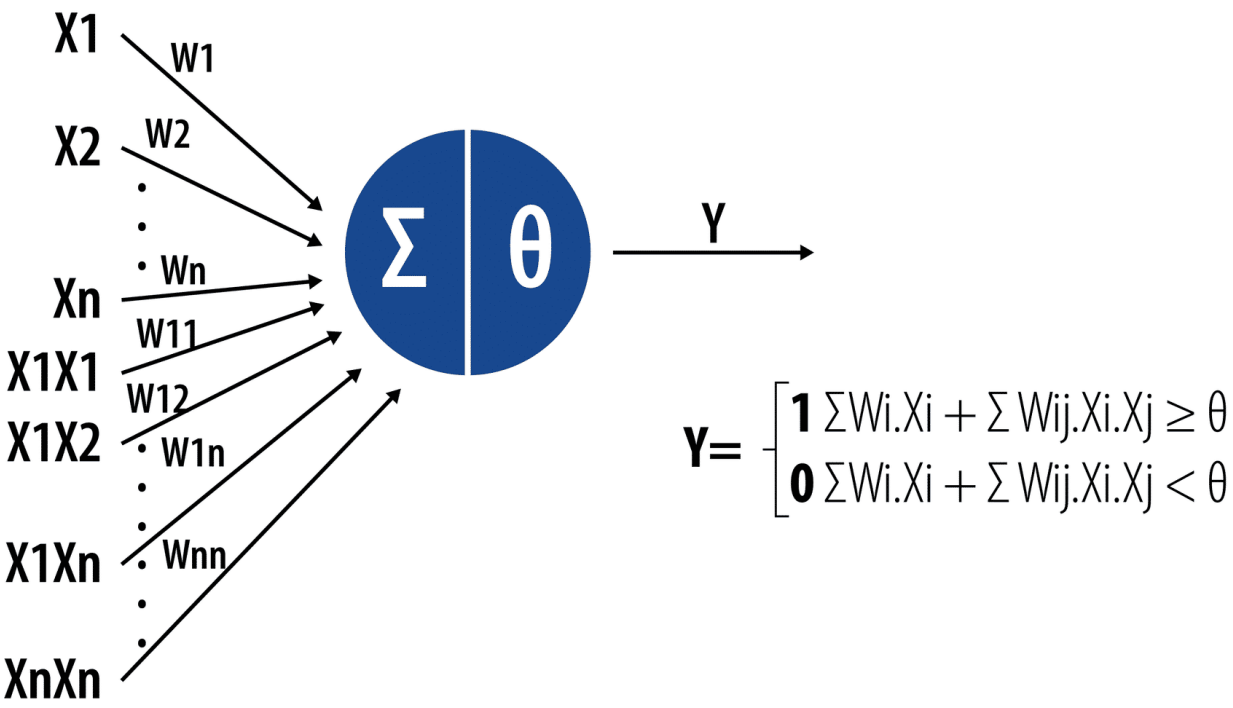

Figura 4 – Neurônio com porta de limiar quadrática
Fonte: Adaptada de MCCULLOCH; PITTS

As RNA podem ser classificadas pelo número de camadas, ou seja, em redes de camada única ou de múltiplas camadas. Em uma rede de múltiplas camadas poderão existir dois ou mais nodos – neurônios – entre uma entrada e uma saída da rede, conforme mostrado nos exemplos *b*, *c* e *d* da Figura 5.

As RNA podem ser elencadas ainda considerando as conexões entre os nodos, entre condições cíclicas ou acíclicas. Em uma RNA cíclica – com *feedback* –, a saída de um nodo na camada i poderá ser conectada a um nodo em uma camada de posição menor ou igual a *i* – tal como mostrado nos exemplos *d* e *e* da Figura 5. Ademais, a rede de Hopfield é um exemplo de condição cíclica.

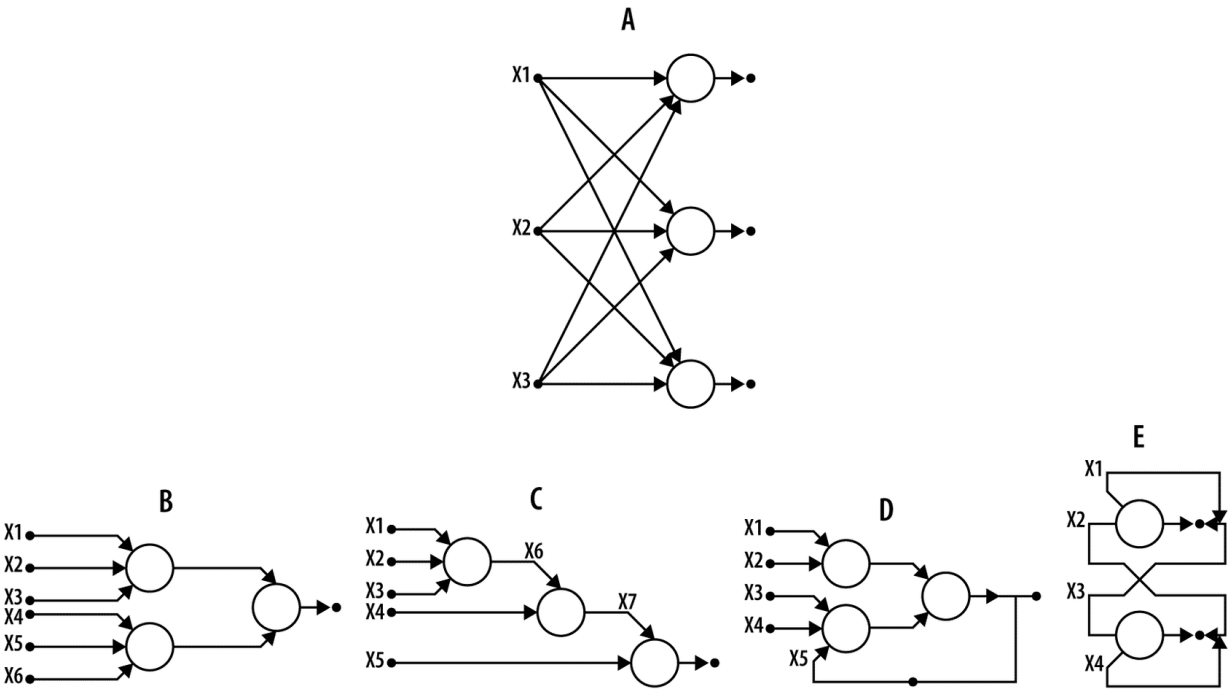

Figura 5 – Arquiteturas de rede

As redes *MultiLayer Perceptron* (*MLP*) – RNA de tipo perceptron multicamadas – apresentam vários nodos – neurônios, perceptrons – entre uma entrada e uma saída.

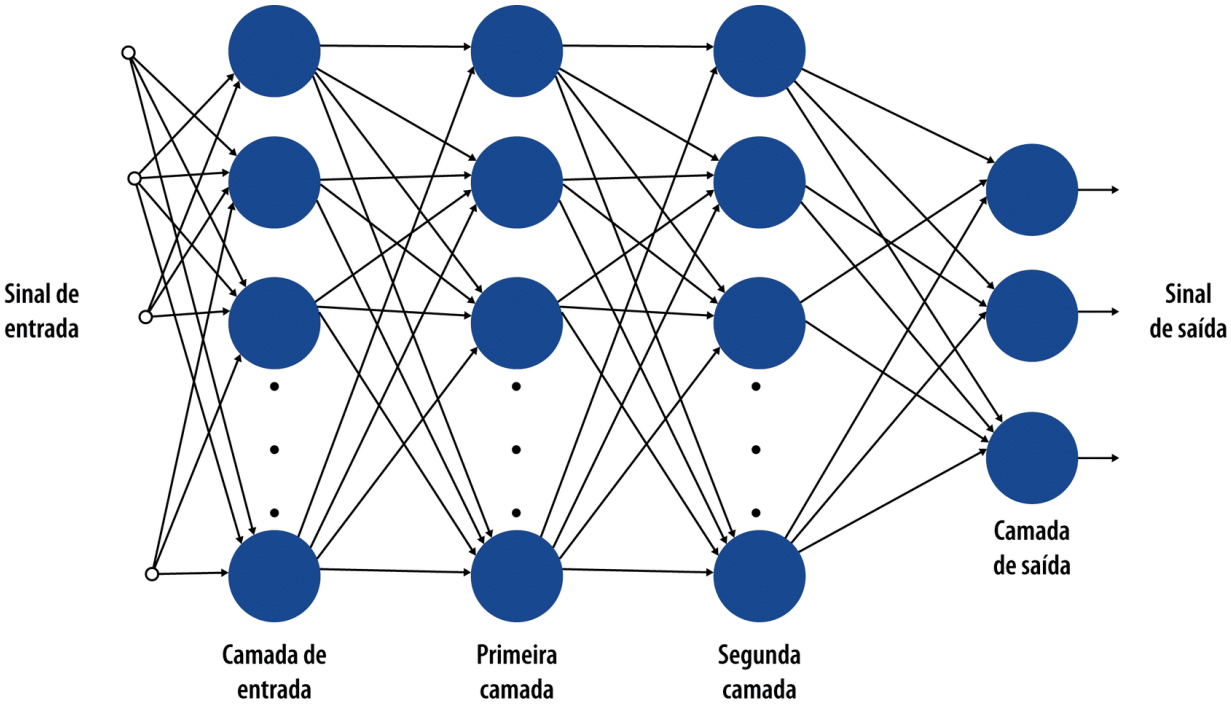

Figura 6 – Redes MLP

As redes MLP possuem um poder computacional maior que as redes com camada simples e podem resolver problemas não linearmente separáveis – mais complexos, em geral. Por este motivo, as redes de múltiplas camadas são as mais efetivas e utilizadas em aplicações com RNA.

Saiba Mais

Aprofunde-se nos temas de treinamento de redes neurais artificiais e nos algoritmos de aprendizado com RNA. Por exemplo, procure mais

informações sobre o algoritmo de aprendizado mais utilizado e referenciado nas redes MLP, o algoritmo *back-propagation*.

Lógica *Fuzzy*

Esta será a última forma de armazenar conhecimento e raciocinar que estudaremos nesta Unidade.

A lógica *fuzzy* lida com a imprecisão, com o conhecimento vago ou a falta de exatidão e sua utilização emprega como base os conjuntos difusos e a lógica difusa.

Importante!

Antes de estudarmos o que são os conjuntos difusos e a lógica *fuzzy*, precisamos observar que não se deve confundir raciocínio aproximado, vago ou impreciso, com o raciocínio incerto ou a incerteza. Afinal, o raciocínio incerto está relacionado à teoria das probabilidades, com a utilização das redes bayesianas para raciocinar em relação à incerteza.

A teoria dos conjuntos difusos foi introduzida por L. A. Zadeh, em 1965, e serve como base para a lógica difusa referenciada a partir da década de 1970. Existem alguns sinônimos importantes, a saber: lógica difusa, lógica nebulosa, lógica *fuzzy* ou até *fuzzy logic* – em inglês – e lógica borrosa – em espanhol.

Como mencionado, a lógica difusa está relacionada ao raciocínio aproximado ou impreciso, isto é, lida com a imprecisão.

A teoria de conjuntos difusos e a lógica difusa criam as bases para o **raciocínio aproximado** – mais uma vez, diferente de raciocínio probabilístico.

Em conjuntos difusos, o valor de Alto(Pedro) não é binário 0 ou 1, tal como aconteceria nas lógicas proposicional ou de predicados, sendo um valor real entre 0 e 1, como 0,52 ou 0,8, por exemplo, e que interpretaríamos como uma medida da pertinência a determinado conjunto difuso.

Veja que este significado também é diferente do resultado binário de pertinência na teoria clássica de conjuntos – onde temos apenas os resultados 0 ou 1, ou seja, pertence ou não pertence. Dito de outra forma, a função característica em conjuntos ordinários retorna um valor do conjunto discreto $\{0, 1\}$, enquanto que a função de pertinência em conjuntos nebulosos ou difusos retornará um valor real no intervalo de valores $[0, 1]$.

Assim, um estudo sobre conjuntos ordinários – ou *crisp sets*, conjuntos abruptos – seria interessante antes de estudar os conjuntos nebulosos ou difusos.

De forma geral, o valor de pertinência antes mencionado é determinado por uma função de pertinência que mostra o grau em que um elemento pertence a um conjunto difuso. A "função característica" dos conjuntos ordinários é substituída por uma "função de pertinência". Logo, um conjunto difuso (A, μA) pode ser assim descrito:

- $\mu A: S \rightarrow [0, 1]$ μA retorna valores no intervalo de 0 a 1;

- $\mu A(x) \rightarrow [0, 1]$ $\mu A(x)$ é a função de pertinência de um valor x ao conjunto difuso.

A função $\mu A(x)$ retorna o grau de pertinência de um elemento x ao conjunto difuso $(A, \mu A)$, sendo $x \in$ ao universo S e $A \subseteq S$.

Portanto, se $\mu A(x) = 0$ é porque x não pertence ao conjunto, e se $\mu A(x) = 1$ é porque x pertence totalmente. Valores intermediários entre 0 e 1 representam graus ou níveis de pertinência de x ao conjunto nebuloso $(A, \mu A)$.

Ademais, as "funções de pertinência" poderiam ter qualquer forma, mas a bibliografia cita como as mais utilizadas as funções gaussianas, triangulares e trapezoidais.

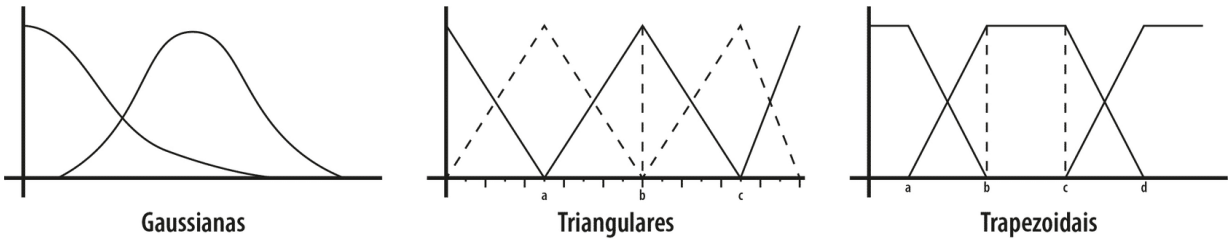

Figura 7 – Funções de pertinência mais utilizadas

Mas como utilizar os conjuntos difusos para raciocinar e tomar decisões? Vejamos alguns exemplos:

Exemplo 1:

Na Figura 8 mostramos vários conjuntos difusos com funções de pertinência que retornam um valor entre 0 e 1, dependendo do valor da temperatura corporal de uma pessoa. Ora, uma temperatura de 41 °C pertencerá, sem dúvidas, ao conjunto difuso de temperatura "muito alta".

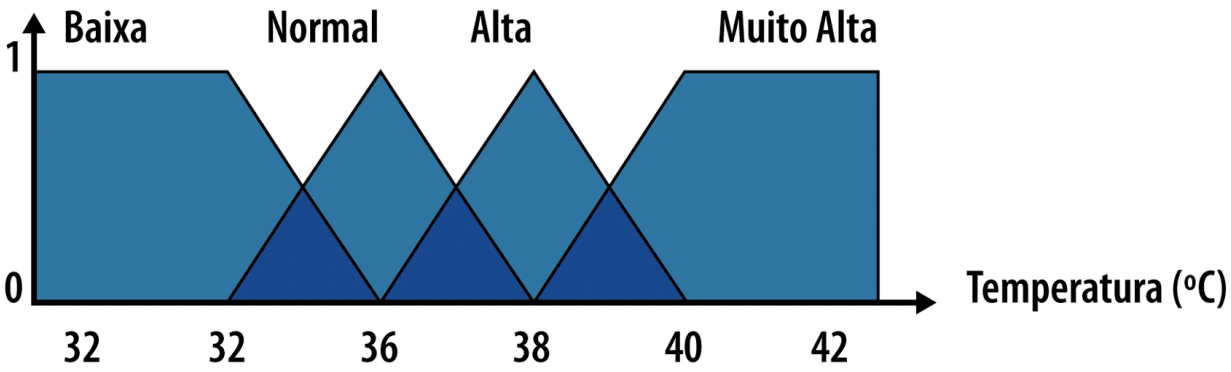

Figura 8

Exemplo 2:

Na Figura 9 mostramos três conjuntos difusos com funções de pertinência que retornam um valor entre 0 e 1, dependendo do valor da idade de uma pessoa. Tais funções de pertinência permitem determinar se uma pessoa é jovem, madura ou velha. Ou seja, para uma idade de 50 anos, temos um valor 1,0 – máximo – de pertinência ao conjunto difuso "maduro", mas para uma idade de 28 anos, a função de pertinência ao conjunto difuso "maduro" retornará um valor praticamente zero.

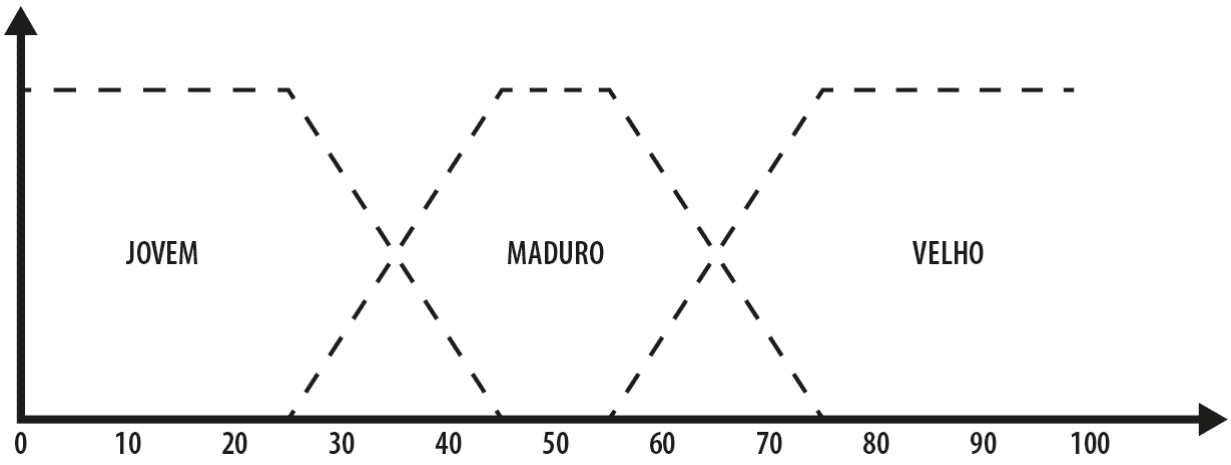

Figura 9

Um sistema de raciocínio baseado em lógica nebulosa contará com as características dessa lógica para efetuar cálculos avançados. Por exemplo, observe as propriedades a seguir, as quais apresentam algumas operações básicas com conjuntos difusos:

- **Negação:** $\mu \neg A(x) = 1 - \mu A(x)$;
- **União:** $\mu A \cup B(x) = \text{máx.}[\mu A(x), \mu B(x)]$;
- **Interseção:** $\mu A \cap B(x) = \text{min.}[\mu A(x), \mu B(x)]$.

Exemplo 3:

Sejam dois conjuntos difusos A e B, sendo $\mu A(x) = 0{,}3$ e $\mu B(x) = 0{,}6$:

- $\mu \neg A(x) = 1 - 0{,}3 = 0{,}7$;
- $\mu A \cup B(x) = 0{,}6$ que é o máx. $[\mu A(x), \mu B(x)]$;
- $\mu A \cap B(x) = 0{,}3$ que é o min. $[\mu A(x), \mu B(x)]$;

A lógica difusa é amplamente utilizada em diferentes situações práticas da indústria, em equipamentos eletrônicos, aparelhos domésticos etc. Os sistemas difusos de tipo Mamdani ou controladores difusos são os mais empregados nas aplicações práticas, por permitirem a manipulação de entradas/saídas reais, ou seja, podendo receber valores analógicos medidos com sensores, sendo transformados em valores para processamento pela lógica *fuzzy* e que poderiam retornar, novamente, uma saída analógica.

Figura 10 – Sistemas difusos

Material Complementar

Indicações para saber mais sobre os assuntos abordados nesta Unidade:

Leitura

Redes Neurais Artificiais

Como Material complementar desta Unidade, leia o artigo intitulado Redes neurais artificiais, de Emerson Alecrim, o qual traz o tema com uma abordagem simples e clara.

Clique no botão para conferir o conteúdo.

ACESSE

Lógica *Fuzzy*: Conceitos e Aplicações

Clique no botão para conferir o conteúdo.

ACESSE

Introdução à Lógica Difusa

Clique no botão para conferir o conteúdo.

ACESSE

Redes Neurais Artificiais

Clique no botão para conferir o conteúdo.

ACESSE

Referências

ALENCAR FILHO, E. de. **Iniciação à lógica matemática**. São Paulo: Nobel, 2006.

BRAGA, A. P.; LUDERMIR, T. B.; CARVALHO, A. C. P. **Redes neurais artificias:** teoria e aplicações. Rio de Janeiro: LTC, 2000.

FERREIRA, A. B. de H. **Novo dicionário eletrônico Aurélio – versão 6.0**. Curitiba, PR: Positivo, 2009.

GANASCIA, J. **Inteligência artificial**. São Paulo: Ática, 1997.

HAUSER, L. *Artificial intelligence*. ***Internet Encyclopedia of Philosophy***, [20--]. Disponível em: <http://www.iep.utm.edu/art-inte>. Acesso em: 30/10/2017.

HAYKIN, S. **Redes neurais** – princípios e prática. 2. ed. Porto Alegre, RS: Bookman, 2001.

LUGER, G. F. **Inteligência artificial**. 6. ed. São Paulo: Pearson Education do Brasil, 2013.

RICH, E.; KNIGHT, K. **Inteligência artificial**. 2. ed. São Paulo: McGraw-Hill do Brasil, 1994.

RIGNEL, D. G. de S.; CHENCI, G. P.; LUCAS, C. A. Uma introdução à lógica *fuzzy*. **Revista Eletrônica de Sistemas de Informação e Gestão Tecnológica**, v. 1, n. 1, 2011. Disponível em: <http://www.logicafuzzy.com.br/wp-content/uploads/2013/04/uma_introducao_a_logica_fuzzy.pdf>. Acesso em: 8/11/2017.

ROSA, J. L. G. **Fundamentos da inteligência artificial**. Rio de Janeiro: LTC, 2011.

RUSSELL, S. J.; NORVIG, P. **Inteligência artificial:** referência completa para cursos de Computação. 2. ed. Rio de Janeiro: Elsevier, 2004.

WINSTON, P. H. **Inteligência artificial**. Rio de Janeiro: LTC, 1988.