UNIVERSIDADE FEDERAL DA PARAÍBA
CENTRO DE CIÊNCIAS E TECNOLOGIA
COORDENAÇÃO DOS CURSOS DE PÓS-GRADUAÇÃO EM INFORMÁTICA

Sistema de Controle Inteligente para um Braço Robótico

por

ESDRAS FERREIRA SALES JÚNIOR

Dissertação apresentada à Coordenação de Pós-Graduação em Informática - COPIN - da Universidade Federal da Paraíba - UFPB, como parte dos requisitos necessários à obtenção do grau de Mestre em Informática.

Prof. Dr. José Homero Feitosa Cavalcanti
Orientador

Prof. Dr. Pablo Javier Alsina
Orientador

Campina Grande, Dezembro - 1997

G667a Gorgônio, Flavius da Luz e.
Uma arquitetura para sistemas inteligentes de suporte ao usuário / Flavius da Luz e Gorgônio. - Campina Grande, 1999.
84 f.

Dissertação (Mestrado em Informática) - Universidade Federal da Paraíba, Centro de Ciências e Tecnologia, 1999.
"Orientação : Prof. Dr. Edilson Ferneda".
Referências.

1. Inteligência Artificial. 2. Sistemas Baseados em Conhecimento. 3. Help Desk. 4. Dissertação - Informática. I. Ferneda, Edilson. II. Universidade Federal da Paraíba - Campina Grande (PB). III. Título

CDU 004.8(043)

SISTEMA DE CONTROLE INTELIGENTE PARA UM BRAÇO ROBÓTICO

ESDRAS FERREIRA SALES JÚNIOR

DISSERTAÇÃO APROVADA EM 30.12.1997

PROF. JOSÉ HOMERO FEITOSA CAVALCANTI, D.Sc
Presidente

PROF. PABLO JAVIER ALSINA, D.Sc
Examinador

PROF. JOSÉ FELÍCIO DA SILVA, Dr.
Examinador

PROF. EDILSON FERNEDA, Dr.
Examinador

CAMPINA GRANDE - PB

À minha **família.**

Agradecimentos

Sou muito grato:

Aos meus orientadores Prof. Dr. José Homero Feitosa Cavalcanti e Prof. Dr. Pablo Javier Alsina, pela orientação, paciência, incentivos e apoio sempre presentes.

A todos os meus amigos, em especial os que vivenciaram comigo quase que diariamente uma parte do trabalho, demostrando incentivos e cooperação: Mário Ernesto pelas opiniões, incentivos e discussão a respeito deste trabalho; Juracy Emanuel pelas perguntas, incentivos e cooperação; Everaldo pela cooperação na construção do robô, Alexandre pela ajuda e apoio em software; Geovani e Ricardo pelas correções feitas no trabalho.

A minha mãe Creuza pelos incentivos e apoio durante a realização deste trabalho.

A minha esposa Carmem Helena pela compreensão, apoio e cooperação na correção gramatical deste trabalho.

À Escola Técnica Federal do Ceará, em especial ao Departamento de Desenvolvimento do Ensino da Unidade Juazeiro do Norte, pela dispensa de algumas atividades letivas para que eu tivesse mais tempo para a pesquisa.

À sociedade brasileira principal financiadora deste trabalho.

Sumário

Lista de Figuras

Capítulo II. Neurônios e Redes Neurais Artificiais

Capítulo III. Controladores usando Redes Neurais Multicamadas

Capítulo IV. Sistemas Nebulosos

Capítulo V. Posicionamento do Pêndulo Invertido

Lista de Tabelas e Algoritmos

TABELAS

ALGORITMOS

Lista de Símbolos e Abreviações

SÍMBOLOS

Caracteres latinos:

$D(t)$ - Velocidade de referência do eixo do motor

E - Índice de desempenho de adaptação da RNMC

$E_p(t)$ - Erro proporcional

f - Função de Ativação do Perceptron

$f(.)$ - Função de mapeamento da dinâmica direta da planta

$g(.)$ - Função de mapeamento da dinâmica inversa da planta

$I_a(t)$ - Corrente de armadura do motor

$j1$ – Joelho do pêndulo

$J_D(t)$ - Índice de desempenho

L – Comprimento do pêndulo

net_{ij} - Ativação de cada unidade i para o padrão j

o_{pj} - Saída atual no nódulo

p.u. - por unidade

$Q1$ - Primeiro quadrante

$Q2$ - Segundo quadrante

$Q3$ - Terceiro quadrante

$Q4$ - Quarto quadrante

$T(t)$ - Saída desejada da RNMC

$Tl(t)$ - Torque de Carga do pêndulo

$Tm(t)$ - Torque gerado pelo motor CC

t_{pj} - Saída desejada no nódulo j para o padrão p

$U(t)$ - Sinal de controle da tensão de armadura do motor CC

$U^*(t)$ - Valor de entrada da planta capaz de reduzir o índice de desempenho

W_i -Pesos das conexões entre os neurônios

X_i -Entradas do Perceptron

$Y(t)$ - Saída intermediária do Perceptron ou Saída de uma planta não linear

$Yr(t)$ - Valor de referência da planta

$Z(t)$ - Saída atual do Perceptron

Caracteres gregos:

Δ - Erro entre a saída desejada e a saída intermediária do Perceptron

χ - Fator de adaptação do controlador neural

χ_T - Fator de adaptação do controlador neural da estratégia TREIN

χ_O - Fator de adaptação do controlador neural da estratégia OSCIL

χ_A - Fator de adaptação do controlador neural da estratégia ATRAC

χ_m - Valor nebuloso do fator de adaptação do controlador neural

η - Ganho positivo para controlar a taxa de adaptação do Perceptron ou fator de convergência no algoritmo de propagação retroativa

$\Omega(t)$ - Velocidade angular do eixo do motor

$\theta(t)$ – Ângulo entre o braço do pêndulo e a normal

θm – Função de pertinência dos quadrantes de $\theta(t)$

θr – Ângulo de referência

β – Controle da declividade da sigmóide

ζ – Fator de adaptação do controlador neural direto

ξ – Propensão de disparo do neurônio

θ – Intensidade da sigmóide

T - Deslocamento da sigmóide sobre o eixo y

$\delta_{p\varphi}$ -Alteração do erro do nódulo p para o nódulo j

$\mu A(x)$ - Função de pertinência de um elemento x a um conjunto A

ABREVIAÇÕES

APRE - Algoritmo de Propagação Retroativa do Erro

ATRAC - Estratégia para atração à posição invertida do pêndulo

CC - Corrente Contínua

cda - Função que retorna o centro da área da função inferida

DIR - Estratégia direta de posicionamento do pêndulo

ETR - Escalonador em tempo real

IA - Inteligência Artificial

INICIO - O robô está inicialmente no ponto de menor energia potencial

LF - Lógica Fuzzy

M - Médio

max - Função que retorna o maior elemento de um conjunto

mdm - Função que retorna a média dos maiores elementos de um conjunto

N - Negativo

NG - Negativo Grande

NO - Número de oscilações

NOf - Número de oscilações "fuzzificado"

OSCIL - Estratégia oscilante para posicionamento do pêndulo

P - Pequeno

PARADA - O robô 1 e 2 são desligados

PG - Positivo grande

RNA - Rede Neural Artificial

RNMC - Rede Neural Multicamadas

SCI - Sistema de Controle Inteligente

SCIBR - Sistema de Controle Inteligente para Braços Robóticos

SDP - Sensibilidade dinâmica do pêndulo

SE - Sistemas Especialistas

sp - Sem peso

st - Status da tarefa

tp - Tempo de distensão

tl - Tempo de lançamento

tp - Tem peso

TR - Tempo de resposta

TREIN - Estratégia de treinamento para posicionamento do pêndulo

Z - Zero

Resumo

Esta dissertação descreve o projeto e a implementação de um Sistema de Controle Inteligente para Braços Robóticos que pode ser utilizado para testes de diferentes estratégias e algoritmos de controle inteligente. O sistema é baseado em alguns paradigmas da Inteligência Artificial (IA), tais como: redes neurais artificiais e lógica nebulosa. O sistema desenvolvido foi utilizado para posicionar um manipulador robótico experimental no Laboratório de Redes Neurais (NEUROLAB) do Departamento de Sistemas de Computação da UFPB Campus II. Enfatizou-se o estudo do problema de controle de um braço robótico no ato de agarrar uma escada, com objetivo de demonstração das diversas dificuldades encontradas pelo robô e as correspondentes estratégias de controle inteligente utilizadas para solucioná-las.

Abstract

This dissertation describes the project and the implementation of a System for Intelligent Control of Robotic Arms that can be used for tests of different intelligent control strategies and algorithms. The system is based on some paradigms of the Artificial Intelligence (AI), such as: artificial neural networks and fuzzy logic. The developed system was utilized for the positioning of an experimental robotic manipulator in Neural network's Laboratory (NEUROLAB) of the Department of Computer Systems of UFPB Campus II. The study of the problem of controlling a Robotic Arm to catch a stairway is emphasized with objective of demonstration of the several difficulties found by the robot and the corresponding strategies of intelligent control used to solve them.

Capítulo I

Introdução

1.1 Os Sistemas Neuro-nebulosos

A Inteligência Artificial (IA) representa uma ciência emergente com fundamentos em muitas disciplinas. Ela pode ser definida como a emulação da inteligência através de software e hardware. Os principais objetivos da IA são entender os mecanismos da inteligência e realizá-los nas máquinas, as quais deverão ter, em princípio, para serem consideradas sistemas inteligentes, a capacidade de aprender, adaptar-se e evoluir [FUK 97].

O exemplo mais característico da aplicação da inteligencia artificial é o sistema especialista (SE), um sistema que procura capturar as habilidades de um especialista humano numa área restrita de conhecimento. Os SEs têm o objetivo de solucionar problemas onde a computação convencional não é eficiente. Tais problemas devem apresentar um domínio bem definido para garantir um SE com complexidade razoável. Os SEs são indicados onde o conhecimento é predominantemente heurístico, impreciso, incerto ou ambíguo, nas situações em que o conhecimento pode ser baseado na experiência.

Em sistemas onde tarefas elementares de aprendizagem são necessárias, como no caso dos sistemas de controle inteligentes, é freqüente a associação dos SEs com as redes neurais artificiais para suprir algumas fraquezas dos sistemas simbolistas (sistemas de símbolos físicos que são uma máquina que produz ao longo do tempo uma coleção progressiva de estruturas de símbolos e que, por hipótese, têm os meios necessários e suficientes para a ação inteligente em geral [RIC 91]). Os sistemas simbolistas, ao contrário das redes neurais artificiais, não são vantajosos para realizar algumas tarefas relativamente simples do ponto de vista humano, tal como reconhecer um padrão de entrada ou estimar uma saída desejada.

Com o progresso da computação, vários métodos referentes à Inteligência Artificial têm sido propostos: O Modelo de Hopfield, a Máquina de Boltzmann, os Mapas Ordenados de Kohonem, o *Competitive Learning*, os sistemas ART de Grossberg, o Discriminador de RAN's, as Redes Neurais Booleanas Seqüenciais, entre outros [ADE 91].

As redes neurais artificiais são muito aplicadas em robótica devido a sua capacidade de adaptação, tolerância a ruído, velocidade de processamento e facilidade de implementação, entre outros, tornando-se assim, indicadas em aplicações que exigem processamento em tempo real.

Em robótica, o uso de controladores baseados em Lógica Nebulosa (lógica baseada na teoria dos conjuntos nebulosos a qual manipula informações qualitativas de forma rigorosa) apresenta como vantagem a facilidade de descrição do comportamento de um robô com regras heurísticas, definidas, *a priori*, por um especialista humano [UEB 95].

A combinação de redes neurais e controladores nebulosos, chamada de modelamento neuro-nebuloso, pode melhorar o desempenho do controlador robótico pelo uso de algoritmos especiais de aprendizagem.

Os sistemas neuro-nebulosos constituem uma emulação da inteligência na manipulação de sistemas e problemas complexos, tendo uma ampla difusão no Japão e na indústria européia [NAU 94]. A aplicação das redes neurais artificiais e a lógica dos sistemas neuro-nebulosos são apropriadas quando se dispõe apenas de um conhecimento parcial do processo e se esse conhecimento puder ser representado na forma de conjuntos e regras nebulosas, além disso, desde que o treinamento da rede neural seja possível. Os Sistemas neuro-nebulosos inteligentes são importantes para manipulação de sistemas não lineares complexos, tais como os manipuladores robóticos.

A principal idéia do controle nebuloso é construir um modelo de controle de um perito humano, que é capaz de controlar a planta sem a necessidade de um modelo matemático. O especialista especifica suas ações de controle na forma de regras lingüísticas. As regras de controle são traduzidas numa estrutura da teoria de conjuntos nebulosos provendo um cálculo que pode simular o comportamento do especialista humano. [GOM 94]

A especificação de regras lingüísticas adequadas depende do conhecimento do especialista humano. A tradução dessas regras dentro da teoria de conjuntos nebulosos

não é formalizada. Por exemplo a função "membership" (ou função de pertinência), usada para representar as variáveis nebulosas, não necessita ser representada de uma forma específica. Tais incertezas no processo de construção de um controlador nebuloso resultam num processo de sintonização heurístico para superar os erros iniciais de projeto.

Os pêndulos são sistemas não lineares que dependem da força da gravidade para serem postos em movimento oscilatório. Eles são utilizados freqüentemente para demonstrar e verificar experimentalmente o desempenho de algoritmos de controle de sistemas não lineares. Os braços dos robôs podem ser tratados como pêndulos simples ou compostos.

Os sistemas de controle inteligentes (SCI) em especial, os neuro-nebulosos cooperativos, são usados no controle inteligente de robôs. Sistemas mecânicos (como o braço de um robô) que têm fricção nas articulações, são adequados para testes de estratégias de controle baseado em aprendizagem. As técnicas de SCI normalmente são usadas no desenvolvimento dos sistemas de controle de robôs considerando os efeitos da gravidade agindo sobre cada elo do robô em movimento.

1.2 Os Objetivos do trabalho

Esta dissertação tem como meta principal descrever o projeto e a implementação de um Sistema de Controle Inteligente para Braços Robóticos. Ele é baseado em redes neurais artificiais e lógica nebulosa e está sendo usado com a função de posicionar um braço de robô no Laboratório de Redes Neurais do Departamento de Sistemas de Computação da UFPB - Campus II. Como objetivos secundários, destacamos:

- Descrever o neurônio biológico. Mostrar os diversos tipos de neurônios artificiais. Descrever o algoritmo de aprendizagem do Perceptron. Descrever alguns conhecimentos básicos sobre redes neurais multicamadas, explicar suas principais características, descrever o algoritmo de aprendizagem retroativa do erro e fazer sugestões para se iniciar um projeto geral com redes neurais.
- Descrever o projeto e implementação de um protótipo experimental (hardware) que pode ser usado para teste de diferentes estratégias de controle inteligente.

- Descrever um executivo em tempo real, um software usado para escalonar as tarefas de controle do braço robótico. O executivo é uma ferramenta de software desenvolvida em linguagem C para operar em microcomputadores compatíveis com o IBM/PC sob sistemas operacionais compatíveis com o MS-DOS, dotado de interface gráfica, para simulação e controle de um protótipo de braço de robô com dois graus de liberdade. O executivo atua sobre o protótipo experimental para o controle dos diversos movimentos permitidos pelo braço robótico experimental.
- Verificar que essas aplicações do sistema, devido à sua não-linearidade, são excelentes bancadas de testes para ferramentas usadas em controle inteligente tais como redes neurais e lógica nebulosa.
- Fazer uma descrição das principais ferramentas usadas nesse trabalho, de forma a abranger os principais tópicos relacionados com o SCI.
- Descrever como são utilizadas as RNAs em controladores, exemplificar alguns tipos de neurocontroladores.
- Descrever os fundamentos da Lógica Nebulosa e explicar a utilização dos Sistemas Nebulosos em controle.
- Descrever a mecânica do pêndulo e como utilizar um controlador neural para controlá-lo, posicionando-o na posição invertida. Descrever o robô experimental em desenvolvimento. Dar ênfase ao movimento do robô sobre os degraus de uma escada, usando uma analogia especial do protótipo com um braço humano.
- Apresentar alguns resultados obtidos através da simulação do posicionamento do braço do robô com dois graus de liberdade, como também apresentar alguns resultados experimentais obtidos no laboratório.

1.3 Organização da dissertação

No capítulo I descreve-se o envolvimento da IA, dos Sistemas Especialistas (SE), da Lógica Nebulosa (LF) e das Redes Neurais Multicamadas com os sistemas de controle inteligente, dando ênfase ao controle de braços robóticos. Descrevem-se os principais objetivos da dissertação, introduzem-se os conhecimentos preliminares sobre o

hardware e o software do sistema e descrevem-se sucintamente os demais capítulos da dissertação.

No capítulo II descrevem-se as principais características dos neurônios biológicos e de sua equivalência tecnológica, o Perceptron. Mostram-se os diversos tipos de neurônios artificiais, descreve-se o modo de treinamento padrão do Perceptron. A seguir, descreve-se o Perceptron Multicamadas e enumera-se uma série de sugestões para se iniciar um projeto com RNA.

No capítulo III apresentam-se diversos tipos de controladores neurais e algumas implementações para controle de uma planta.

No capítulo IV descrevem-se as principais idéias da Lógica Nebulosa, os seus fundamentos, a forma de aprendizagem dos sistemas neuro-nebulosos, como utilizar os Sistemas neuro-nebulosos para controle e faz-se uma breve classificação dos sistemas neuro-nebulosos.

No capítulo V são listados alguns dos principais trabalhos relacionados com o controle neural do pêndulo invertido. Descreve-se o sistema físico do pêndulo simples. Descreve-se como posicionar o pêndulo usando redes neurais e lógica nebulosa. Descreve-se uma solução neuro-nebulosa para colocar o pêndulo na posição invertida a partir das estratégias DIR , OSCIL e, por fim, a partir de uma estratégia híbrida.

No capítulo VI, inicialmente discutem-se os SCI's e apresenta-se um sistema de teste: um manipulador robótico de dois graus de liberdade. Em seguida, para exemplificar a atuação do SCI para um braço robótico, apresentam-se as estratégias desenvolvidas para controle do SCI no movimento de agarrar uma escada. Apresenta-se um escalonador em tempo real para controlar as operações executadas pelo SCI. Finalmente, apresentam-se os resultados experimentais obtidos com o SCI para controle do movimento do robô.

No capítulo VII apresentam-se as conclusões da dissertação.

No anexo A descrevem-se o hardware do SCI e o controle do motor CC.

No anexo B descreve-se o algoritmo de aprendizagem por propagação retroativa do erro.

Capítulo II

Neurônios e Redes Neurais Artificiais

2.1 Introdução

Neste capítulo, inicialmente apresentam-se sucintamente as características do neurônio biológico. A seguir, após a apresentação do neurônio artificial de MacCulloch & Pitts, apresentam-se os tipos básicos de neurônios artificiais. Em seguida apresentam-se o Perceptron e algoritmo para o seu treinamento. Finalmente, apresentam-se as RNMC e seu algoritmo de treinamento baseado na propagação retroativa do erro.

2.2 O Neurônio biológico

A unidade básica do cérebro biológico é o *neurônio*. Apesar de nenhum neurônio ser igual a qualquer outro, eles compartilham certas características estruturais básicas que são relativamente simples. A Fig. 2.1 mostra a representação simplificada de um neurônio biológico.

O *corpo celular* ou *soma* é responsável pela síntese de algumas moléculas necessárias para manter a vida do neurônio. Presos ao corpo celular, estão uns filamentos longos e de forma irregular chamados *dendritos*, os quais fornecem o caminho para as entradas dos sinais eletroquímicos (vindos de outros neurônios) para o corpo celular.

Um outro tipo de elemento preso ao corpo celular é o *axônio*. Ele é eletricamente ativo tal como um dendrito e serve como canal de saída do neurônio. A informação é transmitida de um neurônio a outro através de espaços de conexão chamados *sinapses* que ocorrem freqüentemente entre o axônio de um neurônio e o dendrito de outro. Nas sinapses a transmissão de sinais ocorre de forma eletroquímica através da difusão de moléculas chamadas de *neurotransmissores*.

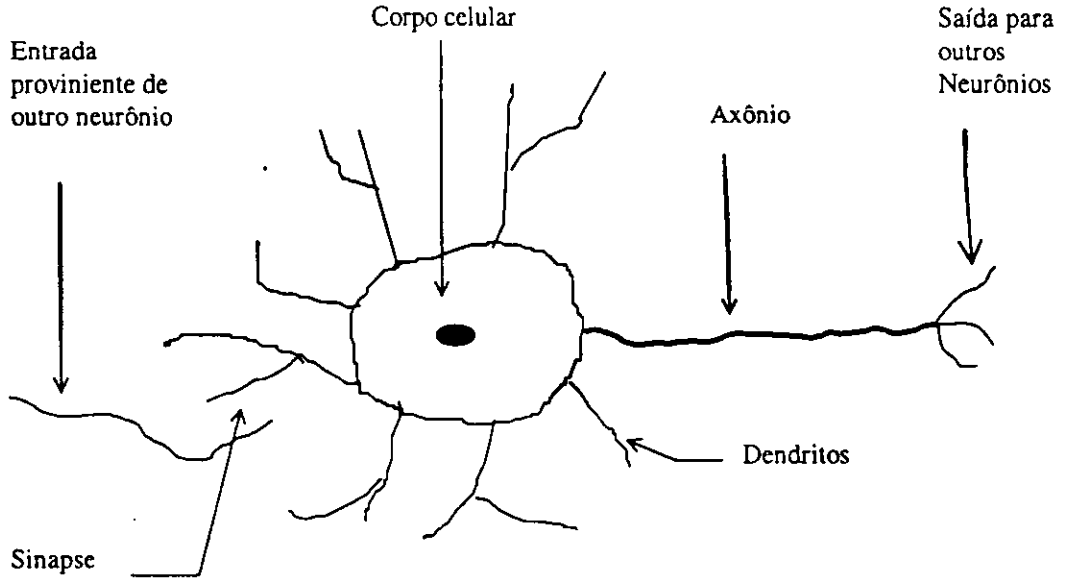


Fig. 2.1 – Representação de um neurônio biológico

Segundo [BEA 91], ocorre um processo de integração quando os sinais de entrada chegam no soma. Quando esta integração atinge um determinado limite, pode ser gerado um impulso elétrico que é transmitido pelo axônio para outros neurônios. O neurônio, continuamente, ou está disparando neurotransmissões ou está inativo. Este mecanismo é definido como a característica *tudo ou nada* dos neurônios.

A função básica de um neurônio biológico é, estimulado por suas entradas, produzir um valor na sua saída. Quando o valor de ativação das suas entradas for maior do que um determinado valor (valor *limiar*). As suas sinapses alteram a eficiência com a qual o sinal é transmitido. Algumas sinapses são consideradas boas junções quando permitem a passagem ampla de um sinal, já outras sinapses são consideradas junções muito fracas por dificultarem a passagem de um sinal [HEB 49].

2.3 O Neurônio Artificial

A Fig. 2.2 apresenta o diagrama do neurônio artificial proposto por McCulloch & Pitts em 1942 [MCC 43]. O neurônio de Mculloch-Pitts emula o neurônio biológico de uma forma simplificada fazendo uma soma ponderada sobre suas entradas. A saída tem o valor 1 quando a soma ponderada for maior do que um determinado limiar ajustável. A saída gerada tem valor 0 quando a soma ponderada for menor do que o valor limiar.

Os pesos das conexões $(w_1,w_2,...,w_n)$ e as entradas $(x_1,x_2,...,x_n)$ ilustrados na Fig. 2.2 podem assumir quaisquer valores reais. Um peso maior do que a média aritmética

dos pesos modela uma sinapse mais eficiente (que transmite mais o sinal) e um peso menor do que essa média modela uma sinapse mais fraca. A aprendizagem, neste caso, é o processo de modificar os valores dos pesos. Isto é conhecido como treinamento sináptico [HEB 49].

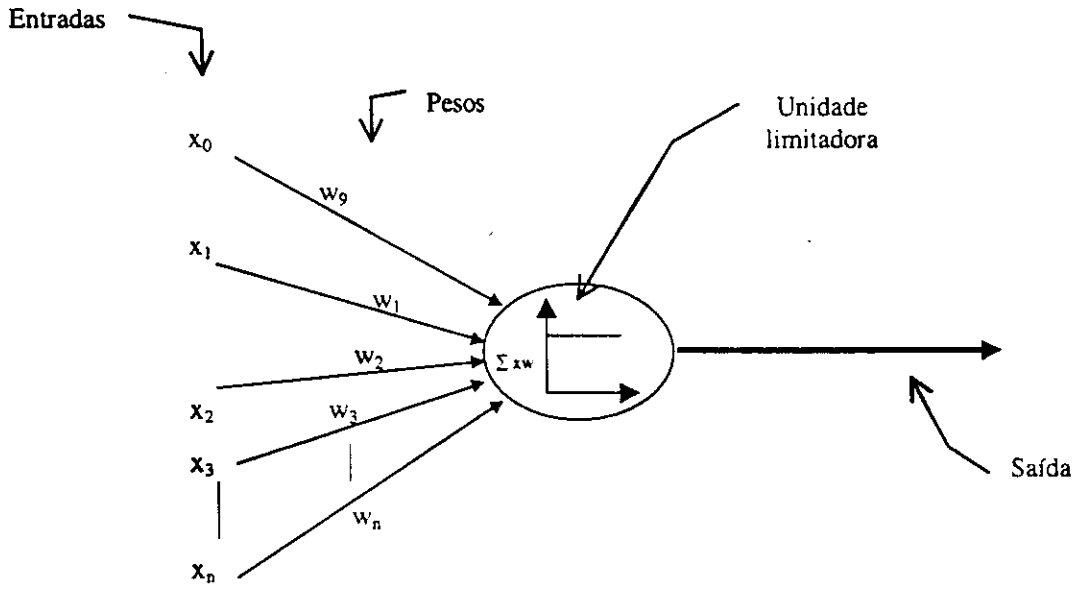


Fig. 2.2 – O Neurônio artificial

2.4 Os Tipos básicos de neurônios artificiais

Mostra-se na Fig. 2.3 o neurônio artificial Perceptron, proposto por Rosenblat em 1957 [ROS 58], o qual é capaz de aprender de acordo com as regras sugeridas por Hebb [HEB 49]. O Perceptron possui três partes: entrada, parte intermediária e saída. As entradas do Perceptron são representadas por X_i, e os pesos das suas entradas são representados por W_i. A saída da parte intermediária é representada por:

$$Y(t) = \sum_i X_i W_i \tag{2.1}$$

O valor de saída do neurônio é dado por:

$$Z(t) = f(Y(t)) \tag{2.2}$$

A função $f(.)$, função de ativação, geralmente é não-linear.

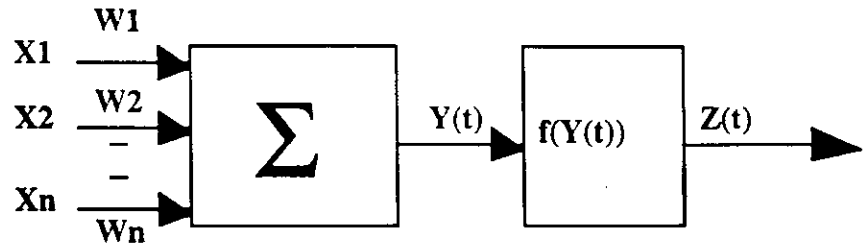


Fig. 2.3 - O Perceptron

Um outro tipo de neurônio muito usado em problemas de otimização é o neurônio de Hopfield. A Fig. 2.4 ilustra uma representação do neurônio de Hopfield com suas partes: entradas, parte intermediária e saída. Sua principal característica é possuir, na sua saída, um integrador que permite uma memorização dos estados anteriores.

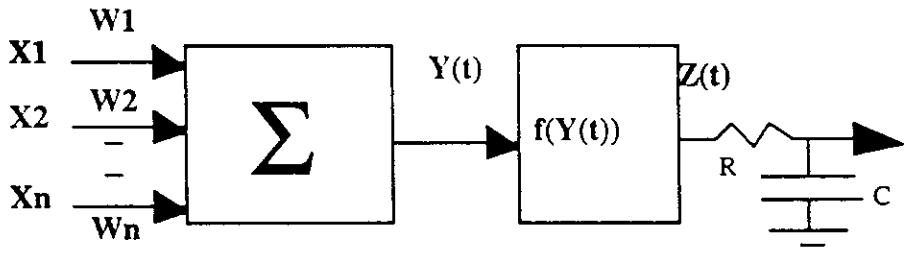


Fig. 2.4 - O Neurônio de Hopfield

2.5 Funções de ativação do neurônio artificial

A função de ativação determina o tipo do neurônio. Dependendo do seu tipo o neurônio pode ser classificado em: linear, lógico, sigmóide, sinal, logarítmico simétrico, entre outros.

Descrevemos a seguir os três primeiros tipos:

2.5.1 O Neurônio Linear

O neurônio linear tem como função de ativação uma função linear do tipo:

$$Z(t) = f(Y(t)) = \alpha \Sigma X_i(t)\, W_i(t) \text{ para } -1 \leq Y \leq 1,\ \alpha = \text{constante}$$

Z é mostrado em destaque na Fig. 2.5.

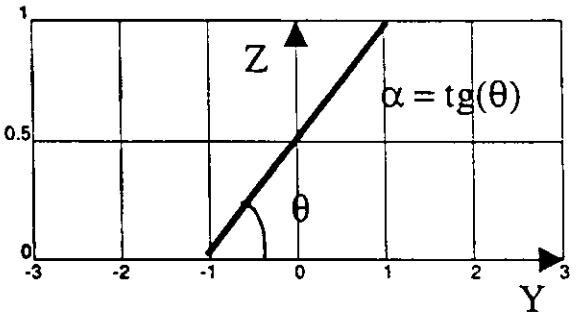


Fig. 2.5 - A Função de Ativação Linear

2.5.2 O Neurônio Lógico

A função de ativação do neurônio lógico, ou neurônio binário, é a função degrau: $Z(t)=1$ se $Y(t)>0$, onde $Y(t)=\Sigma X_i W_i$ e $Z(t)=0$ em caso contrário. A Fig. 2.6 mostra o comportamento da função lógica deste tipo de neurônio.

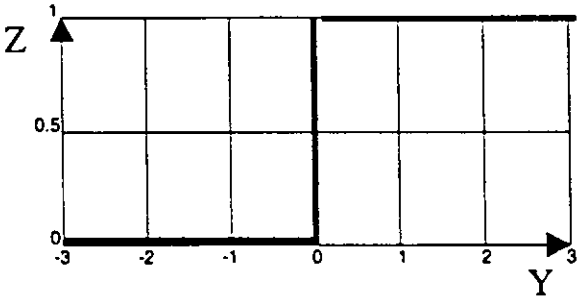


Fig. 2.6- A Função de Ativação Lógica

2.5.3 O Neurônio Sigmóide

O neurônio do tipo sigmóide tem uma função de ativação sigmóide, que é uma função contínua, diferenciável e limitada. Ela pode ser descrita pela Eq. 2.3:

$$Z(t) = \frac{\theta}{[1 + e^{-\beta Y(t)}]} \tag{2.3}$$

onde

$$Y(t) = \Sigma X_i(t) W_i(t) + T \tag{2.4}$$

A Fig.2.7 mostra o seu comportamento para $\theta = 1$, $\beta = 1$ e $T = 0$. A curva de sua derivada é mostrada na Fig. 2.8.

Uma vantagem de usar a função sigmóide como função de ativação não-linear é que, para determinados valores de seus parâmetros, pode-se aproximar da função degrau em qualquer grau de aproximação desejado, permanecendo diferenciável.

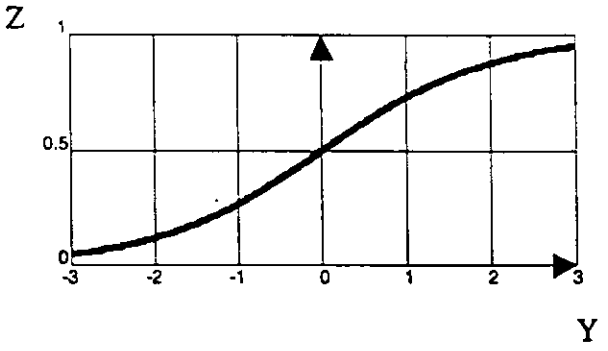


Fig. 2. 7 - A Função de Ativação Sigmóide

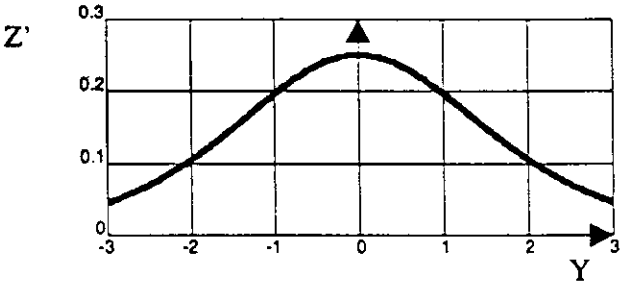


Fig. 2.8- A Derivada da Função de Ativação Sigmóide

Os parâmetros β,θ e T especificam a forma da função. O parâmetro β controla a declividade da curva: valores grandes de β comprimem a função até quando $\beta \rightarrow \infty$, $Z(Y) \rightarrow$ função degrau. O parâmetro θ é a valor base da sigmóide. O parâmetro T é o deslocamento da curva sigmóide sobre o eixo Y.

2.6 Treinamento do Perceptron

Admitindo-se o Perceptron com uma função ativação degrau f conforme o modelo exibido na Fig. 2.3, a Eq. 2.5 representa a saída do Perceptron.

$$z(t) = f\left[\sum_{i=0}^{n} w_i(t)x_i(t) - T\right], \text{ onde } f(t) = 1 \text{ se } t > 0 \text{ e } f(t) = 0 \text{ se } t \leq 0. \qquad (2.5)$$

Definindo-se $w_i(t)$ $(0 \leq i \leq n)$ como sendo o peso da entrada i no tempo t, e T como o valor limiar, o erro Δ pode ser definido na Eq. 2.6.

$$\Delta = d(t) - z(t) \qquad (2.6)$$

onde $d(t)$ representa a saída desejada.

Define-se η $(0 \leq \eta \leq 1)$ como um ganho positivo para controlar a taxa de adaptação.

O procedimento de aprendizagem ou algoritmo de treinamento do Perceptron pode ser descrito como assim:

Algoritmo 2.1

1) Inicialize os pesos $w_i(0)$ e o limite $w_0 = -\xi$ com pequenos valores aleatórios. Faça $x_0 = 1$ sempre.

2) Apresente as entradas $(x_1, x_2, ..., x_n)$ e saída desejada $d(t)$ de uma forma padronizada.

3) Calcule a saída atual $z(t)$ (Eq. 2.5) ou, se a soma ponderada das entradas ultrapassar o valor de limiar do neurônio T, o sinal de saída é 1, caso contrário, é 0. Em seguida compare com a saída desejada correspondente àquela entrada.

4) Adapte os pesos da seguinte forma: reforce as decisões corretas (aumentando os pesos) e desencoraje as decisões erradas (diminuindo os pesos) da seguinte forma:

   - $\Delta = d(t) - z(t)$
   - $w_i(t+1) = w_i(t) + \eta \cdot \Delta \cdot x_i(t)$

5) Volte para o passo (2) até que o erro seja mínimo.

2.7 A Rede Neural Multicamadas

Os neurônios artificiais são os elementos que compõem as RNAs. Eles realizam de forma conjunta algumas das funções mais elementares da inteligência, em analogia com os cérebros biológicos. Essas funções são utilizadas para a realização automática de atividades consideradas "inteligentes".

As RNAs podem modelar o Sistema Nervoso e simular Sistemas Físicos. As RNAs têm alcançado muito sucesso em tarefas perceptuais, tais como a fala e a visão.

Elas se fundamentam na idéia de que pequenas unidades de processamento simples (neurônios), interagindo entre si, terão um elevado poder computacional. Elas são estruturas de processamento paralelas, onde um grande número de unidades participam do processamento. Tal fato, associado a um certo grau de redundância no conhecimento distribuído por toda a RNA, faz com que ela tenha um comportamento, até certo ponto, tolerante a falhas.

As RNAs podem aprender por experiência, possuem tolerância a falhas, são adaptáveis, podem fazer abstrações, são capazes de generalizar e discriminar padrões, podendo ser utilizadas para fazer previsões e controlar processos, entre outras aplicações.

As RNAs representam uma área com fronteiras muito amplas. Seus estudos e desenvolvimento estão relacionados com as engenharias e com a Ciência Cognitiva, incluindo Matemática, Neurologia, Psicologia, Física, Computação e Biologia. Essa área interdisciplinar vem crescendo muito desde o início dos anos 80. Atualmente, as RNAs estão sendo utilizadas no reconhecimento de Padrões, nos Problemas de Otimização, nas Memórias Associativas e na Modelagem dos Processos de Cognição, entre outras aplicações.

2.8 Treinamento da RNMC

Um único neurônio artificial possui relativamente uma pequena capacidade de processamento. A interligação de vários neurônios entre si, em estruturas paralelas e/ou seriais, como em uma rede, aumenta abruptamente o poder computacional coletivo. Esse tipo de estrutura denomina-se rede neural artificial (RNA). O exemplo mais popular de uma RNA é o Perceptron Multicamadas. Sua estrutura básica é mostrada na Fig. 2.9, onde "L" representa neurônios com função de ativação linear e "S" representa neurônios com função de ativação sigmóide.

O modelo mostrado na Fig. 2.9 tem três camadas, uma camada de entrada, uma camada de saída e uma camada entre essas duas, não conectada diretamente à entrada ou à saída, chamada de camada escondida ("hidden"). Cada neurônio da camada escondida ou da camada de saída de dados é um Perceptron, porém a função de ativação é uma

função sigmóide e não uma função degrau dos primeiros Perceptrons. Os neurônios da camada de entrada, com função de ativação linear, servem para distribuir os valores de entrada para a camada escondida.

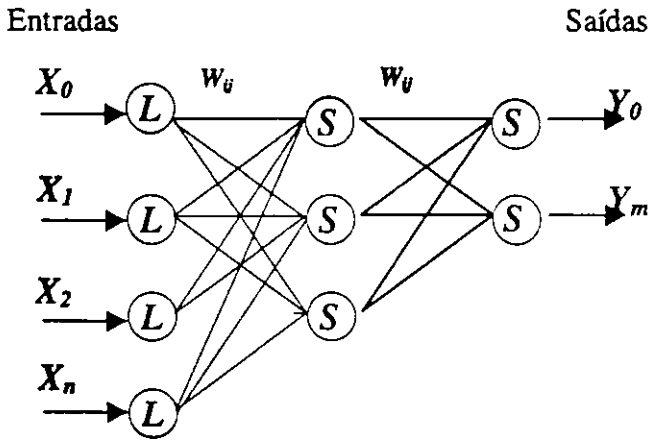


Fig. 2.9 - As Redes Neurais Artificiais Multicamadas

A aprendizagem das RNAs é feita aplicando-se um método sistemático para ajuste de informações internas, com o intuito de modificar a forma de operação de cada ou de alguns neurônios. Basicamente existem dois tipos de aprendizagem em redes neurais artificiais: Aprendizagem Supervisionada e Aprendizagem Não-Supervisionada.

A aprendizagem supervisionada requer um vetor de dados de entrada e o vetor de dados de saída desejada para a rede. O vetor de saída da rede é calculado a partir do vetor de entrada e em seguida é feita uma comparação do vetor de saída da rede com o vetor de saída desejada. O vetor diferença ou vetor erro, é representado pela diferença entre valores dos vetores da saída desejada e os valores do vetor de saída da rede, com ele a rede ajusta os pesos de cada neurônio segundo um algoritmo que tenta minimizar o vetor erro.

A aprendizagem não-supervisionada não exige a apresentação do vetor dos valores de saída desejada e sim apenas o vetor dos valores de entrada. A rede neste caso, durante o treinamento, extrai propriedades do conjunto de vetores de entrada agrupando os vetores similares em classes. Portanto, o vetor de valores de saída corresponde à classe a que pertence o vetor de valores de entrada que está sendo proposto.

Os vetores dos valores de entrada são escolhidos de forma que sejam exemplos representativos do problema. Eles costumam ser chamados de padrões de entrada. Esses

padrões podem assumir várias formas: seqüência de sons (padrão auditivo), pixels de uma imagem (padrão visual), variações de tempo (padrões temporais), entre outros.

A decisão de que um padrão pertence ou não a uma determinada classe é o objetivo clássico das redes neurais artificiais. Modelar um problema em redes neurais é basicamente encontrar os padrões representativos desse problema para apresentar a rede, pois cada problema a ser modelado tem as suas características implícitas nos seus padrões. Após treinar uma rede com tais padrões a mesma será capaz de reconhecer padrões desconhecidos como pertencentes ou não ao problema em questão.

O Perceptron Multicamadas ou Redes Neurais Multicamadas (RNMCs) têm sido aplicados a diversos problemas de classificação e de controle. A RNMC utiliza um algoritmo muito popular conhecido como "*Backpropagation Error Correction Algorithm*", ou Algoritmo de Propagação Retroativa do Erro (APRE, ver Anexo B). O APRE funciona em dois estágios distintos. No estágio de propagação, um padrão é apresentado à camada de entrada e as unidades de processamento calculam as saídas dos neurônios, camada após camada, até alcançar a camada de saída. No estágio de retropopagação, os erros encontrados na camada de saída são usados para atualizar os pesos das conexões da rede, um a um e de trás para frente (da camada de saída para a camada de entrada).

A operação da rede é similar à operação de um único Perceptron no que se refere à apresentação de um padrão e cálculo da sua resposta. A comparação com a saída desejada permite alterar os pesos das conexões para que a rede possa possivelmente produzir uma saída mais acurada da próxima vez. A regra de aprendizagem provê um método para ajustar os pesos na rede.

É óbvio que a regra de aprendizagem da RNMC é mais complexa que a regra usada para treinamento de um único Perceptron. Contudo, pode-se compreendê-la melhor considerando a forma como a rede se comporta quando se apresentam padrões a ela. Quando se apresentam padrões de entrada a uma rede não treinada, ela produz uma saída aleatória. Define-se uma função do erro que representa a diferença entre a saída atual da rede e aquela que se quer que ela produza. Esse tipo de aprendizagem é chamado de aprendizagem supervisionada porque necessita-se conhecer o padrão correto a ser aprendido. Para que a rede aprenda sucessivamente, tem-se que reduzir o

valor da função erro até que a rede aprenda a gerar a saída desejada. Isso é conseguido ajustando os pesos das conexões entre os neurônios.

Na área das redes neurais artificiais, o método mais utilizado para a redução do erro médio quadrático é o descendente por etapas ("steepest descent"), negativo do gradiente, ou regra delta generalizada (Eq. 2.7).

$$W_{ik+1}=W_{ik}+\mu(-\nabla(I(W_k))) \tag{2.7}$$

Nesta equação, μ é um parâmetro que controla a estabilidade e a razão de convergência do treinamento, e $\nabla(I(W_k))$ é o valor do gradiente do índice de desempenho no ponto W_k. O erro deve ser reduzido a partir da modificação dos pesos das conexões da rede neural. No método descendente por etapas, o usuário deve atribuir valores iniciais aleatórios aos pesos W_{i0}. Os pesos são iterativamente calculados pela Eq. 2.7.

A regra delta generalizada faz isso calculando o valor da função erro para uma entrada em particular e, então, retropropagando o erro de uma camada para a anterior. Cada unidade da rede tem seus pesos reajustados para reduzir o valor da função erro. Para as unidades de saídas, seus pesos e suas saídas desejadas são conhecidos. Portanto, ajustar os seus pesos é uma tarefa relativamente simples, equivalente ao treinamento do Perceptron. Porém, para a camada do meio o ajuste não é tão óbvio, pois as saídas desejadas não são conhecidas (ver Anexo B).

2.8 Características básicas para implementação de uma rede neural artificial

Segundo [ADE 91], duas propriedades muito importantes e, até certo ponto, antagônicas, desejadas nos sistemas de reconhecimento de padrões são: a *generalização* e a *discriminação*. Entende-se por generalização a capacidade de considerar padrões um tanto diferentes do conjunto usado no treinamento como pertencentes à mesma classe. Já a discriminação, entende-se como a capacidade de considerar padrões bem diferentes do conjunto usado durante o treinamento como não pertencentes à classe.

São listadas, abaixo, algumas características atrativas de Redes Neurais [VAS 95]:

- "Soluções eficientes para muitos problemas do mundo real.

- Capacidade de adaptação (aprender por exemplos e mudar de acordo com o ambiente).
- Generalização como uma propriedade emergente.
- São bem adequadas à situações onde não há como se definir regras.
- Não assume conhecimento a priori sobre a distribuição das classes de padrões.
- Estruturas inerentemente paralelas.
- Baixos requerimentos de memória (normalmente).
- Rápido tempo de resposta.
- Representação do conhecimento de forma distribuída (tolerância a falhas)."

Listam-se também, algumas características na escolha de um modelo [VAS 95]:

- " Capacidade de generalização
- Taxas de erros de classificação
- Requerimentos de memória
- Tempo de treinamento
- Tempo de classificação
- Capacidade de armazenamento
- Facilidade de implementação em hardware
- Facilidades de programação em computadores seriais e paralelos
- Facilidade de mapeamento entre complexidade de um classificador e quantidade de dados existentes
- Habilidade de realizar treinamento não supervisionado
- Facilidade de interpretação das representações definidas
- Capacidade de detectar e se adaptar a mudanças no ambiente."

Para a fase de projeto [VAS 95]:

- "Identificação da natureza do problema
  - Tarefas e sub-tarefas a serem executadas.
- Escolha do modelo (ou modelos) apropriado para cada tipo de tarefa
  - Características dos modelos.
- Composição de um conjunto estatisticamente representativo de exemplos para treinamento

- Pré-processamento e extração de características (se for empregada).
- Treinamento utilizando um critério de parada
  - Ex.: Erro médio, Cross-validação.
- Teste de performance com um conjunto de teste representativo **independente** do conjunto de treinamento."

Na preparação de uma rede neural artificial para aplicar a um determinado problema surge a necessidade de um pré-processamento para tratar os padrões. Por exemplo, depois de captar os padrões, uma fase praticamente obrigatória se refere ao tratamento da informação que é captada pelos sensores, a qual ainda está num estado bruto, tendo o objetivo de filtrar ou minimizar os ruídos e distorções que venham a surgir durante o processo de aquisição do sinal de entrada. Existe também, em muitas aplicações, a necessidade de separar os padrões que porventura tenham sido apresentados ligados uns aos outros (segmentação) e como também a necessidade de reduzir o sinal original à forma normal das classes de padrões para atender às restrições de limite do espaço de entrada (normalização). Portanto, tarefas de captação, segmentação, discretização, amostragem, codificação etc., fazem parte desta preparação dos padrões. Supondo-se que já se tenha escolhido uma rede adequada para o problema que se quer resolver, a preparação normalmente envolve os seguintes passos [ADE 91]:

1."Extração do conjunto de padrões de treinamento
2. Treinamento da rede
3. Seleção do conjunto de padrões de teste
4. Utilização da rede em regime de teste
5. Utilização da rede em regime permanente"

No caso de se querer resolver um problema de controle, como um sistema de controle inteligente para controlar um pêndulo, são propostos os seguintes passos:

1. Extração do conjunto de padrões de treinamento pela inversa da dinâmica da planta fazendo o pêndulo operar apenas pela ação da gravidade.
2. Treinamento da rede ("off-line") com a apresentação dos sinais experimentais.
3. Operar (controlar) "on-line" em regime de teste.
4. 4. Operar em regime permanente

Capítulo III

Controladores Usando Redes Neurais Artificiais Multicamadas

3.1 Introdução

Neste capítulo, inicialmente discute-se o uso de RNA em controle e apresenta-se o controlador neural inverso. Finalmente, apresentam-se esquemas para treinamento de RNAs usadas como controladores.

3.2 Aplicações das RNAs em Controladores

A teoria de controle automático desde o seu nascimento, há cinco décadas (Brown e Campbell 1948 - citado por [NAR 95]) tem sido fortemente influenciada pela demanda de uma tecnologia emergente para obter controladores mais rápidos, mais acurados e mais robustos. Os diversos tipos de aplicações em controle se deparam facilmente com três tipos de dificuldades *a priori*: (1) – complexidade computacional, (2) – presença de não-linearidades, (3) – presença de incertezas. As RNAs, em princípio, são capazes de superar essas dificuldades: (1) - A arquitetura massivamente paralela das RNAs permite processar em altas velocidades. (2) - Já que elas podem aproximar funções não-lineares para qualquer grau de acurácia elas possuem ao menos a capacidade de identificar (e então controlar) sistemas dinâmicos não-lineares. (3) – Segundo [NAR 95], o fato de que várias regras têm de ser avaliadas simultaneamente na construção de uma rede neural, possibilita o tratamento de incertezas pelas RNA. Desse ponto de vista, as RNAs são fortes candidatas para a construção de sistemas de controle adaptativos complexos, sendo muito apropriadas para controle, identificação, adaptação e sintonia de controladores.

3.3 Controlador neural inverso

Nos problemas de controle encontra-se a tarefa de projetar um controlador que gere um sinal, baseado em todas as informações disponíveis até um determinado instante, que ajuste os parâmetros do controlador de tal modo que a saída da planta (um objeto que atua no ambiente local e que tem uma ou mais variáveis a serem controladas) eventualmente siga o modelo de referência projetado (controle com retroalimentação) [DAZ 88].

Por mais de 20 anos duas abordagens distintas têm sido usadas nos sistemas de controle para controlar uma planta adaptativamente:

1. Controle direto: nele os parâmetros do controlador são ajustados para reduzir o valor absoluto do erro.

2. Controle indireto: nele os parâmetros da planta são estimados indiretamente pelo controlador.

Recentemente, foram publicados diversos trabalhos [HUN 92] sobre redes neurais usando uma perspectiva de sistemas de controle. Diversos pesquisadores, principalmente [NAR 90] e [TAN 91], têm utilizado diferentes estruturas de controladores neurais baseados na inversa da dinâmica da planta. Eles mostraram que a inversa pode ser obtida por meio de redes neurais para plantas lineares e não lineares.

Considerando-se a planta não-linear, descrita pela Eq. 3.1, com única entrada $U(t)$ e única saída $Y(t)$, onde $f(.)$ representa uma função não-linear; se a planta é inversível, deve existir uma função $g(.)$ tal que, $U(t)$ possa ser calculado pela Eq. 3.2. Essa função $g(.)$ é conhecida como a função da dinâmica inversa da planta.

$$Y(t+1) = f(Y(t),...Y(t-n+1);U(t),...,U(t-n+1)) \tag{3.1}$$

$$U(t) = g(Y(t),...Y(t-n+1);U(t-1),...,U(t-n+1)) \tag{3.2}$$

A Fig.3.1 mostra o esquema do controlador neural usando a inversa da planta. A rede neural gera na sua saída o valor $U(.)$ para controlar a planta. Observe-se que $Yr(t)$ é um valor de entrada usado como valor de referência para o controlador. A saída da planta deve seguir o valor $Yr(t)$

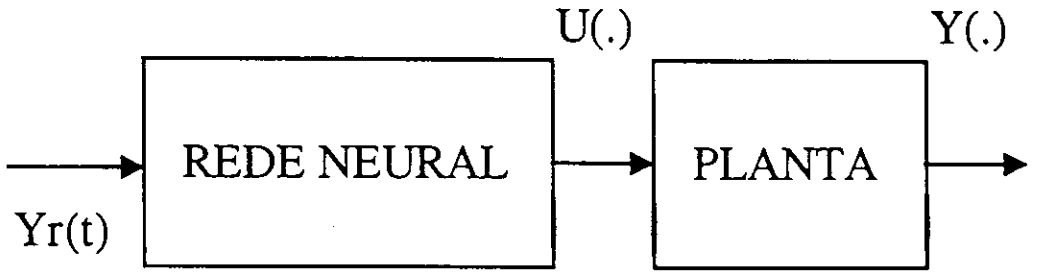


Fig. 3.1 - O Controlador baseado em Rede Neural Artificial

3.4 Controlador neural direto inverso

Na Figura 3.2 é mostrado o esquema geral do controlador neural direto inverso como descrito por [TAN 91], [KHA 92] e [CAV 94a]. Ele se caracteriza por ser um sistema aberto e o seu treinamento deve ser feito "off-line". O controlador neural direto inverso é usado quando se assume que a planta é inversível e já se deve ter um conhecimento prévio do Jacobiano do sistema.

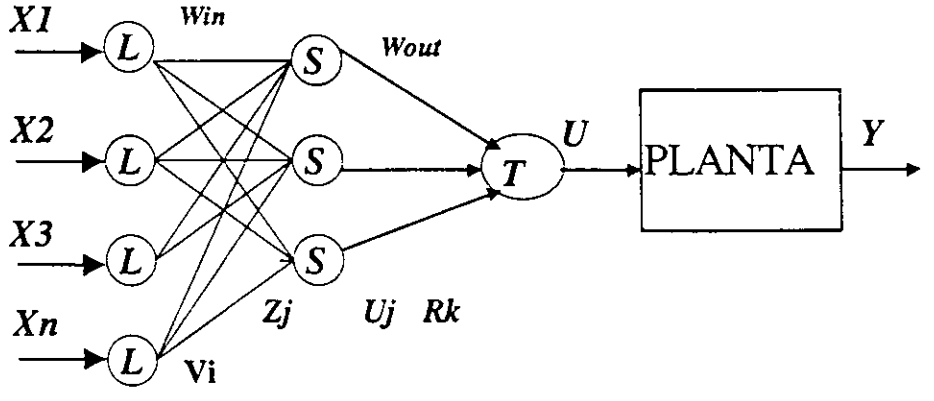


Fig.3.2 - Controlador neural direto inverso

O controlador neural direto inverso passa por duas fases: a fase de treinamento "off-line" e a fase de operação ou controle, "on-line". Durante a fase de treinamento, mostrada na Fig.3.3, a RNMC é treinada com a apresentação dos sinais experimentais de entrada e saída da planta, $U(t)$ e $Y(t)$. Na Fig.3.3, $U_i(t)$ representa o valor da saída da

RNMC. A Eq. 3.3 representa o índice de desempenho usado para treinamento dos pesos e parâmetros da RNMC [CAV 97].

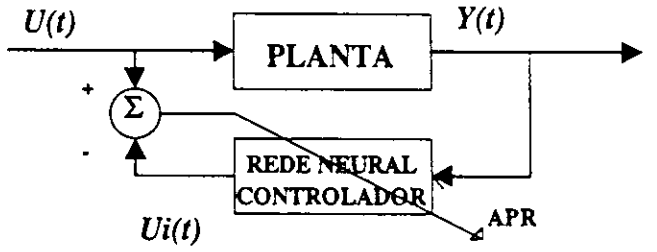


Fig. 3.3 - O controlador neural direto inverso - Treinamento.

$$J(t+1) = (½)e(t+1)^2 = (½)[U(t)-U_i(t)]^2 \qquad (3.3)$$

3.5 Treinamento "on-line" do controlador neural adaptativo direto

Vários esquemas de controladores adaptativos digitais têm sido propostos nas últimas décadas [ÄST 89]. O projeto e configuração desses controladores dependem primariamente de um conhecimento antecipado da dinâmica da planta sob controle e, para sintonização do controlador em tempo real, requer-se a determinação "on-line" da dinâmica de operação da planta.

Na Fig. 3.4 é mostrado o esquema geral do controlador neural adaptativo direto, como descrito por [TAN 91] e [KHA 92].

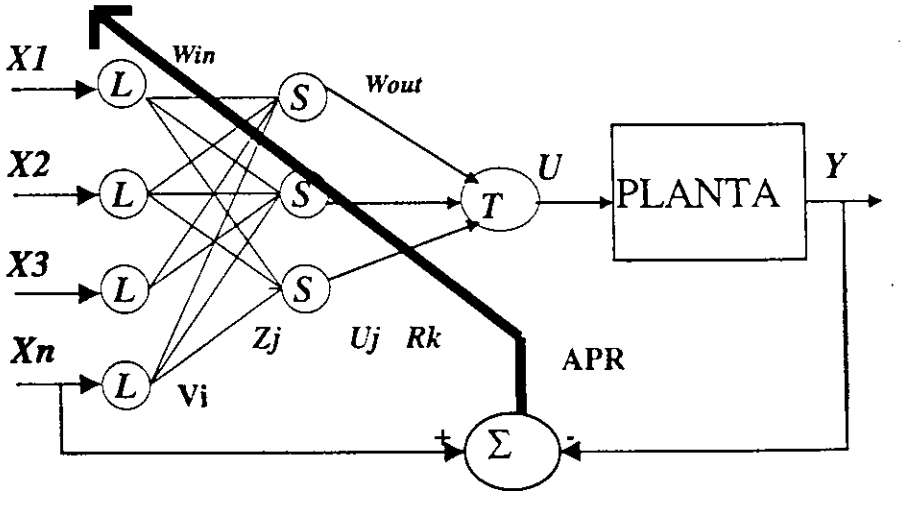


Fig. 3.4 - Controlador neural adaptativo direto.

O controlador neural adaptativo direto, mostrado na Fig. 3.4, durante a fase de controle tem com entradas [$X1$ = $U(t\text{-}1)$, $X2$ = $Y(t\text{-}1)$, $X3$ = $Y(t)$ e X_n =$D(t+1)$], com $D(t+1)$ representando a saída desejada. A rede neural gera a saída $U(t)$ que é usada para controlar a planta.

Ele se caracteriza por ser um sistema de malha fechada em que a RNMC pode ser treinada "off-line", como no controlador neural direto inverso, e sob algumas condições, ela pode ser treinada "on-line". Durante a fase de controle, o sinal na saída da planta (Y($t+1$)) deve seguir um sinal desejado ($D(t+1)$) modificando $U(t)$, seguido do treinamento da RNMC, de forma a minimizar o índice de desempenho $J_D(t+1)$ definido na Eq. 3.4 (ver [TAN 91]).

$$J_D(t+1) = \tfrac{1}{2}E(t+1)^2 = \tfrac{1}{2}[D(t+1)\text{-}Y(t+1)]^2 \tag{3.4}$$

No esquema do controlador neural adaptativo direto define-se $U_a(t)$ como o valor de saída atual da RNMC controladora e $U^*(t)$ como o valor de entrada da planta capaz de reduzir o índice de desempenho da Eq. 3.4. $U^*(t)$ pode ser calculado usando a regra delta generalizada como mostrado na Eq. 3.5. Nessa equação, define-se ζ como o fator de adaptação do controlador neural direto ($0<\zeta<1$). O jacobiano da planta é definido como $\partial\Omega(t+1)/\partial U(t)$. Geralmente o jacobiano da planta é desconhecido e deve ser calculado indiretamente. Para o caso particular de controle de um motor CC, o incremento da saída da RNMC controladora é definido como mostrado na Eq. 3.7, onde Ω(t) representa a velocidade angular do eixo do motor CC.

$$U^*(t) = U(t) + (-\zeta \nabla J_D w(t)) \tag{3.5}$$

$$U^*(t) = U_a(t) - \zeta \frac{\partial J_D(t+1)}{\partial U(t)} = U_a(t) + \zeta E(t+1) \frac{\partial \Omega(t+1)}{\partial U(t)} \tag{3.6}$$

$$\Delta U(t) = U^*(t) - U_a(t) \tag{3.7}$$

Capítulo IV

Sistemas Nebulosos

4.1 Introdução

Neste capítulo, inicialmente define-se a Lógica Nebulosa e compara-se a mesma com a Lógica Booleana. A seguir, apresentam-se os fundamentos da lógica Nebulosa. Mostra-se como é possível expressar o conhecimento usando Lógica Nebulosa. Após a apresentação sucinta do controlador nebuloso padrão define-se a classificação dos sistemas neuro-nebulosos existentes.

4.2 Lógica Nebulosa

A Lógica Nebulosa (LN), lógica difusa ou lógica "Fuzzy", foi proposta em 1965 por Lotfi A. Zadeh, professor de Ciência da Computação da Universidade da Califórnia em Berkeley [ZAD 73]. A LN utiliza uma faixa contínua de valores no intervalo [0, 1]. A LN pode ser considerada um desenvolvimento da Lógica Booleana que só admite dois valores (verdadeiro ou falso). A LN é utilizada com êxito onde existe um sistema de difícil modelagem exata. Ela avalia um modelo inexato de um sistema e permite que o sistema seja manipulado por um operador humano ou um especialista. O sistema pode ser ambíguo e impreciso.

A LN foi desenvolvida para representar os modos de raciocínio usando um sistema de variáveis lingüísticas mais próxima das formas de expressões humanas. Por exemplo, "positivo grande", "negativo pequeno" etc. Na Lógica Nebulosa o raciocínio exato corresponde a um caso limite do raciocínio aproximado.

A LN possibilita a manipulação de informações vagas, ambíguas ou imprecisas. Dessa forma ela pode ser usada como base no desenvolvimento de métodos e algoritmos tanto para modelagem como para controle de processos. Ela possibilita a redução da complexidade de projeto e implementação, tornando-se uma solução para situações de controle e modelagem que são difíceis de serem abordadas pelas técnicas clássicas.

Na teoria clássica de conjuntos, um elemento pertence ou não pertence a um determinado conjunto. Já na teoria dos conjuntos nebulosos, um dado elemento pode pertencer parcialmente a um determinado conjunto, existindo um grau de pertinência relativa do elemento ao conjunto. Por exemplo, a variável lingüística *altura* poderá assumir como valor qualquer um dos membros do conjunto{*baixa, média, alta*}. Para se atribuir um significado aos termos lingüísticos, é associado a cada um destes um conjunto nebuloso definido sobre um universo de discurso comum. Na Fig. 4.1 x, na abscissa, representa a altura em metros e $\mu(x)$, na ordenada, representa a função de pertinência correspondente. $\mu(x)$ pode representar a variável lingüística altura como Baixa, Média ou Alta.

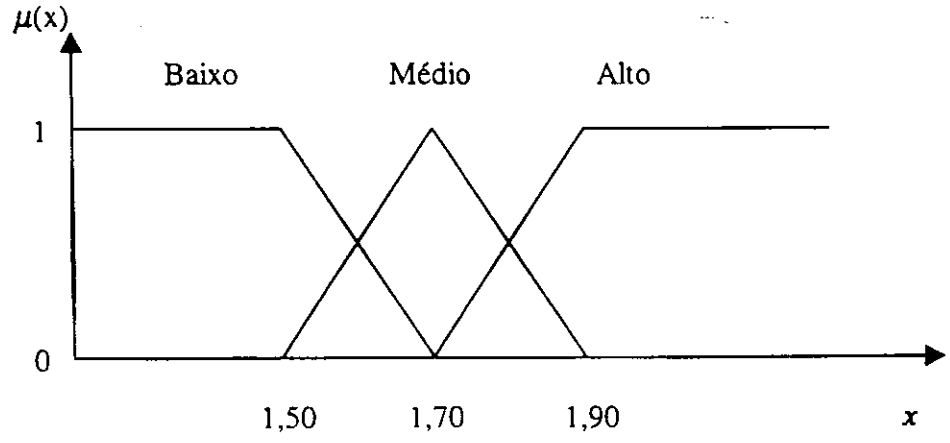


Fig. 4.1 – A Variável lingüística altura

Essas atribuições representam a noção que o homem possui das grandezas físicas, apesar de ele não saber o quanto exatamente a altura é média, ele é capaz de tomar decisões baseado nesse conhecimento "difuso" da grandeza física em análise.

Nas teorias de controle clássicas e modernas, normalmente, derivar o modelo matemático que descreve o sistema, é o primeiro passo para se implementar o controle

do sistema [GOM 94]. Isso requer um conhecimento pormenorizado do sistema a ser controlado. Se o sistema for muito complicado, esse conhecimento pode não ser disponível. Para uma grande quantidade de processos bem definidos, se aplicam muito bem as teorias de controle clássicas, tais como: controle ótimo, sistemas lineares estocásticos, controle linear multivariável, algumas classes de problemas não-lineares determinísticos, entre outras. Contudo, estas técnicas geralmente não são capazes de resolver problemas reais de modelagem. Exemplificando: em diversas situações só é conhecida uma grande quantidade de informação essencial de forma qualitativa, estando disponíveis apenas critérios de desempenho em termos lingüísticos. Este panorama leva à imprecisão e falta de exatidão que inviabilizam a maioria das teorias de controle clássicas utilizadas até agora.

4.3 Fundamentos da lógica nebulosa

A LN possibilita uma caracterização mais ampla que a lógica booleana, na medida em que sugere que alguns elementos são mais membros de um conjunto do que de outros. O grau de pertinência de um elemento a um conjunto nebuloso pode então assumir qualquer valor entre 0 e 1. O valor 1 representa completa pertinência e o valor 0 indica uma completa exclusão. Essa generalização aumenta significativamente o poder de expressão da função característica (uma função que diz o grau de pertinência $\mu A(x)$ de um elemento x pertencente a um universo U com respeito a um conjunto A, onde $A \subseteq U$).

Assim, se Y é uma variável que toma valores num universo U, então a distribuição de possibilidade associada com Y pode ser vista como uma condição elástica nos valores que podem ser atribuídos a Y. Por exemplo, se F é um subconjunto nebuloso de U caracterizado pela função de pertinência μf: $U \rightarrow [0, 1]$, então a declaração "Y é F" é traduzido numa distribuição de possibilidade para Y sendo igual a F. Pode-se escrever:

O conjunto nebuloso A é definido como o par ordenado $A=\{x, \mu A(x)\}$

Onde $x \in X$ e $0 \leq \mu A(x) \leq 1$. A função de pertinência $\mu A(x)$ descreve o grau a que o objeto x pertence ao conjunto A. Quando $\mu A(x) = 0$, representa nenhuma pertinência e $\mu A(x) = 1$, representa pertinência total.

Por exemplo, seja X a representação da idade de uma pessoa. O subconjunto A de X que representa as pessoas jovens ($\mu A(x)$) é um conjunto nebuloso com a função de pertinência mostrada na Fig.4.2. Observe-se que $\mu A(x) \equiv 1$ para pessoas com idade até 25 anos. A partir de 25 anos, $\mu A(x)$ começa a decrescer. Para $X = 75$ anos, $\mu A(x) \equiv 0.1$.

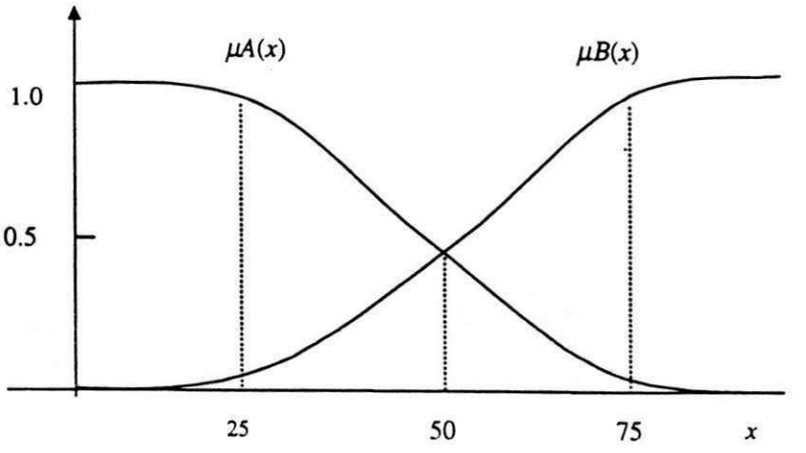


Fig.4.2 A Função de pertinência

Um conjunto nebuloso pode ter mais de uma representação, assim podemos descrever a idade de uma pessoa também por um subconjunto B de X representando as pessoas velhas ($\mu B(x)$), cuja função de pertinência é mostrada na Fig. 4.2. Observe que $\mu B(x) \cong 0.9$ para pessoas com idade acima de 75 anos. Também observe que para X = 50 anos, $\mu A(x) = \mu B(x)$, ou seja, com 50 anos uma pessoa tem o mesmo grau de juventude ou velhice.

As operações no conjunto nebuloso são extensões das operações usadas em conjuntos tradicionais. Assumindo $A \in X$ e $B \in X$. O símbolo $\in$ significa "pertence" e o símbolo $\forall$ significa "para todos".

Comparação: $A = B$

$A = B$ se $\mu A(x) = \mu B(x)\ \forall\ x \in X$

Contido: $A \subset B$

$A \subset B$ se $\mu A(x) < \mu B(x)\ \forall\ x \in X$

União: $A \cup B$

$\mu A(x) \cup \mu B(x) = max(\mu A(x), \mu B(x))\ \forall\ x \in X$

A Fig.4.3 mostra as curvas das funções de pertinência $\mu A(x)$e $\mu B(x)$. A curva em negrito representa $A \cup B$.

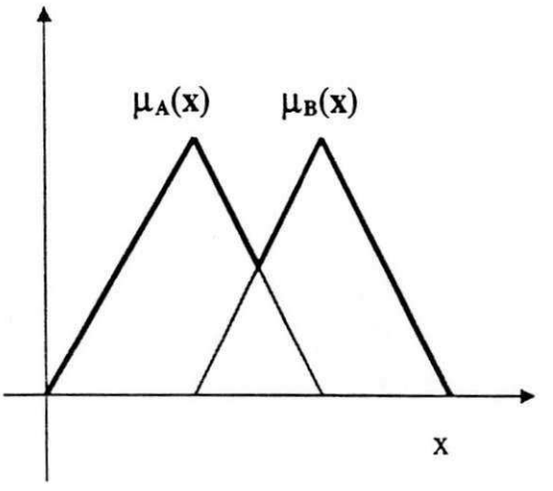


Fig.4.3 - A União.

Interseção: $A \cap B$

$\mu A(x) \cap \mu B(x) = min(\mu A(x), \mu B(x)) \; \forall \; x \in X$

A Fig.4.4 mostra as curvas das funções de pertinência $\mu A(x)$ e $\mu B(x)$. A curva em negrito representa $A \cap B$.

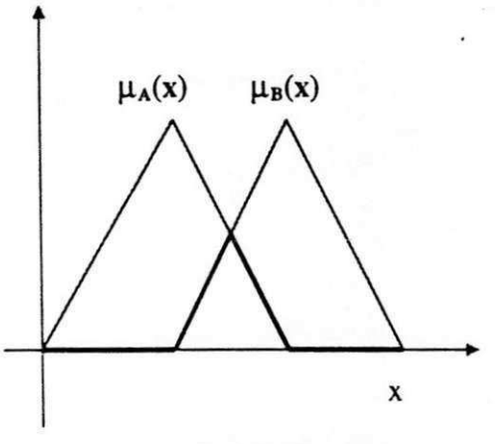


Fig.4.4 - A Intersecção.

Complemento: A'

$\mu A'(x) = 1 - \mu A(x) \; \forall \; x \in X$

A Fig.4.5 mostra as curvas das funções de pertinência $\mu A(x)$ e $\mu A'(x)$. A curva em negrito representa A'.

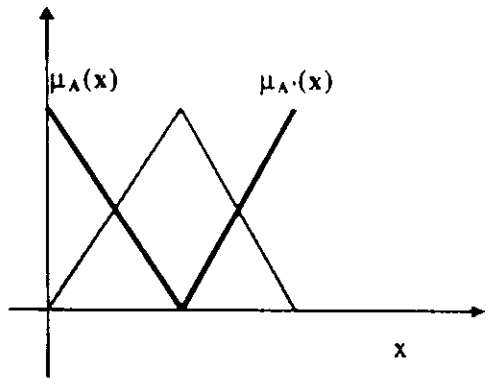


Fig.4.5 - O Complemento

4.4 Expressão Nebulosa do conhecimento

Para expressar conceitos é muito comum o uso de elementos qualitativos ao invés de valores quantitativos [GOM 94]. Elementos típicos incluem "mais ou menos", "alto", "não muitos", "médio", entre outros. Estas idéias são capturadas pela definição de variável lingüística. Uma variável lingüística tem por característica assumir valores dentro de um conjunto de termos lingüísticos, ou seja, palavras ou frases. Assim, ao invés de assumir instâncias numéricas, uma variável lingüística assume instâncias lingüísticas. Por exemplo, uma variável lingüística temperatura poderá assumir um dos membros do conjunto {baixa, média, alta}. Para se atribuir um significado aos termos lingüísticos, associa-se a cada um deles um conjunto nebuloso definido sobre um universo de discurso comum .

Uma das formas mais comuns de expressar o conhecimento é por meio de regras do tipo condição-ação. Exemplificando, um conjunto de condições que descrevem uma parcela observável das saídas do processo são associadas com uma ação de controle que irá manter ou levar o processo às condições de operações desejadas. A idéia aqui é representar o conhecimento por meio de um conjunto de regras nas quais as condições são dadas a partir de um conjunto de termos lingüísticos associados a variáveis de saída/entrada do processo (as quais são entradas do controlador). As ações de controle ou as saídas são expressas de modo similar para cada variável de controle (saída). Regras do tipo *se-então-senão* são freqüentemente chamadas de regras de controle nebulosas. Tipicamente, uma proposição lingüística sobre o valor das variáveis de entrada é, por exemplo: *o erro é grande e positivo*. De modo análogo, uma típica ação de controle é uma descrição lingüística, como por exemplo: *aumente um pouco a tensão de armadura.*

4.5 Sistemas de Controle Nebulosos

No controle Nebuloso não é necessário modelar o processo, e sim, modelar as ações a partir do conhecimento de um especialista. Essa é, portanto, uma abordagem diferente dos métodos convencionais de controle de processos [ÄST 89], pois os mesmos são desenvolvidos via modelagem matemática dos processos de modo a derivar as ações de controle como função do estado do processo.

A estrutura básica de um controlador nebuloso é mostrado na figura 4.6.

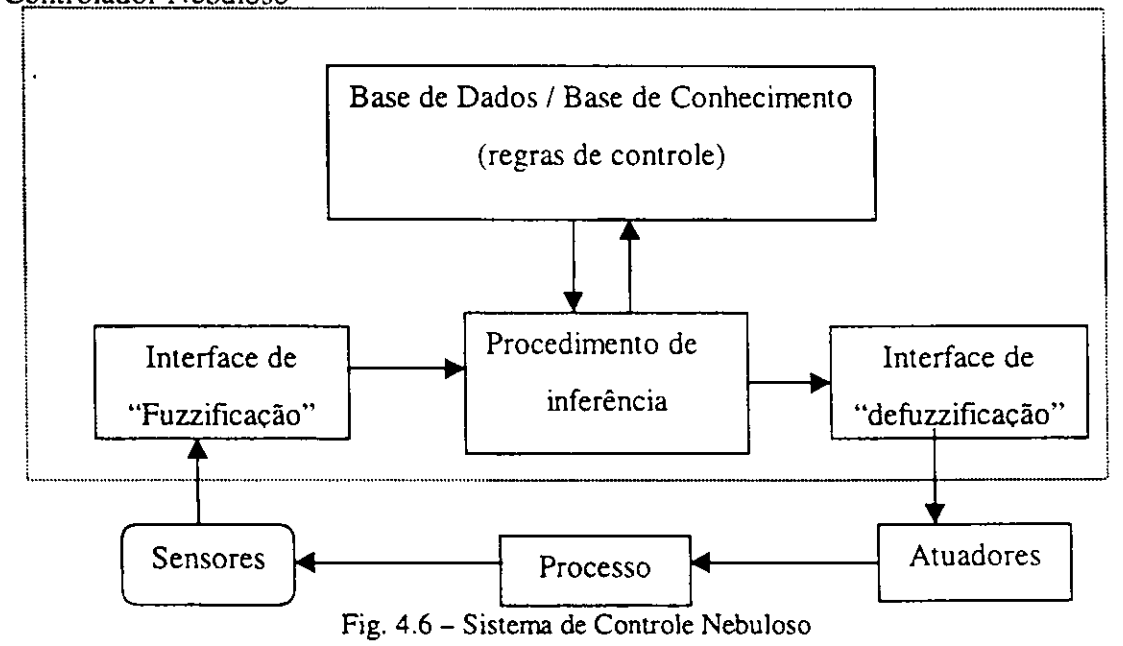


Fig. 4.6 – Sistema de Controle Nebuloso

O núcleo do controlador nebuloso só analisa variáveis nebulosas. As informações têm que ser transformadas na forma Nebulosa ou "fuzzyficadas" (transformadas em conjuntos nebulosos). A interface de "fuzzyficação" recebe os valores das variáveis de entrada (vindos dos sensores), faz um escalonamento para dimensionar os valores a universos de discursos normalizados e "fuzzyfica" os valores (transformando números em conjuntos nebulosos), para torná-los instâncias de variáveis lingüísticas. A base de conhecimento consiste de um conjunto de regras que caracterizam as estratégias de controle e seus objetivos. A base de dados armazena as informações necessárias sobre as discretizações, as normalizações dos universos de discurso, as partições nebulosas dos espaços de entrada e saída e as definições das funções de pertinência. O procedimento de inferência atua sobre os dados nebulosos de entrada, juntamente com as regras, para

inferir as ações de controle nebulosas, usando o operador de implicação nebulosa e as regras de inferência da lógica nebulosa. A interface de "defuzzificação" atua sobre as ações de controle nebuloso inferidas, transformando-as em ações de controle não-nebuloso, efetuando, em seguida, um escalonamento para compatibilizar os valores normalizados vindos do passo anterior com os valores dos universos de discursos reais das variáveis.

Deve-se determinar uma ação de controle não-nebuloso para ser enviada ao controle logo após se inferir a ação de controle nebuloso. A ação de controle não-nebuloso escolhida deve ser a que represente melhor a decisão nebulosa. Não há nenhum procedimento sistemático para escolher a estratégia de "defuzzificação", as mais comuns são: o critério do máximo (MAX), o qual escolhe o ponto onde a função inferida tem o seu máximo, a média dos máximos (MDM), a qual representa o valor médio dentre todos os pontos de máximos se existir mais de um máximo e o método de centro de área (CDA) o qual retorna as coordenadas do centro da área da função inferida.

4.6 Classificação dos Sistemas Neuro-nebulosos

Segundo [NAU 94], quando se usa sistemas nebulosos em automatização, surge a tarefa de definir conjuntos nebulosos e regras nebulosas. Isso pode ser difícil em alguns domínios. Nesses domínios o controlador resultante tem de ser refinado. Os modelos neuro-nebulosos podem ajudar nesse processo de refinamento pela adaptação de conjuntos nebulosos e criação de regras nebulosas.

Um possível tipo de combinação entre RNA e controladores nebulosos utiliza ambos os modelos independentemente. A combinação é determinada pelo fato de que uma rede neural está aprendendo ou otimizando diversos parâmetros de um controlador nebuloso antes de o controlador ser usado ("off-line"), ou enquanto ele opera ("on-line"). Esse tipo de combinação é chamada de controladores neuro-nebulosos cooperativos.

Alguns tipos de modelos neuro-nebulosos [NAU 94]:

a) A rede neural deriva as funções de pertinência ("membership") do treinamento dos dados. Isso pode ser feito pela aprendizagem de parâmetros especiais ou pelo uso de diversas redes neurais para aproximar as funções diretamente. Os conjuntos nebulosos

são aprendidos "off-line" e são usadas regras nebulosas predefinidas para implementar um controlador nebuloso.

b) A rede neural deriva regras de controle lingüístico dos dados de treinamento. Isso é feito freqüentemente pelo uso de algoritmo de "clustering", tais como: mapas de feição de Kohonem ou arquiteturas neurais similares. A aprendizagem é feita "off-line", e os conjuntos nebulosos devem ser definidos de algum outro modo.

c) A rede neural adapta parâmetros dos conjuntos nebulosos "online", isto é, enquanto o controlador nebuloso opera. É necessário conhecer as regras nebulosas e os conjuntos nebulosos iniciais. Além do mais, uma medida de erro tem de ser definida para que guie o processo de aprendizagem.

Além desses tipos de modelos cooperativos pode-se pensar em abordagens onde as redes neurais são utilizadas para pré-processar ou pós-processar as saídas de um controlador nebuloso, como o sistema de controle inteligente descrito nos Capítulos V e VI. Essas abordagens não cuidam da otimização do controlador em si, mas são concernentes acerca da otimização do desempenho de controle de todo o sistema. A aprendizagem toma lugar apenas na parte da rede neural, o controlador nebuloso permanece imodificável. Por essa razão, essas abordagens não são chamadas de modelos neuro-nebulosos. Elas podem ser chamadas de abordagens de pré ou pós-processamento neural.

Um outro tipo de combinação entre redes neurais e controladores nebulosos é chamado de modelo neuro-nebuloso híbrido. Essa abordagem cria uma arquitetura homogênea que é costumeiramente orientada pela rede neural. Isso pode ser feito interpretando um controlador nebuloso como uma rede neural especial.

Capítulo V

Posicionamento do Pêndulo invertido

5.1 Introdução

Neste capítulo, inicialmente discute-se a utilização dos sistemas físicos tipo pêndulo e descreve-se o modelo matemático do pêndulo simples. Em seguida apresenta-se o controle neuro-nebuloso do pêndulo invertido. Finalmente apresentam-se um SCI para controlar o pêndulo, as estratégias utilizadas e os resultados experimentais do equilíbrio do pêndulo.

5.2 Controle neural de um pêndulo invertido

Os sistemas físicos do tipo pêndulo são utilizados freqüentemente para demonstrar e verificar experimentalmente o desempenho de algoritmos de controle de sistemas não-lineares. A escolha do pêndulo deve-se às suas características não-lineares e pelo seu comportamento que apresenta modos de equilíbrio instáveis e estáveis.

Alguns autores já estudaram como posicionar o pêndulo na posição vertical para cima (estabilizar o sistema pêndulo invertido). Batur & Kasp [BAT 91] mostraram alguns resultados obtidos a partir de simulação de um sistema nebuloso usado para estabilizar um sistema carro e pêndulo invertido, sujeito a uma pequena variação no ângulo θ do pêndulo em relação à normal. Ishida et alii [ISH 91] mostraram uma RNMC usada para estabilizar um sistema carro e pêndulo invertido também sob pequenas variações no ângulo entre o pêndulo e a normal. Lin et alii [LIN 92] mostraram resultados experimentais de um sistema híbrido, nebuloso e convencional, capaz de estabilizar um sistema carro e pêndulo invertido. Eles definiram os quadrantes de posicionamento do braço do pêndulo e mostraram como são desenvolvidas as regras do controlador nebuloso. Eles usaram um controlador convencional com realimentação para

a sintonização fina do posicionamento do pêndulo quando o braço do pêndulo estiver na posição para cima. Furuta et alii [FUR 91] apresentaram um sistema experimental de um pêndulo invertido e um controlador capaz de posicionar o braço do pêndulo na posição para cima. Eles usaram um modelo não-linear a fim de desenvolver o controlador para deslocar o braço do pêndulo até próximo da posição para cima, e usando um modelo linearizado do sistema pêndulo invertido, desenvolveram um controlador linear capaz de movimentar o pêndulo e posicionar o seu braço junto ao ponto de maior instabilidade.

Outros autores estudaram o pêndulo de duplo braço. Buchner et alii [BUC 88] apresentam resultados obtidos com simuladores para o posicionamento para cima de um pêndulo invertido de dois braços. Jim et alii [JIN 92] apresentam resultados de simulação obtidos para o posicionamento para cima, usando controladores neurais, de um pêndulo invertido de duplo braço.

Cavalcanti & Lima & Deep [CAV 93] e [CAV 94b] mostraram que sob certas condições, um controlador neural direto pode ser treinado (ou adaptado) em tempo real sem qualquer treinamento prévio, "off-line". Recentemente, Cavalcanti et alii [CAV 94a] apresentaram resultados experimentais obtidos no posicionamento para cima de um pêndulo invertido na sua posição de máxima instabilidade usando um controlador neural direto.

5.3 O pêndulo simples

Na Fig. 5.1 é mostrado o esquema geral de um pêndulo simples. A equação 5.1, refere-se ao torque do pêndulo, onde, P representa o peso e M a massa, g aceleração da gravidade, L comprimento do braço do pêndulo, e θ representa o ângulo entre o braço e a normal ao suporte do pêndulo que passa pelo joelho j1 do pêndulo que é acoplado ao motor CC.

$$Tl(t) = (P/g)\, L^2\, d^2\theta(t)/dt^2 + P\, L\, sen(\theta(t)) \qquad (5.1)$$

Na Fig. 5.2 é mostrado o esquema de montagem de um braço rígido, com um peso a ele acoplado (círculo menor na Fig. 5.2) e ao eixo do motor CC (círculo maior da Fig. 5.2). Nessa montagem o eixo do motor é visto de perfil. No esquema da Fig. 5.2 são mostradas 4 posições especiais do braço do pêndulo, representadas pelos ângulos θ

entre o braço do pêndulo e a normal. Um pêndulo é dito ser um pêndulo invertido se é permitido o seu posicionamento no ângulo $\theta=\pi$ (ou $|\theta|>\pi/2$), como mostrado na Fig. 5.2a. Considera-se que os ângulos são medidos no sentido horário.

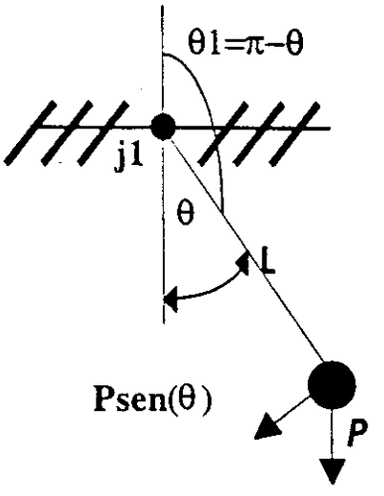


Fig. 5.1 - Esquema geral do pêndulo simples.

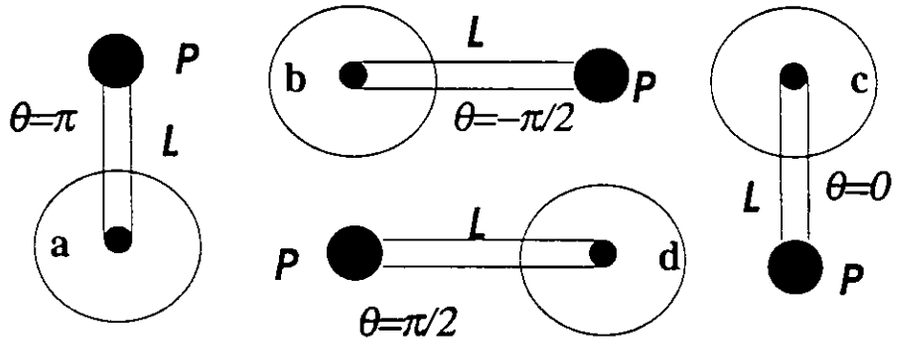


Fig. 5.2 - O Posicionamento do eixo do motor CC.

Define-se t como a representação discreta do tempo. Assume-se que $\theta r(t+1)$ representa o valor de referência ou o valor desejado de $\theta(t+1)$ para a posição do eixo do motor CC. $\theta r(t+1)$ e $\theta(t+1)$ são representados em p.u. com valor de base 2π rd, $U(t)$ (tensão de armadura) representado em p.u. com valor de base 12 V, e $\Omega(t)$ (velocidade angular) representado no sistema p.u. com valor de base 2 rps. Na Fig. 5.3 é mostrado o controlador neural adaptativo direto de posição do eixo do motor CC com $X1=U(t)$, $X2=\theta(t)$ e $X3=\theta r(t)$. W_{inp} representa os pesos entre os neurônios de entradas e os neurônios da camada escondida e W_{out} representa os pesos entre os neurônios da camada escondida e os neurônios da camada de saída.

2) Com o pêndulo em repouso, o torque $|Tl(t)|$ será máximo para $|\theta(t)| = \pi/2$ rd (ver Figura 5.2b e 5.2d).

3) O pêndulo estará no estado estável com $\theta(t) = 0$ rd (Fig. 5.2c). Qualquer movimento no pêndulo em torno de $\theta(t) = 0$ rd o fará retornar ao ponto $\theta(t) = 0$ rd.

4) O módulo do torque $Tm(t)$ gerado pelo motor CC deverá aumentar para estabilizar (posicionar e imobilizar) o pêndulo quando $|\theta(t)|$ crescer de 0 a $\pi/2$.

5) O módulo do torque $Tm(t)$ gerado pelo motor CC deverá diminuir para estabilizar o pêndulo quando $\theta(t)$ crescer de $|\theta(t)| = \pi/2$ a $|\theta(t)| = \pi$.

Na Fig. 5.4 é mostrado um círculo com quatro quadrantes representando o torque de carga $Tl(t)$ gerado pelo pêndulo (ver [LIN 92]) para diferentes ângulos $\theta(t)$ do pêndulo. O torque $Tm(t)$ gerado pelo motor CC (diretamente proporcional à tensão da armadura do motor CC) para estabilizar o pêndulo deve ter a mesma intensidade de $Tl(t)$ mas com sinal contrário. Os quadrantes foram definidos a partir do ponto de instabilidade em $\theta(t)=\pi$. Q1 foi definido no intervalo $-\pi \leq \theta(t) < -\pi/2$, Q2 foi definido para $-\pi/2 < \theta(t) < 0$, Q3 foi definido no intervalo $0 < \theta(t) < \pi/2$, e Q4 foi definido no intervalo $\pi/2 < \theta < \pi$. Os quadrantes podem ser vistos na função de pertinência θm de $\theta(t)$ mostrada na Fig. 5.5. (θm representa a função de pertinência dos quadrantes de $\theta(t)$ e $\Delta U(t)$ representa o incremento em $U(t)$).

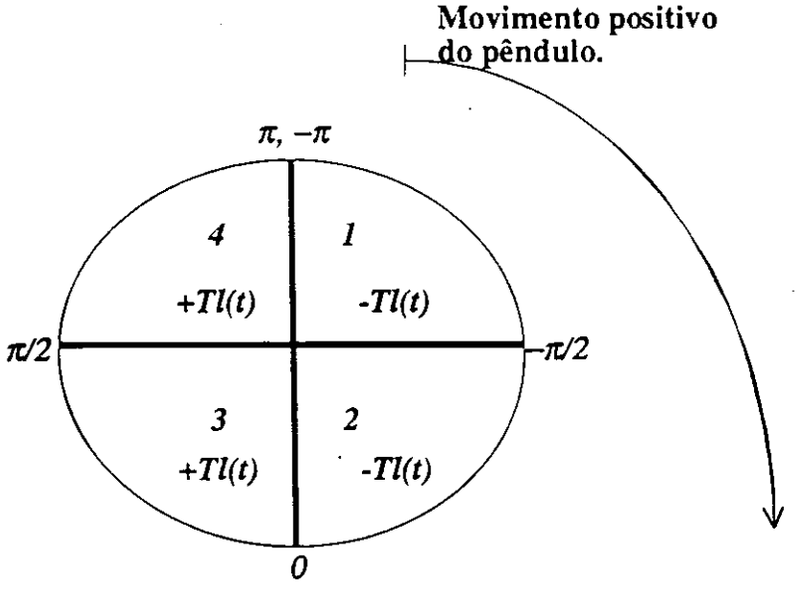


Fig. 5.4 - O Círculo de Torque

O índice de desempenho $I(u)$ usado para a adaptação da RNMC do sistema motor CC e pêndulo invertido é definido na Eq. 5.2.

$$I(U) = \frac{1}{2}(e_{t+1})^2 = \frac{1}{2}[\theta r(t+1)-\theta(t+1)]^2 \qquad (5.2)$$

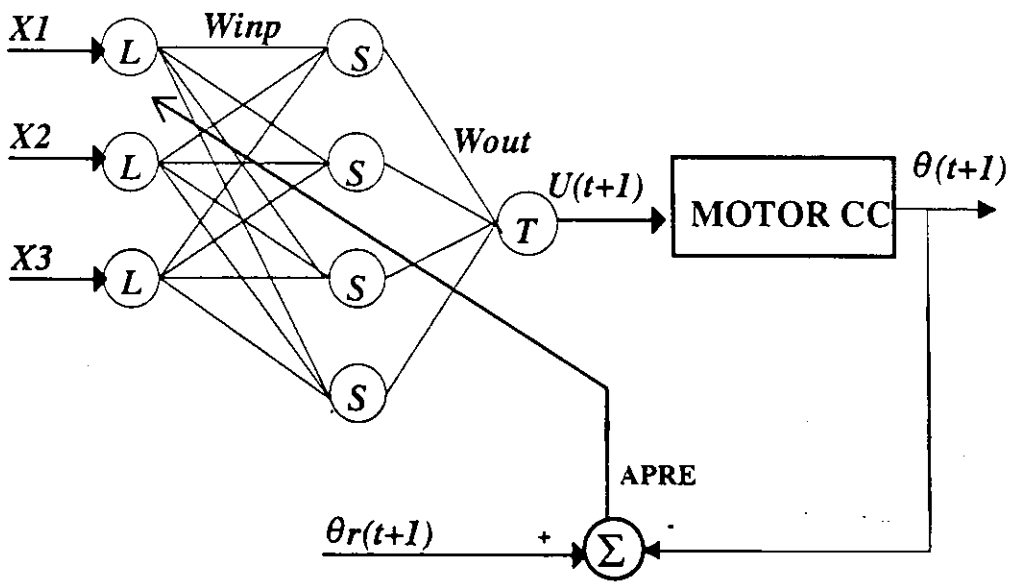


Fig. 5.3 - Controlador neural de posição do eixo do motor CC.

5.4 - Posicionamento do pêndulo invertido.

Na implementação experimental usou-se um pêndulo com comprimento $L = 0.1$ m e peso $P = 0.1$ Kg a fim de que, para determinados valores do ângulo $\theta(t)$, o torque $Tl(t)$ seja maior que o atrito viscoso do eixo do motor CC. Assim, quando o torque do motor for igual a zero, o pêndulo é movido para a posição em que $\theta = 0$.

Na Eq. 5.1, o torque da carga $Tl(t)$ depende do deslocamento e também da aceleração do pêndulo. A partir das Fig. 5.1 e 5.2, e considerando o pêndulo em repouso (estado estável) podem ser observadas as seguintes características do sistema pêndulo invertido:

1) O pêndulo estará no estado instável em $\theta(t) = \pi$ rd (Figura 5.2a). Qualquer deslocamento do pêndulo fará $Tl(t) \neq 0$, levando o pêndulo ao ponto $\theta(t) = 0$ rd

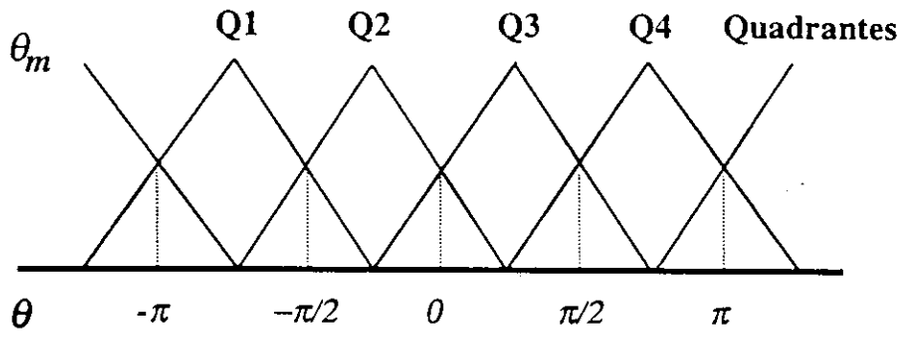


Fig. 5.5 - Função de pertinência do círculo

Considerou-se o torque *Tl(t)* positivo nos quadrantes Q3 e Q4 e torque *Tl(t)* negativo, nos quadrantes Q1 e Q2. Isto é, o torque *Tm(t)*, a ser gerado pelo motor CC para estabilizar o pêndulo, deve ser positivo quando o braço do pêndulo estiver posicionado nos quadrantes Q1 e Q2 e deve ser negativo quando o braço do pêndulo estiver posicionado nos quadrantes Q3 e Q4.

O posicionamento $\theta(t)$ do eixo do motor CC segue uma referência $\theta r(t)$. No caso do pêndulo invertido, para cada quadrante do círculo de torque da Fig. 5.4, existem diferentes valores de *U(t)* capazes de sustentar o pêndulo invertido em $\theta(t) = \theta r(t)$. O valor exato de *U(t)* não é conhecido. Baseado nas características do sistema pêndulo invertido representadas pela equação 5.1, no círculo de torque da Fig. 5.4, e na função de pertinência da Fig. 5.5, foram desenvolvidas as seguintes regras nebulosas para o treinamento seguido de controle neural.

Regra (1): Quando o braço do pêndulo estiver parado em Q1 ou Q2, *U(t)* deve ser positivo;

if θm == Q1 **or** θm == Q2 **then** $U(t)>0$;

Regra (2): Quando o braço do pêndulo estiver parado em Q3 ou Q4, *U(t)* deve ser negativo;

if θm == Q3 **or** θm == Q4 **then** $U(t)<0$;

Regra (3) e (4): Quando o braço do pêndulo estiver em Q1 ou Q4, $|\Delta U(t)|$ será crescente para posicionar e sustentar o pêndulo de $|\theta| = \pi$ rd a $|\theta| = \pi/2$ rd

if θm == Q4 **then** $\Delta U(t)<0$;

if θm == Q1 **then** $\Delta U(t)>0$;

Regra (5) e (6): Quando o braço do pêndulo estiver em Q2 ou Q3, $|\Delta U(t)|$ será decrescente para posicionar e sustentar o pêndulo de $|\theta| = \pi/2$ rd a $|\theta| = 0$ rd;

if θm == Q2 **then** $\Delta U(t)>0$;

if θm == Q3 **then** $\Delta U(t)<0$;

Durante o treinamento do controlador neural foram seguidas algumas regras adicionais, que não podem ser consideradas necessariamente regras nebulosas, obtidas do círculo de torque da Fig. 5.4. Essas regras, consideradas como regras de apoio, foram indiretamente utilizadas no desenvolvimento das diferentes estratégias de controle do pêndulo invertido.

1) Não foram usados como referência os valores de $|\theta r(t)| = \pi/2$ *rd* que é o ponto de maior torque gerado pelo pêndulo. Nesses pontos, a corrente na armadura do motor CC se aproxima do seu limite máximo.

2) Treinou-se a RNMC do controlador neural com valores do ângulo de referência próximo do ponto de estabilidade do pêndulo, ou $\theta r(t) \cong 0$ rd (ver a Fig. 5.4)

3) Cavalcanti et alii [CAV 94a] desenvolveram o conceito de estado passivo para permitir o treinamento em tempo real da RNMC. Basicamente, eles supuseram que o sistema representado pelo conjunto motor mais controlador neural (sistema global) tem pelo menos um estado de equilíbrio, e sem perda de generalidade, a origem pode ser considerada um desses estados. Representando-se o sistema motor CC na forma discreta por $x=f(x,u)$, eles definiram:

Definição 1 - Um ponto $x_e \in \Re^n$ é um ponto de equilíbrio de $x = f(x,u)$ se existir uma entrada u_e tal que: $x_e = f(x_e,u_e)$.

Assim, particularmente, a RNMC da Fig. 5.3 pode ser treinada, usando-se a tensão de excitação da armadura u e o estado x_e, para fornecer na sua saída o valor u_e

correspondente ao estado de equilíbrio. Baseado nessa estratégia, Cavalcanti et alii [CAV 94a] definiram o Estado Passivo do sistema global como:

Definição 2 - Um sistema dinâmico está no estado passivo quando estiver no ponto de equilíbrio com $x_e = f(x_e, u_e)$ e o controlador neural garante a excitação de controle u_e.

Os valores de $\theta r(t)$ foram gerados para garantir os estados passivos durante o treinamento da RNMC. Isso garantiu que o pêndulo se movimentasse lentamente o que permitiu a simplificação na geração dos sinais usados no treinamento da RNMC.

O Algoritmo 5.1 foi desenvolvido a partir das características do sistema pêndulo invertido, para treinamento seguido de controle da RNMC.

Algoritmo 5.1

Treinamento da RNMC com a dinâmica do pêndulo

ETAPA 1 - Inicialização dos parâmetros da RNMC.
1)Geram-se aleatoriamente os pesos Winp e Wout entre os limites 0.33 e -0.33.
2) Nos parâmetros da sigmóide, faz-se: $\Theta = \beta = 1$ *e* $T = 0$.
3)Atribui-se $\mu = 0.1$ (fator de treinamento) e $\chi = 0.5$ (fator de adaptação do controlador).
ETAPA 2 - Treinamento em torno do ponto de estabilidade representado com $\theta = 0$.
4)Obtém $\theta r(t)$.
5)Usa $\theta r(t)$ capaz de calcular o novo $U(t)$ do ponto passivo da RNMC.
6)Calcula: $\Delta U(t) = \chi e(t+1)$ *e* $U(t+1) = U(t) + \Delta U(t)$. Treina a RNMC.
7)Repetem-se as etapas 5 e 6 até que $\lvert\theta r(t) - \theta(t)\rvert < \varepsilon$. (ε pequeno)

A Fig. 5.6a mostra as posições de treinamento do pêndulo em torno de $\theta(t) = 0$. A Fig. 5.6b mostra o pêndulo próximo da sua posição invertida.

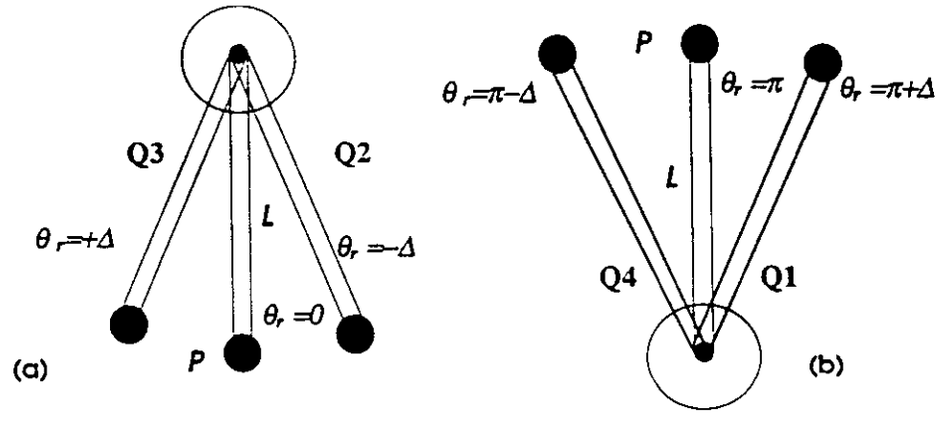


Fig. 5.6 - Posições de referência do braço do pêndulo para treinamento da RNMC.

As regras nebulosas 7, 8, 9 e 10 foram desenvolvidas para o posicionamento com segurança do eixo do motor CC. Para essas regras, definiu-se a variável nebulosa θp com a função de pertinência $\theta p = PG$ quando $\theta(t) > \pi$ (ou $\theta(t) > 1$ p.u.), e $\theta p = NG$ quando $\theta(t) < -\pi$ (ou, em P.U., $\theta(t) < -1$). Para a velocidade, definiu-se a variável nebulosa p com a função de pertinência $\Omega_p = PG$ quando $\Omega(t) > 1$ rps, e $\Omega p = NG$ quando $\Omega(t) < -1$ rps (ver Fig. 5.7).

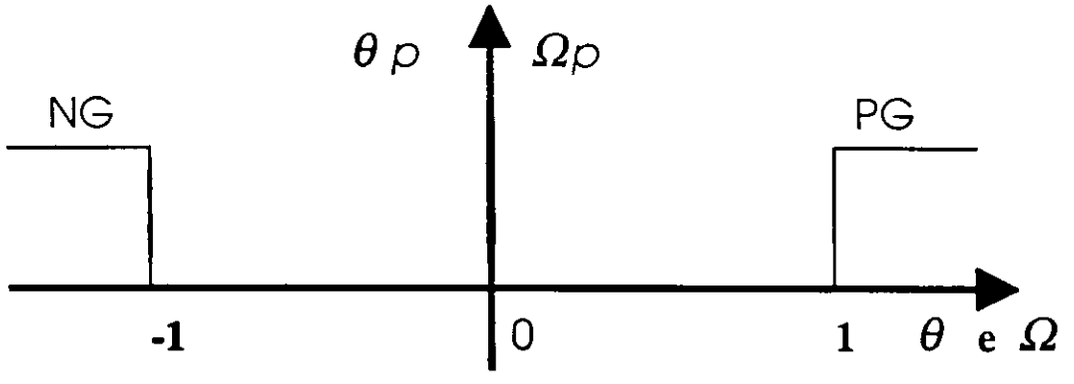


Fig. 5.7 - Função de pertinência de posição $\theta(t)$ e velocidade angular $\Omega(t)$.

Regra (7): Limite no valor do ângulo $\theta(t)$ em p.u.

if $\theta p == PG$ **then** $\theta(t) = \theta(t) - 1$

Regra (8): Limite no valor do ângulo $\theta(t)$ em p.u.

if $\theta p == NG$ **then** $\theta(t) = \theta(t) + 1$

Regra (9): Limite na velocidade do braço do pêndulo.

if $\Omega p == PG$ **or** $\Omega p == N$ **then** $U(t)=0$

5.5 O SCI e o Círculo das Estratégias

A partir das regras nebulosas de número 1 a 9 Cavalcanti et alii [CAV 95b] sugeriram a utilização de um conjunto de três estratégias para representar as posições do pêndulo e permitir o seu controle. Eles propuseram, baseados no Controlador nebuloso da Fig. 4.6 (Cap. 4), um SCI mostrado na Fig. 5.8.

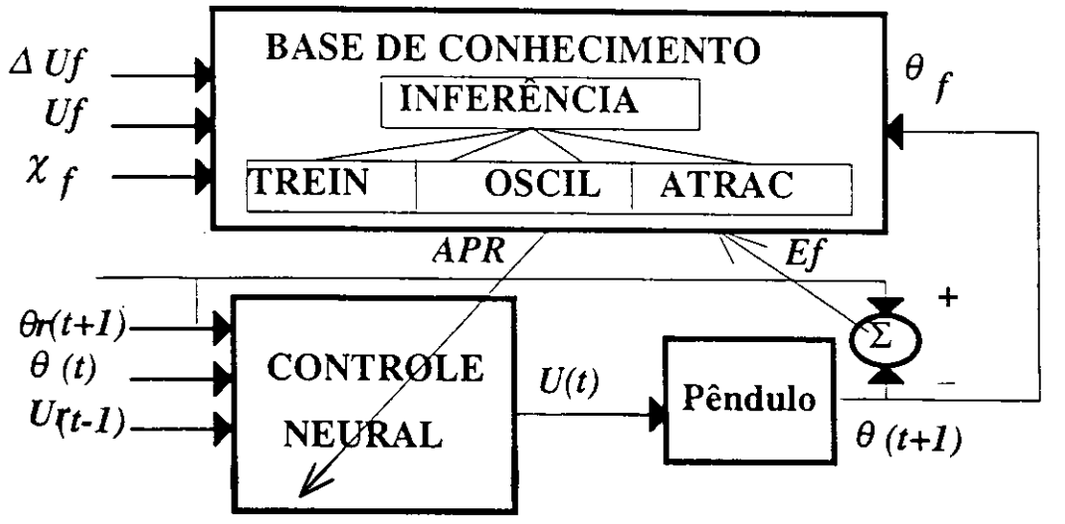


Figura 5.8 - Esquema geral do SCI.

No diagrama de estados, Fig. 5.9, é apresentado o círculo das estratégias desenvolvido para o SCI. Nesse círculo são mostradas três estratégias: TREIN, OSCIL e ATRAC.

Essas estratégias são ativadas de acordo com o estado do sistema pêndulo invertido (posição e velocidade do pêndulo). Inicialmente, a estratégia TREIN (treinamento) é ativada. Após a RNMC ter sido treinada, as demais estratégias são ativadas.

Essas estratégias são a base do treinamento da RNMC, de acordo com as posições de referências do braço do pêndulo.

TREIN - treinamento oscilante, é usada quando $\theta \cong 0$. Consiste em variar θ_r em torno do ponto de maior estabilidade ($\theta = 0$ = ponto passivo), de modo a treinar a rede neural.

OSCIL - posicionamento oscilante do braço do pêndulo. Ela é ativada nas seguintes condições: (1) - depois do estado de TREIN e (2) - O ângulo de referência é $\theta r(t) \cong 0$. Ela consiste em variar θ_r alternadamente com o seu módulo crescente, de

modo que o pêndulo oscile e ganhe energia o suficiente para atingir os quadrantes superiores.

ATRAC - posicionamento zero do braço do pêndulo. Ela é ativada nas seguintes condições: (1) - depois do estado de OSCIL; (2) - Quando $|\theta(t)| \cong \pi$ e $|\Omega(t)| \cong 0$. Ela consiste em impor a referência $\theta r = \pi$ de modo a atrair o pêndulo para a posição de equilíbrio instável, que é o objetivo final de controle.

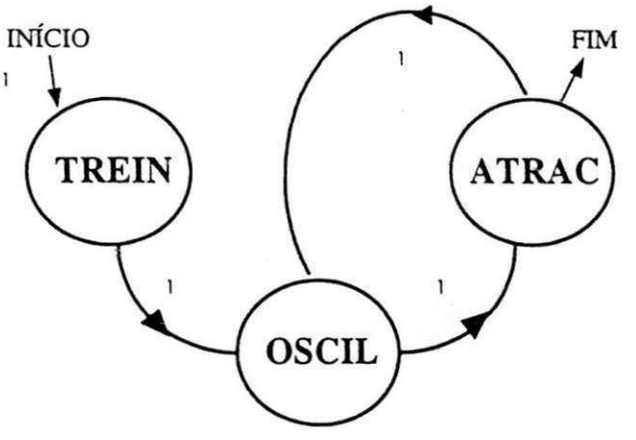


Fig.5.9 - O círculo das estratégias

A variável *NO* (número de oscilações), representada como NO_f na forma nebulosa (Fig. 5.10), é usada para indicar o fim da fase de treinamento. NOf possui três funções de pertinência: Z, M e G.

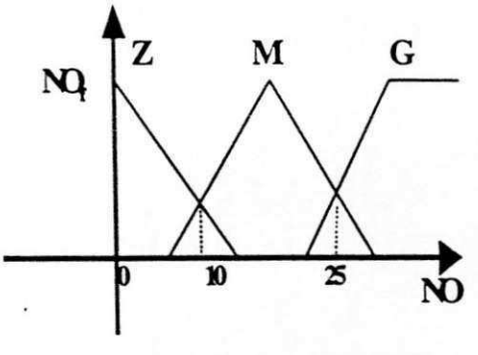


Fig. 5.10 - Funções de pertinência de NO.

Inicialmente no SCI, as variáveis booleanas que indicam a mudança de estados no círculo das estratégias têm os seguintes valores: INICIO="1" e TREIN = OSCIL = ATRAC = FIM = "0".

A regra 10 dispara a estratégia TREIN:

Regra (10): if INICIO **then** TREIN=1

A estratégia TREIN é representada pelas regras nebulosas 11 e 12 mostradas abaixo. δ é o valor do ângulo referência. A regra 13, que dispara a estratégia OSCIL, é executada quando o valor de *NO* é médio (*M* na Fig. 5.10). Isto é, o operador sugere o número de operações durante o treinamento.

Regra (11): if TREIN and $\theta(t) \approx \delta$ **then** $\theta r(t) = -\delta$ **and** $NO = NO + 1$

Regra (12): if TREIN and $\theta(t) \approx -\delta$ **then** $\theta r(t) = \delta$ **and** $NO = NO + 1$

Regra (13): if TREIN and $NOf == M$ **then** TREIN=0 **and** $OSCIL = 1$

A regra nebulosa 14 é usada na estratégia OSCIL. φ representa o incremento do ângulo referência. Define-se a variável nebulosa $\theta m = MAX$ quando $| \theta(t) | \cong \pi$ (pêndulo próximo da sua posição invertida).

Regra (14): if *OSCIL* and $\theta(t) \approx \theta r(t)$ **then** $\theta r(t) = -\theta r(t) - \varphi \cdot SGN(\theta r(t))$ **and** $NO = NO + 1$

A estratégia de atração (ATRAC) é acionada sob as condições mostradas na regra nebulosa 15. Isto é, quando o braço do pêndulo se aproximar do objetivo.

Regra (15): if *OSCIL* **and** $\theta m = MAX$ **then** $ATRAC = 1$ **and** $OSCIL = 0$ **and** $| \theta_r(t) | = \pi$.

Durante a estratégia ATRAC a variável *NO* é incrementada quando o pêndulo atravessa, com velocidade não nula, o ponto $\theta(t) \approx \theta r(t)$ (ver regra nebulosa 16).

Regra (16): if *ATRAC* **and** $\theta(t) \approx \theta r(t)$ **and** $\theta(t) - \theta(t-1) \neq 0$ **then** $NO = NO + 1$

A regra nebulosa 17 é usada para finalizar o posicionamento do pêndulo.

Regra (17): if $ATRAC == 1$ **and** $\theta == \theta r(t)$ **and** $\theta(t) - \theta(t-1) = 0$

then $FIM = 1$ **and** $ATRAC = 0$

Regra (18): if $ATRAC == 1$ **and** $NO_f == G$ **then** $OSCIL = 1$ **and** $ATRAC = 0$

A regra 18 é usada para, quando ultrapassar o número de oscilações permitidas, acionar a estratégia OSCIL.

A Fig. 5.11 mostra resultados experimentais obtidos durante o posicionamento do pêndulo invertido usando o SCI (Fig. 5.8). Os resultados da Fig. 5.11 foram obtidos com: (1) Fator de treinamento da RNMC $\mu=0.3$; (2) Número de oscilações da estratégia de treinamento $NOt=10$; (3) Peso $Pm=P$ no eixo do motor CC.

Na Fig. 5.11, inicialmente em $t=0$, a estratégia TREIN é acionada pela regra 10. Na estratégia TREIN, usando o Algoritmo de Controle e as regras nebulosas 11 e 12 a RNMC é treinada com valores de referência $\theta r(t)=\pm 18^{o}$ e $\chi_T = 0.9$ (Fator de adaptação do controlador neural da estratégia TREIN) $(\chi m=G)$ (intervalo $0=<t<=23s$ da Fig. 5.11). A seguir, a estratégia OSCIL é acionada (regra nebulosa 13). A estratégia OSCIL utiliza a regra nebulosa 14 e $\chi_O=0.3$ (Fator de adaptação do controlador neural da estratégia OSCIL) $(\chi m=P)$ (intervalo $23s=<t<=57s$ da Fig. 5.10). Aproximadamente em $t=57s$, o pêndulo estará próximo do objetivo e, usando a regra nebulosa 15, a estratégia ATRAC é acionada. A estratégia ATRAC usa $\chi_A = 0.3$ (Fator de adaptação do controlador neural da estratégia ATRAC) $(\chi_A=P)$. A estratégia ATRAC consegue posicionar o pêndulo na posição invertida (em $t\cong 70s$).

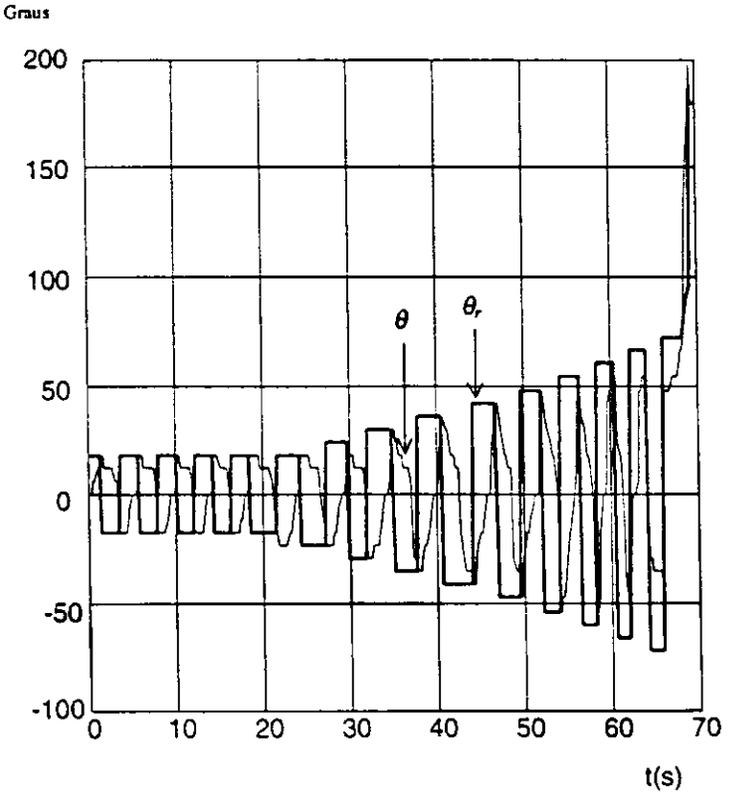


Fig. 5.11 - Resultados experimentais, Pm=P.

Capítulo VI

O SCI para o Problema de Agarrar um degrau

6.1 Introdução

Neste capítulo, inicialmente discute-se sobre os SCI's e apresenta-se um sistema de teste: um manipulador robótico de dois graus de liberdade. Em seguida, apresentam-se as estratégias de controle do SCI no movimento de agarrar o degrau de uma escada. Apresenta-se um escalonador em tempo real para controlar as operações executadas pelo SCI. Finalmente, apresentam-se os resultados experimentais obtidos com o SCI para controle do movimento do robô.

6.2 Sistemas de controle inteligente

Durante a década de 50, K. S. Fu introduziu o termo "Controle Inteligente" [SHO 91] ou "Sistema de Controle Inteligente" (SCI). SCI é um sistema que possui a habilidade de sentir o seu ambiente, processar as informações para reduzir as incertezas nos parâmetros do processo, planejar, gerar e executar ações de controle. Um SCI deve ser capaz de distribuir as tarefas de decisão entre um conjunto de executores de tarefas, usando intensivamente os computadores disponíveis para inferir o estado atual do sistema e detectar mudanças no estado interno e circunvizinho do sistema. Os SCIs são necessários em sistemas com variações na sua circunvizinhança, onde há mudanças nos modelos de referência, onde utilizam-se diferentes critérios de desempenho, e em sistemas que podem estar sujeitos a falhas de componentes.

Um SCI pode, de uma forma autônoma, atingir um objetivo de alto nível, enquanto os seus componentes, tais como objetivos de controle e modelo da planta, não são completamente definidos. Um SCI pode emular os processos mentais da adaptação e aprendizagem.

Para o desenvolvimento de um SCI necessita-se de um sistema de teste que possa ser controlado de uma forma inteligente. Um sistema simples, um braço de robô com dois graus de liberdade, programado para a tarefa de agarrar o degrau de uma escada, fornece um problema suficientemente complexo que precisa de um SCI para ser solucionado.

6.3 O Manipulador Robótico

Neste trabalho, considera-se o comportamento do robô com dois graus de liberdade como sendo semelhante ao de dois pêndulos simples conectados em série. Na Fig. 6.1 é mostrado o diagrama simplificado do manipulador robótico que se compõe de dois pêndulos conectados em série, um deles conectado à extremidade do outro. A cinemática e a dinâmica de manipuladores semelhantes ao protótipo desenvolvido são descritas por [CRA 86].

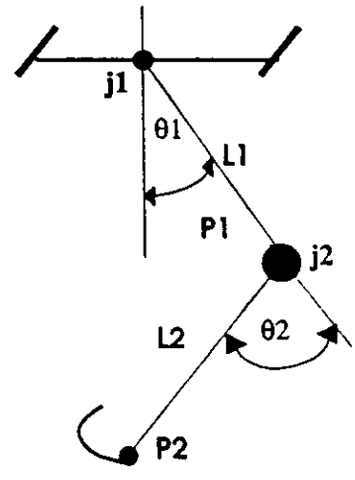


Fig. 6.1 - O Robô com Dois Graus de Liberdade.

6.4 Características do manipulador robótico

O pêndulo 1, controlado por um motor CC com um detetor de posição, com liberdade de rotação de 360 graus, possui peso P1, comprimento L1, junta j1 e está posicionado no ângulo θ1 em relação à vertical. O pêndulo 2, controlado por um motor CC em malha aberta e com liberdade de giro de 180 graus, com três posições permitidas (solto (θ2=0), θ2= π/2 ou θ2= −π/2), possui peso P2, comprimento L2 e junta j2, e está posicionado no ângulo θ2 em relação ao pêndulo 1. O pêndulo 2 possui um gancho fixo

na sua extremidade. Esta estrutura foi escolhida para permitir que o gancho possa agarrar uma escada.

6.5 Estratégias de Controle

[CAV 95a] propuseram um Sistema de Controle Inteligente (SCI) baseado em 3 estratégias para o posicionamento de um pêndulo invertido. A partir da observação dos movimentos (analogia) que um ginasta executa para agarrar um trapézio e baseado no SCI descrito no Capítulo 5 desta dissertação, foram desenvolvidas para controle do robô no movimento de agarrar a escada cinco estratégias para o novo SCI, denominadas: CONTRAÇÃO, POSICIONAMENTO, LANÇAMENTO, DISTENSÃO e ACOPLAMENTO.

Por analogia com os movimentos do ginasta considerou-se que essas estratégias podem ser acionadas seqüencialmente como mostrado na Fig. 6.2.

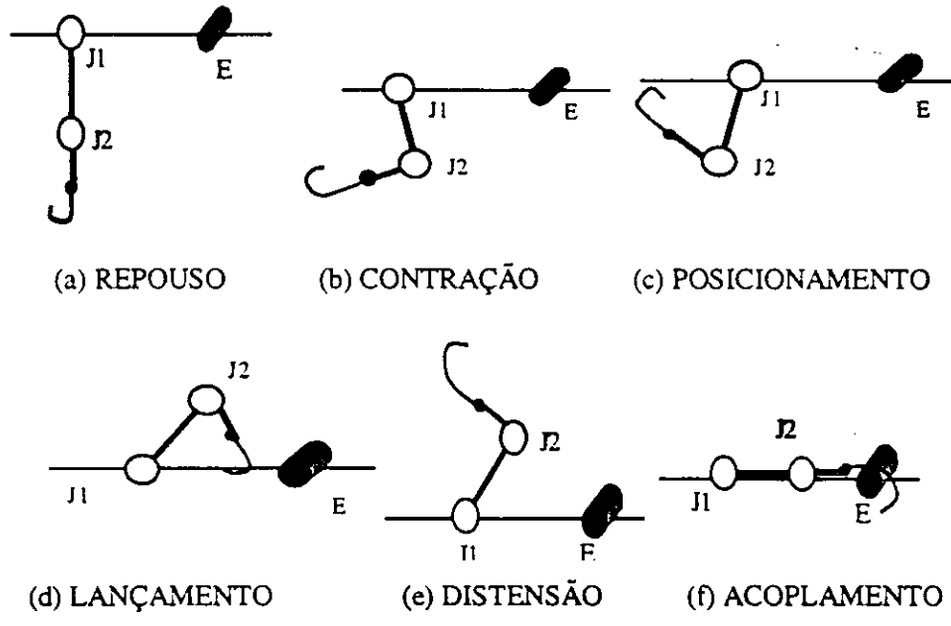


Fig. 6.2 - Seqüência de estratégias para agarrar a escada.

Antes dos movimentos do manipulador robótico a rede neural artificial que controla o motor base tem que ser treinada com a dinâmica do sistema. O treinamento é feito de uma forma semelhante à estratégia TREIN descrita na seção 5.5 do capítulo anterior. Inicialmente, o manipulador é posicionado da forma mostrada na estratégia CONTRAÇÃO em que o pêndulo 2 é forçado a se posicionar em $\theta 2=\pi/2$ e a variável

booleana que indica a mudança de estados no diagrama das estratégias é modificada para TREIN=1. A seguir, o ângulo referência é colocado em θr=π/3. Usando as regras nebulosas 18, 19 e 20, o manipulador é posicionado em ±π/3 durante 10 oscilações. No início o número de oscilações é zerado NO=0. A cada iteração NO é incrementado e quando NOf=M (quando NO>10 como mostrado na Fig. 5.10), a variável booleana da estratégia CONT é modificada para CONT=1 e a variável booleana da estratégia TREINO é modificada para TREIN=0. A Fig. 6.3 ilustra a fase de treinamento da rede neural que controla o motor base do manipulador robótico. Após a fase de treinamento o manipulador robótico é posicionado com θ1=θ2=0 como mostrado na estratégia REPOUSO da Fig. 6.2.

Regra (19): **if** TREIN and $\theta(t) \approx \pi/3$ **then** $\theta r(t)=-\pi/3$ **and** $NO=NO+1$

Regra (20): **if** TREIN and $\theta(t) \approx -\pi/3$ **then** $\theta r(t)=\pi/3$ **and** $NO=NO+1$

Regra (21): **if** TREIN and $NOf==M$ **then** TREIN=0 **and** $REP=1$

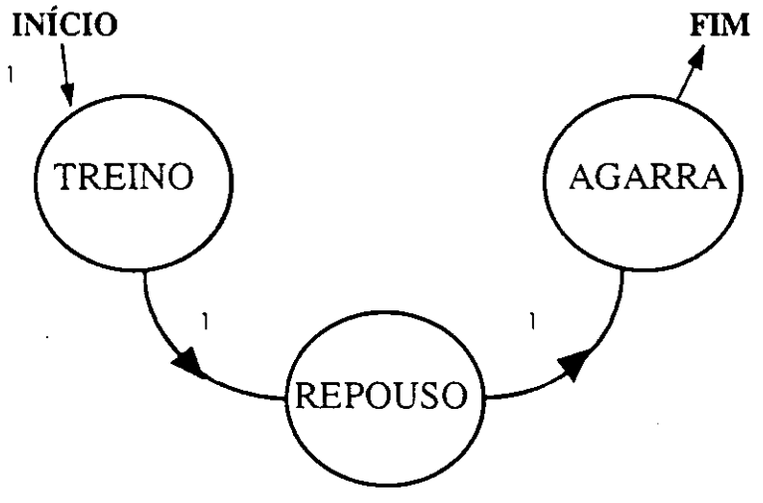


Fig. 6.3 - Fase de treinamento da rede neural artificial

O manipulador na estratégia REPOUSO, com os dois braços do robô como mostrado na Fig. 6.2a, está no modo de menor energia potencial com θ1=θ2=0 rad. Nesta posição o manipulador robótico pode ser movimentado para agarrar a escada. Para simplificar a representação das estratégias, criou-se uma estratégia complexa, denominada AGARRA, mostrada na Fig. 6.5.

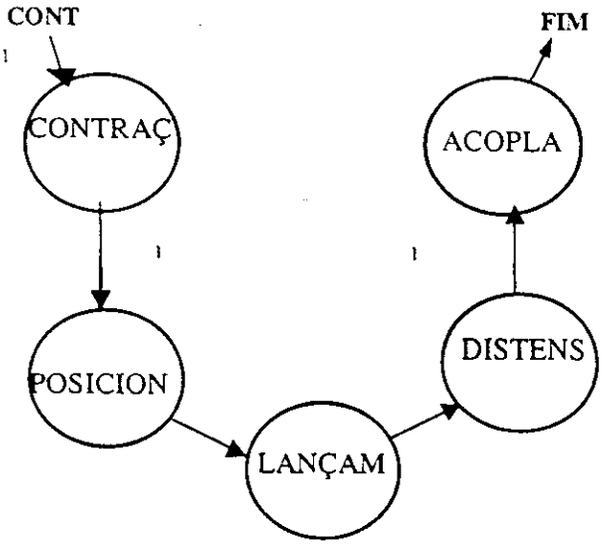


Fig. 6.4 - Estratégia complexa AGARRA

A estratégia AGARRA é inicializada pelo operadoR com CONT=1 e é representado pelas regras nebulosas 22 e 23. Na estratégia CONTRAÇÃO, o robô contrai o segundo braço no sentido horário para no movimento seguinte poder passar pela escada, ver Fig. 6.2b. O motor 2 é acionado de maneira a que $\theta 2=\pi/2$ rad. Nestas condições, o ângulo $\theta 1$ dependerá da carga sustentada pelo braço. Após o posicionamento do pêndulo 1, a estratégia POSICIONAMENTO é ativada.

Regra (22): if CONT **then** $\theta r2(t)=\pi/2$

Regra (23): if CONT **and** $\theta 2(t)=\pi/2$ **then** $POSIC=1$

Na estratégia POSICIONAMENTO, representada pelas regras nebulosas 24 e 25, o robô após deslocar o segundo braço no sentido horário, como mostrado na Fig. 6.2b, ele desloca o primeiro braço até a posição mostrada na Fig. 6.2c. Este movimento é feito para o manipulador ganhar energia potencial, para auxiliá-lo durante a passagem sobre o degrau da escada($\theta 2=\pi/2$ rad e $\theta 1=\pi/6$ rad). Nesta configuração, o robô eleva-se partindo do estado de menor energia potencial a um ponto próximo da máxima energia potencial permitida pelos motores para o peso do braço. A seguir a estratégia LANÇAMENTO é ativada.

Regra (24): if POSIC **then** $\theta r1(t)=\pi/6$

Regra (25): if POSIC **and** $\theta 1(t)=\pi/6$ **then** $LANÇ=1$

Na estratégia LANÇAMENTO, representada pelas regras nebulosas 26 e 27, o robô desloca no sentido anti-horário o seu primeiro braço, como mostrado na Fig. 6.2d.

durante um intervalo de tempo pré-determinado (t_l) mantendo o seu segundo braço fixo em $\theta 2=\pi/2$ rad. Em seguida a estratégia DISTENSÃO é ativada.

Regra (26): if LANÇ **then** $\theta r1(t)=-\pi/2$

Regra (27): if LANÇ **and** $t==t_l$ **then** $DIST=1$

Na estratégia DISTENSÃO, representada pelas regras nebulosas 28 e 29, o robô joga o seu segundo braço no sentido anti-horário até que $\theta 2=-\pi/2$ rad e o seu primeiro braço atinja a posição $\theta 1\cong -3\pi/4$ rad em um tempo pré-determinado (t_d), como mostrado na Fig. 6.2e. Em seguida é ativada a estratégia ACOPLAMENTO.

Regra (28): if DIST **then** $\theta r2(t)=-\pi/2$

Regra (29): if DIST **and** $t==t_d$ **then** $ACOP=1$

Na estratégia ACOPLAMENTO, usando a força gravitacional, os dois braços do robô ficam na posição mostrada na Fig. 6.2f. Para isso, desligam-se sucessivamente a alimentação dos motores dos braços 1 e 2 do robô (U1(t) e U2(t)), como pode ser visto na regra nebulosa número 30. Quando $\theta 1(t)>-\pi/2$ termina a fase AGARRAR do robô (regra nebulosa 31).

Regra (30): if ACOP **then** $U1(t)=0$ **and** $U2(t)=0$

Regra (31): if ACOP **and** $\theta 1(t)\cong\pi/2$ **then** FIM=1

6.6 Escalonador em tempo real

O controle e posicionamento do manipulador devem ser feitos em tempo real pelo SCI. A cada conjunto de operações executadas pelo SCI foi atribuída uma tarefa. Desenvolveu-se um Escalonador em Tempo Real (ETR) para acionamento das tarefas dos dois motores. Cada tarefa possui o seu próprio descritor. Cada descritor possui 4 campos: identificação (id), que identifica a tarefa; status (st), que indica o estado da tarefa, o qual pode ser *executando* (st=-1), *bloqueado* (st=0) e *pronto* (st=1); freqüência (freq) de ativação da tarefa em ms; contador (conta) das interrupções, uma a cada 1ms, do computador.

O ETR foi implementado na linguagem C (Borland 4.5) e o seu pseudocódigo é mostrado abaixo. A cada interrupção os parâmetros dos descritores de todas as tarefas são atualizados. A seguir, todas as tarefas prontas para executar são acionadas. Após a sua execução, cada tarefa terá st=0.

```
a cada interrupção
{
faça {
    se conta !=0 então conta=conta-1;
    se conta==0 então
      {
       conta=freq;
       st=1;
      }
  } (para todos os descritores)
faça {
    se status==1 então
     {
      status=-1;
      aciona tarefa;
      status=0;
     }
  } (para todos os descritores)
}
```

6.7 Tarefas do SCI

O manipulador robótico possui dois braços. Cada braço possui um motor de corrente contínua com o seu circuito de acionamento. Existe um detetor ótico de posição acoplado ao eixo do motor que movimenta o braço base (ver anexo A). A cada estratégia de posicionamento do manipulador robótico foi associada uma tarefa, conforme mostrado na TABELA 6.1 (tarefas 1, 2, 3, 4, 5 e 6). O código das tarefas refletem as regras nebulosas mostradas na seção 6.5.

TABELA 6.1 - LISTA DE TAREFAS

N	NOME	DESCRIÇÃO
1	REPOUSO	Modo de menor energia potencial, $\theta1=\theta2=0$
2	CONTRAÇÃO	O motor 2 gira no sentido horário o segundo braço até que $\theta2=0$
3	POSICIONAMENTO	O motor 1 gira no sentido horário até que $\theta1=\pi/6$, permanecendo o segundo braço em $\theta2=\pi/2$
4	LANÇAMENTO	O motor 1 gira no sentido anti-horário durante 1200 ms permanecendo o segundo braço em $\theta2=\pi/2$
5	DISTENSÃO	O motor 2 gira no sentido anti-horário até que $\theta2=-\pi/2$ e $\theta1=-3\pi/2$
6	ACOPLAMENTO	A alimentação do motor 1 e o motor 2 são desligadas, U1=U2=0 V
7	TREINO	Consiste em variar θr em torno dos ângulos $\theta1=\pm\pi/6$

6.8 Resultados Experimentais

O relógio em tempo real usado no ETR foi de 1 ms (uma interrupção a cada 1ms). A cada 1 interrupção (1ms) é verificada a posição do motor base (motor 1) pela leitura do codificador ótico (ver Apêndice A). A cada 20 ms são acionadas as tarefas (ver tabela 6.1) prontas para executar.

Não são mostradas as curvas experimentais obtidas durante o treinamento da rede neural que controla o motor de corrente contínua do braço 1 do manipulador robótico. Nas Fig. 6.5 (movimento do braço 2) e Fig. 6.6 (movimento do braço 1) são mostradas as curvas experimentais obtidas com o SCI para controle do robô no movimento de agarrar a escada. Os instantes de acionamento das tarefas estão indicados por t1, t2, t3, t4 e t5.

Inicialmente em $t<t1$, o robô encontra-se em repouso, representado pelo ponto t1 nas Fig. 6.5 e Fig. 6.6. No instante t1 é acionada a tarefa CONTRAÇÃO que faz o motor 2 girar no sentido horário (ver Fig. 6.5) e ocasiona um pequeno giro no sentido anti-horário do primeiro braço, que pode ser vista na Fig. 6.6.

A seguir ($t2 \geq t \geq t1$), a tarefa POSICIONAMENTO é acionada, o braço robótico permanece erguido tentando se posicionar em $\theta1=\pi/6$ rad (ver pequenas variações na curva de θ na Fig. 6.6).

Em t=t2 a tarefa LANÇAMENTO é ativada, o primeiro braço inicia um giro no sentido anti-horário, com o segundo braço fixo em $\theta2=\pi/2$ rad, com $t_l = 1200$ ms até que $\theta1 \cong -\pi/4$.

Em t=t3 a tarefa DISTENSÃO é ativada, o segundo braço se distende indo para $\theta2=-\pi/2$ rad e o primeiro braço se posiciona em $\theta1 \cong -3\pi/2$ rad. Observe-se a pequena variação em $\theta1$ na Fig. 6.6.

Entre t=t3 e t=t4 é ativada a estratégia ACOPLAMENTO, o conjunto desce lentamente por falta de força no motor 1 alcançando, finalmente, a escada (ou suas imediações) em t=t5. Após aproximadamente 500 ms o manipulador para de funcionar, repousando sobre a escada.

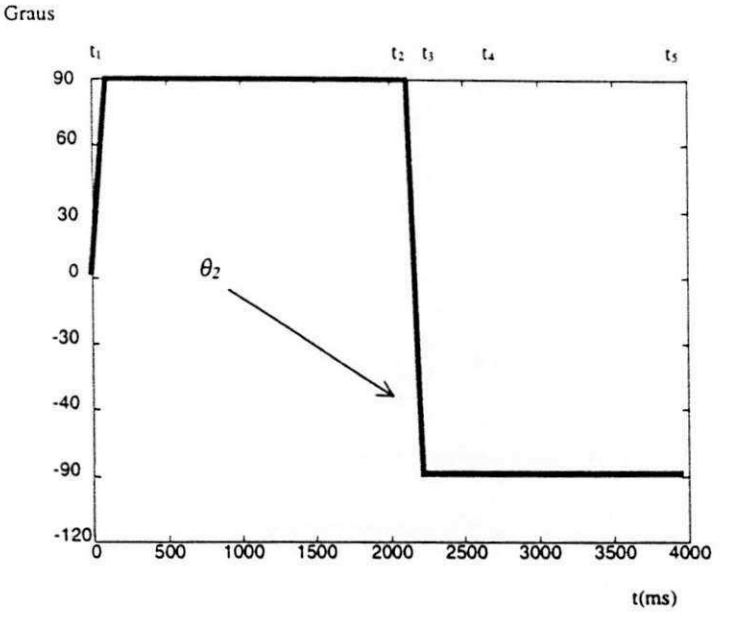


Fig. 6.5 - Resultados experimentais para o segundo braço.

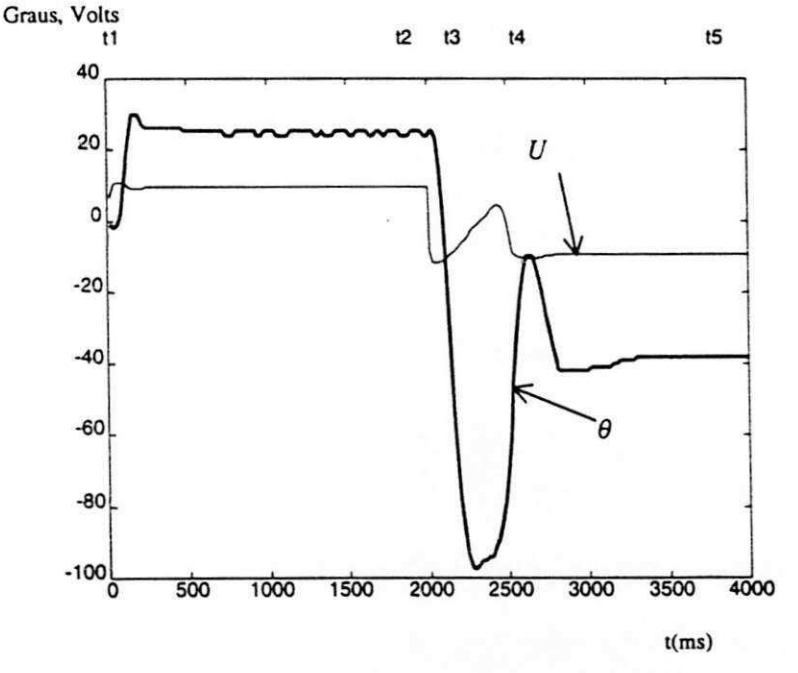


Fig. 6.6 - Resultados experimentais para o primeiro braço.

Capítulo VII

Conclusão

Nesta dissertação, são descrevem-se dois paradigmas da inteligência artificial usados na construção de um SCI: Redes neurais artificiais e Lógica Nebulosa. Apresentam-se suas principais características e os conceitos básicos para uma aplicação na área de controle. São enfatizados os modelos que foram usados na construção do SCI.

Apresentam-se considerações de projeto de um Sistema de Controle Inteligente baseado em regras Nebulosas e redes neurais artificiais, bem como detalhes de sua implementação e alguns resultados experimentais no controle do braço robótico no movimento de agarrar uma escada.

Através da construção e operação de um robô protótipo de simples montagem baixo custo e muito flexível para diversos tipos de tarefas conseguiu-se testar e implementar diversos paradigmas da inteligência artificial: Lógica Nebulosa, Redes Neurais, entre outros. Essas abordagens tornaram a tarefa de construir um SCI para controlar um sistema altamente não-linear, como um braço robótico, bastante independente do conhecimento da planta do sistema. Além do mais, o sistema tornou-se de fácil operação e adaptação a novas situações.

Usou-se, como bancada de testes, um manipulador robótico desenvolvido com 2 pêndulos, controlados por uma RNA, para emular um animal agarrando um degrau de uma escada. Com base em trabalhos anteriores, para o movimento do robô, foram criadas algumas etapas. A cada etapa foi associada uma tarefa. Foram apresentados e comentados resultados experimentais obtidos com o manipulador robótico agarrando uma escada.

Para controlar e sincronizar o acionamento das tarefas foi desenvolvido um escalonador de tarefas.

Pontos positivos do SCI:

1. Um sistema que pode ser descrito qualitativamente pelo usuário.
2. Pode ser programado a nível de tarefas.
3. Representado por estratégias.
4. Como controlador, usa Redes Neurais Artificiais, que pode ser treinada usando resultados obtidos da lógica nebulosa.

Pontos negativos do SCI:

1. A sua programação é difícil, o operador deve conhecer completamente o sistema.
2. Entendimento da dinâmica em tempo real das tarefas é difícil.
3. Por ser uma pesquisa há escassez de bibliografia e manuais de sistemas implementados.
4. Existe uma grande dificuldade de eliminar possíveis falhas nos objetivos das tarefas.

Sugestões para projetos futuros:

1. Reprojetar o escalonador de tarefas para torná-lo mais inteligente, levando em consideração, no algoritmo de controle, não só o acionamento das tarefas como também a posição real do braço robótico.
2. Desenvolver uma interface homem máquina mais adequada para o desenvolvimento de novos SCI's, melhorando a existente.
3. Reprogramar o SCI para que o movimento seja sempre com sucesso. Isto pode ser feito usando Algoritmo Genético.
4. Aumentar, através de novas abordagens da IA, a inteligência do SCI para torná-lo mais útil e fácil de usar.

Anexo A

Hardware de Acionamento do Motor CC

O sistema motor CC, mostrado na Figura A.1, é composto de um motor CC acoplado a um microcomputador pessoal. Usa uma placa com interface padrão para a saída da impressora do microcomputador pessoal IBM PC. A Fig. A.1 ilustra o circuito de acionamento do motor CC.

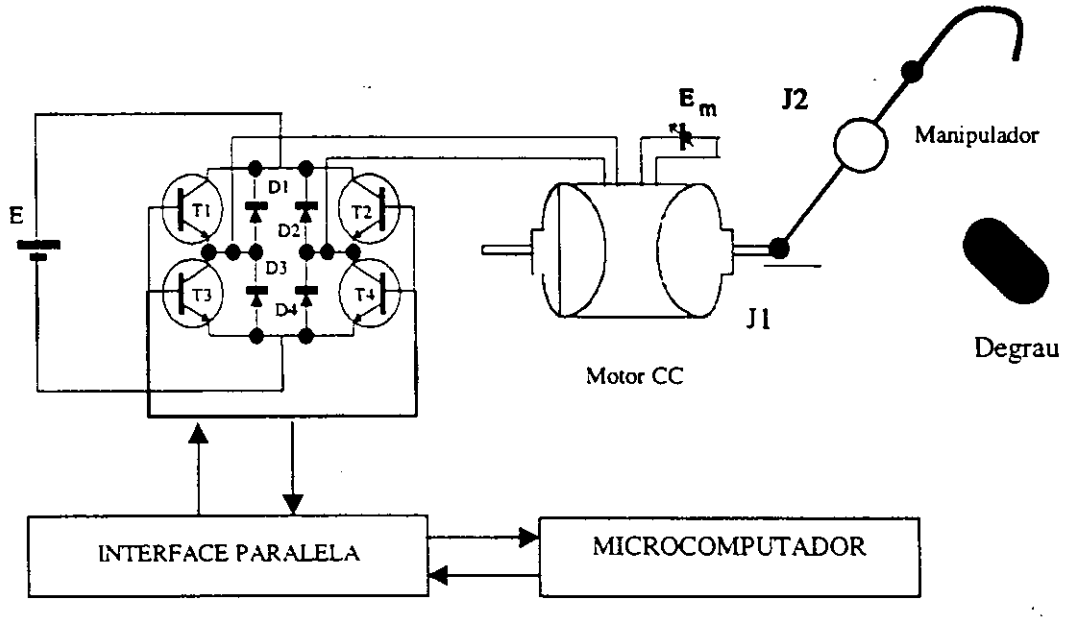


Fig. A.1 - O "hardware" do manipulador robótico

Tem-se dois motores principais (motor 1 e o motor 3). Ambos são motores CC a ímã permanente. Neste caso o campo magnético é produzido por um ímã permanente e é constante, fazendo com que as características torque-velocidade do motor sejam relativamente lineares [KUO 92]. Cada motor possui um disco com 60 orifícios (ver Fig. A.2) onde dois codificadores óticos fazem leitura em tempo real.

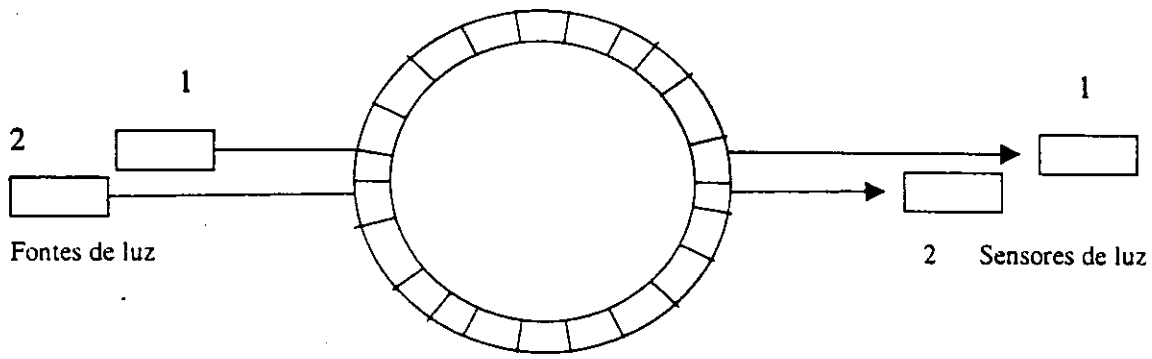


Fig. A.2 – O Codificador ótico

Estes codificadores óticos são do tipo incremental que são de simples implementação e baixo custo. Eles convertem deslocamento rotativo em sinais codificados digitalmente ou pulsos (0s e 1s). O codificador ótico usado tem cinco partes: duas fontes de luz, um disco rotativo e dois sensores (ver Fig. A.2).

Através da contagem dos pulsos gerados pelo codificador ótico pode-se determinar a qualquer instante a posição e a velocidade de cada um dos motores. Para isso, realimenta-se em tempo real o sistema de controle do manipulador robótico com as informações das contagens dos orifícios descritas a seguir:

- Se a luz proveniente da fonte 1 (ver Fig. A.2) desaparecer primeiro que a luz proveniente da fonte 2 é sinal de que o motor está se deslocando no sentido horário. Assim, se o codificador digital constatar os seguintes trens de pulsos (Fig. A.3), o motor estará rodando no sentido horário.

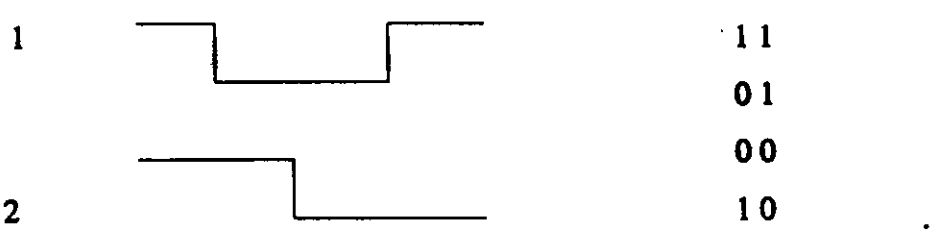


Fig. A.3 – Seqüência de trens de pulsos

- Se a luz proveniente da fonte 2 (ver Fig. A.2) desaparecer primeiro que a luz proveniente da fonte 1 é sinal de que o motor está se deslocando no sentido anti-

horário. Assim, se o codificador digital constatar os seguintes trens de pulsos (Fig. A.4), o motor estará rodando no sentido anti-horário.

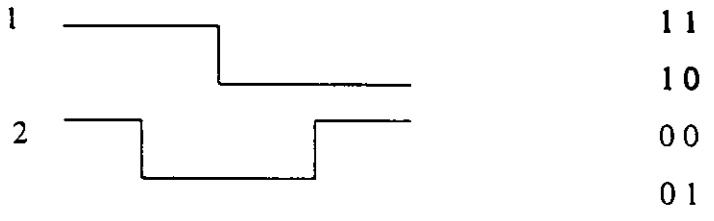


Fig. A.4 – Outra seqüência de trens de pulsos

Exemplo do cálculo do deslocamento rotativo e da velocidade angular:

- Se o número de orifícios contados pelo codificador ótico for, por exemplo, 20, tem-se o seguinte deslocamento angular:

$$\frac{20 \text{ (orifícios)} \times 360 \text{ (graus)}}{60 \text{ (número total de orifícios)}} = 120 \text{ graus}$$

Através do uso do relógio de 1000 Hz (1000 pulsos por segundo), se o contador conta 400 pulsos, por exemplo, enquanto o codificador registra a contagem de 20 buracos, tem-se a seguinte velocidade angular:

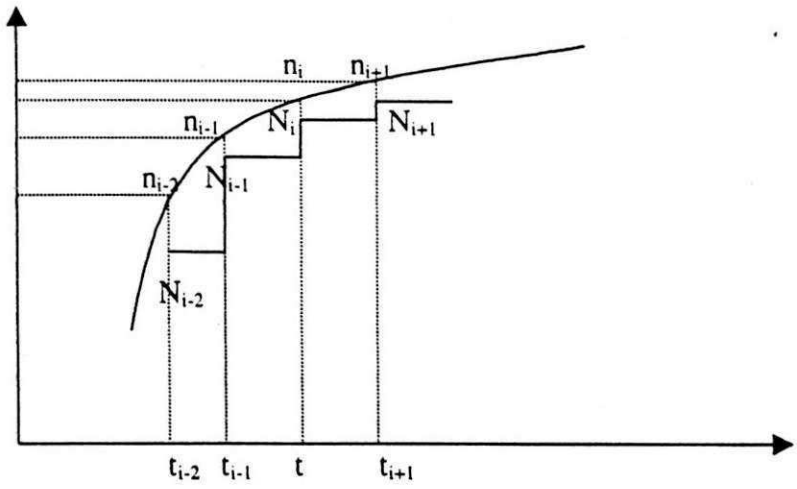
$$\frac{1000 \text{ (pulsos por segundo)} \times 120 \text{ (graus)}}{400} = 300 \text{ graus/seg}$$

$$= \frac{60 \text{ (Seg / min)} \times 300 \text{ (graus/seg)}}{360 \text{ (graus)}} = 50 \text{ rpm}$$

O método descrito acima não detecta a velocidade instantânea do motor, e sim, detecta a velocidade média do motor num certo intervalo de tempo. Porém esse sistema de controle necessita de uma resposta muito precisa para melhorar a estabilidade do braço robótico, por isso, para obter-se a velocidade usa-se um método de estimação da velocidade instantânea proposto por [SAI 85], o qual descreve-se:

Supondo-se que a velocidade do motor varia conforme a figura A.5:

Velocidade do Motor



A.5 – Relação entre velocidade instantânea e velocidade detectada do motor

A velocidade detectada N é obtida conforme o método anteriormente descrito (contando os trens de pulsos de saída num certo intervalo de tempo). Nominalmente, a velocidade detectada é obtida da seguinte fórmula (desprezando a quantização):

$$N_i = \frac{1}{T}\int_{t_{i-1}}^{t_i} n(t)dt \qquad \text{(A.1)}$$

Onde T é o período de amostragem da velocidade de controle

$$T = t_i - t_{i-1} \qquad \text{(A.2)}$$

Supondo-se que a velocidade do motor n(t) entre dois pontos de amostragem varia de acordo com a função de primeira ordem seguinte:

$$n(t) = at + b \qquad \text{(A.3)}$$

pela Eq. A.1 a velocidade detectada pode ser escrita da seguinte forma:

$$N_{i-1} = a\frac{(t_{i-2} + t_{i-1})}{2} + b \qquad \text{(A.4)}$$

$$N_i = a\frac{(t_{i-1} + t_i)}{2} + b \tag{A.5}$$

da Eq. A.3 a velocidade instantânea pode ser escrita:

$$n_i = at_i + b \tag{A.6}$$

eliminando as constantes a e b das fórmulas (A.4) a (A.6) a velocidade estimada $\hat{n}_{i1}$ da velocidade instantânea, por uma estimação de primeira ordem, é como segue:

$$\hat{n}_{i1} = \frac{3N_i - N_{i-1}}{2} \tag{A.7}$$

A Eq. A.7 é uma aproximação da velocidade instantânea bem melhor que a velocidade detectada pela contagem de pulsos.

Anexo B

Algoritmo de Propagação Retroativa do Erro

Usa-se a seguinte notação: E_p é a função erro para o padrão p, t_{pj} representa a saída desejada no nódulo para o padrão p no nódulo j, enquanto o_{pj} representa a saída atual no nódulo,. w_{ij} é o peso do nódulo i para o nódulo j.

Define-se a função erro proporcional ao quadrado da diferença entre a saída desejada e a saída atual para todos os padrões aprendidos.

$$E_p = \frac{1}{2}\sum_j (t_{pj} - o_{pj})^2 \qquad (B.1)$$

A ativação de cada unidade j, para o padrão p, pode ser escrita como

$$net_{pj} = \sum_i w_{ij} o_{pi} \qquad (B.2)$$

Isto é uma soma ponderada simples, como no Perceptron de uma só camada.

A saída de cada unidade j é a função limiar f_j atuando sobre a soma ponderada. No Perceptron, a função de ativação é a função degrau. No Perceptron Multicamadas, costumeiramente é a função de ativação sigmóide, embora qualquer função monotônica diferenciável possa ser usada.

$$o_{pj} = f_j(net_{pj}) \qquad (B.3)$$

Pode-se escrever usando a regra da cadeia:

$$\frac{\partial E_p}{\partial w_{ij}} = \frac{\partial E_p}{\partial net_{pj}} \frac{\partial net_{pj}}{\partial w_{ij}} \qquad (B.4)$$

Observando-se para o segundo membro à direita da Eq. *B*.4 e substituindo em (*B*.2)

$$\frac{\partial net_{pj}}{\partial w_{ij}} = \frac{\partial}{\partial w_{ij}} \sum w_{kj} o_{pk}$$

$$= \sum \frac{\partial w_{ik}}{\partial w_{ij}} o_{pk}$$

$$= o_{pi} \qquad (B.5)$$

desde que

$$\frac{\partial w_{kj}}{\partial w_{ij}} = 0$$

exceto quando $k = i$, quando então ele é igual a 1.

Podemos definir as alterações nos erros como uma função das alterações nas ativações de entradas numa unidade como

$$-\frac{\partial E_p}{\partial net_{pj}} = \delta_{pj} \qquad (B.6)$$

E por (*B*.4) vem a ser

$$-\frac{\partial E_p}{\partial w_{ij}} = \delta_{pj} o_{pi} \qquad (B.7)$$

Concluímos de (*B*.7) que decrementar o valor de E_p significa fazer os pesos mudarem proporcionalmente a $\delta_{pj}\, o_{pi}$, i.e.,

$$\Delta_p w_{ij} = \eta \delta_{pj} o_{pi} \qquad (B.8)$$

Necessita-se conhecer δ_{pj} para cada unidade, pois com ele pode-se descrever E. Usando

$$\delta_{pj} = -\frac{\partial E_p}{\partial net_{pj}} = -\frac{\partial E_p}{\partial o_{pj}} \frac{\partial o_{pj}}{\partial net_{pj}} \qquad (B.9)$$

Essa equação representa a mudança na função erro em relação aos pesos na rede. Isso provê um método para alterar a função erro a fim de reduzi-la. Essa função é proporcional ao erro δ_{pk} na unidade subsequente, assim o erro tem de ser calculado primeiro na unidade de saída (dado por (B.12)) e, então, passado para trás através da rede para permitir que as unidades de entrada alterarem os pesos de suas conexões. Assim, passando para traz os valores do erro é que a rede é referida como "*backpropagation network*". As equações (B.12) e (B.15) juntas definem como treinar a rede muticamadas.

A função sigmóide é definida como,

$$f(net) = \frac{1}{(1+e^{-knet})}$$

E tem a faixa $0<f(net)<1$. A maior razão para seu uso é que ela possui como derivada uma função simples que facilita a implementação dos algoritmos de treinamento com retropropagação.

Dado que a saída de uma unidade, o_{pj} é

$$o_{pj} = f(net) = \frac{1}{(1+e^{-knet})}$$

A derivada dessa unidade $f'(net)$, é dada por

$$f'(net) = \frac{ke^{-knet}}{(1+e^{-knet})^2}$$

$$= kf(net)(1-f(net))$$

$$= ko_{pj}(1-o_{pj})$$

Segundo esse procedimento, o algoritmo de aprendizagem do Perceptron Multicamadas pode ser descrito como segue:

Algoritmo B.1

1. Inicialize pesos e limiares:

Atribua pequenos valores aleatórios aos pesos e limiares.

2. Apresente um padrão de entrada e a saída desejada. Apresente a entrada $X_p = x_0, x_1, x_2, \ldots, x_{n-1}$ e saída desejada $T_p = t_0, t_1, t_2, \ldots, t_{m-1}$ onde n é o número de nódulos de entrada

e m é o número de nódulos de saída. Faça w_0 ser $-\xi$, a propensão de disparo independente das entradas e x_0 ser sempre 1. Para associação de padrões, X_p e T_p representam os padrões a serem associados. Para a classificação, T_p é sempre igualado a zero, exceto para um elemento ele é igualado a 1 o qual corresponde à classe em que X_p está.

3. Calcule a saída atual:

Em cada camada calcule

$$z_{pj} = f(\sum_{i=0}^{n-1} w_i x_i)$$

E passe como entrada para a próxima camada. Os valores da saída da última camada são o_{pj}.

4. Adapte os pesos:

Comece da última camada e vá retrocedendo para as primeiras

$$w_{ij}(t+1) = w_{ij}(t) + \eta \delta_{pj} o_{pj}$$

$w_{ij}(t)$ representa o peso do nódulo i para o nódulo j no tempo t, η é o termo de ganho, e δ_{pj} é um termo de erro do padrão p no nódulo j.

Para as unidades de saída

$$\delta_{pj} = k o_{pj}(1 - o_{pj})(t_{pj} - o_{pj})$$

Para as unidades escondidas

$$\delta_{pj} = k o_{pj}(1 - o_{pj}) \sum_k \delta_{pk} w_{pk}$$

onde a soma é sobre k nódulos na camada acima do nódulo j.

5. Volte para o passo (2) até que o erro seja mínimo.

Bibliografia

[ADE 91] ADEODATO, P. J. L. **O Mecanismo de Histerese na Adaptação de Redes Neurais Artificiais**, Dissertação (mestrado), Recife, 1991.

[ÄST 89] ÄSTRÖM, K. J.; HÄGGLUND, T. **Automatic Tuning of PID Controllers**, Instrument Society of America, USA, 1988.

[BAT 91] BATUR, C.; KASPARIAN, V. **Adaptive Expert Control**, International J. Control, Vol.54, No.4, April 1991, pp.867-881.

[BEA 91] BEALE, R.; JACKSON, T. **Neural Computing: An Introduction**. In: The Multilayer Perceptron. Bristol, Inglaterra: J W Arrowsmith Ltd, 1991. Chap.4, pp.73-74.

[BUC 88] BUCHNER, H. J.; HEMAMI, H. **Servocompensation of Disturbances in Robotics Systems**, International J. Control, Vol.48, No.1, April 1988, pp.273-288.

[CAV 93] CAVALCANTI, J. H. F.; LIMA, A. M. N.; DEEP, G. S., **Controlador Fuzzy de Plantas Não Lineares**, Anais do I Simpósio Brasileiro de Automação Inteligente, UNESP, Rio Claro, São Paulo, Brasil, setembro de 1993, pp.534-542.

[CAV 94a] CAVALCANTI, J. H. F.; LIMA, A. M. N.; DEEP, G. S., **Posicionamento de um Pêndulo Invertido usando Redes Neuronais**, 23' JAIIO, 23a Jornadas Argentinas de Informatica e Investigacion Operativa, Buenos Aires, Argentina, 6-8 de sept. 1994.

[CAV 94b] CAVALCANTI, J. H. F.; LIMA, A. M. N.; DEEP, G. S., **On-line Training of Adaptative Neural Network controllers**, IEEE Industrial Electronics Society IECON'94, Proceedings, Bologna, Italy, setembro 1994, pp. 92-114.

[CAV 95a] CAVALCANTI, J. H. F.; FERNEDA, E., **Intelligent Control of an Inverted Pendulum - Evolution**, SBAI'95, Curitiba PR, 10-12 setembro de 1995, pp.121-126.

[CAV 95b] CAVALCANTI, J. H. F.; SALES. E. F. Jr.; FERNEDA, E. "**Intelligent Control of an Inverted Pendulum**, IISBRN'95, São Carlos, São Paulo, outubro 1995.

[CRA 86] CRAIG, J. J. **Introduction to Robotics, Mechanics & Control**, Addison Wesley Pub. Comp., USA, 1986.

[DAZ 88] D'AZZO, J. J; HOUPIS, C. H., **Análise e Projeto de Sistemas de Controle Lineares**, Editora Guanabara, 1988.

[FUK 97] FUKUDA, T. **Learning, Adaptation and Evolution for Intelligent System.** ISTE'97 Proceedinges of the IEEE Internatioanl Symposium on Industrial Electronics. Guimarães – Portugal University of Minho July 7-11, 1997 Vol 1, TU9-TU14

[FUR 91] FURUTA, K.; YAMAKITA, M. **Swing Up Control of Inverted Pendulum,** IECON'91, Proceedings, October 1991, Kobe - Japan, pp.2193-2198.

[GOM 94] GOMIDE. F. A. C.; GUDWIN, R. R. **Modelagem, Controle, Sistemas e Lógica Fuzzy.** SBA Controle & Automação. Vol.4 n 3. Setembro-outubro 1994. Florianópolis - Santa Catarina, 1994.

[HEB 49] HEBB, D. O. **The organization of Behavior: A Neupsychological Theory.** New York: Wiley, 1949.

[HUN 92] HUNT, K. J.; SBARBARO, D.; ZBIKOWSKI, R.; GAWTHROP, P.J. **Neural Networks for Control Systems - A Survey.** International Federation of Automatic Control, Automatica, Vol.28, No.6, 1992, pp.1083-1112.

[ISH 91] ISHIDA, T.; SHIOKAWA, N.; NAGADO, T.; SHINICHI, G. **Learning Control of a Inverted Pendulum Using a Neural Network,** IECON'91, Proceedings, October 1991, Kobe - Japan, pp. 1401-1404.

[JIM 92] JIM, Y.; PIPE, T.; WINFIELD, A. **DBP Learning Law for Real Time Neural Control,** ICARCV'92,1992, pp.CO.8.2-CO-8.8.

[KHA 92] KHALID, M.; OMATU, S. **A Neural Network Controller for a Temperature Control System,** IEEE Control System, june 1992, pp.58-64.

[KUO 92] KUO, B. C. Sistemas de Controle Automático, quarta edição. In: **Modelagem Automática de Sistemas Físicos.** Rio de Janeiro: Printice-Hall do Brasil, 1982. Chap.4, p.162.

[LIN 92] LIN, C. E.; SHEU, Y. R. **A Hybrid-Control Approach for Pendulum-Car Control**, IEEE Transactions on Industrial Electronics , Vol.39, No.3, June 1992, pp.208-214.

[MCC 43] MCCULLOCH, W. S.; PITTS, W. **Logical Calculus of the Ideas Immanent in Nervous Activity.** Publicado inicialmente em 1943 e reeditado por James A. Anderson and Edward Rosenfeld. "Nerocomputing Foundations of Research, The MIT - Press, Lodon England, 1989, pp 18-27.

[NAR 90] KUMPATI, S. N.; PARTHASARATHY, K. **Identification and Control of Dynamical Sistems Using Neural Networks.** IEEE trans. Neural Networks vol 1, nº 1, Mar. 1990, pp 4-27

[NAR 95] KUMPATI. S. N. **The Handbook of Brain and Neural Networks.** Massachusetts Institute of Technology, 1995, pp 69-73

[NAU 94] NAUCK, D.; KRUSE, R. **Choosing Appropriate Neuro-Fuzzy Models.** EUFIT'94, Aachen 1994, pp. 552-557.

[RIC 91] RICH, E.; KNIGHT, K. **Inteligência Artificial.** Makron Books. São Paulo, 1991, pp. 39-74.

[ROS 58] ROSENBLATT, F. **The Perceptron: a Probabilistic Model for Information Storage and Organization in the Brain.** 1958, pp 92-114.

[SAI 85] SAITO, K.; OHMAE, T.; KAMIYAMA, k.; MATSUDA, Y. **A Microprocessor-Controlled Speed Regulator with Instantaneous Speed Estimation for Motor Drives.** IECON'85, pp 755-760. CH2160-0/85/0000-0755 © 1985 IEEE.

[SHO 91] SHOURESHI, R. **The Mystique of Intelligent Control,** IEEE Control Systems Magazine, January 1991, pp.33-33.

[TAN 91] TANOMARU, J.; OMATU, S. **Towards Effective Neuromorphic Controllers**, IECON'91, Proceedings, October 1991, Kobe - Japan, pp.1395-1400.

[UEB 95] UEBEL, L. F.; BOTELHO, S. S. C.; SIMÕES, E. V.; BARONE, D. A. C. **Controle Inteligente de Robôs Móveis Autônomos.** RAM x Fuzzy. II SBRN São Carlos 18 a 20 de outubro de 1995.

[VAS 95] VASCONCELOS, G. C. **Redes Neurais e Reconhecimentos de Padrões**. In: II SBRN, 1995, São Carlos.

[ZAD 73] ZADEH, L. **Outline of a new approach to the analysis of complex systems and decision processes.** IEEE Trans. On Systems Man and Cybernetics, vol SMC-3, pp 28-44, 1973.