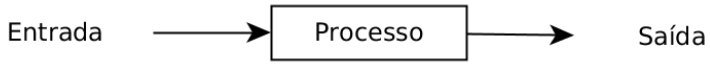
1 Fundamentação Teórica

Neste capítulo, são abordados os temas necessários para a compreensão do trabalho desenvolvido. Na Seção 1.1 é apresentada uma introdução aos sistemas de controle, na 1.2, é discutido o que é uma Rede Neural Artificial (RNA), na 1.3 se trata da lógica fuzzy e, por fim, na Seção 1.4 é apresentada a rede neuro-fuzzy.

1.1 Sistemas de Controle

Um sistema de controle é, segundo Dorf (2011, p. 2), uma interconexão de componentes formando uma configuração de sistema que vai fornecer uma resposta desejada. Todo sistema de controle tem, como objetivo, atuar sobre um determinado processo, o qual pode ser representado por um bloco que representa a relação entre entrada e saída do sistema, como mostrado na Figura 1.

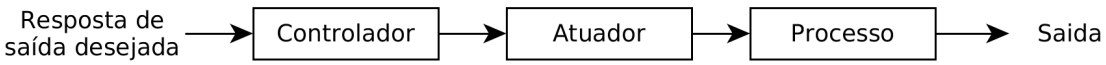
Figura 1 – Diagrama representando um processo



Fonte: Adaptado de Dorf (2011, p. 2)

O controle de um processo pode assumir duas formas distintas: malha aberta ou malha fechada, sendo que um sistema de controle em malha aberta é composto, além do processo, por um controlador e um atuador para se obter a resposta desejada sem o uso de realimentação (DORF, 2011, p. 2). Um exemplo de sistema deste tipo é mostrado na Figura 2.

Figura 2 – Diagrama representando um sistema de controle em malha aberta

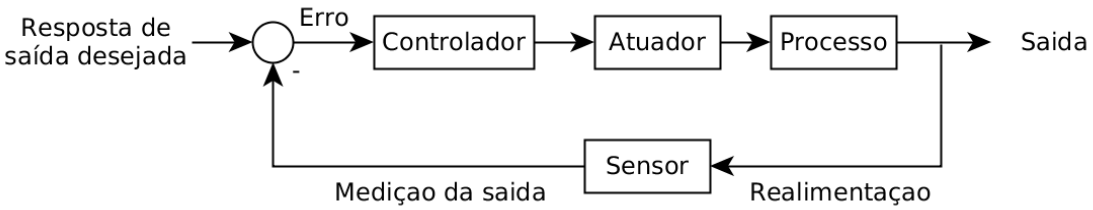


Fonte: Adaptado de Dorf (2011, p. 2)

Em oposição a um sistema de controle em malha aberta, um em malha fechada incorpora, além dos componentes que aquele inclui, uma medição dos estados atuais para serem comparados com os valores desejados para o processo. Um exemplo de um sistema de controle em malha fechada simples é mostrado na Figura 3. Os sistemas em malha fechada apresentam várias vantagens sobre os em malha aberta como, por exemplo, a capacidade de rejeitar distúrbios externos e melhorar a atenuação de ruídos

nas medições, elementos estes que são inevitáveis em aplicações no mundo real (DORF, 2011, p. 3).

Figura 3 – Diagrama representando um sistema de controle em malha fechada



Fonte: Adaptado de Dorf (2011, p. 3)

Como já foi visto, um processo representa a relação entre a entrada e a saída de um sistema sendo que há diferentes formas de fazê-lo. Uma forma de representar sistemas contínuos e é utilizando o espaço de estados.

1.1.1 Espaço de Estados

A representação de um sistema dinâmico linear no espaço de estados descreve um sistema a partir das seguintes equações :

$$\dot{x} = Ax + Bu$$
$$y = Cx + Du$$

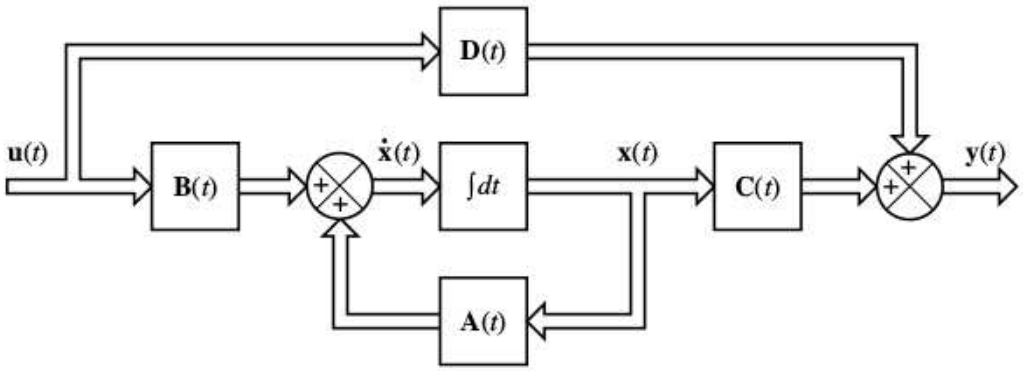
em que A é a matriz de estados, B a matriz de entrada, C a matriz de saída, D a matriz de transmissão direta, x é o vetor de variáveis de estado, u o vetor de entradas e y o vetor de saídas. A Figura 4 exibe um diagrama de blocos para a representação em espaço de estados definida.

1.2 Redes Neurais Artificiais

Redes Neurais Artificiais (RNAs) são modelos computacionais bioinspirados no sistema neurológico humano. A motivação para o desenvolvimento e uso destes modelos é a grande complexidade do cérebro humano, definido por Haykin (1998) como um computador altamente complexo, não linear e paralelo. O autor ainda define uma RNA como"*a machine that is designed to* model *the way in which the brain performs a particular task or function of interest*"[1]. Seguindo o modelo biológico do cérebro humano,

[1] "uma máquina que é desenvolvida para *modelar* a forma como o cérebro desempenha uma tarefa específica ou função de interesse" (tradução nossa).

Figura 4 – Diagrama de blocos de um sistema linear e contínuo tempo representado no espaço de estados

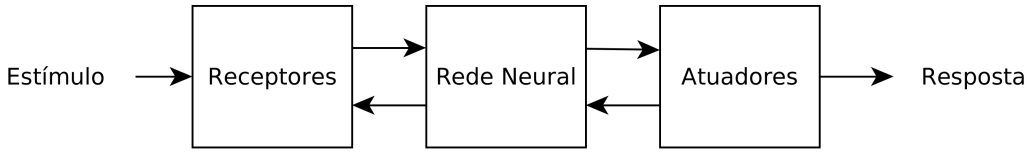


Fonte: Ogata (2010, p. 32)

uma RNA é composta por neurônios artificiais e pelas interações existentes entre estes neurônios: as sinapses.

A Figura 5 ilustra, em um diagrama de blocos, o sistema nervoso como um sistema de três estágios. A Rede Neural representa o cérebro em si, que recebe informações continuamente, as percebe e toma as decisões apropriadas para cada uma delas (HAYKIN, 1998, p. 24).

Figura 5 – Representação em digrama de blocos do sistema nervoso



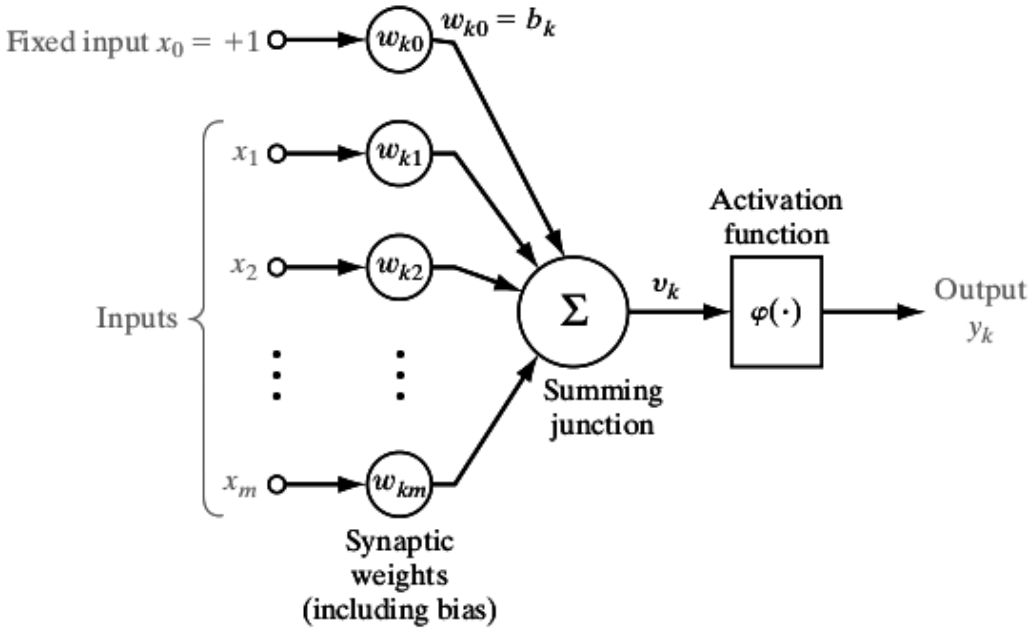
Fonte: Adaptado de Haykin (1998, p. 28)

Toda esta comunicação se dá a partir de sinapses, que são unidades estruturais e funcionais que mediam a interação entre neurônios. Seu funcionamento simplificado é o seguinte: o processo pré-sináptico libera uma substância transmissora que é difundida através da junção sináptica entre neurônios e então age sobre o processo pós-sináptico. Então, a sinapse converte o sinal elétrico pré-sinápitico em um sinal químico e, por fim, de volta a um sinal elétrico pós-sináptico (HAYKIN, 1998, p. 28). É por meio deste processo de comunicação entre neurônios que adquirimos novos conhecimentos e relacionamos estímulos a respostas. É também devido a ele que podemos fazer associações diversas com acontecimentos no passado, evento que chamamos de *memória*. O imenso poder que os neurônios possuem inspirou modelagens computacionais capazes de atuar em situações em que se deseja obter respostas adequadas a diferentes

estímulos, e em outras relacionadas à memória e aprendizado. Um modelo neural comumente aplicado na literatura em problemas relacionados às RNAs é o perceptron.

Um perceptron é um modelo computacional de um neurônio não linear e é ilustrado na Figura 6, em que y_k é a saída do sistema obtido após o processamento neural relacionado às entradas x_i, cada qual contribuindo com um peso w_{ki} para a junção de soma representada pelo bloco $\sum$. O processamento neural envolve ainda a função de ativação ϕ que, de acordo com o valor obtido na junção de soma, define o valor da saída y_k. Se o valor v_k for maior que um limiar pré-determinado, a saída do sistema é ativada e terá o valor 1, caso contrário assumirá o valor 0, simulando assim o processo de transmissão ou não de impulsos elétricos que ocorrem nos neurônios biológicos.

Figura 6 – Modelo não linear de um neurônio



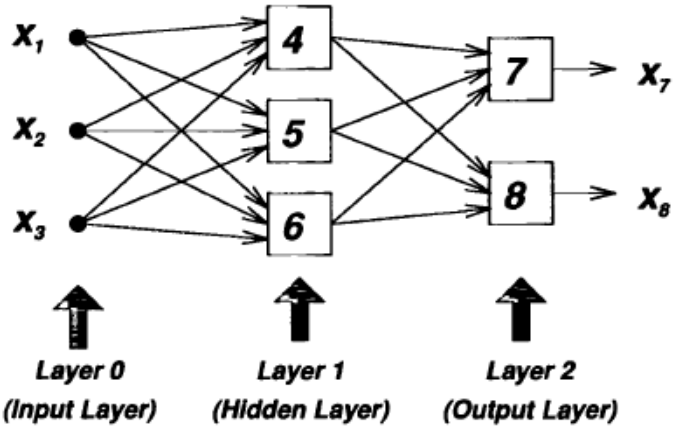
Fonte: Adaptado de Haykin (1998, p. 33)

O modelo simples de um neurônio representado por perceptrons permite a simulação das já definidas sinapses, possibilitando a criação de RNAs complexas formadas por múltiplos neurônios, organizados em cadeias, como é mostrado na Figura 7. Neste exemplo, x_1, x_2 e x_3 são as entradas do sistema; os blocos 4, 5 e 6 representam, cada qual, um neurônio numa camada escondida; os blocos 7 e 8 representam os dois neurônios que compõem a camada de saída; e, por fim, x_7 e x_8 são as saídas de RNA. Redes como essa, que possuem mais de uma camada de neurônios, são denominadas *Multilayer Perceptron* (MLP[2]). As diferentes combinações que se podem obter distribuindo os

[2] Perceptron Multicamada (tradução nossa).

neurônios de uma RNA de diferentes maneiras fazem com que estas redes possam ser utilizadas em aplicações diversas relacionadas à IA e IC.

Figura 7 – Modelo de um *MLP*



Fonte: Jang et al. (1997, p. 205)

A principal característica de uma rede neural artificial que a torna interessante para aplicações computacionais é a sua capacidade de aprendizado. O procedimento usado para efetuar o processo de aprendizado é chamado *algoritmo de aprendizado*, cuja função é modificar os pesos sinápticos da rede de forma ordenada para atingir um objetivo desejado (HAYKIN, 1998, p. 24). É a partir deste aprendizado que as RNAs alcançam uma ótima taxa de generalização, fazendo delas fortes aliadas, por exemplo, para reconhecimento de padrões. Controladores que utilizam técnicas relacionadas a RNAs se beneficiam justamente destas capacidades de aprendizado e generalização conferidas por elas.

O controlador desenvolvido neste trabalho utiliza RNAs com treinamento supervisionado, que são treinadas a partir de um conjunto de entradas mapeadas no valor de suas respectivas saídas, resultando na obtenção de um conjunto de retas com parâmetros ajustados de acordo com os dados do treinamento. São as retas obtidas após este treinamento que conferem à rede o poder de generalização, permitindo que novas entradas sejam mapeadas para saídas que condizem com o cenário em questão.

Além disto, como o controlador projetado é do tipo Neuro-*Fuzzy*, as retas ajustadas após o treinamento se enquadram num grupo especial correspondendo cada qual a uma função de pertinência de conjuntos *Fuzzy*, assunto este que é abordado na seção seguinte.

1.3 Lógica Fuzzy

A lógica *fuzzy* é uma alternativa à lógica convencional[3], que permite uma abordagem diferente relacionada à pertinência de elementos a conjuntos implementando a possibilidade de se obter uma pertinência parcial para a definição desses conjuntos. A principal aplicação da lógica *fuzzy* são os sistemas de inferência *fuzzy*, tema que será abordado na Seção 1.3.4. As Seções 1.3.1, 1.3.2 e 1.3.3 referentes a conjuntos, regras e raciocínio *fuzzy* respectivamente tratam dos diferentes componentes utilizados nestes sistemas.

1.3.1 Conjuntos *Fuzzy*

Os conjunto *fuzzy* são os componentes elementares dos sistema de inferência *fuzzy* e se contrapõem àqueles definidos pela lógica tradicional, que restringe, aos valores 0 e 1, o grau de pertinência μ de um elemento u a um conjunto A. Na lógica tradicional, este grau é definido da seguinte forma:

$$\mu_A(u) = 1, \text{ se } u \text{ é um elemento do conjunto } A, \text{ e}$$
$$\mu_A(u) = 0, \text{ se } u \text{ não é um elemento do conjunto } A$$

Com isto, ou um elemento pertence a um conjunto ou não. Já os conjuntos *fuzzy* permitem um grau de flexibilidade acerca do grau pertinência de cada elemento ao conjunto, sendo este grau definido por uma *função de pertinência*. A definição formal de conjuntos *fuzzy* e funções de pertinência é dada por Jang et al. (1997, p. 14) como:

$$A = \{(x,\mu_A(x))|x \in X\}$$

Nesta definição, um conjunto *fuzzy* A é composto pelos pares $(x,\mu_A(x))$ de cada elemento x pertencente a um conjunto X, em que x é o elemento em si e $\mu_A(x)$ é o grau de pertinência de x ao conjunto A, que pode assumir qualquer valor entre 0 e 1, em que 0 representa a não-pertinência total do elemento ao conjunto e 1 representa a pertinência total a ele.

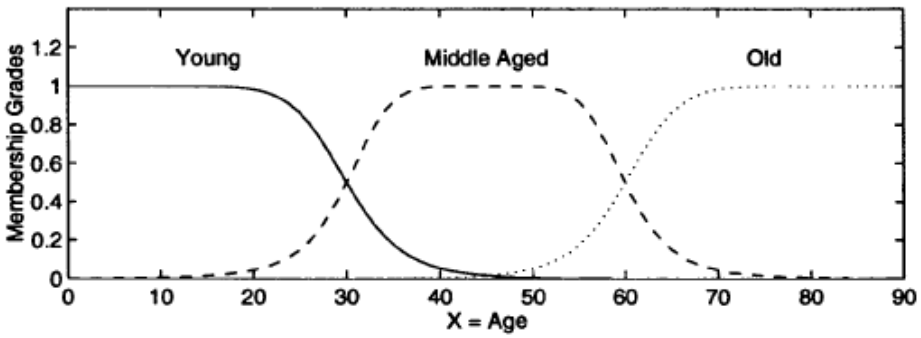
Como se pode perceber, a definição de um conjunto *fuzzy* é uma simples extensão da referente a um conjunto clássico, no qual a função de pertinência (ou função característica) apenas pode assumir os valores 0 e 1. Se uma função de pertinência $\mu_A(x)$ de um conjunto *fuzzy* A é restrita a assumir os valores 0 ou 1, então ele é reduzido a um conjunto clássico (JANG et al., 1997, p. 14).

Para melhor exemplificar a definição de conjuntos *fuzzy*, tome como exemplo a Figura 8. Neste caso, a variável idade (*age*) é dividida em três subconjuntos *fuzzy*, *young*, *middle aged* e *old*[4], representados pelas funções de pertinência que os descrevem. Cada

[3] lógica aristotélica.
[4] jovem, meia idade e velho, respectivamente (tradução nossa).

idade não precisa possuir necessariamente grau de pertinência 1 para algum conjunto e pode ainda pertencer simultaneamente a mais de um. A idade 30, por exemplo pertence ao conjunto jovem com grau de pertinência de 0,5 e também pertence ao conjunto meia idade com o mesmo grau.

Figura 8 – Funções de pertinência representando três conjuntos *fuzzy* para a variável idade



Fonte: Jang et al. (1997, p. 17)

Além dos conjuntos *fuzzy*, há outro aspecto fundamental para o funcionamento desta lógica alternativa e poderosa: as regras *fuzzy*.

1.3.2 Regras *Fuzzy*

As regras *fuzzy* são os componentes de um sistema de inferência *fuzzy* responsáveis por definir as relações entre as entradas do sistema e suas saídas e assumem a forma

$$\text{se } x \text{ é } A \text{ então } y \text{ é } B$$

sendo "x é A" denominada premissa e "y é B", consequente da regra em que x e y são *variáveis linguísticas* de entrada e saída respectivamente e A e B são os valores que elas assumem, representados por *termos linguísticos*.

Uma variável linguística, segundo Jang et al. (1997, p. 54), é caracterizada por uma quíntupla $(x,T(x),X,G,M)$ em que x é o nome da variável; $T(x)$ é o conjunto de termos de x, que é o conjunto de seus valores linguísticos ou termos linguísticos; X é o universo de discurso; G é uma regra sintática que gera os termos em $T(x)$; e M é uma regra semântica que associa a cada termo linguístico A seu respectivo $M(A)$, em que $M(A)$ denota um conjunto *fuzzy* em X.

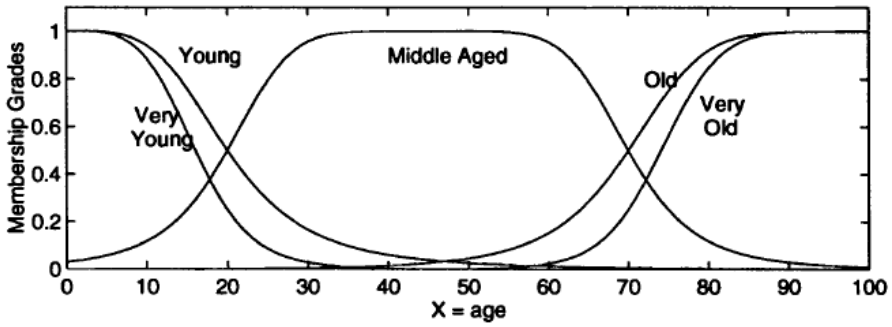
Para facilitar a compreensão das definições relacionadas às variáveis linguísticas, o seguinte exemplo foi dado (JANG et al., 1997, p. 55): se *idade* é interpretado como uma

variável linguística, então seu conjunto de termos $T(idade)$ poderia ser dado por:

T(idade) = { *novo, não-novo, muito novo, não muito-novo, . . . ,*
meia-idade, não de meia-idade, . . . ,
velho, não-velho, muito velho, mais ou menos velho, não muito velho, . . . ,
não muito novo e não muito velho, . . . },

Em que cada termo em $T(idade)$ é caracterizado por um conjunto *fuzzy* de um universo de discurso $X = [0,100]$, como mostrado na Figura 9. Geralmente, se diz "idade é jovem" para denotar a atribuição do valor linguístico jovem à variável linguística idade. A regra sintática se refere à forma como os valores linguísticos no conjunto de termos $T(idade)$ são gerados. A regra semântica define a função de pertinência de cada valor linguístico do conjunto de termos. A Figura 9 mostra algumas das funções de pertinência típicas para a variável linguística idade.

Figura 9 – Exemplo de função de pertinência do conjunto de termos T(idade)



Fonte: Jang et al. (1997, p. 55)

Diversas regras *fuzzy* fazem parte do nosso cotidiano. Exemplos possuindo a variável linguística *idade* como variável linguística de entrada e saída respectivamente incluem:

se *idade* é *jovem* então *energia* é *alta*
se *sabedoria* é *grande* então *idade* é *velha*

Um outro componente dos sistemas de inferência *fuzzy* agrega muito valor ao uso de regras *fuzzy*, permitindo a aplicação aproximada delas: o raciocínio *fuzzy*.

1.3.3 Raciocínio *Fuzzy*

O processo de raciocínio *fuzzy*, também conhecido como raciocínio aproximado, é um procedimento de inferência que deriva conclusões de um conjunto de regras

fuzzy se-então como fatos conhecidos (JANG et al., 1997, p. 62) e é a partir dele que se pode fazer uma generalização a partir da regra básica na lógica tradicional com duas variáveis, denominada *modus ponens*.

De acordo com a regra *modus ponens*, podemos inferir a verdade da proposição B a partir da verdade de A e a implicação $A \rightarrow B$ (se A então B). Por exemplo, se A é identifica por "o tomate é vermelho" e B por "o tomate está maduro", então se é verdade que "o tomate é vermelho", é também verdade que "o tomate está maduro".

Contudo, o raciocínio humano emprega constantemente o modus ponens em uma maneira aproximada. Por exemplo, usando a mesma regra de implicação "se o tomate é vermelho, então ele está maduro", e sabemos que o "o tomate está mais ou menos vermelho" (A'), então podemos inferir que "o tomate está mais ou menos maduro" (B'), em que A' é próximo de A e B' é próximo de B. Quando A, B, A' e B' são conjuntos *fuzzy* do universo adequado, o procedimento de inferência descrito é chamado raciocínio aproximado ou raciocínio *fuzzy*, podendo ser também chamado *modus ponens* generalizado (GMP[5]) (JANG et al., 1997, p. 65).

A definição formal do raciocínio aproximado (raciocínio *fuzzy*) é dada por Jang et al. (1997, p. 65) como: sejam A, A', e B conjuntos *fuzzy* de X, X e Y respectivamente; assuma que a implicação *fuzzy* $A \rightarrow B$ é expressa como uma relação R em $X \times Y$. Então, o conjunto *fuzzy* B induzido por "x é A" e a regra *fuzzy* "se x é A então y é B" é definida por:

$$\begin{aligned} \mu_{B'}(y) &= \max_x \min[\mu_{A'}, \mu_R(x,y)] \\ &= \vee_x[\mu_{A'}(x) \wedge \mu_R(x,y)], \end{aligned}$$

ou, equivalentemente,

$$B' = A' \circ R = A' \circ (A \rightarrow B).$$

Assim, podemos usar o procedimento de inferência do raciocínio *fuzzy* para derivar conclusões, dado que a implicação *fuzzy* $A \rightarrow B$ é definida como uma relação *fuzzy* binária apropriada.

Uma vez definidos os conjuntos, regras e raciocínio *fuzzy*, pode-se mostrar como eles são utilizados em conjunto pelos sistemas de inferência *fuzzy*.

1.3.4 Sistema de Inferência *Fuzzy*

Um sistema de inferência *fuzzy* (FIS, do nome em inglês *Fuzzy Inference System*) é uma ferramenta computacional popular e poderosa baseada nos conceitos de teoria de conjuntos *fuzzy*, regras *fuzzy* e raciocínio *fuzzy*. Há uma grande variedade de aplicações

[5] Do inglês, *Generalized Modus Ponens.*

para sistemas de inferências *fuzzy* tais como controle automatizado, classificação de dados, análise de decisão, sistemas especialistas, predição de séries temporais, robótica e reconhecimento de padrões (JANG et al., 1997, p. 73).

A estrutura básica de um FIS consiste de três componentes conceituais: uma base de regras, que contém uma seleção de regras *fuzzy*; um dicionário (ou base de dados), que define as funções de pertinência usadas nas regras *fuzzy*; e o mecanismo de raciocínio, que realiza o procedimento de inferência (geralmente o raciocínio *fuzzy*) sobre as regras e fatos dados para obter uma saída ou conclusão razoável (JANG et al., 1997, p. 73). Sistemas de inferência *fuzzy* diferentes implementam estruturas ligeiramente diferentes entre si, havendo, dois tipos principais de sistemas que podem ser aplicados aos mais diversos casos.

Os dois principais tipos de FIS são os Mamdani e Sugeno e a diferença entre eles reside basicamente no consequente de suas regras *fuzzy* (JANG et al., 1997, p. 74).

No sistema de inferência *fuzzy* Mamdani, o consequente das regras *se-então* são conjuntos *fuzzy*. Desta forma, um sistema de inferências *fuzzy* Mamdani possui regras do seguinte tipo para um sistema com duas entradas e uma saída:

$$\text{se } x \text{ é } A \text{ e } y \text{ é } B, \text{ então } z \text{ é } C$$

em que x, y e z são variáveis linguísticas nos universos de discurso X, Y e Z, respectivamente e A, B e C são valores (termos) linguísticos. Com isto, um exemplo de sistema de inferências *fuzzy* Mamdani é formado pelas seguintes regras *fuzzy*:

$$\begin{cases} \text{Se } X \text{é pequeno e } Y \text{é pequeno então } Z \text{ é muito negativo} \\ \text{Se } X \text{é pequeno e } Y \text{é grande então } Z \text{ é pouco negativo} \\ \text{Se } X \text{é grande e } Y \text{é pequeno então } Z \text{ é pouco positivo} \\ \text{Se } X \text{é grande e } Y \text{é grande então } Z \text{ é muito positivo} \end{cases}$$

Já no sistema de inferência *fuzzy* Sugeno, também conhecido como TSK (Takagi-Sugeno-Kang), o consequente é uma função envolvendo as variáveis de entrada. Com isto, uma regra *fuzzy* típica de um modelo *fuzzy* Sugeno apresenta a seguinte forma:

$$\text{se } x \text{ é } A \text{ e } y \text{ é } B, \text{ então } z = f(x,y),$$

em que A e B são conjuntos *fuzzy* do antecedente, enquanto $z = f(x,y)$ é uma função no consequente. Geralmente, $f(x,y)$ é uma função polinomial sobre as variáveis x e y, mas pode ser qualquer função, contanto que seja capaz de descrever apropriadamente a saída do modelo dentro da região *fuzzy* especificada pelo antecedente da regra. Um

exemplo de sistema de inferências *fuzzy* TSK com duas entradas e uma saída é formado pelas seguintes regras *fuzzy*:

$$\begin{cases} \text{Se } X \text{é pequeno e } Y \text{é pequeno então } z = -x + y + 1 \\ \text{Se } X \text{é pequeno e } Y \text{é grande então } z = -y + 3 \\ \text{Se } X \text{é grande e } Y \text{é pequeno então } z = -x + 3 \\ \text{Se } X \text{é grande e } Y \text{é grande então } z = x + y + 2 \end{cases}$$

Esta propriedade dos sistemas de inferência *fuzzy* Sugeno de possuírem, como consequente, funções paramétricas que relacionam as premissas resulta em várias possibilidades para sua aplicação. Uma delas é implementada por uma técnica que alia o poder das RNAs ao do FIS: o sistema de inferência neuro-*fuzzy*.

1.4 Redes Neuro-*Fuzzy*

Aliando o poder das RNAs, que reconhecem padrões e se adaptam para lidar com mudanças no ambiente, aos sistemas de inferência *fuzzy*, que incorporam o conhecimento humano e executam inferências e tomadas de decisão, surgiram os sistemas de inferência neuro-*fuzzy* (ANFIS, acrônimo para *Adaptive Neuro-Fuzzy Inference System*), uma nova ferramenta computacional poderosa amplamente utilizada em diversos contextos (JANG et al., 1997, p. 1).

1.4.1 ANFIS: *Adaptive Neuro-Fuzzy Inference Systems*

Segundo Jang et al. (1997, p. 335), do ponto de vista funcional, praticamente não há limitações para a gama de funções, no sentido matemático, de uma rede neuro-*fuzzy*, exceto pela exigência de ela ser diferenciável por partes. Devido às mínimas restrições, redes neuro-*fuzzy* podem ser empregadas diretamente em uma grande variedade de aplicações de modelagem, tomadas de decisão, processamento de sinal e controle.

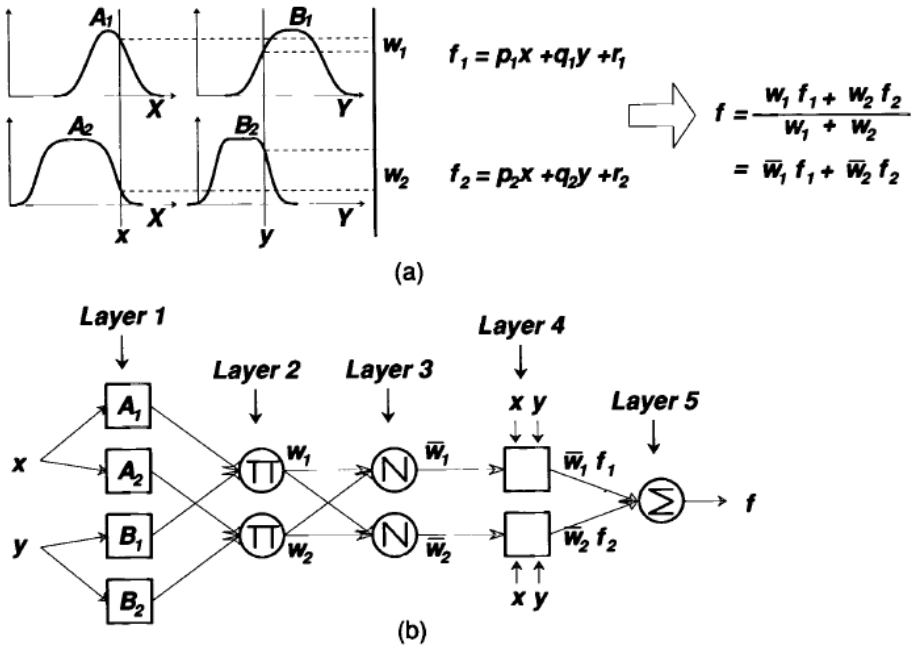
Para simplificar, a arquitetura ANFIS[6] será descrita considerando-a dotada de duas entradas x e y e uma saída z. Para um modelo *fuzzy* Sugeno de primeira ordem, um conjunto comum de regras do tipo *se-então* é o seguinte:

$$\begin{aligned} &\text{Regra 1: Se } x \text{ é } A_1 \text{ e } y \text{ é } B_1, \text{ então } f_1 = p_1 x + q_1 y + r_1, \\ &\text{Regra 2: Se } x \text{ é } A_2 \text{ e } y \text{ é } B_2, \text{ então } f_2 = p_2 x + q_2 y + r_2. \end{aligned}$$

A Figura 10 ilustra um mecanismo de raciocínio para o modelo Sugeno e a arquitetura ANFIS equivalente, em que nós de uma mesma camada têm funções similares, como descrito a seguir.

[6] Sistema de Inferência Neuro-*Fuzzy* Adaptativo (tradução nossa).

Figura 10 – Equivalência entre modelo *fuzzy* Sugeno e ANFIS; (a) Um modelo *fuzzy* Sugeno com duas entradas de primeira ordem com duas regras; (b) arquitetura ANIFS equivalente



Fonte: (JANG et al., 1997, p. 336)

Na camada 1 (*Layer 1*), todo nó i é um nó adaptativo com uma função de nó do tipo:

$$O_{1,i} = \mu_{A_i}(x), \text{ para } i = 1, 2, \text{ ou}$$
$$O_{1,i} = \mu_{B_{i-2}}(y), \text{ para } i = 3, 4$$

em que x (ou y) é a entrada para o nó i e A_i (ou B_{i-2}) é um termo linguístico (como "pequeno" ou "grande") associado a este nó. Em outras palavras, $O_{1,i}$ é o grau de pertinência de um conjunto *fuzzy* A (A_1, A_2, B_1 ou B_2) e especifica o grau em que a entrada x (ou y) satisfaz o quantificador A. Aqui, a função de pertinência A pode ser qualquer função de pertinência parametrizada apropriada. Parâmetros nesta camada são chamados parâmetros de premissa (JANG et al., 1997, p. 336).

Na camada 2 (*Layer 2*), cada nó é um nó fixo rotulado $\prod$, cuja saída é o produto de todos sinais de entrada:

$$O_{2,i} = w_i = \mu_{A_i}(x)\mu_{B_{i-2}}(y), i = 1, 2.$$

Cada nó de saída representa a força de ativação de uma regra. Em geral, qualquer outro operador norma-T[7] que desempenhe a operação *AND fuzzy* pode ser usado como função de nó nesta camada (JANG et al., 1997, p. 337).

[7] Operador de interseção (JANG et al., 1997, p. 337)

Na camada 3 (*Layer 3*), cada nó é um nó fixo rotulado N. O i-ésimo nó calcula a taxa da força de ativação da i-ésima regra pela soma da força de ativação de todas as regras:

$$O_{3,i} = \bar{w}_i = \frac{w_i}{w_1 + w_2}, i = 1, 2.$$

Por conveniência, saídas desta camada são chamadas forças de ativação normalizadas (JANG et al., 1997, p. 336).

Na camada 4 (*Layer 4*), todo nó i é um nó adaptativo com uma função nodal

$$O_{4,i} = \bar{w}_i f_i = \bar{w}_i(p_i x + q_i y + r_i),$$

em que $\bar{w}_i$ é um taxa da força de ativação normalizada da camada 3 e $\{p_i, q_i, r_i\}$ é o conjunto de parâmetros deste nó. Parâmetros nesta camada são denominados parâmetros consequentes (JANG et al., 1997, p. 336).

O único nó na camada 5 (*Layer 5*) é um nó fixo denominado $\sum$, e é responsável por computar a saída global como a soma ponderada de todos os sinais de entrada:

$$\text{saída global} = O_{5,i} = \sum_i \bar{w}_i f_i = \frac{\sum_i w_i f_i}{\sum_i w_i}$$

Com esta arquitetura, constrói-se uma rede neuro-*fuzzy* (ANFIS) que é funcionalmente equivalente a um modelo *fuzzy* Sugeno (JANG et al., 1997). Com isto, sistemas de inferência neuro-*fuzzy* podem operar, por exemplo, utilizando o treinamento supervisionado de uma RNA para ajustar o valor dos parâmetros das funções de resposta dos sistemas de inferência *fuzzy* Sugeno para uma dada operação. Um exemplo de aplicação para este tipo de rede seria o controle de sistemas não lineares.

1.4.2 Controladores *Fuzzy* e Neuro-*Fuzzy*

Um controlador lógico fuzzy (FLC)[8] é um sistema de inferência fuzzy projetado especificamente para atuar como controlador de um sistema que se deseja controlar. Este FIS portanto possui, como entradas, sinais provenientes do sistema e, como saídas, sinais de atuação sobre ele.

Com o avanço dos microprocessadores e hardware, tem se criado uma diversidade ainda maior de domínios para aplicação de controladores lógicos fuzzy, que vão de eletroeletrônicos à indústria automobilística. De fato, para sistemas complexos ou mal definidos que não são facilmente sujeitados a métodos de controle convencionais, os FLCs proveem uma alternativa viável, uma vez que eles podem capturar os aspectos qualitativos aproximados do raciocínio de um humano e o processo de tomada de

[8] Do inglês, *Fuzzy Logic Controller*.

decisão. De qualquer forma, sem a capacidade adaptativa, a performance de um FLC se baseia exclusivamente em dois fatores: a disponibilidade de humanos com expertise e as técnicas de aquisição de conhecimento para converter esta expertise humana nas apropriadas regras fuzzy *se-então* e funções de pertinência. Estes dois fatores restringem substancialmente o domínio de aplicações dos FLCs (JANG et al., 1997, p. 451).

Nesse contexto, o uso de controladores neuro-fuzzy passou a se mostrar muito interessante, tendo em vista que eles eliminam os dois fatores citados que restringem bastante o domínio de aplicações dos FLCs, uma vez que os sistemas ANFIS preenchem as lacunas deixadas pelos FIS, eliminando a necessidade de humanos com expertise sobre o sistema e encorporando técnicas de aquisição de conhecimento.

Com isto, podem-se identificar algumas propriedades particulares aos controladores ANFIS que os tornam ótimas opções para diversos casos (JANG et al., 1997, p. 458):

1. Habilidade de aprendizado;
2. Operação paralela;
3. Representação do conhecimento estruturado;
4. Melhor integração com outros métodos de controle.

Como comparativo, é importante salientar que uma RNA possui as propriedades 1 e 2, mas não as 3 e 4, que são devidas à contribuição dos sistemas de inferência fuzzy (JANG et al., 1997, p. 458). Um ANFIS, por aliar as duas técnicas, possui todas essas propriedades.

Ainda segundo o autor, há diversas formas diferentes de se projetar controladores neuro-fuzzy. A maioria deles são não lineares. Assim sendo, uma análise rigorosa para sistemas de controle neuro-fuzzy é difícil e continua sendo uma área desafiadora para outras investigações. Por outro lado, um controlador neuro-fuzzy geralmente contém um grande número de parâmetros, o que o torna mais versátil do que um controlador linear ao lidar com características não lineares de plantas. Desta forma, controladores neuro-fuzzy quase sempre superam controladores puramente lineares se devidamente projetados.

Referências

DORF, R. **Modern control systems**. 12. ed. Upper Saddle River, N.J: Pearson Prentice Hall, 2011. ISBN 978-0136024583. Citado 2 vezes nas páginas 1 e 2.

HAYKIN, S. **Neural Networks: A Comprehensive Foundation**. NJ, USA: Prentice Hall PTR Upper Saddle River, 1998. ISBN 0132733501. Citado 4 vezes nas páginas 2, 3, 4 e 5.

JANG, J.; SUN, C.; MIZUTANI, E. **Neuro-fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence**. [S.l.]: Prentice Hall, 1997. ISBN 9780132610667. Citado 10 vezes nas páginas 5, 6, 7, 8, 9, 10, 11, 12, 13 e 14.

OGATA, K. **Modern Control Engineering**. 5. ed. [S.l.]: Prentice Hall, 2010. ISBN 978-0136156734. Citado na página 3.

Apêndices

APÊNDICE A – Código para Criação de Modelo Neuro-Fuzzy para Altitude e Definição de Dados para Treinamento

```
% le arquivo fis referente ao controle de altitude
fismat = readfis('fis_altitude.fis');

% define numero de casos a serem avaliados (treinamento + teste)
n = 300;
% define conjunto de n entradas aleatorias para o sistema fuzzy
% respeitando o range de cada entrada
input = zeros(n,2);
for i=1:n
    z_value = rand * 2 - 1;
    z_dot_value = rand * 10 - 5;
    input(i,:) = [ z_value z_dot_value ];
end

% avalia resposta fuzzy para cada entrada
output= evalfis(input,fismat);

% define data como vetor relacionando cada conjunto de entradas ...
   a saida
% - obtida pelo sistema fuzzy
data = [];
for i=1:n
   data(i,:) = [ input(i,:) output(i) ];
end

% define que 2/3 dos dados obtidos serao usasdos para treinamento
% e 1/3 sera usado para teste da rede
train = data(1:2*n/3,:);     % dados para treinamento
test = data(2*n/3+1:n,:);    % dados para validacao do sistema ...
   treinado

% gera modelo fuzzy Sugeno a partir do Mamdani modelado
sugFIS = mam2sug(fismat);
% salva modelo Sugeno em disco com o nome fis_altitude_neuro.fis
writefis(sugFIS, 'fis_altitude_neuro.fis');
```

APÊNDICE B – Código para Criação de Modelo Neuro-Fuzzy para Atitude e Definição de Dados para Treinamento

```
% le arquivo fis referente ao controle de atitude
fismat = readfis('fis_atitude.fis');

% define numero de casos a serem avaliados (treinamento + teste)
n = 300;
% define conjunto de n entradas aleatorias para o sistema fuzzy
% respeitando o range de cada entrada
input = zeros(n,2);
for i=1:n
    phi_value = rand * 4 - 2;
    phi_dot_value = rand * 3 - 1.5;
    input(i,:) = [ phi_value phi_dot_value ];
end

% avalia resposta fuzzy para cada entrada
output= evalfis(input,fismat);

% define data como vetor relacionando cada conjunto de entradas ...
   a saÃda
% obtida pelo sistema fuzzy
data = [];
for i=1:n
   data(i,:) = [ input(i,:) output(i) ];
end

% define que 2/3 dos dados obtidos serao usasdos para treinamento
% e 1/3 sera usado para teste da rede
train = data(1:2*n/3,:);     % dados para treinamento
test = data(2*n/3+1:n,:);    % dados para validaÃ§Ã£o do sistema ...
   treinado

% gera modelo fuzzy Sugeno a partir do Mamdani modelado
sugFIS = mam2sug(fismat);
% salva modelo Sugeno em disco com o nome fis_atitude_neuro.fis
writefis(sugFIS, 'fis_atitude_neuro.fis');
```