Universidade Estadual de Campinas
Faculdade de Engenharia Elétrica e de Computação
Departamento de Sistemas e Controle da Enérgia
Laboratório de Sistemas Modulares Robóticos
LSMR-DSCE-FEEC-UNICAMP

LSMR

Análise de Uma Classe de Neurônios Artificiais para Aplicações em Sistemas Robóticos

Autor: Eng. Jés de Jesus Fiais Cerqueira
Orientador: Prof. Dr. Álvaro Geraldo Badan Palhares

Dissertação submetida à Faculdade de Engenharia Elétrica e de Computação da Universidade Estadual de Campinas, para preenchimento dos pré-requisitos parciais para obtenção do título de
Mestre em Engenharia Elétrica

Campinas, 24 de Outubro de 1996.

Este exemplar corresponde à redação final da tese defendida por Jés de Jesus Fiais Cerqueira e aprovada pela Comissão Julgadora em 24 / 10 / 97.

Orientador

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DA ÁREA DE ENGENHARIA - BAE - UNICAMP

C335a

Cerqueira, Jés de Jesus Fiais
Análise de uma classe de neurônios artificiais para aplicações em sistemas robóticos / Jés de Jesus Fiais Cerqueira.--Campinas, SP: [s.n.], 1996.

Orientador: Álvaro Geraldo Badan Palhares.
Dissertação (mestrado) - Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Redes neurais (Computação). 2. Robótica. 3. Controle de processo. 4. Sistema de controle ajustável. 5. Liapunov, Funcões de. I. Badan Palhares, Álvaro Geraldo II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

Universidade Estadual de Campinas
Faculdade de Engenharia Elétrica e de Computação
Departamento de Sistemas e Controle da Enérgia
Laboratório de Sistemas Modulares Robóticos
LSMR-DSCE-FEEC-UNICAMP

LSMR

Título: Análise de uma Classe de Neurônios Artificiais para Aplicações em Sistemas Robóticos

Autor: Eng. Ele. Jés de Jesus Fiais Cerqueira

Orientador: Prof. Dr. Álvaro Geraldo Badan Palhares

Aprovada em 24 de outubro de 1996 pela banca examinadora

Prof. Dr. Álvaro Geraldo Badan Palhares - UNICAMP (Presidente)

Prof. Dr. José Cláudio Geromel - UNICAMP

Prof. Dr. Aluízio Fausto Ribeiro Araujo - USP/São Carlos

"Ainda que a própria figueira não floresça e não haja produção nas videiras, o trabalho da oliveira realmente resulte em fracasso e os próprios socalcos realmente não produzam alimento, o rebanho seja separado do redil e não haja manada nos currais; ainda assim, no que se refere a mim, vou rejubilar com o próprio Jeová; vou jubilar com o Deus de minha salvação.

Jeová, o Soberano Senhor, é minha energia vital; e Ele fará meus pés semelhantes aos das corças e me fará pisar nos meus altos."

Habacuque 3:17-19 *Tradução do Novo Mundo das Escrituras Sagradas*

Agradecimentos

Esta talvez seja a parte mais difícil para mim na realização deste trabalho, não que haja algo de errado em se fazer agradecimentos, muito pelo contrário, são em momentos como este que um homem mostra o seu caráter. É que, quando parei por um momento e olhei para trás em minha vida, para ver as pessoas a quem agradecer, observei que desde o meu nascimento até os dias de hoje existiram, e existem, pessoas as quais eu tenho gratidão pelas coisas que faço.

Umas das frases mais significativas que já ouvi é:"Melhor que se fazer justiça é não se fazer injustiça". Assim, para não cometer a injustiça de deixar de citar algumas dessas pessoas, as quais sou grato, que todas essas, que sabem de suas contribuições, sintam-se agradecidas, e assim farei apenas algumas citações de pessoas e instituições que foram fundamentais no contexto deste trabalho.

- À Fundação Escola Politécnica da Bahia pela ajuda financeira nos meses iniciais do curso de mestrado.
- À CAPES pela concessão de uma bolsa de estudos durante a realização do curso.
- À UNICAMP por ter proporcionado esta oportunidade de trabalho.
- Ao Prof. Dr. Luís Gimeno Latre por ter me aceitado como seu orientado no início do curso.
- Aos membros da banca examinadora desta dissertação, o Prof. Dr. José Cláudio Geromel e o Prof. Dr. Aluízio Fausto Ribeiro Araujo, por seus comentários e sugestões para a versão final do texto e continuidade de futuros trabalhos.
- Ao Prof. Dr. Álvaro Geraldo Badan Palhares por ter aceito a minha proposta de trabalho, pela excelente orientação e por ter "vestido a camisa da tese".
- À toda a minha família, pelos incentivos e por terem suportado a minha ausência, em especial aos meus irmãos Wedem e William e aos meus tios Miguel e Virgínia.

- Um profundo agradecimento aos meus pais Manuel e Angelina por tudo que me proporcionaram na vida e por sempre acreditarem e confiarem em mim.
- Por fim àquele ao qual, não apenas eu, mas todos nós, devemos ser gratos por todas as coisas, ao Único e Verdadeiro Deus cujo nome é Jeová.

Resumo

Neste trabalho são analisadas as estruturas recursivas e não recursivas da classe *perceptron* de Redes Neurais Artificiais. A lei de aprendizado utilizada é baseada no gradiente com a inclusão da retro-propagação (back-propagation) de sinais através da rede. É apresentada uma formulação vetorial para os modelos e estabelecido um critério para análise de convergência baseado no segundo método de Lyapunov.

Como exemplo, uma rede que mistura neurônios recursivos e não recursivos é testada por simulação em um sistema de controle para o acionamento das juntas de um robô de dois graus de liberdade, sendo propostos um identificador e um controlador adaptativo por modelo referência, com os resultados apresentados que mostram a eficiência do método.

Palavras Chaves: Redes neurais (Computação), Robótica, Controle de processo, Sistema de controle ajustável, Funções Liapunov.

Abstract

This work analyses the recurrent and no recurrent dynamic structures of the *perceptron* class of artificial neural networks. The learning law used is based on the gradient with the inclusion of back-propagation of signals through the network. A vectorial formulation to modelsis presented and a criterion for convergence analysis based on the second Lyapunov's method is established.

As on example, a mix network is tested through simulation in a control system for the activation of the joints of a two degree of freedom robot, and an identifier and an adaptive controller by reference model are proposed, the results showing the efficiency of the method being presented.

Key Words: Neural Networks (Computation), Robotics, Process Control, Adaptive Control System, Lyapunov Function.

Conteúdo

Lista de Figuras

Lista de Tabelas

Prefácio

A humanidade observa neste século 20, um espantoso e crescente desenvolvimento tecnológico. Atividades que no século passado, ou até mesmo há décadas atrás, eram feitas com grandes esforços físicos e altos custos são, hoje em dia, realizadas por máquinas e, até mesmo, sem a presença do homem no local.

Por exemplo, atividades de colheitas agrícolas, que há tempos passados eram feitas por mãos humanas, hoje podem ser realizadas por máquinas colheitadeiras. Essas máquinas , por sua vez, quando começaram a ser produzidas, eram feitas por mãos humanas e hoje são feitas por sofisticados robôs industriais.

No entanto, o desenvolvimento tecnológico sempre traz consigo novos desafios, pois a solução de problemas, geralmente, provoca o surgimento de outros novos, ou a reevidência dos antigos, aos quais não se havia dado a devida atenção.

O surgimento dos primeiros sistemas robóticos nas décadas de 40 e 50 [66], motivaram a solução de diversos problemas, como por exemplo, a produção em manufatura de larga escala, como a de automóveis, de caminhões e das acima citadas colheitadeiras. Contudo, os sistemas robóticos apresentavam problemas, e ainda apresentam alguns que não foram completamente solucionados.

Inspirados na anatonia do corpo humano, ou de animais, os robôs são dirigidos principalmente para atividades de manufatura que originalmente são realizadas pelo homem. Em geral são atividades onde se exige precisão e repetibilidade.

Muitos dos problemas existentes nos primórdios dos sistemas robóticos já foram solucionados ou obtiveram soluções satisfatórias para determinadas aplicações. Dentre tantos, podemos citar, problemas relacionados com o controle de movimentos de um robô.

Para a maioria das aplicações existentes, no que diz respeito ao controle dos movimentos de juntas, uma solução muito utilizada é a base da classe de controladores **PID**, que são de baixo custo e em geral de fácil projeto. No entanto este paradigma não consegue atender satisfatoriamente a critérios de desempenho pré-estabelecidos, principalmente

quando as interações entre as diversas variáveis do sistema são muito evidentes. Muito esforço tem sido dispendido por pesquisadores na elaboração de novos paradigmas mais sofisticados e eficientes com base em teoria de controle ótimo e adaptativo [20, 50], no nível hierárquico dos movimentos de juntas, e até mesmo no nível de controle e planejamento de trajetórias (controle hierárquico [51, 52]), etc. No entanto, essas técnicas ainda não alcançaram expressiva aceitação nos meios industriais, talvez porque elas requeiram um conhecimento mais especializado por parte de quem as estejam utilizando e especificando, o que não é muito comum para a maioria dos profissionais no cotidiano das indústrias, ou ainda pela falta de confiabilidade que essas novas soluções possam apresentar.

Já por alguns anos tem sido feitas pesquisas e aplicações na área de sistemas por aprendizado para controle e automação [43, 47, 45]. Esses sistemas são concebidos para aprender e operar eficientemente sistemas dinâmicos não lineares, interativos e com alto grau de incertezas, já existindo inclusive aplicações em sistemas robóticos [1, 2, 15, 17, 18, 26, 27, 29, 35, 38, 42, 65, 74]. Com isso tornam-se muitas vezes fáceis para se operar pois requerem pouco conhecimento especializado por parte dos usuários.

Outra área cujas pesquisas recentes tem provocado relativo entusiasmo é a dos sistemas neurais artificiais ou simplesmente redes neurais artificiais (**RNA's**). Também inspirada no ser humano, esta nova área de pesquisa objetiva obter equivalentes artificiais dos sistemas cerebrais que controlam e gerenciam o corpo e suas atividades. Modelos para células nervosas e sobre suas formas de organização foram desenvolvidos [1, 2, 7, 41, 61, 68, 71]. Assim como os humanos realizam suas atividades porque em algum momento as aprenderam, os modelos de sistemas neurais também utilizam-se de leis de aprendizado para ajustar a maneira em que devem realizar as suas tarefas [1, 2, 23, 32, 61, 64].

Este trabalho surgiu nesse contexto, da necessidade de obtenção de novas pespectivas de soluções de problemas robóticos, no caso, problemas de controle de movimento, com a utilização de técnicas que funcionem por aprendizado, mais especificamente, à base de **RNA's**.

Tendo isso por base, podemos citar os objetivos principais deste trabalho como sendo:

1. Obtenção de uma proposta de solução do problema de controle de movimento em sistemas robóticos com base em aprendizado e **RNA's**.
2. Obtenção de conhecimento básico necessário, que possibilite no futuro, a utilização de sistemas por aprendizado e com **RNA's** na solução de outros problemas de robótica, dentre outros.
3. Ter disponível uma infra-estrutura mínima, em termos de rotinas computacionais,

que permitam o desenvolvimento de pesquisas e aplicações de sistemas por aprendizado com **RNA's** em sistema diversos, que sejam de fácil utilização e confiáveis.

O presente trabalho esta organizado em seis capítulos. No capítulo 1 é feita uma introdução a uma classe de **RNA's** e ao processo utilizado em seu aprendizado, no qual adotaremos um método baseado no gradiente. É analisada a classe *perceptrons* e introduzida uma nova formulação, vetorial, que possibilita uma melhor visualização de todo o processo de treinamento da rede e permite a obtenção de informações a respeito de sensibilidade paramétrica, algo que a formulação atualmente utilizada, baseada na regra delta generalizada, fazendo o uso de somatórios, formulada originalmente por Windrow e Hoff [72] e generalizada por Rumelhart *et. all.* [64], não favorece.

No capítulo 2 é feita a análise de estabilidade local do processo de aprendizado, baseado no segundo método de Lyapunov, e sua aplicação às classes *perceptrons* de **RNA's**. É obtido um critério para a avaliação da convergência local do aprendizado e elaborado um algoritmo adaptativo derivado do mesmo, delegando assim relativa confiabilidade a todo o processo.

No capítulo 3 é analisado a forma como as **RNA's** são aplicadas em sistemas de controle de um modo genérico. É apresentado inicialmente uma visão geral de sistemas de controle, seguindo as aplicações em identificação e controle. É feita também a análise de estabilidade para o processo de aprendizado em controladores e apresentado uma extensão do algoritmo adaptativo para o caso presente.

No capítulo 4 é feita uma descrição matemática de um sistema robótico, apresentada a análise cinemática e uma dinâmica com a inclusão do modelo de acionadores elétricos, seguido de um método para o projeto de controladores **PID's**.

No capítulo 5 são feitas as aplicações de **RNA's** no controle de sistemas robóticos. São propostos um neuro-identificador e um neuro-controlador para estes sitemas. O método de controle proposto é aplicado sobre o modelo simulado de um robô de 2 **DOF**. Três experimentos são realizados: identificação off-line; sintonia on-line do neuro-controlador; e comparação entre o neuro-controlador e o controlador **PI**.

Conclusões, comentários sobre os resultados obtidos e sugestões para trabalhos futuros são apresentados no capítulo 6.

Capítulo 1

Introdução às Redes Neurais: Uma Formulação Vetorial Baseada na Lei de Aprendizado pelo Gradiente

Este capítulo apresenta o desenvolvimento de um modelo matemático com tratamento vetorial para *Redes Neurais Artificiais* (**RNA's**) com base nas estruturas *perceptrons* recursivas e não recursivas, juntamente com os respectivos algoritmos de aprendizado baseados em uma lei pelo gradiente com a inclusão da retro-propagação (back-propagation) de sinais através da rede.

1.1 Introdução

O cérebro humano é superior a qualquer computador em termos de processamento de informações. Um bebê de apenas um ano de idade é capaz de reconhecer objetos e pessoas, como por exemplo, sua mãe [24], enquanto que para um computador, seria necessário um sistema avançado de inteligência artificial (IA) sendo executado em um sistema de grande porte.

Possui ainda outras características que são desejáveis em sistemas computacionais:

- É flexível, podendo facilmente aprender ou aprimorar novos conceitos;
- É capaz de processar informações consideradas nebulosas, probabilísticas, ruidosas ou inconsistentes;
- É altamente paralelo;

- É compacto e dissipa pouca energia;
- É capaz de interpretar sinais de sensores tais como visão, tato, olfato, paladar, proximidade, deslizamento, força, etc;
- É capaz de emitir comandos de movimento, emoção, som, força, etc;
- Tem grande capacidade de análise e armazenamento de informações.

Dotar sistemas computacionais de caracteríticas como estas são as reais motivações para o estudo dos sistemas por aprendizado, e particularmente da *Neurocomputação*, sendo esta inspirada no conhecimento obtido desde a Neurociência. As **RNA's** tornam-se assim um novo paradigma computacional alternativo para aplicações em ciência da computação e engenharia.

As pesquisas atuais em neurocomputação são fortemente motivadas pela possibilidade de construção de **RNA's** em computadores. Os modelos implementados são extremamente simplificados quando analisados do ponto de vista biológico. As limitações existentes em comparação com a capacidade do cérebro humano fica por conta dos limites tecnológicos dos elementos de memória artificial.

Estima-se que cérebro humano é composto de aproximadamente 10^{11} neurônios, ou em outras palavras, células nervosas. Elas estão agrupadas em redes com formas de árvores. McCulloch e Pitts [41] propuseram um modelo matemático simples para um neurônio. O modelo proposto era concebido como uma soma de entradas modificadas por pesos e a saída do neurônio assumindo forma binária, valores *zero* ou *um* de acordo com o somatório estar abaixo ou acima de certo valor de limiar (função de ativação):

$$y(k) = g\left(\left(\sum_{i=1}^{N} w_i\, x_i\right) - \mu\right) = g\left(\mathbf{w}'\,\mathbf{x}(k) - \mu\right)$$

para $k \in \mathbb{N}$ e onde:

$$g(k) = \begin{cases} 1 & \text{se } (\mathbf{w}'\,\mathbf{x}(k) - \mu) \geq 0 \\ 0 & \text{se } (\mathbf{w}'\,\mathbf{x}(k) - \mu) < 0 \end{cases}$$

Na figura 1.1 é apresentado um neurônio de *McCulloch-Pitts* com duas entradas. Observando a figura e supondo um neurônio com N entradas, temos que:

- $y(k) \in \{0, 1\} \subset \mathbb{N}$ é o sinal de saída;
- $\mathbf{x}(k) \in \mathbb{R}^N$ é o vetor de entrada;
- $\mathbf{w} \in \mathbb{R}^N$ é o vetor de pesos de conexões entrada-saída;
- $g(\cdot) \in \{0, 1\} \subset \mathbb{N}$ é a função de ativação;

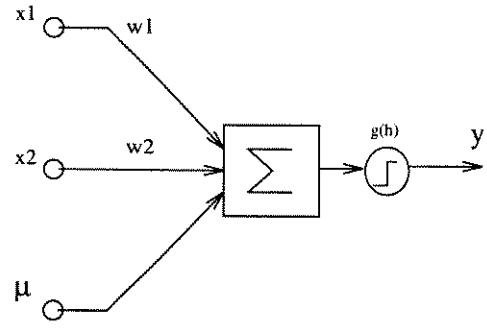

Figura 1.1: Neurônio de McCulloch e Pitts.

- $\mu \in \mathbb{R}$ é o valor de limiar.

Devido a sua simplicidade, o neurônio de *McCulloch-Pitts* é computacionalmente muito potente para uma escolha correta dos pesos w_j.

1.2 Histórico

A base da neurocomputação pode ser traçada a partir do trabalho de McCulloch e Pitts [41] que introduziu o modelo descrito acima. Por volta dos anos 60, as atividades de pesquisas na área foram dirigidas a se encontrar métodos para a obtenção dos pesos corretos na execução de uma dada tarefa. Rosenblatt [61] estudou uma rede chamada de *perceptron*, em que os neurônios são organizados em camadas com conexões *feedforward* de uma camada de neurônios para outra. Um exemplo é mostrado na figura 1.2.

Uma simples classe de *perceptron* sem camadas intermediárias foi apresentada por Rosenblatt [61] com algoritmo de aprendizado convergente, causando entusiasmo na comunidade científica. Entende-se como algoritmo de aprendizado, os métodos matemáticos estabelecidos para determinação dos pesos em uma dada tarefa em que o neurônio esteja sendo utilizado, também chamado de mapeamento entrada-saída. No aprendizado os pesos são ajustados sem a formulação e solução de alguma equação matemática específica. Neste sentido, o aprendizado de uma rede pode ser feito de duas maneiras:

Aprendizado Supervisionado: Que é feito com base na comparação direta da saída da rede com a saída desejada.

Aprendizado não Supervisionado: A saída desejada não é especificada em termos de exemplos corretos. A rede então cria categorias com as correlações das entradas

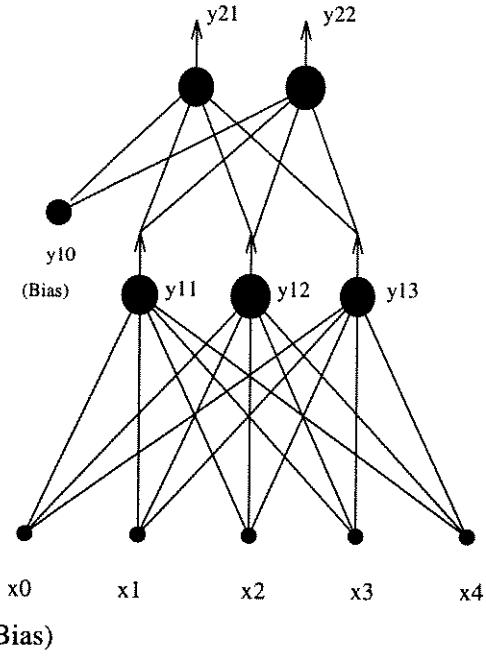

Figura 1.2: *Perceptron* com duas camadas.

para produzir sinais de saída correspondentes. A técnica de aprendizado possui a capacidade de fazer agrupamentos de informações com características similares, sendo muitas vezes chamadas de auto-organizáveis [15, 18, 32].

Todavia, Minsky e Papert [44] mostraram que a rede de uma camada de Rosenblatt não era capaz de aprender algumas funções elementares, como por exemplo a função lógica **XOR** (veja apêndice A).

Rosenblatt também tinha estudado estruturas com mais de uma camada e acreditava que elas poderiam vencer as limitações das estruturas de uma camada apenas. Porém, não existia um algoritmo de aprendizado para determinar os pesos de uma rede deste tipo. Isto desmotivou a comunidade da ciência da computação a desenvolver o paradigma da rede neural por aproximadamente 20 anos.

Alguns grupos continuaram as pesquisas durante os anos 70. A maioria dos trabalhos foram na área de memória associativa (*associative content-addressable memory*), em que diferentes padrões de entrada são associados com outros de suficiente similaridade (aprendizado não supervisionado). Isso tinha sido proposto inicialmente por Taylor [68] e redescoberto por Anderson [7], Willshaw [71], Koehonen [32].

Um método conhecido como *back-propagation* que, pelo que consta, foi proposto

por Werbos [70] e independentemente redescoberto por Rumelhart *et al* [64], é hoje bastante conhecido e utilizado para o aprendizado em **RNA's**.

1.3 O *Perceptron* de Única Camada

O *perceptron* é uma rede do tipo *feedforward* inicialmente estudada por Rosenblatt [61]. Na figura 1.3 temos um exemplo de um *perceptron* de única camada com três neurônios possuindo quatro entradas. A rede *perceptron* será tratada aqui no contexto do aprendizado supervisionado. Essa seção está restrita ao caso de única camada. Assumiremos também que os neurônios não possuem uma função de ativação especifica.

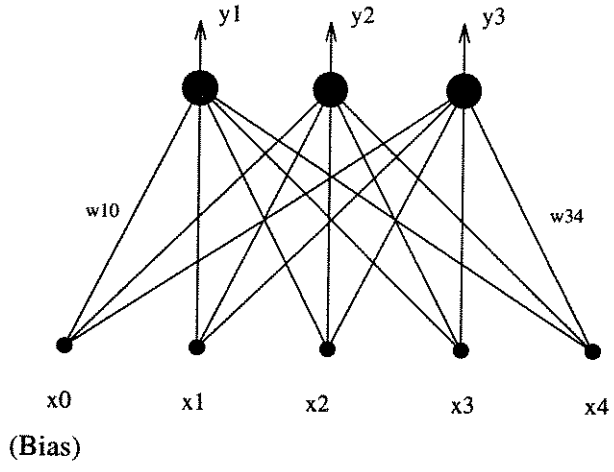

Figura 1.3: *Perceptron* de única camada.

Em geral costuma-se adicionar uma entrada x_0 à rede com valor constante, e com peso w_{i0}, proporcionando assim um *bias* ou *grau de liberdade* ao aprendizado do neurônio. Usando a figura 1.3 e supondo uma camada com M neurônios e N entradas, será adotada aqui a seguinte convenção:

- $\mathbf{y}(k) \in \mathbb{R}^M$ é o vetor de saídas da rede com componentes y_i;
- $\mathbf{x}(k) \in \mathbb{R}^{(N+1)}$ é o vetor de entradas da rede com componentes x_j que incluem o *bias*;
- $\mathbf{w} \in \mathbb{R}^{M\times(N+1)}$ é a matriz de peso das conexões entrada-saída com componentes w_{ij};
- $\mathbf{g}(\cdot) \in \mathbb{R}^M$ é vetor das funções de ativação da rede com componentes de $g_i(\cdot)$;
- $\mathbf{h}(\cdot) \in \mathbb{R}^M$ é vetor de entradas ponderadas com componentes $h_i(\cdot)$.

O sinal de saída da rede, $\mathbf{y}(k)$, pode ser descrito por:

$$\begin{cases} \mathbf{h}(k) &= \mathbf{w}\,\mathbf{x}(k) \\ \mathbf{y}(k) &= \mathbf{g}\,(\mathbf{h}(k)) \end{cases} \tag{1.1}$$

O vetor de funções $\mathbf{g}(\cdot)$ é usualmente constituído de funções não lineares, sendo mais adiante discutida em particular.

Observe que na equação 1.1, o vetor $\mathbf{x} \in \mathbb{R}^{N+1}$ tem a forma $\mathbf{x} = [x_0 \;\; x_1 \;\; \cdots \;\; x_N]'$, e que a matriz $\mathbf{w} \in \mathbb{R}^{M\times(N+1)}$ com componentes w_{ij}, tem a forma:

$$\mathbf{w} = \begin{bmatrix} w_{10} & w_{11} & \cdots & w_{1N} \\ w_{20} & w_{21} & \cdots & w_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ w_{M0} & w_{M1} & \cdots & w_{MN} \end{bmatrix}_{M\times(N+1)}$$

ou, alternativamente:

$$\mathbf{w} = \begin{bmatrix} \mathbf{w}_1' \\ \mathbf{w}_2' \\ \vdots \\ \mathbf{w}_M' \end{bmatrix}_{M}$$

onde os $\mathbf{w}_i \in \mathbb{R}^{(N+1)}$ são vetores de pesos de cada neurônio individual da rede.

1.3.1 Funções de Ativação

É conhecido que neurônios localizados em diferentes partes do sistema nervoso possuem diferentes características. Por exemplo, os neurônios do sistema motor ocular possuem um comportamento semelhante ao de uma função não linear chamada de *sigmoid*, enquanto que os do sistema visual possuem comportamento *gaussiano* [25].

Assim, existem diversas opções para a função de ativação $g(\cdot)$. Algumas dessas funções possuem base biológica estabelecida, como por exemplo a função *sigmoid*, outras não, como por exemplo a função *logarítmica*. A tabela 1.1 apresenta a forma de algumas delas. As principais características *desejáveis* para elas são:

- $g(h(k))$ ser limitada para $h(k) \longrightarrow \pm\infty$;
- $\exists\, \alpha \in \mathbb{R}\; \exists\, x \in \mathbb{R} \mid g(h(x), k) = \alpha \;\forall\, k$
- $\partial g/\partial h$ existir para todo $h(k)$;
- $\partial g/\partial h = 0$ para $h(k) \longrightarrow \pm\infty$;

Nome	*Função*
Threshold	$+1$ se $h(k) > 0$, caso contrário -1
Sigmoid	$\frac{1}{1+\exp(-h(k))}$
Tangente Hiperbólica	$\tanh h(k)$
Gaussiana	$\exp\left(\frac{-h^2(k)}{\sigma^2}\right)$, $\sigma \neq 0$
Linear	$\alpha\, h(k)$, $\alpha \neq 0$

Tabela 1.1: Funções $g(\cdot)$.

1.3.2 A Lei de Aprendizado Baseada no Gradiente

O algoritmo de aprendizado baseado no gradiente é desenvolvido a partir da função vetorial de erro entre o vetor saídas desejadas para a rede, o qual convencionaremos como $\mathbf{y}_d$, e o vetor saídas da rede, $\mathbf{y}$. Definimos então um vetor $\mathbf{y}_d = [y_{d_1} \;\; y_{d_2} \;\; \cdots \;\; y_{d_M}]'$, com $\mathbf{y}_d \in \mathbb{R}^M$, onde y_{d_i} é a saída desejada para o i-ésimo neurônio. Também um vetor erro na forma $\mathbf{e} = (\mathbf{y}_d - \mathbf{y})$, com $\mathbf{e} \in \mathbb{R}^M$ e e_i o erro correspondente a i-ésima saída.

Podemos então definir uma função de custo baseada no erro quadrático de saída da rede, na forma:

$$\begin{aligned} J(k, \widehat{\mathbf{w}}) &= \frac{1}{2}\, \mathbf{e}'(k)\, \mathbf{e}(k) \\ &= \frac{1}{2}\left\{e_1^2(k) + e_2^2(k) + \cdots + e_M^2(k)\right\} \\ &= J_1(k) + J_2(k) + \cdots + J_M(k) \\ &= \sum_{i=1}^{M} J_i(k) \end{aligned} \tag{1.2}$$

Note que a equação 1.2 é escalar, e $J(k, \widehat{\mathbf{w}}) \in \mathbb{R}$ é função de um conjunto

de parâmetros ajustáveis, w_{ij}, e que sua minimização depende apenas desses parâmetros ajustáveis. Definimos assim um vetor de parâmetros para esta função, $\widehat{\mathbf{w}} \in \mathbb{R}^{np}$, com $np = M(N+1)$, contemplando todos os parâmetros ajustáveis da rede, na forma:

$$\widehat{\mathbf{w}} = \begin{bmatrix} \mathbf{w}_1 \\ \mathbf{w}_2 \\ \vdots \\ \mathbf{w}_M \end{bmatrix}_{np}$$

Observe que $\widehat{w}_1 = w_{10}$, que $\widehat{w}_{(N+1)} = w_{1N}$, que $\widehat{w}_{(N+2)} = w_{20}$, e assim sucessivamente até $\widehat{w}_{M(N+1)} = w_{MN}$.

Definimos também um vetor de diferenças de pesos na forma:

$$\Delta\widehat{\mathbf{w}}(k) = \widehat{\mathbf{w}}(k+1) - \widehat{\mathbf{w}}(k) \tag{1.3}$$

O algoritmo do gradiente sugere a obtenção do vetor de diferenças de pesos a partir do vetor gradiente da função de custo, $J(k, \widehat{\mathbf{w}})$, em relação aos parâmetros ajustáveis na forma:

$$\Delta\widehat{\mathbf{w}}(k) = -\,\eta\,\nabla J(k, \widehat{\mathbf{w}}) \tag{1.4}$$

onde η é um fator de ponderação, no nosso caso chamado de *taxa de aprendizado*, e ∇ é o operador gradiente aplicado à função objetivo na forma:

$$\begin{aligned} \nabla J(k, \widehat{\mathbf{w}}) &= \frac{\partial J(k, \widehat{\mathbf{w}})}{\partial \widehat{\mathbf{w}}} \in \mathbb{R}^{np} \\ &= \begin{bmatrix} \frac{\partial J(k,\widehat{\mathbf{w}})}{\partial \widehat{w}_1} \\ \frac{\partial J(k,\widehat{\mathbf{w}})}{\partial \widehat{w}_2} \\ \vdots \\ \frac{\partial J(k,\widehat{\mathbf{w}})}{\partial \widehat{w}_{np}} \end{bmatrix}_{np} \end{aligned} \tag{1.5}$$

Este vetor gradiente pode ser obtido a partir da equação 1.2 como:

$$\begin{aligned} \nabla J(k, \widehat{\mathbf{w}}) &= \left[\frac{\partial \mathbf{e}(k)}{\partial \widehat{\mathbf{w}}}\right]' \mathbf{e}(k) \\ &= -\left[\frac{\partial \mathbf{y}(k)}{\partial \widehat{\mathbf{w}}}\right]' \mathbf{e}(k) \end{aligned} \tag{1.6}$$

O termo $\partial \mathbf{y}(k)/\partial \widehat{\mathbf{w}} \in \mathbb{R}^{M \times np}$ é chamado de *Matriz Jacobiana* ou *Matriz de Sensibilidade* da saída da rede em relação ao vetor de parâmetros ajustáveis $\widehat{\mathbf{w}}$, possuindo a forma:

$$\frac{\partial \mathbf{y}(k)}{\partial \widehat{\mathbf{w}}} = \begin{bmatrix} \frac{\partial y_1(k)}{\partial \widehat{w}_1} & \frac{\partial y_1(k)}{\partial \widehat{w}_2} & \cdots & \frac{\partial y_1(k)}{\partial \widehat{w}_{np}} \\ \frac{\partial y_2(k)}{\partial \widehat{w}_1} & \frac{\partial y_2(k)}{\partial \widehat{w}_2} & \cdots & \frac{\partial y_2(k)}{\partial \widehat{w}_{np}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial y_M(k)}{\partial \widehat{w}_1} & \frac{\partial y_M(k)}{\partial \widehat{w}_2} & \cdots & \frac{\partial y_M(k)}{\partial \widehat{w}_{np}} \end{bmatrix}_{M \times np}$$

ou na forma de um vetor linha:

$$\frac{\partial \mathbf{y}(k)}{\partial \widehat{\mathbf{w}}} = \begin{bmatrix} \frac{\partial \mathbf{y}(k)}{\partial \widehat{w}_1} & \frac{\partial \mathbf{y}(k)}{\partial \widehat{w}_2} & \cdots & \frac{\partial \mathbf{y}(k)}{\partial \widehat{w}_{np}} \end{bmatrix}_{np}$$

A matriz jacobiana pode ser obtida a partir da equação 1.1 como:

$$\frac{\partial \mathbf{y}(k)}{\partial \widehat{\mathbf{w}}} = \frac{\partial \mathbf{g}(k)}{\partial \mathbf{h}} \frac{\partial \mathbf{h}(k)}{\partial \widehat{\mathbf{w}}} \tag{1.7}$$

Perceba que o termo $\partial \mathbf{g}(k)/\partial \mathbf{h} \in \mathbb{R}^{M \times M}$ é uma matriz diagonal, com componentes $\partial g_{ii}(k)/\partial h_{ii}$, e que o termo $\partial \mathbf{h}(k)/\partial \widehat{\mathbf{w}} \in \mathbb{R}^{M \times np}$, possui a mesma dimensão do jacobiano, tendo a forma:

$$\frac{\partial \mathbf{h}(k)}{\partial \widehat{\mathbf{w}}} = \begin{bmatrix} x_0 & x_1 & \cdots & x_N & 0 & 0 & \cdots & 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & x_0 & x_1 & \cdots & x_N & \cdots & 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & \cdots & x_0 & x_1 & \cdots & x_N \end{bmatrix}_{M \times np} \tag{1.8}$$

ou, se escrita na forma de matriz pseudo-diagonal:

$$\frac{\partial \mathbf{h}(k)}{\partial \widehat{\mathbf{w}}} = \begin{bmatrix} \mathbf{x} & 0_{(N+1)} & \cdots & 0_{(N+1)} \\ 0_{(N+1)} & \mathbf{x} & \cdots & 0_{(N+1)} \\ 0_{(N+1)} & 0_{(N+1)} & \cdots & 0_{(N+1)} \\ \vdots & \vdots & \ddots & \vdots \\ 0_{(N+1)} & 0_{(N+1)} & \cdots & \mathbf{x} \end{bmatrix}_{M \times M} \tag{1.9}$$

Definindo-se uma matriz $\widehat{\mathbf{x}} \in \mathbb{R}^{M \times np}$ na forma:

$$\widehat{\mathbf{x}}(k) = \frac{\partial \mathbf{h}(k)}{\partial \widehat{\mathbf{w}}}$$

e substituindo a equação 1.7 na equação 1.6 teremos:

$$\nabla J(k, \hat{\mathbf{w}}) = - \left[\frac{\partial \mathbf{g}(k)}{\partial \mathbf{h}} \hat{\mathbf{x}}(k)\right]' \mathbf{e}(k) \tag{1.10}$$

A partir da equação 1.3, a atualização dos pesos da rede segue a forma:

$$\begin{aligned} \hat{\mathbf{w}}(k+1) &= \hat{\mathbf{w}}(k) + \Delta\hat{\mathbf{w}}(k) \\ &= \hat{\mathbf{w}}(k) + (-\eta \nabla J(k)) \\ &= \hat{\mathbf{w}}(k) + \eta \left[\frac{\partial \mathbf{g}(k)}{\partial \mathbf{h}} \hat{\mathbf{x}}(k)\right]' \mathbf{e}(k) \end{aligned} \tag{1.11}$$

Cada peso individual w_{ij} da rede é atualizado por um Δw_{ij} expresso por:

$$\Delta w_{ij} = \eta\, e_i(k) \frac{\partial g_{ii}(k)}{\partial h_{ii}} x_j(k) \tag{1.12}$$

A sensibilidade da i-ésima saída em relação a cada peso individual w_{ij} é expresso por:

$$\frac{\partial y_i(k)}{\partial w_{ij}} = \frac{\partial g_{ii}(k)}{\partial h_{ii}} x_j(k) \tag{1.13}$$

1.4 O *Perceptron* de Múltiplas Camadas

Para análise do *perceptron* com múltiplas camadas, consideraremos, sem perda de generalidade, uma rede com duas camadas como visto na figura 1.2.

O exemplo da figura 1.2 é de uma rede de duas camadas, com dois neurônios na camada de saída, a segunda, três neurônios na primeira camada e quatro sinais de entrada. Observe a existência de *bias* na(s) camada(s) intermediária(s). Esta rede é também conhecida como *Feedforward Neural Network* (**FNN**).

Supondo uma rede com L camadas, N entradas e cada camada possuindo M_l neurônios, será adotada a seguinte convenção:

- $\mathbf{y}(k) \in \mathbb{R}^{M_L}$ é o vetor de saída da última camada da rede, possuindo componentes y_i;
- $\mathbf{y}_l(k) \in \mathbb{R}^{(M_l+1)}$ é o vetor de saída de cada camada da rede com a inclusão do *bias*, possuindo componentes y_{l_j};

- $\mathbf{y}_l^-(k) \in \mathbb{R}^{M_l}$ é o vetor de saída de cada camada da rede sem a inclusão do *bias*, possuindo componentes $y_{l_j}^-$;
- $\mathbf{x}(k) \in \mathbb{R}^{(N+1)}$ é o vetor de entrada da rede com componentes x_n;
- $\mathbf{w}_l \in \mathbb{R}^{M_l \times (M_{(l-1)}+1)}$ é a matriz de conexões entrada-saída entre camadas com a inclusão do *bias* de entrada, possuindo componentes $w_{l_{ij}}$;
- $\mathbf{w}_l^- \in \mathbb{R}^{M_l \times M_{(l-1)}}$ é a matriz de conexões entrada-saída sem a inclusão do *bias* de entrada, possuindo componentes $w_{l_{ij}}$;
- $\mathbf{g}_l(\cdot) \in \mathbb{R}^{M_l}$ é o vetor das função de ativação de cada camada com componentes $g_{l_i}(\cdot)$;
- $\mathbf{h}_l(\cdot) \in \mathbb{R}^{M_l}$ é o vetor de entradas ponderadas com componentes $h_{l_i}(\cdot)$.

O sinal de saída da última camada de uma rede com $L = 2$, $\mathbf{y}(k)$, pode ser descrito por:

$$\begin{cases} \mathbf{h}_1(k) &= \mathbf{w}_1\, \mathbf{x}(k) \\ \mathbf{y}_1^-(k) &= \mathbf{g}_1\left(\mathbf{h}_1(k)\right) \\ \mathbf{h}_2(k) &= \mathbf{w}_2\, \mathbf{y}_1(k) \\ \mathbf{y}(k) &= \mathbf{g}_2\left(\mathbf{h}_2(k)\right) \end{cases} \tag{1.14}$$

Observe que na equação 1.14, o vetor $\mathbf{x} \in \mathbb{R}^{N+1}$ tem a forma $\mathbf{x} = [x_0 \;\; x_1 \;\; \cdots \;\; x_N]'$, o vetor $\mathbf{y}_l \in \mathbb{R}^{M_l}$ a forma $\mathbf{y}_l = [y_{l_0} \;\; y_{l_1} \;\; \cdots \;\; y_{l_{M_l}}]'$ e o vetor $\mathbf{y}_l^- \in \mathbb{R}^{M_l}$ a forma $\mathbf{y}_l^- = [y_{l_1}^- \;\; \cdots \;\; y_{l_{M_l}}^-]'$. A matriz $\mathbf{w}_l \in \mathbb{R}^{M \times (N+1)}$, tem forma:

$$\mathbf{w}_l = \begin{bmatrix} w_{l_{10}} & w_{l_{11}} & \cdots & w_{l_{1M_{(l-1)}}} \\ w_{l_{20}} & w_{l_{21}} & \cdots & w_{l_{2M_{(l-1)}}} \\ \vdots & \vdots & \ddots & \vdots \\ w_{l_{M_l0}} & w_{l_{M_l1}} & \cdots & w_{l_{M_lM_{(l-1)}}} \end{bmatrix}_{M_l \times (M_{(l-1)}+1)}$$

ou alternativamente:

$$\mathbf{w} = \begin{bmatrix} \mathbf{w}'_{l_1} \\ \mathbf{w}'_{l_2} \\ \vdots \\ \mathbf{w}'_{l_{M_l}} \end{bmatrix}_{M_l}$$

onde os $\mathbf{w}_{l_i} \in \mathbb{R}^{(M_l+1)}$ são vetores de pesos de cada neurônio individual de cada camada da

rede com a inclusão do *bias*. A matriz $\mathbf{w}_l^-$, com componentes $w_{l_{ij}}$, possui a forma:

$$\mathbf{w}_l^- = \begin{bmatrix} w_{l_{11}} & \cdots & w_{l_{1M_{(l-1)}}} \\ w_{l_{21}} & \cdots & w_{l_{2M_{(l-1)}}} \\ \vdots & \ddots & \vdots \\ w_{l_{M_l1}} & \cdots & w_{l_{M_lM_{(l-1)}}} \end{bmatrix}_{M_l \times M_{(l-1)}}$$

1.4.1 O Aprendizado pelo Gradiente com Inclusão da Retro-Propagação de sinais através da Rede

Para uma rede com mais de uma camada existe um problema na aplicação direta do algoritmo descrito na seção 1.3.2. É que não se sabe qual o valor desejado nas saídas das camadas intermediárias.

O algoritmo de *retro-propagação* é bastante utilizado atualmente, visto que o mesmo transpõe este problema. Ele foi desenvolvido independentemente diversas vezes por Bryson e Ho [14], Parker [54], Rumelhart *et al* [64] e Werbos [70]. O algoritmo permite a propagação do erro através da rede tendo por base a regra de derivadas em cadeias, possibilitando assim o uso da *lei de aprendizado* pelo gradiente, apresentado na seção 1.3.2, para uma rede com L camadas.

Definimos então, como na seção 1.3.2, um vetor $\mathbf{y}_d = [y_{d_1} \;\; y_{d_2} \;\; \cdots \;\; y_{d_{M_L}}]'$, com $\mathbf{y}_d \in \mathbb{R}^{M_L}$, onde y_{d_i} é a saída desejada para o i-ésimo neurônio da L-ésima camada da rede, no nosso caso, a segunda. Também um vetor erro na forma $\mathbf{e} = (\mathbf{y}_d - \mathbf{y})$, com $\mathbf{e} \in \mathbb{R}^{M_L}$, com componentes os e_i. Para o erro propagado à saida da l-ésima camada definimos um vetor $\mathbf{e}_l \in \mathbb{R}^{M_l}$.

Podemos então definir uma função de custo para a saída da rede com a mesma forma da equação 1.2. Note que a mesma é escalar, em função de um conjunto de parâmetros ajustáveis $\widehat{\mathbf{w}}$, e que a sua minimização depende apenas desses parâmetros. Neste caso particular o vetor $\widehat{\mathbf{w}}$ contempla os pesos da primeira e da segunda camada, $w_{1_{jn}}$ e $w_{2_{ij}}$. Subdividiremos então este vetor em dois vetores, $\widehat{\mathbf{w}}_1 \in \mathbb{R}^{np_1}$ para os parâmetros da primeira camada, e $\widehat{\mathbf{w}}_2 \in \mathbb{R}^{np_2}$ para os parâmetros da segunda, sendo eles montados da mesma maneira que a descrita na seção 1.3.2. Assim, o vetor $\widehat{\mathbf{w}}$ possui composição:

$$\widehat{\mathbf{w}} = \begin{bmatrix} \widehat{\mathbf{w}}_2 \\ \widehat{\mathbf{w}}_1 \end{bmatrix}_{np}$$

onde $np_1 = (M_1(N+1))$, $np_2 = (M_2(M_1+1))$ e $np = np_1 + np_2$.

Adotaremos também as mesmas definições expressas pelas equações 1.3, 1.4, 1.5 e 1.6. Note que o jacobiano da saída da rede, $\mathbf{y}$, em relação ao vetor $\widehat{\mathbf{w}}$ pode ser expresso na forma:

$$\frac{\partial \mathbf{y}(k)}{\partial \widehat{\mathbf{w}}} = \left[\begin{array}{cc} \frac{\partial \mathbf{y}(k)}{\partial \widehat{\mathbf{w}}_2} & \frac{\partial \mathbf{y}(k)}{\partial \widehat{\mathbf{w}}_1} \end{array} \right]_{M_2 \times np}$$

Com a Definição de uma matriz $\widehat{\mathbf{x}}_2 \in \mathbb{R}^{M_2 \times np_2}$

$$\widehat{\mathbf{x}}_2(k) = \frac{\partial \mathbf{h}_2(k)}{\partial \widehat{\mathbf{w}}_2}$$

o jacobiano em relação ao subvetor $\widehat{\mathbf{w}}_2$, a partir da equação 1.14, possuirá a forma:

$$\frac{\partial \mathbf{y}(k)}{\partial \widehat{\mathbf{w}}_2} = \frac{\partial \mathbf{g}_2(k)}{\partial \mathbf{h}_2} \widehat{\mathbf{x}}_2(k) \tag{1.15}$$

Observe que a equação 1.15 tem a mesma forma da equação 1.7. O vetor de entrada para a segunda camada é o de saída da primeira. Assim, no termo $\partial \mathbf{h}_2(k)/\partial \widehat{\mathbf{w}}_2$ no lugar das componentes x_n aparecerão as componentes y_{1_j} do vetor de saída da primeira camada.

O jacobiano em relação ao subvetor $\widehat{\mathbf{w}}_1$, a partir da equação 1.14, possui a forma:

$$\frac{\partial \mathbf{y}(k)}{\partial \widehat{\mathbf{w}}_1} = \frac{\partial \mathbf{g}_2(k)}{\partial \mathbf{h}_2} \frac{\partial \mathbf{h}_2(k)}{\partial \mathbf{g}_1} \frac{\partial \mathbf{g}_1(k)}{\partial \mathbf{h}_1} \frac{\partial \mathbf{h}_1(k)}{\partial \widehat{\mathbf{w}}_1} \tag{1.16}$$

Perceba que na equação 1.16, $\partial \mathbf{h}_2(k)/\partial \mathbf{g}_1 = \mathbf{w}_2^-$ visto que, de fato, o *bias* de uma camada não se propaga para uma camada anterior. A partir dessa equação podemos definir uma matriz de propagação da saída da segunda camada até a saída da primeira na forma:

$$\begin{aligned} \boldsymbol{\lambda}_1(k) &= \frac{\partial \mathbf{g}_2(k)}{\partial \mathbf{h}_2} \frac{\partial \mathbf{h}_2(k)}{\partial \mathbf{g}_1} \\ &= \frac{\partial \mathbf{g}_2(k)}{\partial \mathbf{h}_2} \mathbf{w}_2^- \end{aligned} \tag{1.17}$$

A matrizes de propagação $\boldsymbol{\lambda}_l(k) = \partial \mathbf{y}/\partial \mathbf{y}_l^- \in \mathbb{R}^{M_L \times M_l}$, com componentes λ_{ij}, propagam o erro, ou outro sinal qualquer, da saída da rede para a saída de alguma camada intermediária ou para a entrada da rede. No nosso caso, temos a matriz λ_1, que propaga a saída da rede à saída da primeira camada. Esta é a essência do algoritmo de retro-propagação, e assim permite a utilização do algoritmo do gradiente no aprendizado de redes de mais de uma camada. Se definirmos uma matriz de propagação para a camada de saída

igual a matriz identidade, $\boldsymbol{\lambda}_L(k) = \mathbf{I}_{M_L \times M_L}$, poderemos fazer a seguinte generalização para a sua obtenção:

$$\boldsymbol{\lambda}_l(k) = \begin{cases} \mathbf{I}_{M_L} & \text{se } l = L \\ \boldsymbol{\lambda}_{(l+1)} \frac{\partial \mathbf{g}_{(l+1)}(k)}{\partial \mathbf{h}_{(l+1)}} \mathbf{w}^{-}_{(l+1)} & \text{se } l \neq L \end{cases} \tag{1.18}$$

Convencionaremos a matriz de propagação até à entrada da rede como $\boldsymbol{\lambda}_0$. Note que $\boldsymbol{\lambda}_0$ corresponde ao jacobiano da saída da rede em relação às suas entradas.

Para o erro propagado à saida da l-ésima camada teremos:

$$\mathbf{e}_l = [\boldsymbol{\lambda}_l(k)]' \, \mathbf{e}(k) \tag{1.19}$$

Definindo-se uma matriz $\widehat{\mathbf{x}}_1 \in \mathbb{R}^{M_2 \times np_1}$ na forma:

$$\widehat{\mathbf{x}}_1(k) = \frac{\partial \mathbf{h}_1(k)}{\partial \widehat{\mathbf{w}}_1}$$

a matriz jacobiana pode ser reescrita na como:

$$\frac{\partial \mathbf{y}(k)}{\partial \widehat{\mathbf{w}}} = \left[\left[\boldsymbol{\lambda}_2(k) \frac{\partial \mathbf{g}_2(k)}{\partial \mathbf{h}_2} \widehat{\mathbf{x}}_2(k) \right]_{M_2 \times np_2} \quad \left[\boldsymbol{\lambda}_1(k) \frac{\partial \mathbf{g}_1(k)}{\partial \mathbf{h}_1} \widehat{\mathbf{x}}_1(k) \right]_{M_2 \times np_1} \right]_{M_2 \times np} \tag{1.20}$$

Cada peso individual $w_{2_{ij}}$ da segunda camada da rede é atualizado por um $\Delta w_{2_{ij}}$ expresso por:

$$\Delta w_{2_{ij}} = \eta \, e_i(k) \frac{\partial g_{2_{ii}}(k)}{\partial h_{2_{ii}}} y_{1_j}(k) \tag{1.21}$$

A sensibilidade da i-ésima saída em relação a cada peso individual $w_{2_{ij}}$ é expresso por:

$$\frac{\partial y_i(k)}{\partial w_{2_{ij}}} = \frac{\partial g_{2_{ii}}(k)}{\partial h_{2_{ii}}} y_{1_j}(k) \tag{1.22}$$

Cada peso individual $w_{1_{jn}}$ da segunda camada da rede é atualizado por um $\Delta w_{1_{jn}}$ expresso por:

$$\Delta w_{1_{jn}} = \eta \left(\sum_{i=1}^{M_L} e_i(k) \, \lambda_{1_{ij}}(k) \right) \frac{\partial g_{1_{jj}}(k)}{\partial h_{1_{jj}}} x_n(k) \tag{1.23}$$

A sensibilidade da i-ésima saída em relação a cada peso individual $w_{1_{jn}}$ é expresso por:

$$\frac{\partial y_i(k)}{\partial w_{1_{jn}}} = \lambda_{1_{ij}}(k) \frac{\partial g_{1_{jj}}(k)}{\partial h_{1_{jj}}} x_n(k) \tag{1.24}$$

1.5 O *Perceptron* Recursivo

Almeida [3, 4] e Pineda [55, 56] introduziram uma extensão para a rede a *perceptron* dotando-a de recorrência ou realimentação saída-entrada. As redes discutidas nas seções 1.3 e 1.4 são chamadas de redes a neurônios *não recursivos* e nesta seção discutiremos o neurônio *recursivos*, que pode apresentar uma realimentação saída-entrada como mostrado na figura 1.4.

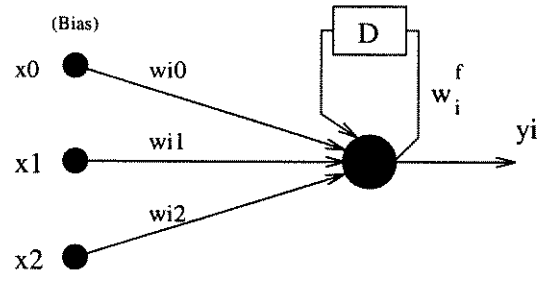

Figura 1.4: *Perceptron* recursivo.

O neurônio apresentado na figura 1.4 possui duas entradas externas e uma de realimentação saída-entrada (*feedback*). Na figura o bloco **D** significa *Delay* ou deslocamento para trás no tempo, sendo a sua *Transformada* $\mathcal{Z}$ igual a z^{-1} [28]. Assim como no *perceptron não recursivo*, ao *perceptron recursivo* costuma-se adicionar uma entrada com valor constante, o *bias*. Supondo uma rede de única camada com M neurônios e N entradas, adotaremos a seguinte convenção:

- $\mathbf{y}(k) \in \mathbb{R}^M$ é o vetor de saída com componentes y_i;
- $\mathbf{x}(k) \in \mathbb{R}^{N+1}$ é o vetor de entrada com componentes x_j;
- $\mathbf{w} \in \mathbb{R}^{M \times (N+1)}$ é a matriz de pesos de conexões entrada-saída com componentes w_{ij};
- $\mathbf{w}^f \in \mathbb{R}^{M \times M}$ é a matriz de recursividade saída-entrada com componentes w_{is}^f;
- $\mathbf{g}(\cdot) \in \mathbb{R}^M$ é o vetor de funções de ativação com componentes $g_i(\cdot)$;
- $\mathbf{h}(\cdot) \in \mathbb{R}^M$ é o vetor de entradas ponderadas com componentes $h_i(\cdot)$

O sinal de saída da rede pode ser descrito por:

$$\begin{cases} \mathbf{h}(k) &= \mathbf{w}\,\mathbf{x}(k) + \mathbf{w}^f\,\mathbf{y}(k-1) \\ \mathbf{y}(k) &= \mathbf{g}(\mathbf{h}(k)) \\ \mathbf{y}(0) &= 0_M \end{cases} \tag{1.25}$$

Observe que na equação 1.25 o vetor $\mathbf{x} \in \mathbb{R}^{N+1}$ tem a forma $\mathbf{x} = [x_0 \;\; x_1 \;\; \cdots \;\; x_N]'$, o vetor $\mathbf{y} \in \mathbb{R}^M$ a forma $\mathbf{y} = [y_1 \;\; y_2 \;\; \cdots \;\; y_M]'$. A matriz $\mathbf{w} \in \mathbb{R}^{M\times(N+1)}$ tem a forma $\mathbf{w} = [\mathbf{w}_1 \;\; \mathbf{w}_2 \;\; \cdots \;\; \mathbf{w}_M]'$, onde $\mathbf{w}_i$ correspode aos pesos do i-ésimo neurônio individual da rede. Logo $\mathbf{w}$ pode ser expressa como:

$$\mathbf{w} = \begin{bmatrix} w_{10} & w_{11} & \cdots & w_{1N} \\ w_{20} & w_{21} & \cdots & w_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ w_{M0} & w_{M1} & \cdots & w_{MN} \end{bmatrix}$$

e a matriz $\mathbf{w}^f$, com $\mathbf{w}^f \in \mathbb{R}^{M\times M}$, na forma:

$$\mathbf{w}^f = \begin{bmatrix} w^f_{11} & w^f_{12} & \cdots & w^f_{1M} \\ w^f_{21} & w^f_{22} & \cdots & w^f_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ w^f_{M1} & w^f_{M2} & \cdots & w^f_{MM} \end{bmatrix} = \begin{bmatrix} \mathbf{w}^f_1 \\ \mathbf{w}^f_2 \\ \vdots \\ \mathbf{w}^f_M \end{bmatrix}$$

onde w^f_{is} corresponde aos pesos de *feedback* do neurônio s para o neurônio i. Neste caso, temos a recursividade total entre os neurônios da camada. Se tivéssemos feito os elementos de fora da diagonal principal de $\mathbf{w}^f$ iguais a zero teríamos o caso de uma camada apenas com neurônios auto-recursivos. Sem perda de generalidade, abordaremos aqui apenas o segundo caso, com neurônios auto-recursivos, e portanto $\mathbf{w}^f$ será diagonal e referir-nos-emos aos seus elementos por w^f_{ii}.

1.5.1 Aprendizado em *Perceptrons* Recursivos

O algoritmo de aprendizado para o *perceptron* recursivo também é baseado no gradiente. Definimos então, como nas seções 1.3.2 e 1.4.1, um vetor $\mathbf{y}_d \in \mathbb{R}^M$, onde y_{d_i} é a saída desejada para o i-neurônio da rede e também um vetor de erro na forma $\mathbf{e} = (\mathbf{y}_d - \mathbf{y})$, com $\mathbf{e} \in \mathbb{R}^M$, onde os e_i são os elementos individuais do vetor de erro.

Podemos definir uma função de custo para a saída da rede igual à equação 1.2, escalar e função dos parâmetros ajustavéis da rede, dos quais depende sua minimização. Neste caso particular, o nosso vetor $\widehat{\mathbf{w}}$ contempla os parâmetros w_{ij} e w^f_{ii}. Novamente dividiremos o vetor $\widehat{\mathbf{w}}$ em dois outros, $\widehat{\mathbf{w}}_1 \in \mathbb{R}^{np_1}$ para todos os parâmetros w_{ij} e $\widehat{\mathbf{w}}^f_1 \in \mathbb{R}^{np^f_1}$ para todos os parâmetros w^f_{ii}. Assim, o vetor $\widehat{\mathbf{w}}$ terá a seguinte composição:

$$\widehat{\mathbf{w}} = \begin{bmatrix} \widehat{\mathbf{w}}_1 \\ \widehat{\mathbf{w}}^f_1 \end{bmatrix}_{np}$$

onde $np_1 = (M(N+1))$, $np_1^f = M$ e $np = np_1 + np_1^f$.

Adotaremos também as mesmas definições expressas pelas equações 1.3, 1.4, 1.5 e 1.6.

O jacobiano em relação ao vetor de pesos $\widehat{\mathbf{w}}_1^f$, obtido a partir da equação 1.25, tem a forma:

$$\frac{\partial \mathbf{y}(k)}{\partial \widehat{\mathbf{w}}_1^f} = \frac{\partial \mathbf{g}(k)}{\partial \mathbf{h}} \frac{\partial \mathbf{h}(k)}{\partial \widehat{\mathbf{w}}_1^f} + \frac{\partial \mathbf{g}(k)}{\partial \mathbf{h}} \mathbf{w}^f(k) \frac{\partial \mathbf{y}_1(k-1)}{\partial \widehat{\mathbf{w}}_1^f} \tag{1.26}$$

Note que $\partial \mathbf{g}(k)/\partial \mathbf{h}$ é diagonal e que $\partial \mathbf{h}(k)/\partial \widehat{\mathbf{w}}_1^f$ é diagonal na forma:

$$\frac{\partial \mathbf{h}(k)}{\partial \widehat{\mathbf{w}}^f} = \begin{bmatrix} y_1 & 0 & \cdots & 0 \\ 0 & y_2 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & y_M \end{bmatrix}_{M \times M} \tag{1.27}$$

O termo $\partial \mathbf{y}(k-1)/\partial \widehat{\mathbf{w}}_1^f$ possui a forma:

$$\frac{\partial \mathbf{y}(k-1)}{\partial \widehat{\mathbf{w}}_1^f} = \frac{\partial \mathbf{g}(k-1)}{\partial \mathbf{h}} \frac{\partial \mathbf{h}(k-1)}{\partial \widehat{\mathbf{w}}_1^f} + \frac{\partial \mathbf{g}(k-1)}{\partial \mathbf{h}} \mathbf{w}^f(k-1) \frac{\partial \mathbf{y}(k-2)}{\partial \widehat{\mathbf{w}}_1^f} \tag{1.28}$$

Definindo-se uma matriz $\widehat{\mathbf{x}}^f$ na forma:

$$\begin{cases} \widehat{\mathbf{x}}^f & = \frac{\partial \mathbf{h}(k)}{\partial \widehat{\mathbf{w}}_1^f} + \mathbf{w}^f(k) \frac{\partial \mathbf{y}(k-1)}{\partial \widehat{\mathbf{w}}_1^f} \\ \frac{\partial \mathbf{y}(0)}{\partial \widehat{\mathbf{w}}_1^f} & = 0_{M \times np_1^f} \end{cases} \tag{1.29}$$

teremos para o jacobiano em relação a $\widehat{\mathbf{w}}_1^f$:

$$\frac{\partial \mathbf{y}(k)}{\partial \widehat{\mathbf{w}}_1^f} = \frac{\partial \mathbf{g}(k)}{\partial \mathbf{h}} \widehat{\mathbf{x}}^f(k) \tag{1.30}$$

O jacobiano em relação ao vetor de pesos $\widehat{\mathbf{w}}_1$, obtido a partir da equação 1.25, tem a forma:

$$\frac{\partial \mathbf{y}(k)}{\partial \widehat{\mathbf{w}}_1} = \frac{\partial \mathbf{g}(k)}{\partial \mathbf{h}} \frac{\partial \mathbf{h}(k)}{\partial \widehat{\mathbf{w}}_1} + \frac{\partial \mathbf{g}(k)}{\partial \mathbf{h}} \mathbf{w}^f(k) \frac{\partial \mathbf{y}(k-1)}{\partial \widehat{\mathbf{w}}_1} \tag{1.31}$$

Observe que o termo $\partial \mathbf{h}(k)/\partial \widehat{\mathbf{w}}_1$ tem a mesma composição apresentada nas equações 1.8 e 1.9. Temos ainda que:

$$\frac{\partial \mathbf{y}(k-1)}{\partial \widehat{\mathbf{w}}_1} = \frac{\partial \mathbf{g}(k-1)}{\partial \mathbf{h}} \frac{\partial \mathbf{h}(k-1)}{\partial \widehat{\mathbf{w}}_1} + \frac{\partial \mathbf{g}(k-1)}{\partial \mathbf{h}} \mathbf{w}^f(k-1) \frac{\partial \mathbf{y}(k-2)}{\partial \widehat{\mathbf{w}}_1} \tag{1.32}$$

Definindo-se uma matriz $\widehat{\mathbf{x}}_1$ na forma:

$$\begin{cases} \widehat{\mathbf{x}} & = & \frac{\partial \mathbf{h}(k)}{\partial \widehat{\mathbf{w}}_1} + \mathbf{w}^f(k) \frac{\partial \mathbf{y}(k-1)}{\partial \widehat{\mathbf{w}}_1} \\ \frac{\partial \mathbf{y}(0)}{\partial \widehat{\mathbf{w}}_1} & = & 0_{M \times np_1} \end{cases} \tag{1.33}$$

teremos para o jacobiano em relação a $\widehat{\mathbf{w}}$:

$$\frac{\partial \mathbf{y}(k)}{\partial \widehat{\mathbf{w}}_1} = \frac{\partial \mathbf{g}(k)}{\partial \mathbf{h}} \widehat{\mathbf{x}}(k) \tag{1.34}$$

Assim, a matriz jacobiana pode ser reescrita na forma:

$$\frac{\partial \mathbf{y}(k)}{\partial \widehat{\mathbf{w}}} = \frac{\partial \mathbf{g}(k)}{\partial \mathbf{h}} \left[\begin{array}{cc} \widehat{\mathbf{x}} & \widehat{\mathbf{x}}^f \end{array} \right]_{M \times np} \tag{1.35}$$

A matriz de propagação através de uma camada recursiva, $(l+1)$, propagada para uma camada qualquer l, anterior à camada recursiva, tem o formato:

$$\begin{cases} \boldsymbol{\lambda}_l(k) & = & \boldsymbol{\lambda}_{(l+1)}(k) \frac{\partial \mathbf{g}_{(l+1)}(k)}{\partial \mathbf{y}_l^-} \\ \frac{\partial \mathbf{g}_{(l+1)}(k)}{\partial \mathbf{y}_l^-} & = & \frac{\partial \mathbf{g}_{(l+1)}(k)}{\partial \mathbf{h}_{(l+1)}} \left(\mathbf{w}_{(l+1)}^- + \mathbf{w}^f \frac{\partial \mathbf{g}_{(l+1)}(k-1)}{\partial \mathbf{y}_l^-} \right) \\ \frac{\partial \mathbf{g}_{(l+1)}(0)}{\partial \mathbf{y}_l^-} & = & 0_{M_L \times M_l} \end{cases} \tag{1.36}$$

Note que, em todos os desenvolvimentos feitos nessa seção, se fizermos $\mathbf{w}^f = 0_{M \times M}$ teremos as expressões obtidas para os perceptrons não recursivos. O algoritmo de retro-propagação como expresso na equação 1.36, é chamado de retro-propagação dinâmica.

Cada peso individual w_{ij} da rede é atualizado por um Δw_{ij} expresso por:

$$\Delta w_{ij} = \eta \, e_i(k) \frac{\partial g_{ii}(k)}{\partial h_{ii}} \widehat{x}_{im}(k) \tag{1.37}$$

onde $m = (M\, i + j)$.

A sensibilidade da i-ésima saída em relação a cada peso individual $w_{2_{ij}}$ é expresso por:

$$\frac{\partial y_i(k)}{\partial w_{ij}} = \frac{\partial g_{2_{ii}}(k)}{\partial h_{ii}} \widehat{x}_{im}(k) \tag{1.38}$$

Cada peso individual w_{ii}^f da rede é atualizado por um Δw_{ii}^f expresso por:

$$\Delta w_{ii}^f = \eta \, e_i(k) \frac{\partial g_{ii}(k)}{\partial h_{ii}} \widehat{x}_{ii}(k) \tag{1.39}$$

A sensibilidade da i-ésima saída em relação a cada peso individual w_{ii}^f é expresso por:

$$\frac{\partial y_i(k)}{\partial w_{1_{jn}}} = \frac{\partial g_{ii}(k)}{\partial h_{ii}} \, \hat{x}_{ii}(k) \tag{1.40}$$

1.6 Rede de Múltiplas Camadas com *Perceptrons* Recursivos

Nesta seção apresentaremos uma rede que mistura *perceptrons recursivos e não recursivos*. Os não recursivos são utilizados na camada de saída e os recursivos nas intermediárias. Sem perda de generalidade, restrigir-nos-emos ao caso de uma rede de duas camadas sendo, a primeira dinâmica e a segunda estática. Esta rede é a mesma apresentada por Ku e Lee [34] e chamada de *Diagonal Recurrent Neural Network* (**DRNN**).

Supondo uma rede com M_2 neurônios na segunda camada, M_1 na primeira e N entradas, vetor de saída desta rede tem a forma:

$$\begin{cases} \mathbf{h}_1(k) &= \mathbf{w}_1 \, \mathbf{x}(k) + \mathbf{w}_1^f \, \mathbf{y}^-(k-1) \\ \mathbf{y}_1^-(k) &= \mathbf{g}_1 \left(\mathbf{h}_1(k) \right) \\ \mathbf{h}_2(k) &= \mathbf{w}_2 \, \mathbf{y}_1(k) \\ \mathbf{y}(k) &= \mathbf{g}_2 \left(\mathbf{h}_2(k) \right) \\ \mathbf{y}^-(0) &= 0_{M_1} \end{cases} \tag{1.41}$$

O aprendizado para esta rede pode ser feito com os processos descritos nas seções 1.4.1 e 1.5.1, observando que neste caso a primeira camada é composta de neurônios recursivos. Aqui, o vetor $\widehat{\mathbf{w}}$ possui três componentes, $\widehat{\mathbf{w}}_1 \in \mathbb{R}^{np_1}$ para os parâmetros $w_{1_{jn}}$ da primeira camada, $\widehat{\mathbf{w}}_1^f \in \mathbb{R}^{np_1^f}$ para os parâmetros $w_{1_{jj}}^f$, e $\widehat{\mathbf{w}}_2 \in \mathbb{R}^{np_2}$ para os parâmetros $w_{2_{ij}}$ da segunda camada. Assim o vetor $\widehat{\mathbf{w}}$ possui a forma:

$$\widehat{\mathbf{w}} = \begin{bmatrix} \widehat{\mathbf{w}}_2 \\ \widehat{\mathbf{w}}_1 \\ \widehat{\mathbf{w}}_1^f \end{bmatrix}_{np}$$

onde $np_1 = (M_1(N+1))$, $np_1^f = M_1$, $np_2 = (M_2(M_1+1))$ e $np = np_1 + np_1^f$.

A matriz jacobiana tem portanto a forma:

$$\frac{\partial \mathbf{y}(k)}{\partial \widehat{\mathbf{w}}} = \begin{bmatrix} \frac{\partial \mathbf{y}(k)}{\partial \widehat{\mathbf{w}}_2} & \frac{\partial \mathbf{y}(k)}{\partial \widehat{\mathbf{w}}_1} & \frac{\partial \mathbf{y}(k)}{\partial \widehat{\mathbf{w}}_1^f} \end{bmatrix}_{M_2 \times np}$$

$$\frac{\partial \mathbf{y}(k)}{\partial \widehat{\mathbf{w}}} = \begin{bmatrix} \left[\frac{\partial \mathbf{g}_2(k)}{\partial \mathbf{h}_2} \, \widehat{\mathbf{x}}_2 \right] & \left[\boldsymbol{\lambda}_1(k) \, \frac{\partial \mathbf{g}_1(k)}{\partial \mathbf{h}_1} \, \widehat{\mathbf{x}}_1 \right] & \left[\boldsymbol{\lambda}_1(k) \, \frac{\partial \mathbf{g}_1(k)}{\partial \mathbf{h}_1} \, \widehat{\mathbf{x}}_1^f \right] \end{bmatrix}_{M_2 \times np} \tag{1.42}$$

$$\widehat{\mathbf{x}}_2 = \frac{\partial \mathbf{h}_2(k)}{\partial \widehat{\mathbf{w}}_2} \tag{1.43}$$

$$\begin{cases} \widehat{\mathbf{x}}_1 & = \frac{\partial \mathbf{h}_1(k)}{\partial \widehat{\mathbf{w}}_1} + \mathbf{w}_1^f(k)\, \frac{\partial \mathbf{y}_1^-(k-1)}{\partial \widehat{\mathbf{w}}_1} \\ \frac{\partial \mathbf{y}_1^-(0)}{\partial \widehat{\mathbf{w}}_1} & = 0_{M_1 \times np_1} \end{cases} \tag{1.44}$$

$$\begin{cases} \widehat{\mathbf{x}}_1^f & = \frac{\partial \mathbf{h}_1(k)}{\partial \widehat{\mathbf{w}}_1^f} + \mathbf{w}_1^f(k)\, \frac{\partial \mathbf{y}_1^-(k-1)}{\partial \widehat{\mathbf{w}}_1^f} \\ \frac{\partial \mathbf{y}_1^-(0)}{\partial \widehat{\mathbf{w}}_1^f} & = 0_{M_1 \times np_1^f} \end{cases} \tag{1.45}$$

A matriz de propagação $\boldsymbol{\lambda}_1(k)$ é descrita pela equação 1.17. Cada peso individual $w_{2_{ij}}$ da segunda camada da rede é atualizado por um $\Delta w_{2_{ij}}$ expresso por:

$$\Delta w_{2_{ij}} = \eta\, e_i(k)\, \frac{\partial g_{2_{ii}}(k)}{\partial h_{2_{ii}}}\, y_{1_j}(k) \tag{1.46}$$

A sensibilidade da i-ésima saída em relação a cada peso individual $w_{2_{ij}}$ é expresso por:

$$\frac{\partial y_i(k)}{\partial w_{2_{ij}}} = \frac{\partial g_{2_{ii}}(k)}{\partial h_{2_{ii}}}\, y_{1_j}(k) \tag{1.47}$$

Cada peso individual $w_{1_{jn}}$ da segunda camada da rede é atualizado por um $\Delta w_{1_{jn}}$ expresso por:

$$\Delta w_{1_{jn}} = \eta \left(\sum_{i=1}^{M_L} e_i(k)\, \lambda_{1_{ij}}(k) \right) \frac{\partial g_{1_{jj}}(k)}{\partial h_{1_{jj}}}\, \widehat{x}_{jm}(k) \tag{1.48}$$

onde $m = (M_1\, j + n)$.

A sensibilidade da i-ésima saída em relação a cada peso individual $w_{1_{jn}}$ é expresso por:

$$\frac{\partial y_i(k)}{\partial w_{1_{jn}}} = \lambda_{1_{ij}}(k)\, \frac{\partial g_{1_{jj}}(k)}{\partial h_{1_{jj}}}\, \widehat{x}_{jm}(k) \tag{1.49}$$

Cada peso individual w_{jj}^f da rede é atualizado por um Δw_{jj}^f expresso por:

$$\Delta w_{jj}^f = \eta \left(\sum_{i=1}^{M_L} e_i(k)\, \lambda_{1_{ij}}(k) \right) \frac{\partial g_{jj}(k)}{\partial h_{1_{jj}}}\, \widehat{x}_{jj}(k) \tag{1.50}$$

A sensibilidade da i-ésima saída em relação a cada peso individual w_{jj}^{f} é expresso por:

$$\frac{\partial y_i(k)}{\partial w_{1_{jn}}} = \lambda_{1_{ij}}(k)\,\frac{\partial g_{jj}(k)}{\partial h_{1_{jj}}}\,\widehat{x}_{ii}(k) \tag{1.51}$$

1.7 Sumário

Neste capítulo foi feito uma introdução às **RNA**'s com o desenvolvimento de modelos matemáticos vetoriais para a classe *perceptron*. Inicialmente o *perceptron* não recursivo foi apresentado e modelado, sendo depois apresentado o *perceptron* recursivo e feita uma extensão do modelo vetorial para o mesmo.

As leis de aprendizado desenvolvidas para os ajustes dos parâmetros das redes foram baseados no método de otmização pelo gradiente, com a inclusão da retro-propagação do erro para redes de múltiplas camadas. Neste ponto podemos destacar a matriz de propagação, $\boldsymbol{\lambda}_l$, equações 1.18 e 1.36, que propaga sinais da saída da rede para camadas intermediárias. Quando a matriz de propagação é obtida em relação à entrada da rede, $\boldsymbol{\lambda}_0$, a mesma corresponde ao jacobiano da saída da rede em relação as sua entrada.

Alguns desenvolvimentos feitos neste capítulo serão levados para capítulos posteriores. Podemos destacar os modelos vetoriais para as redes de múltiplas camadas, equações 1.14 e 1.41, as matrizes de propagação, equações 1.17 e 1.18, e as equações 1.8, 1.9, 1.20, 1.29, 1.30, 1.33 e 1.34, que seraão utilizadas no capítulo seguinte para a análise de estabilidade da lei de aprendizado.

Capítulo 2

Análise de Estabilidade da Lei de Aprendizado pelo Gradiente e Aplicação às RNA's a *Perceptrons*

Neste capítulo é feita a análise de estabilidade local do processo de aprendizado, baseado no segundo método de Lyapunov, e sua aplicação às classes *perceptrons* de **RNA's**. É obtido um critério para a avaliação da convergência local do aprendizado e elaborado um algoritmo adaptativo derivado do mesmo, delegando assim relativa confiabilidade a todo o processo.

2.1 Introdução

A *Lei de Aprendizado pelo Gradiente*, expressa pela equação 1.4, apresenta um fator de ponderação, η, chamado de *taxa de aprendizado*. Assim, há a necessidade de se avaliar os valores de η para os quais a lei de aprendizado é estável, garantindo assim a convergência do algoritmo.

Uma rede neural é um sistema de considerável complexidade pois, além de ser em geral não linear, pode conter uma grande quantidade de parâmetros organizados em camadas. Os primeiros trabalhos no sentido de se estabelecer regras para a obtenção dos limites do parâmetro η através de análise de estabilidade começaram na década de 90, e algumas das primeiras tentativas foram com Polycarpou e Ioannou [57, 58, 59], com base na Teoria de Estabilidade de Lyapunov. Também podemos citar os trabalhos de Rovithakis *et al* [62, 63], Kosmatopoulos *et al* [33], Lewis *et al* [38] e Ku e Lee [34]. Entretanto, esses trabalhos tratavam casos particulares ou justificavam apenas redes de única camada ou com

única saída.

Para alguns desenvolvimentos que faremos neste capítulo usaremos a norma *Euclidiana* para vetores, que é definida como $\| \mathbf{x} \|^2 := \sum_i x_i^2 = \mathbf{x}^T \mathbf{x}$, com $x_i \in \mathbb{R}$. Para matrizes adotaremos a norma *Frobenius*, definida como $\| \mathbf{A} \|^2 := \sum_{i,j} a_{i,j}^2 = tr(\mathbf{A}\,\mathbf{A}')$, onde $tr(\cdot)$ denota o *traço* de uma matriz. Ainda, faremos que $x_{\max} = \max \mid x_i \mid$, e $a_{\max} = \max \mid a_{ij} \mid$, com $x_i \in \mathcal{X}$, $\mathcal{X} \subset \mathbb{R}$, $a_{ij} \in \mathcal{A}$, e $\mathcal{A} \subset \mathbb{R}$, denotam o valor máximo, max, e $\mid \cdot \mid$ é o valor absoluto de x_i ou de a_{ij}.

A seguir, será apresentada uma análise de estabilidade local baseada na *Função Discreta de Lyapunov*, para a lei de aprendizado pelo gradiente descendente e a aplicaremos às **RNA**'s.

2.2 Análise de Estabilidade da Lei de Aprendizado pelo Gradiente

Inicialmente, definiremos um sistema qualquer, como mostrado na figura 2.1, regido por uma função vetorial $\phi(\cdot)$, possuindo um vetor $\mathbf{x}$, com $\mathbf{x} \in \mathbb{R}^{ne}$, como sinais externos ou de entrada, um vetor $\widehat{\mathbf{w}}$, com $\widehat{\mathbf{w}} \in \mathbb{R}^{np}$, de parâmetros internos que são ajustados por uma *Lei de Aprendizado* baseada no *gradiente*, de modo que a saída do sistema, um vetor $\mathbf{y}$, seja igual ou esteja próximo a um vetor $\mathbf{y}_d$, sendo admitido um erro, ou diferença entre o valor desejado e a saída do sistema, expresso na forma $\mathbf{e} = (\mathbf{y}_d - \mathbf{y})$, com $\mathbf{e}$, $\mathbf{y}_d$ e $\mathbf{y}$ pertencentes ao espaço $\mathbb{R}^{ns}$, e ne, np e ns pertencentes a $\mathbb{N}$. Assim temos que:

$$\mathbf{y}(k, \widehat{\mathbf{w}}, \mathbf{x}) = \phi(k, \widehat{\mathbf{w}}, \mathbf{x}) \tag{2.1}$$

Definindo-se uma função objetivo a ser minimizada para a saída desse sistema na forma:

$$J(k, \widehat{\mathbf{w}}, \mathbf{x}) = \frac{1}{2}\, \mathbf{e}'(k)\, \mathbf{e}(k) \tag{2.2}$$

e um vetor de diferenças de pesos para o sistema, sendo ele obtido através da lei de aprendizado pelo gradiente descendente, teremos:

$$\begin{aligned} \Delta \widehat{\mathbf{w}}(k) &= \eta \left(-\frac{\partial J(k)}{\partial \widehat{\mathbf{w}}} \right) \\ &= \eta \left[\frac{\partial \mathbf{y}(k)}{\partial \widehat{\mathbf{w}}} \right]' \mathbf{e}(k) \end{aligned} \tag{2.3}$$

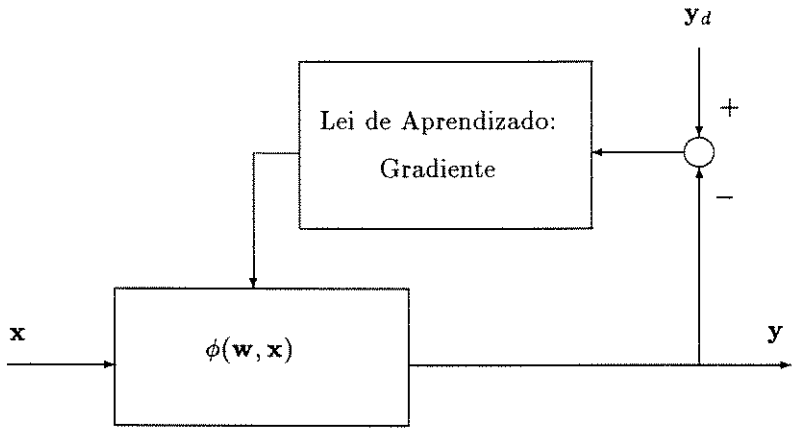

Figura 2.1: Sistema qualquer, regido por uma Lei de Aprendizado baseada no Gradiente.

onde η é o fator de ponderação, no nosso caso chamado de *taxa de aprendizado*, e ∇ é o operador gradiente aplicado à função objetivo na forma:

$$\begin{aligned} \nabla J(k, \widehat{\mathbf{w}}) &= \frac{\partial J(k, \widehat{\mathbf{w}})}{\partial \widehat{\mathbf{w}}} \\ &= \begin{bmatrix} \frac{\partial J(k,\widehat{\mathbf{w}})}{\partial \widehat{w}_1} \\ \frac{\partial J(k,\widehat{\mathbf{w}})}{\partial \widehat{w}_2} \\ \vdots \\ \frac{\partial J(k,\widehat{\mathbf{w}})}{\partial \widehat{w}_{np}} \end{bmatrix}_{np} \end{aligned} \tag{2.4}$$

Este vetor gradiente pode ser obtido a partir da equação 2.1 como:

$$\begin{aligned} \nabla J(k, \widehat{\mathbf{w}}) &= \left[\frac{\partial \mathbf{e}(k)}{\partial \widehat{\mathbf{w}}}\right]' \mathbf{e}(k) \\ &= -\left[\frac{\partial \mathbf{y}(k)}{\partial \widehat{\mathbf{w}}}\right]' \mathbf{e}(k) \end{aligned} \tag{2.5}$$

O termo $\partial \mathbf{y}(k)/\partial \widehat{\mathbf{w}} \in \mathbb{R}^{ns \times np}$ é a *Matriz Jacobiana* ou *Matriz de Sensibilidade*

da saída da rede em relação ao vetor de parâmetros ajustáveis $\widehat{\mathbf{w}}$, possuindo a forma:

$$\frac{\partial \mathbf{y}(k)}{\partial \widehat{\mathbf{w}}} = \begin{bmatrix} \frac{\partial y_1(k)}{\partial \widehat{w}_1} & \frac{\partial y_1(k)}{\partial \widehat{w}_2} & \cdots & \frac{\partial y_1(k)}{\partial \widehat{w}_{np}} \\ \frac{\partial y_2(k)}{\partial \widehat{w}_1} & \frac{\partial y_2(k)}{\partial \widehat{w}_2} & \cdots & \frac{\partial y_2(k)}{\partial \widehat{w}_{np}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial y_{ns}(k)}{\partial \widehat{w}_1} & \frac{\partial y_{ns}(k)}{\partial \widehat{w}_2} & \cdots & \frac{\partial y_{ns}(k)}{\partial \widehat{w}_{np}} \end{bmatrix}_{M \times np}$$

ou na forma de um vetor linha:

$$\frac{\partial \mathbf{y}(k)}{\partial \widehat{\mathbf{w}}} = \begin{bmatrix} \frac{\partial \mathbf{y}(k)}{\partial \widehat{w}_1} & \frac{\partial \mathbf{y}(k)}{\partial \widehat{w}_2} & \cdots & \frac{\partial \mathbf{y}(k)}{\partial \widehat{w}_{np}} \end{bmatrix}_{np}$$

A *Função de Lyapunov* aplicada à saída do sistema pode ser da forma:

$$\mathcal{V}(k) = \mathbf{e}'(k)\,\mathbf{e}(k) \tag{2.6}$$

A diferença na função de *Lyapunov* entre dois instantes de tempo devido ao processo de aprendizado é dado por:

$$\Delta\mathcal{V}(k) = \mathcal{V}(k+1) - \mathcal{V}(k) = \left(\mathbf{e}'(k+1)\,\mathbf{e}(k+1) - \mathbf{e}'(k)\,\mathbf{e}(k)\right) \tag{2.7}$$

Supondo o tempo decorrido entre os instantes de $(k+1)$ e (k) muito pequeno e constante, a mudança na função erro pode, atraés de uma aproximação de primeira ordem, ser representada por:

$$\begin{aligned} \Delta\mathbf{e}(k) &= \mathbf{e}(k+1) - \mathbf{e}(k) \\ &\cong \frac{\partial \mathbf{e}(k)}{\partial \widehat{\mathbf{w}}}\,\Delta\widehat{\mathbf{w}}(k) \end{aligned} \tag{2.8}$$

A expresão 2.8 é uma aproximação para a variável $\Delta\mathbf{e}(k)$. Assim, para os desenvolvimentos a seguir estaremos em busca de uma convergência para um *mínimo local*. Ainda, substituindo a equação 2.3 na expressão 2.8 temos:

$$\begin{aligned} \Delta\mathbf{e}(k) &\cong \frac{\partial \mathbf{e}(k)}{\partial \widehat{\mathbf{w}}}\,\eta \left[\frac{\partial \mathbf{y}(k)}{\partial \widehat{\mathbf{w}}}\right]' \mathbf{e}(k) \\ &\cong -\eta \frac{\partial \mathbf{y}(k)}{\partial \widehat{\mathbf{w}}} \left[\frac{\partial \mathbf{y}(k)}{\partial \widehat{\mathbf{w}}}\right]' \mathbf{e}(k) \end{aligned} \tag{2.9}$$

Fazendo-se $\mathbf{A} = \partial \mathbf{y}(k)/\partial \widehat{\mathbf{w}}$ na expressão 2.9 e substituindo-a na expressão 2.7 teremos que:

$$\begin{aligned}
\Delta \mathcal{V}(k) &= (\Delta \mathbf{e}(k) + \mathbf{e}(k))' (\Delta \mathbf{e}(k) + \mathbf{e}(k)) - \mathbf{e}'(k)\,\mathbf{e}(k) \\
&= \Delta \mathbf{e}'(k)\,\Delta \mathbf{e}(k) + \Delta \mathbf{e}'(k)\,\mathbf{e}(k) + \Delta \mathbf{e}'(k)\,\mathbf{e}(k) \\
&= \Delta \mathbf{e}'(k)\,\Delta \mathbf{e}(k) + 2\,\Delta \mathbf{e}'(k)\,\mathbf{e}(k) \\
&= \eta^2\,\mathbf{e}'(k)\,\mathbf{A}\,\mathbf{A}'\,\mathbf{A}\,\mathbf{A}'\,\mathbf{e}(k) + 2\,\eta\,\mathbf{e}'(k)\,\mathbf{A}\,\mathbf{A}'\,\mathbf{e}(k) \\
&= -\,\mathbf{e}'(k)\,\left\{2\,\eta\,\mathbf{A}\,\mathbf{A}' + \eta^2\,\mathbf{A}\,\mathbf{A}'\,\mathbf{A}\,\mathbf{A}'\right\}\,\mathbf{e}(k) \\
&= -\,\eta\,\mathbf{e}'(k)\,\mathbf{A}\,\mathbf{A}'\,\left\{2\,\mathbf{I}_M - \eta\,\mathbf{A}\,\mathbf{A}'\right\}\,\mathbf{e}(k) \qquad (2.10)
\end{aligned}$$

Com base no Segundo Método de Lyapunov para sistemas discretos o sistema é estável se $\Delta \mathcal{V} < 0$. Analisando a equação 2.10 podemos ver que isso impõe, para que o sistema seja estável, $\eta > 0$, satisfazendo o decréscimo do erro, e ainda que:

$$\begin{aligned}
2\,\mathbf{I}_M - \eta\,\mathbf{A}\,\mathbf{A}' &> 0 \\
2 - \eta\,\lambda_{\max}(\mathbf{A}\,\mathbf{A}') &> 0 \qquad (2.11)
\end{aligned}$$

Na expressão 2.11 $\lambda_{\max}$ corresponde ao maior autovalor da matriz $(\mathbf{A}\,\mathbf{A}')$ e $\mathbf{I}_M$ a uma matriz identidade com a dimensão M, número de saídas da rede. Logo, temos que:

$$\begin{aligned}
0 < \eta < \frac{2}{\lambda_{\max}(\mathbf{A}\,\mathbf{A}')} &= \frac{2}{\left\| \frac{\partial \mathbf{y}(k)}{\partial \widehat{\mathbf{w}}} \right\|^2} \\
0 < \eta < \frac{2}{\left\| \frac{\partial \mathbf{y}(k)}{\partial \widehat{\mathbf{w}}} \right\|^2} &\qquad (2.12) \\
0 < \eta < \frac{2}{\sum_{i=1}^{np} \left| \frac{\partial \mathbf{y}(k)}{\partial \widehat{w}_i} \right|^2} &\qquad (2.13)
\end{aligned}$$

Assim, para a estabilidade do algoritmo é necessário que a desigualdade da equação 2.12 seja satisfeita, sendo o parâmetro η admissível em um intervalo inversamente

proporcional à norma da matriz jacobiana ou de sensibilidade da saída do sistema em relação ao parâmetro interno $\widehat{\mathbf{w}}$. Ainda, a equação 2.13 informa que esse intervalo é inversamente proporcional ao somatório das normas da sensibilidade da saída do sistema em relação a cada componente individual do vetor $\widehat{\mathbf{w}}$.

Note que para chegarmos nessas conclusões, e para os desenvolvimentos feitos desde a equação 2.6 até a equação 2.13, não foi necessário conhecer ou definir a função $\phi(\cdot)$, apresentada na equação 2.1. Portanto, o critério expresso pela equação 2.12 pode ser aplicado a qualquer sistema regido pela lei de aprendizado do gradiente, inclusive às redes neurais a *perceptrons*. Ainda, para obtermos o intervalo de admissibilidade do parâmetro η necessitamos apenas conhecer a sensibilidade da saída do sistema em relação a cada um dos seus parâmetros ajustáveis.

As **RNA**'s, apresentadas no capítulo 1, com base nas estruturas *perceptrons recursivas* ou *não recursivas*, são sistemas regidos por uma função vetorial $\phi(\cdot)$, possuido um vetor $\mathbf{x}$, com $\mathbf{x} \in \mathbb{R}^{ne}$, e $ne = N + 1$, como sinais externos ou de entrada, um vetor $\widehat{\mathbf{w}}$, com $\widehat{\mathbf{w}} \in \mathbb{R}^{np}$, de parâmetros internos que são ajustados por uma *Lei de Aprendizado* baseada no *gradiente*, de modo que a saída do sistema, um vetor $\mathbf{y}$, seja igual ou esteja próximo a um vetor $\mathbf{y}_d$, sendo admitido um erro ou diferença entre o valor desejado e a saída do sistema expresso na forma $\mathbf{e} = (\mathbf{y}_d - \mathbf{y})$, com $\mathbf{e}$, $\mathbf{y}_d$ e $\mathbf{y}$ pertencentes ao espaço $\mathbb{R}^{ns}$, e ne, np e ns pertencentes $\mathbb{N}$. Assim temos que:

$$\mathbf{y}(k, \widehat{\mathbf{w}}, \mathbf{x}) = \phi(k, \widehat{\mathbf{w}}, \mathbf{x}) \tag{2.14}$$

Sem perda de generalidade, como exemplo, faremos aplicação do critério acima para os casos particulares das redes **FNN** e **DRNN** a partir das definições apresentadas no capítulo 1. Ambas as redes estarão restritas a duas camadas ($L = 2$), tendo a segunda camada M_2 neurônios, a primeira camada M_1 neurônios e com a **RNA** possuindo N entradas. As equações 1.14 e 1.41 descrevem o funcionamento das redes **FNN** e rede **DRNN** respectivamente.

Para a segunda camada das redes adotaremos uma função de ativação *linear*, definida na tabela 1.1, com coeficiente angular unitário ($\alpha = 1$), logo:

$$g_{2_i}(k) = h_{2_i}(k) \tag{2.15}$$

$$\frac{\partial g_{2_i}(k)}{\partial h_{2_i}} = 1 \tag{2.16}$$

Para a primeira camada adotaremos a função *sigmoid*, na forma:

$$g_{1_j}(k) \;=\; sigmoid(h_{1_j}(k)) - \frac{1}{2}$$

$$= \frac{1}{1+\exp(-h_{1_j}(k))} - \frac{1}{2} \tag{2.17}$$

$$\frac{\partial g_{1_{jj}}(k)}{\partial h_{1_{jj}}} = g_{1_j}(k)\,(1 - g_{1_j}(k)) \tag{2.18}$$

$$\sup \mid g_{1,j}(k) \mid \cong \frac{1}{2} \tag{2.19}$$

$$\sup \left| \frac{\partial g_{1,j}(k)}{\partial h_{1,j}} \right| = \frac{1}{4} \tag{2.20}$$

A escolha de uma função de ativação do tipo *sigmoid* para a primeira camada garante que os neurônios recursivos sejam sempre **BIBO** estáveis [16], para $\forall w^f_{1,jj} \in \mathbb{R}$, pois a função *sigmoid* mapea a sua entrada para o intervalo $(0;1) \subset \mathbb{R}$. No entanto, veremos mais adiante que, para a estabilidade do processo de aprendizado será necessário impor restrições a $w^f_{1,jj}$. Note que caso fosse escolhida como função de ativação para a camada recursiva com características diferentes das pertecentes à função sigmoid, a **BIBO** estabilidade não estaria garantida, sendo necessário então uma análise de estabilidade específica para esses neurônios (veja a seção 1.3.1).

Com a **BIBO** estabilidade dos neurônios da primeira camada, teremos também a da rede como um todo. Isto pois, os neurônios da segunda camada são lineares, e a somatória finita de valores finitos resulta em um valor finito.

Ainda, temos para o vetor de funções de ativação da primeira camada que $y_{1_j}(k) \in (-1/2;1/2) \subset \mathbb{R}$, e que $\max \mid y_{1_j}(k) \mid \cong 1/2$. Note que $\partial g_{1_j}(k)/\partial h_{1_j} \in (0;1/4) \subset \mathbb{R}$, e que $\max \mid \partial g_{1_j}(k)/\partial h_{1_j} \mid = 1/4$. Definiremos intervalos finitos, $\mathcal{X} = (-x_{\max}; x_{\max}) \subset \mathbb{R}$, com $x_n \in \mathcal{X}$ e $\max \mid x_n \mid = x_{n_{\max}}$, $\mathcal{W}_2 = (-w_{2_{\max}}; w_{2_{\max}}) \subset \mathbb{R}$, com $w_{2_{ij}} \in \mathcal{W}_2$ e $\max \mid w_{2_{ij}} \mid = w_{2_{\max}}$, $\mathcal{W}^f_1 = (0; w^f_{1_{\max}}) \subset \mathbb{R}$, com $w^f_{1_j} \in \mathcal{W}^f_1$ e $\max \mid w^f_{1_j} \mid = w^f_{1_{\max}}$.

Na análise de estabilidade do aprendizado das redes **FNN** e **DRNN** calcularemos a norma máxima de seus respectivos jacobianos. Assim, consideraremos que todos os elementos das matrizes e vetores assumem os valores máximos admissíveis.

2.3 Estabilidade para a Rede FNN

Passaremos agora a determinação do intervalo de admissividade do parâmetro η para a rede **FNN**. Pela secção 1.4 o vetor de parâmetros da rede **FNN** tem a forma:

$$\widehat{\mathbf{w}} = \begin{bmatrix} \widehat{\mathbf{w}}_2 \\ \widehat{\mathbf{w}}_1 \end{bmatrix}_{ns}$$

onde $\widehat{\mathbf{w}}_2 \in \mathbb{R}^{np_2}$, com $np_2 = (M_2(M_1+1))$, contempla os pesos da segunda camada, $\widehat{\mathbf{w}}_1 \in \mathbb{R}^{np_1}$, com $np_1 = (M_1(N+1))$, contemplando os pesos da primeira camada e $np = np_1 + np_2$.

A norma máxima do jacobiano da saída em relação aos pesos dessa rede pode ser expressa na forma:

$$\left\| \frac{\partial \mathbf{y}(k)}{\partial \widehat{\mathbf{w}}} \right\|^2_{\max} = \left\| \frac{\partial \mathbf{y}(k)}{\partial \widehat{\mathbf{w}}_2} \right\|^2_{\max} + \left\| \frac{\partial \mathbf{y}(k)}{\partial \widehat{\mathbf{w}}_1} \right\|^2_{\max}$$

Através das equações 1.20, 1.18 e 1.9, teremos:

$$\begin{aligned} \left\| \frac{\partial \mathbf{y}(k)}{\partial \widehat{\mathbf{w}}_2} \right\|^2_{\max} &= \left\| \boldsymbol{\lambda}_2(k) \, \frac{\partial \mathbf{g}_2(k)}{\partial \mathbf{h}_2} \, \frac{\partial \mathbf{h}_2(k)}{\partial \widehat{\mathbf{w}}_2} \right\|^2_{\max} \\ &\leq \sum_{i=1}^{M_2} \left\| \lambda_{2_{ii}}(k) \, \frac{\partial g_{2_{ii}}(k)}{\partial h_{2_{ii}}} \, \mathbf{y}_1(k) \right\|^2_{\max} \\ &\leq M_2 \, (M_1+1) \left| \frac{\partial g_{2_{ii}}}{\partial h_{2_{ii}}} \right|^2_{\max} \left| y_{1_j} \right|^2_{\max} \end{aligned} \tag{2.21}$$

$$\begin{aligned} \left\| \frac{\partial \mathbf{y}(k)}{\partial \widehat{\mathbf{w}}_1} \right\|^2_{\max} &= \left\| \boldsymbol{\lambda}_1(k) \, \frac{\partial \mathbf{g}_1(k)}{\partial \mathbf{h}_1} \, \frac{\partial \mathbf{h}_2(k)}{\partial \widehat{\mathbf{w}}_1} \right\|^2_{\max} \\ &\leq \sum_{i=1,j=1}^{M_2,M_1} \left\| \lambda_{1_{ij}}(k) \, \frac{\partial g_{1_{jj}}(k)}{\partial h_{1_{jj}}} \, \mathbf{x}(k) \right\|^2_{\max} \\ &\leq M_2 \, M_1 \, (N+1) \left| \lambda_{1_{ij}} \right|^2_{\max} \left| \frac{\partial g_{1_{jj}}}{\partial h_{1_{jj}}} \right|^2_{\max} |x_n|^2_{\max} \\ &\leq M_2 \, M_1 \, (N+1) \left| \frac{\partial g_{2_{ii}}}{\partial h_{2_{ii}}} \right|^2_{\max} \left| w_{2_{ij}} \right|^2_{\max} \left| \frac{\partial g_{1_{jj}}}{\partial h_{1_{jj}}} \right|^2_{\max} |x_n|^2_{\max} \end{aligned} \tag{2.22}$$

Logo, temos que:

$$\left\| \frac{\partial \mathbf{y}(k)}{\partial \widehat{\mathbf{w}}} \right\|^2_{\max} \leq M_2 (M_1 + 1) \left| \frac{\partial g_{2_{ii}}}{\partial h_{2_{ii}}} \right|^2_{\max} y^2_{1_{\max}}$$

$$+ M_2 M_1 (N + 1) \left| \frac{\partial g_{2_{ii}}}{\partial h_{2_{ii}}} \right|^2_{\max} w^2_{2_{\max}} \left| \frac{\partial g_{1_{jj}}(k)}{\partial h_{1_{jj}}} \right|^2_{\max} x^2_{n_{\max}} \quad (2.23)$$

No caso particular da nossa rede **FNN** teremos:

$$\left\| \frac{\partial \mathbf{y}(k)}{\partial \widehat{\mathbf{w}}} \right\|^2_{\max} \leq M_2 (M_1 + 1) \frac{1}{4} + M_2 M_1 (N + 1) w^2_{2_{\max}} x^2_{n_{\max}} \frac{1}{16}$$

$$\leq \frac{4 M_2 (M_1 + 1) + M_2 M_1 (N + 1) w^2_{2_{\max}} x^2_{\max}}{16} \quad (2.24)$$

Substituindo a expressão 2.24 na expressão 2.12 teremos para o intervalo de admissibilidade do parâmetro η:

$$0 < \eta < \frac{32}{4 M_2 (M_1 + 1) + M_2 M_1 (N + 1) w^2_{2_{\max}} x^2_{\max}} \quad (2.25)$$

2.4 Estabilidade para a Rede DRNN

Pela seção 1.6 o vetor de parâmetros da rede **DRNN** tem a forma:

$$\widehat{\mathbf{w}} = \begin{bmatrix} \widehat{\mathbf{w}}_2 \\ \widehat{\mathbf{w}}_1 \\ \widehat{\mathbf{w}}_1^f \end{bmatrix}_{np}$$

onde $\widehat{\mathbf{w}}_1 \in \mathbb{R}^{np_1}$, com $np_1 = (M_1(N + 1))$, contempla os pesos diretos da primeira camada, $\widehat{\mathbf{w}}_1^f \in \mathbb{R}^{np_1^f}$, com $np_1^f = M_1$, contemplando os pesos recursivos da primeira camada, $\widehat{\mathbf{w}}_2 \in \mathbb{R}^{np_2}$, com $np_2 = (M_2(M_1 + 1))$, contemplando os pesos diretos da segunda camada e $np = np_2 + np_1 + np_1^f$.

A norma máxima da saída em relação aos parâmetros ajustáveis dessa rede pode ser da forma:

$$\left\| \frac{\partial \mathbf{y}(k)}{\partial \widehat{\mathbf{w}}} \right\|^2_{\max} = \left\| \frac{\partial \mathbf{y}(k)}{\partial \widehat{\mathbf{w}}_2} \right\|^2_{\max} + \left\| \frac{\partial \mathbf{y}(k)}{\partial \widehat{\mathbf{w}}_1} \right\|^2_{\max} + \left\| \frac{\partial \mathbf{y}(k)}{\partial \widehat{\mathbf{w}}_1^f} \right\|^2_{\max}$$

Da seção anterior temos:

$$\left\| \frac{\partial \mathbf{y}(k)}{\partial \widehat{\mathbf{w}}_2} \right\|^2_{\max} \leq M_2 \left(M_1 + 1\right) \left| \frac{\partial g_{2_{ii}}}{\partial h_{2_{ii}}} \right|^2_{\max} y^2_{1_{\max}} \tag{2.26}$$

Com base na equação 1.42 temos ainda:

$$\left\| \frac{\partial \mathbf{y}(k)}{\partial \widehat{\mathbf{w}}_1} \right\|^2_{\max} = \left\| \boldsymbol{\lambda}_1(k) \frac{\partial \mathbf{g}_1(k)}{\partial \mathbf{h}_1} \widehat{\mathbf{x}}_1 \right\|^2_{\max} \tag{2.27}$$

$$\left\| \frac{\partial \mathbf{y}(k)}{\partial \widehat{\mathbf{w}}^f_1} \right\|^2_{\max} = \left\| \boldsymbol{\lambda}_1(k) \frac{\partial \mathbf{g}_1(k)}{\partial \mathbf{h}_1} \widehat{\mathbf{x}}^f_1 \right\|^2_{\max} \tag{2.28}$$

Analisando a convergência do termo $\widehat{\mathbf{x}}_1$, com base nas equações 1.33 e 1.32 teremos:

$$\begin{cases} \widehat{\mathbf{x}}_1 & = & \frac{\partial \mathbf{h}_1(k)}{\partial \widehat{\mathbf{w}}_1} + \mathbf{w}^f_1(k) \frac{\partial \mathbf{y}^-_1(k-1)}{\partial \widehat{\mathbf{w}}_1} \\ & = & \sum_{m=0}^{\infty} \left(\prod \left(\widehat{\mathbf{w}}^f_1(k-q+1) \frac{\partial \mathbf{g}_1(k-1)}{\partial \mathbf{h}_1} \right)^q \right) \frac{\partial \mathbf{h}_1(k-m)}{\partial \widehat{\mathbf{w}}_1} \\ \frac{\partial \mathbf{h}_1(0)}{\partial \widehat{\mathbf{w}}_1} & = & 0_{M_1 \times np_1} \\ \frac{\partial \mathbf{g}_1(0)}{\partial \mathbf{h}_1} & = & 0_{M_1} \end{cases} \tag{2.29}$$

Calculando-se o máximo de convergência de $\widehat{\mathbf{x}}_1$ a partir da equação 2.29 chegaremos a:

$$\widehat{\mathbf{x}}_{1_{\max}} = \left(\sum_{m=0}^{\infty} \left(\prod \left(\widehat{w}^f_{1_{\max}} \left| \frac{\partial g_{1_{jj}}(k)}{\partial h_{1_{jj}}} \right|_{\max} \right)^q \right) \left(\prod (\mathbf{I}_{M_1} \, \mathbf{I}_{M_1})^q \right) \right) \left[\frac{\partial \mathbf{h}_1}{\partial \widehat{\mathbf{w}}_1} \right]_{\max} \tag{2.30}$$

Observe que o produtório de matrizes sempre irá resultar na matriz identidade. A série irá convergir nas seguintes condições:

$$\prod \left(\widehat{w}^f_{1_{\max}} \left| \frac{\partial g_{1_{jj}}(k)}{\partial h_{1_{jj}}} \right|_{\max} \right)^q < 1$$

$$\widehat{w}^f_{1_{\max}} \left| \frac{\partial g_{1_{jj}}(k)}{\partial h_{1_{jj}}} \right|_{\max} < 1$$

Assim, podemos agora impor a seguinte restrição para os parâmetros w^f_1:

$$\widehat{w}^f_{1_{\max}} < \frac{1}{\left| \frac{\partial g_{1_{jj}}}{\partial h_{1_{jj}}} \right|_{\max}} \tag{2.31}$$

Faremos uma aproximação para a convergência da série da equação 2.30 na forma:

$$\widehat{\mathbf{x}}_{1_{\max}} \cong \gamma \left[\frac{\partial \mathbf{h}_1}{\partial \widehat{\mathbf{w}}_1}\right]_{\max} \tag{2.32}$$

onde:

$$\gamma = \frac{1}{1 - \widehat{w}^f_{1_{\max}} \left|\frac{\partial g_{1_{jj}}}{\partial h_{1_{jj}}}\right|_{\max}} \tag{2.33}$$

Calculando-se a norma da equação 2.27 teremos:

$$\begin{aligned}
\left\|\frac{\partial \mathbf{y}(k)}{\partial \widehat{\mathbf{w}}_1}\right\|^2_{\max} &= \left\|\boldsymbol{\lambda}_1(k)\,\frac{\partial \mathbf{g}_1(k)}{\partial \mathbf{h}_1}\,\widehat{\mathbf{x}}_1\right\|^2_{\max} \\
&\leq \sum_{i=1,j=1}^{M_2,M_1} \left\|\lambda_{1_{ij}}(k)\,\frac{\partial g_{1_{jj}}(k)}{\partial h_{1_{jj}}}\,\gamma\,\mathbf{x}(k)\right\|^2_{\max} \\
&\leq M_2\,M_1\,(N+1)\left|\lambda_{1_{ij}}\right|^2_{\max}\left|\frac{\partial g_{1_{jj}}}{\partial h_{1_{jj}}}\right|^2_{\max}\gamma^2\,|x_n|^2_{\max} \\
&\leq M_2\,M_1\,(N+1)\left|\frac{\partial g_{2_{ii}}}{\partial h_{2_{ii}}}\right|^2_{\max}\left|w_{2_{ij}}\right|^2_{\max}\left|\frac{\partial g_{1_{jj}}}{\partial h_{1_{jj}}}\right|^2_{\max}\gamma^2\,|x_n|^2_{\max}
\end{aligned} \tag{2.34}$$

Analisando a convergência do termo $\widehat{\mathbf{x}}^f_1$, com base nas equações 1.30 e 1.29 teremos:

$$\begin{cases}
\widehat{\mathbf{x}}^f_1 &= \frac{\partial \mathbf{h}_1(k)}{\partial \widehat{\mathbf{w}}^f_1} + \mathbf{w}^f_1(k)\,\frac{\partial \mathbf{y}^-_1(k-1)}{\partial \widehat{\mathbf{w}}^f_1} \\
&= \sum_{m=0}^{\infty}\left(\prod\left(\widehat{\mathbf{w}}^f_1(k-q+1)\,\frac{\partial \mathbf{g}_1(k-1)}{\partial \mathbf{h}_1}\right)^q\right)\,\frac{\partial \mathbf{h}_1(k-m)}{\partial \widehat{\mathbf{w}}^f_1} \\
\frac{\partial \mathbf{y}^-_1(0)}{\partial \widehat{\mathbf{w}}^f_1} &= 0_{M_1 \times np^f_1} \\
\frac{\partial \mathbf{g}^-_1(0)}{\partial \mathbf{h}^f_1} &= 0_{M_1}
\end{cases} \tag{2.35}$$

Calculando-se o máximo de convergência de $\widehat{\mathbf{x}}^f_1$ a partir da equação 2.35 chegaremos a:

$$\widehat{\mathbf{x}}_{1_{\max}} = \left(\sum_{m=0}^{\infty}\left(\prod\left(\widehat{w}^f_{1_{\max}}\left|\frac{\partial g_{1_{jj}}(k)}{\partial h_{1_{jj}}}\right|_{\max}\right)^q\right)\left(\prod\left(\mathbf{I}_{M_1}\,\mathbf{I}_{M_1}\right)^q\right)\right)\left[\frac{\partial \mathbf{h}_1}{\partial \widehat{\mathbf{w}}^f_1}\right]_{\max} \tag{2.36}$$

Faremos uma aproximação para a convergência da série da equação 2.36 na forma:

$$\widehat{\mathbf{x}}_{1_{\max}} \cong \gamma \left[\frac{\partial \mathbf{h}_1}{\partial \widehat{\mathbf{w}}_1^f}\right]_{\max} \tag{2.37}$$

onde γ esta definido na equação 2.33.

Calculando-se a norma da equação 2.27 teremos:

$$\begin{aligned}
\left\|\frac{\partial \mathbf{y}(k)}{\partial \widehat{\mathbf{w}}_1^f}\right\|_{\max}^2 &= \left\|\boldsymbol{\lambda}_1(k)\,\frac{\partial \mathbf{g}_1(k)}{\partial \mathbf{h}_1}\,\widehat{\mathbf{x}}_1^f\right\|_{\max}^2 \\
&\leq \sum_{i=1,j=1}^{M_2,M_1} \left\|\lambda_{1_{ij}}(k)\,\frac{\partial g_{1_{jj}}(k)}{\partial h_{1_{jj}}}\,\gamma\,\mathbf{y}_1^-(k-1)\right\|_{\max}^2 \\
&\leq M_2\,M_1\,(N+1)\,\left|\lambda_{1_{ij}}\right|_{\max}^2 \left|\frac{\partial g_{1_{jj}}}{\partial h_{1_{jj}}}\right|_{\max}^2 \gamma^2\,\left|y_{1_j}\right|_{\max}^2 \\
&\leq M_2\,M_1\,(N+1)\,\left|\frac{\partial g_{2_{ii}}}{\partial h_{2_{ii}}}\right|_{\max}^2 \left|w_{2_{ij}}\right|_{\max}^2 \left|\frac{\partial g_{1_{jj}}}{\partial h_{1_{jj}}}\right|_{\max}^2 \gamma^2\,\left|y_{1_j}\right|_{\max}^2
\end{aligned} \tag{2.38}$$

Logo, temos que:

$$\begin{aligned}
\left\|\frac{\partial \mathbf{y}(k)}{\partial \widehat{\mathbf{w}}}\right\|_{\max}^2 &\leq M_2\,(M_1+1)\,\left|\frac{\partial g_{2_{ii}}}{\partial h_{2_{ii}}}\right|_{\max}^2 y_{1_{\max}}^2 \\
&\quad + M_2\,M_1\,(N+1)\,\left|\frac{\partial g_{2_{ii}}}{\partial h_{2_{ii}}}\right|_{\max}^2 w_{2_{\max}}^2 \left|\frac{\partial g_{1_{jj}}(k)}{\partial h_{1_{jj}}}\right|_{\max}^2 \gamma^2\,x_{n_{\max}}^2 \\
&\quad + M_2\,M_1\,(N+1)\,\left|\frac{\partial g_{2_{ii}}}{\partial h_{2_{ii}}}\right|_{\max}^2 w_{2_{\max}}^2 \left|\frac{\partial g_{1_{jj}}(k)}{\partial h_{1_{jj}}}\right|_{\max}^2 \gamma^2\,y_{j_{\max}}^2
\end{aligned} \tag{2.39}$$

No caso particular da nossa rede **DRNN** teremos que $w_{1_{\max}}^f < 4$, e ainda:

$$\begin{aligned}
\left\|\frac{\partial \mathbf{y}(k)}{\partial \widehat{\mathbf{w}}}\right\|_{\max}^2 &\leq M_2\,(M_1+1)\,\frac{1}{4} + M_2\,M_1\,(N+1)\,w_{2_{\max}}^2\,x_{\max}^2\,\gamma_{\max}^2\,\frac{1}{16} \\
&\quad + M_2\,M_1\,w_{2_{\max}}^2\,\gamma_{\max}^2\,\frac{1}{64}
\end{aligned}$$

$$\leq \frac{16\,M_2\,(M_1+1)+M_2\,M_1\,w_{2_{\max}}^2\,\gamma_{\max}^2\,(4\,(N+1)\,x_{\max}^2+1)}{64} \quad (2.40)$$

Substituindo a expressão 2.40 na 2.12 teremos para o intervalo de admissibilidade do parâmetro η:

$$0 < \eta < \frac{128}{16\,M_2\,(M_1+1)+M_2\,M_1\,w_{2_{\max}}^2\,\gamma_{\max}^2\,(4\,(N+1)\,x_{\max}^2+1)} \quad (2.41)$$

2.5 O Aprendizado pelo Gradiente em sua forma Recursiva

O aprendizado pelo o aprendizado pelo gradiente aplicado até o momento, equação 2.3, é conhecido como *aprendizado pelo gradiente em forma não recursiva.* Existe também o chamado *aprendizado pelo gradiente em forma recursiva*, proposto por Jacobs [30], como mostrado abaixo [24]:

$$\begin{cases} \Delta\widehat{\mathbf{w}}(k+1) &= \left(-\,\widehat{\eta}\,\frac{\partial J(k)}{\partial\widehat{\mathbf{w}}}\right)+\alpha\,\Delta\widehat{\mathbf{w}}(k) \\ \Delta\widehat{\mathbf{w}}(0) &= 0_{np} \end{cases} \quad (2.42)$$

A equação 2.42 é chamada de *gradiente com recursividade de primeira ordem* e α é uma constante escalar chamada de *momento de velocidade.* O gradiente em sua forma não recursiva pode apresentar uma convergência oscilatória e a forma dinâmica possibilita o amortecimento dessas oscilações.

Para a equação 2.42 temos que para o instante de tempo k:

$$\Delta\widehat{\mathbf{w}}(k)=\left(-\,\widehat{\eta}\,\frac{\partial J(k-1)}{\partial\widehat{\mathbf{w}}}\right)+\alpha\,\Delta\widehat{\mathbf{w}}(k-1)$$

Para o instante de tempo $k-1$, temos que:

$$\Delta\widehat{\mathbf{w}}(k-1)=\left(-\,\widehat{\eta}\,\frac{\partial J(k-2)}{\partial\widehat{\mathbf{w}}}\right)+\alpha\,\Delta\widehat{\mathbf{w}}(k-2)$$

e assim sucessivamente para instantes de tempo $k-2, k-3, \cdots$.

Assim, podemos reescrevê-la como uma de série numérica na forma:

$$\begin{cases} \Delta\widehat{\mathbf{w}}(k+1) &= -\,\widehat{\eta}\,\sum_{m=0}^{\infty}\left(\alpha^m\,\frac{\partial J(k-m)}{\partial\widehat{\mathbf{w}}}\right) \\ \Delta\widehat{\mathbf{w}}(0) &= 0_{np} \end{cases} \quad (2.43)$$

Esta série numérica convergirá para um valor finito, caso o sistema regido pela lei de aprendizado do gradiente descendente seja **BIBO** estável, e se $\mid \alpha \mid < 1$. Calculando-se o *limite* para a equação 2.43 teremos que:

$$\lim_{m\to\infty}\left(\lim_{z\to 1}\Delta\widehat{\mathbf{w}}(k+1)\right) = -\, f_c\,\widehat{\eta}\,\frac{\partial J(k)}{\partial\widehat{\mathbf{w}}}$$

onde:

$$f_c = \frac{1}{1-\alpha} \tag{2.44}$$

Para evitar uma convergência oscilatória, adotaremos neste caso apenas valores positivos para α. O valor finito f_c, será chamado de *fator de correção* . Assim, teremos a equação 2.43 reescrita na forma:

$$\Delta\widehat{\mathbf{w}}(k+1) = -\, f_c\,\widehat{\eta}\,\frac{\partial J(k)}{\partial\widehat{\mathbf{w}}} \tag{2.45}$$

Substituindo a equação 2.3 pela equação 2.45 e refazendo-se os desenvolvimentos que se seguiram à equação 2.3 teremos que:

$$0 < \widehat{\eta} < \frac{2}{f_c\left\|\frac{\partial\mathbf{y}(k)}{\partial\widehat{\mathbf{w}}}\right\|^2} \tag{2.46}$$

Assim, vemos que na utilização do aprendizado com o gradiente na forma recursiva é necessário se fazer uma correção para o intervalo de admissividade do parâmetro $\widehat{\eta}$ na forma apresentada na equação 2.46.

Ainda, podemos fazer na equação 2.45 a seguinte modificação:

$$\eta = f_c\,\widehat{\eta} \tag{2.47}$$

e a mesma se tornaria equivalente ao aprendizado com gradiente na forma não recursiva, permitindo com isso que as análises de estabilidade e convergência feitas para o caso estático, também valham para o caso recursivo.

2.6 Um Algoritmo Adaptativo para o Aprendizado em RNA's a *Perceptrons*

Analisando-se as seções 2.3 e 2.4 observa-se que os intervalos de admissibilidade das taxas de aprendizado dependem dos valores de $\lambda_{l_{ij}}(k)$, que no caso específico (veja as

equações 2.25 e 2.41) dependem dos valores de $x_{\max}$, $w_{2_{\max}}$ e $w^f_{1_{\max}}$, os quais não temos a priori nenhuma estimativa, pois os parâmetros ajustáveis, no caso os pesos $w_{2_{ij}}$ e $w^f_{1_{jj}}$, se alteram durante o processo de treinamento da rede e $x_{\max}$ devido a algum condicionamento imprevisto das amostras apresentadas ao sistema para o aprendizado. Assim há a necessidade de uma escolha apropriada das taxas de aprendizado de maneira tal que se estabeleçam valores aceitáveis para $x_{\max}$, $w_{2_{\max}}$ e $w^f_{1_{\max}}$, com a finalidade de se fazer com que o processo não passe a divergir inesperadamente.

Na literatura técnica é possível encontrar várias referências de algoritmos adaptativos para aprendizado em **RNA's**. Park *et al* [53] e Bello [11] propuseram algoritmos adaptativos baseados em técnicas de programação não linear. Fukume e Omatu [22] procuram conter a divergência da rede através da adoção de funções de ativação não diferenciáveis para os neurônios. Brent [13] propõe um algoritmo de convergência rápida através de montagens de árvores de decisão. Wang e Chen [69] propuseram um algoritmo de treinamento para o ajuste das camadas em separado. Ku e Lee [34] propuseram um algoritmo adaptativo, com base em análise de estabilidade via função de Lyapunov, válida apenas para redes de única saída e ajuste dos pesos de um neurônio a cada passo de interação.

Apresentaremos nesta seção um algoritmo adaptativo antecipativo à divergências para o aprendizado em **RNA's** a *perceptrons*, e sem perda de generalidade, faremos aplicações às redes **FNN** e **DRNN**, para o qual, utilizaremos as equações 2.25 e 2.41 nos seus limites máximos. Isso possibilita uma convergência mais rápida e, além disso, robustez às necessidades de variações dos parâmetros $x_{\max}$, $w_{2_{\max}}$ e $w^f_{1_{\max}}$.

Como verificado acima, a integridade do processo de aprendizado pode ser violada pelos fatores $x_{\max}$, $\lambda_{l_{ij}}$, e por consequência por $w_{2_{\max}}$ e $w^f_{1_{\max}}$. O algoritmo proposto consiste basicamente em se testar a integridade do processo de aprendizado antes de sua execução. No início do processo de aprendizado são escolhidos valores pequenos para $x_{\max}$, $w_{2_{\max}}$ e $w^f_{1_{\max}}$, os quais chamaremos de $x^0_{\max}$, $w^0_{2_{\max}}$ e $w^{f^0}_{1_{\max}}$. Com esses valores é especificada a taxa de aprendizado inicial, a qual chamaremos de η^0, através das equações 2.25 ou 2.41. A partir deste ponto, a cada par $(\mathbf{x}, \mathbf{y}_d)$ apresentado para treinamento, é verificado antes do processamento se $x_{\max} < x^0_{\max}$, se $w_{2_{\max}} < w^0_{2_{\max}}$ e se $w^f_{1_{\max}} < w^{f^0}_{1_{\max}}$. Caso alguma dessas condições seja *falsa*, as taxas de aprendizado são recalculadas, com os valores $x^0_{\max}$ ou $w^0_{2_{\max}}$ ou $w^{f^0}_{1_{\max}}$ sendo corrigidos por fazê-los iguais aos valores de $x_{\max}$ ou $w_{2_{\max}}$ ou $w^f_{1_{\max}}$, multiplicados por uma constante $\kappa \in (1; +\infty) \subset \mathbb{R}$.

Para valores $x^0_{\max}$, $w^0_{2_{\max}}$ e $w^{f^0}_{1_{\max}}$ previamente escolhidos para o início do proceso de aprendizado, o algoritmo adaptativo pode ser implementado em um **procedimento** *learn*. Abaixo apresentamos a implementação desse algoritmo para os casos particulares das redes **FNN** e **DRNN** em **PDL**, em *Program Design Language* (**PDL**), que é uma liguagem

para projeto de software em que se faz uma descrição do sistema em forma de narrativa idiomática com inclusão de sentenças lógicas e matemáticas. Na parte experimental deste trabalho as redes **FNN** e **DRNN** foram implementas com metodologia **OOD**, *Object Oriented Design*, que a possibilita a criação de estruturas de software com abstração de dados e de funções e com o desacoplamento a detalhes representacionais [60].

Em nossa implementação a abstração e o desacoplamento chegam ao ponto que para o *usuário* utilizá-lo, basta a definição das dimensões da rede, sem preocupação com parâmetros de aprendizados para os treinamentos. Um sistema *orientado ao usuário.*

Algoritmo 2.1 *Estrutura básica do Algoritmo Adaptativo Antecipativo de Divergências em* **RNA***'s a perceptrons.*

procedure $learn(\mathbf{x} \in \mathbb{R}^{ne}, \mathbf{y}_d \in \mathbb{R}^{ns})$
begin
  def var: $x_{\max} \in \mathbb{R}$;
    $w_{2_{\max}} \in \mathbb{R}$;
    $w_{2_{\max}} \in \mathbb{R}$;
  $x_{\max} := get\_max(\mathbf{x})$;
  $w_{2_{\max}} := get\_max(\mathbf{w}_2)$;
  $w^f_{1_{\max}} := get\_max(\mathbf{w}^f_1)$;
  if $(x_{\max} \geq x^0_{\max})$ **or** $(w_{2_{\max}} \geq w^0_{2_{\max}})$ **or** $(w^f_{1_{\max}} \geq w^{f^0}_{1_{\max}})$ **then**
    if $(x_{\max} \geq x^0_{\max})$ **then**
      $x^0_{\max} := \kappa \times x_{n_{\max}}$ **endif**
    if $(w_{2_{\max}} \geq w^0_{2_{\max}})$ **then**
      $w^0_{2_{\max}} := \kappa \times w_{2_{\max}}$ **endif**
    if $(w^f_{1_{\max}} \geq w^{f^0}_{1_{\max}})$ **then**
      $w^{f^0}_{1_{\max}} := \kappa \times w^f_{1_{\max}}$ **endif**
    $get\_learn\_rate(x^0_{\max}, w^0_{2_{\max}}, w^{f^0}_{1_{\max}})$;
  endif
  $apply\_gradient\_descent(\cdot)$;
end

2.7 Sumário

Neste capítulo foi feita a análise de estabilidade local da lei de aprendizado pelo gradiente, baseado no segundo método de Lyapunov, e feitas aplicações às **RNA**'s

perceptrons de múltiplas camadas apresentadas no capítulo 1. Foi obtido um critério para a avaliação da convergência do aprendizado com base na norma do jacobiano da saída da rede em relação a seus parâmetro ajustáveis. Foram obtidas expressões para os intervalos de admissibilidade das taxas de aprendizado das redes **FNN** e **DRNN**, equações 2.25 e 2.41. Foi apresentado também a lei de aprendizado pelo gradiente em sua forma recursiva e feita a extensão da análise de estabilidade para este caso, equação 2.46. Finalmente foi apresentado um algoritmo adaptativo para o aprendizado em **RNA**'s a *perceptrons* baseado nos desenvolvimentos feitos no capítulo.

As equações 2.7 e 2.8 serão utilizadas no capítulo seguinte para a análise de estabilidade local do aprendizado em sistemas de controle. O algoritmo adaptativo para o aprendizado em **RNA**'s a *perceptrons* será utilizado na parte experimental deste trabalho.

Capítulo 3

Redes Neurais em Sistemas de Controle

Neste capítulo será mostrado como o paradigma das redes neurais pode ser aplicado em sistemas de controle. Será apresentada inicialmente uma visão geral de sistemas de controle, sendo depois feita as aplicações de redes neurais na identificação e em estruturas de controle. Também é apresentada a análise de estabilidade para o aprendizado em sistemas de controle a **RNA's** e feita uma extensão do algoritmo adaptativo para o caso presente.

3.1 Introdução

Há bastante tempo a comunidade científica da área de controle e automação desenvolve métodos e algoritmos baseados em teoria matemática de sistemas para a modelagem e controle de sistemas físicos. Muitos dos métodos desenvolvidos procuram tratar os sistemas físicos como lineares, utilizando propriedades de estabilidade para um sistema linear invariante no tempo [16]. Quando as características do sistema não permitem essas abordagens, em geral procura-se tratá-lo com modelagem e técnicas de controle não lineares. No entanto, é muito difícil encontrar-se um procedimento baseado no paradigma não linear que satisfaça simultaneamente a requisitos de estabilidade, robustez e boa resposta dinâmica para algum caso particular [5, 48].

Nas áreas de controle e identificação adaptativos, os sistemas físicos são tratados como possuindo parâmetros desconhecidos. As técnicas fundamentam-se em geral na escolha de estruturas de identificação e controle baseadas em resultados bem estabelecidos desde a teoria de sistemas lineares [5, 9, 40, 46]. As técnicas adaptativas possuem a vantagem de não exigirem, a priore, o conhecimento do modelo dinâmico do sistema físico que se está trabalhando. Entretanto, tais técnicas exigem grande esforço computacional

para identificação paramétrica do sistema físico, sendo muito sensível à precisão numérica e ruído, fazendo com que parâmetros ótimos de controle sejam de difícil obtenção [27, 31].

Em anos recentes algumas pesquisas têm sido desenvolvidas em aplicações de redes neurais a sistemas de controle e automação [25, 48]. Isso têm possibilitado novas técnicas que se inspiram na obtenção de melhores e mais eficientes ações de controle através do paradigma das redes neurais. Essas novas técnicas tentam aliar os benefícios das técnicas de controle ótimo e controle adaptativo.

3.2 Visão Geral de um Sistema de Controle

De uma forma geral, o problema de controle de um processo dinâmico pode ser esboçado como na figura 3.1, sendo este esquema chamado de sistema de controle com realimentação de variáveis de saída.

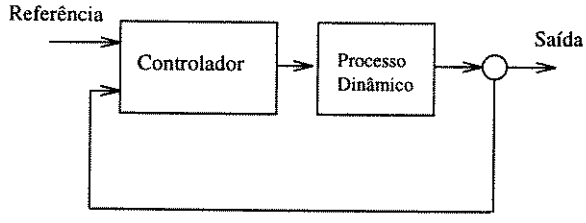

Figura 3.1: Sistema de controle genérico

Um *processo dinâmico* ou *sistema dinâmico* é um conjunto de elementos coordenados entre si , e que formam uma estrutura organizada como se fosse um *"todo unificado"*, e pode ser visto com mais detalhes na figura 3.2. Ele é composto de:

Sistema de Transformação: Responsável por fazer operações de transformações quaisquer. Possui como entradas os sinais de saída dos acionadores e variáveis externas, de perturbação, que podem ser classificadas em mensuráveis e não mensuráveis. Os efeitos das transformações são refletidos nas chamadas variáveis de saída. Os sistemas de transformações apresentam comportamento não lineares e algumas vezes, apesar de terem os seus modelos matemáticos conhecidos teoricamente, esses modelos apresentam parâmetros com valores desconhecidos ou pouco confiáveis e/ou variantes. Porém, muitas vezes, é possível fazer-se linearizações e utilizar o modelo em questão para o projeto das estratégias de controle do sistema.

Sensores: Responsáveis pela medição das variáveis de saída bem como as externas e informá-las ao controlador. Em sistemas físicos reais os sensores apresentam um

ligeiro atraso na medição, não linearidades e sujeição a ruídos, mesmo nos casos de sensores projetados com base em tecnologias avançadas. No entanto, em geral é possível desprezar-se esses efeitos e considerar os sinais apresentados pelos sensores ao controlador em um dado instante de tempo como sendo os valores reais das variáveis de saída.

Acionadores: Responsáveis pela tarefa de receber o sinal de saída do controlador e prover potência necessária à operação do sistema de transformação. Esta tarefa também pode ser feita pela modificação ou atuação em variáveis de entrada do processo dinâmico. Assim como os sensores, os acionadores possuem problemas de atraso de tempo, não linearidades e sujeição a ruídos e distorções de sinais de baixa potência, gerados pelo controlador.

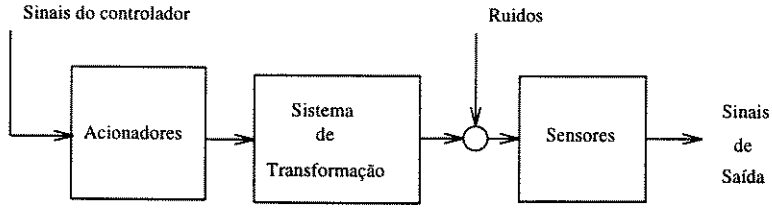

Figura 3.2: Processo Dinâmico

Para efeito de projeto de um sistema de controle, são considerados em conjunto os modelos matemáticos dos sistemas de transformação, dos sensores e dos acionadores. Assim, obtém-se um único modelo que facilita o estudo do processo dinâmico.

O *controlador* é o elemento responsável em fazer com que as variáveis de saída da planta tendam ou rastreem os valores desejados para as mesmas (*rastreamento assintótico*), atendendo também à especificações de desempenho que estejam pré-estabelecidas.

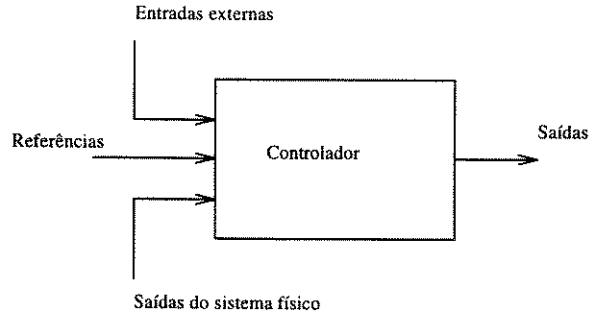

Figura 3.3: Controlador genérico

Na figura 3.3 temos uma visão mais detalhada do controlador. Com base nela e

na figura 3.2 passaremos a adotar as seguintes definições:

- $\mathbf{y}(k) \in \mathbb{R}^m$ é o vetor de sinal de saída da planta com componenentes y_i;
- $\mathbf{r}(k) \in \mathbb{R}^m$ é o vetor de sinal de referência para a saída da planta com componentes r_i;
- $\mathbf{x}(k) = [\mathbf{r}'(k) \;\; \mathbf{y}'(k)] \in \mathbb{R}^{2m}$, é o vetor de entrada do controlador com componentes x_i;
- $\mathbf{u}(k) \in \mathbb{R}^n$, é o vetor de saída do controlador com componentes u_i;
- $\mathbf{v}(k) \in \mathbb{R}^l$, é o vetor de entrada da planta com componentes v_i.

Em geral todos controladores possuem como entradas os sinais de saída da planta, $\mathbf{y}(k)$, e os sinais de referência para as variáveis de saída, $\mathbf{r}(k)$. No entanto, alguns algoritmos de controle possuem entradas externas para a conexão de variáveis que não sejam as saídas e as referências, como por exemplo, variáveis de perturbação (veja, por exemplo, em Newell[49] o algoritmo *feedforward*). Também, na maioria das situações as dimensões dos vetores $\mathbf{u}(k)$ e $\mathbf{v}(k)$ são iguais, e mais, muitas vezes $\mathbf{u}(k) = \mathbf{v}(k)$. Porém, podem existir no processo variáveis de entrada que não sejam sinais de controle, como é o caso das variáveis de perturbação, ou então é necessário, para se aumentar a potência do sinal, que se altere a relação entre $\mathbf{u}$ e $\mathbf{v}$, fazendo-se $\mathbf{v}(k) = \boldsymbol{\alpha}\,\mathbf{u}(k)$, para $\boldsymbol{\alpha} = diag[\alpha_{11}, \alpha_{22}, \cdots, \alpha_{mm}]$, e $\alpha_{ii} \in \mathbb{R}^+$.

Para a formulação discreta do modelo do sistema de controle temos que a cada instante k as saídas do controlador são funções de suas entradas. Assim, definiremos uma função $\mathcal{F}(\cdot)$ para as variáveis de saída do controlador como função das variáveis de entrada na forma:

$$\mathbf{u}(k+1) = \begin{cases} \mathbf{u}_0 & \text{se } k = 0 \\ \mathcal{F}(\mathbf{x}(k)) & \text{se } k > 0 \end{cases} \tag{3.1}$$

A diferença na saída do controlador entre dois instantes de tempo discretos $(k+1)$ e (k) é dado por:

$$\Delta\mathbf{u}(k) = \begin{cases} 0 & \text{se } k = 0 \\ \mathbf{u}(k+1) - \mathbf{u}(k) & \text{se } k > 0 \end{cases} \tag{3.2}$$

Substituindo a equação 3.1 na equação 3.2 teremos:

$$\mathbf{u}(k+1) = \begin{cases} \mathbf{u}_0 & \text{se } k = 0 \\ \mathcal{F}(\mathbf{x}(k)) - \mathcal{F}(\mathbf{x}(k-1)) + \mathbf{u}(k) & \text{se } k > 0 \end{cases} \tag{3.3}$$

Assim, além de representarmos a saída do controlador pela equação 3.1, podemos representá-la também pela equação 3.3, a qual é chamada de equação incremental do controlador, sendo bastante útil quando o controlador trabalha com as diferenças entre os sinais de referência e os sinais de saída do processo.

Existem diversos paradigmas para obtenção de $\mathcal{F}(\cdot)$. No entanto, abordagens neste sentido fogem ao propósito deste trabalho, onde destacaremos apenas o paradigma baseado em redes neurais.

A cada instante as saídas da planta são funções de suas entradas e da sua natureza dinâmica. Assim, definiremos uma função $\mathcal{G}(\cdot)$, dinâmica e não linear, para as variáveis de saída da planta em função das variáveis de entrada na forma:

$$\mathbf{y}(k) = \mathcal{G}(\mathbf{y}(k-1), \cdots, \mathbf{y}(k-p), \mathbf{v}(k), \cdots, \mathbf{v}(k-q+1)) \tag{3.4}$$

onde p e $q \in \mathbb{N}$, e p é a ordem do sistema.

Existem basicamente duas formas distintas de obtenção de $\mathcal{G}(\cdot)$, por *modelagem fenomenológica* ou por *identificação*. A modelagem fenomenológica é feita a partir das leis fundamentais que regem a dinâmica dos processos, observados em montagens experimentais e a identificação é feita a partir de resultados de testes experimentais de entrada e saída, considerando-se apenas variáveis de importância do sistema.

Em geral, adiciona-se à equação 3.4 um termo referente aos ruídos que influenciam o desempenho dinâmico da planta. Define-se uma função $\mathcal{N}(a(k))$, onde $a(k)$ é conhecido como ruído branco e $\mathcal{N}(\cdot)$ a influência do ruído branco no sistema [12].

$$\mathbf{y}(k) = \mathcal{G}(\mathbf{y}(k-1), \cdots, \mathbf{y}(k-p), \mathbf{v}(k), \cdots, \mathbf{v}(k-q+1)) + \mathcal{N}(a(k)) \tag{3.5}$$

3.3 Redes Neurais em Identificação

A identificação consiste da obtenção de uma representação matemática para o processo, equação 3.5, sem fazer uso de leis fudamentais que regem o seu comportamento dinâmico. Para a identificação são usados pares de dados de entrada e saída, $(\mathbf{v}, \mathbf{y})$, e um algoritmo para ajuste de um modelo matemático previamente especificado. A figura 3.4 ilustra o procedimento, onde $\widehat{\mathbf{y}}$ é o vetor de saída do modelo identificado.

Existem, na literatura técnica, numerosos tipos de modelos para a identificação, bem como algoritmos para ajuste de parâmetros [12, 39, 40]. No entanto abordagens neste sentido fogem aos objetivos deste trabalho, onde destacaremos apenas a técnica baseada

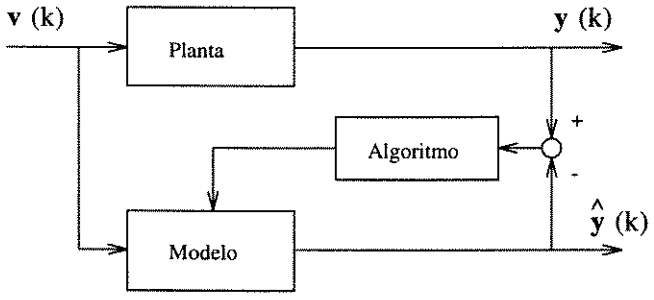

Figura 3.4: Procedimento de identificação.

em redes neurais. Assumiremos, inicialmente, que $\mathcal{N}(\cdot)$ é igual a zero e assim buscaremos apenas um modelo para $\mathcal{G}(\cdot)$ na equação 3.5.

Na identificação por redes neurais o modelo dinâmico do processo pode ser obtido na forma (Veja Narendra[48] e Hunt *et al* [25]):

$$\hat{\mathbf{y}} = \mathcal{G}(\mathbf{y}(k-1), \cdots, \mathbf{y}(k-p), \mathbf{v}(k), \cdots, \mathbf{v}(k-q+1)) \tag{3.6}$$

onde $\mathcal{G}(\cdot)$ é uma rede neural, com $\| \mathbf{y} - \hat{\mathbf{y}} \| \leq \varepsilon$, sendo ε um erro de modelagem aceitável. O vetor de entrada do identificador possui dimensão $(p \times m) + (q \times l)$. Na figura 3.5 temos a representação de um neuro-identificador para um processo de primeira ordem.

Com base na figura 3.4, o valor real para saída do processo é $\mathbf{y}(k)$, sendo $\hat{\mathbf{y}}(k)$ o valor da saída do modelo identificado. Assim, podemos definir uma função de custo para o erro entre as saída do processo e do modelo identificado na forma:

$$J(\mathbf{w}, (k)) = \frac{1}{2} \left[(\mathbf{y}(k) - \hat{\mathbf{y}}(k) \right]' \left[(\mathbf{y}(k) - \hat{\mathbf{y}}(k) \right] \tag{3.7}$$

Como $\hat{\mathbf{y}}(k)$ é igual a saída da rede neural que está servindo para identificar o modelo da planta, podemos comparar diretamente as equações 3.7 e 1.2 e concluir que as mesmas são equivalentes. Assim os desenvolvimentos feitos a partir da equação 1.2, nos capítulos 1 e 2, também são válidos para a equação 3.7 e portanto o algoritmo de aprendizado pelo gradiente pode ser utilizado na identifição sem maiores alterações.

Com o ajuste do modelo para $\mathcal{G}(\cdot)$, a diferença entre $\mathbf{y}(k)$ e $\hat{\mathbf{y}}(k)$ pode ser atribuída a ruídos presentes na planta e erros na definição da rede adotada. No entanto, não se pode fazer uma modelagem direta para $\mathcal{N}(\cdot)$ com base em redes neurais visto não se saber valores para $a(k)$. Assim, se houver necessidade de identificação de um modelo para $\mathcal{N}(\cdot)$, far-se-à então o uso de métodos outros, como por exemplo, os modelos *ARMAX*, *ARAMA*, *ARARX* e *IV*, descritos por Ljung[39, 40].

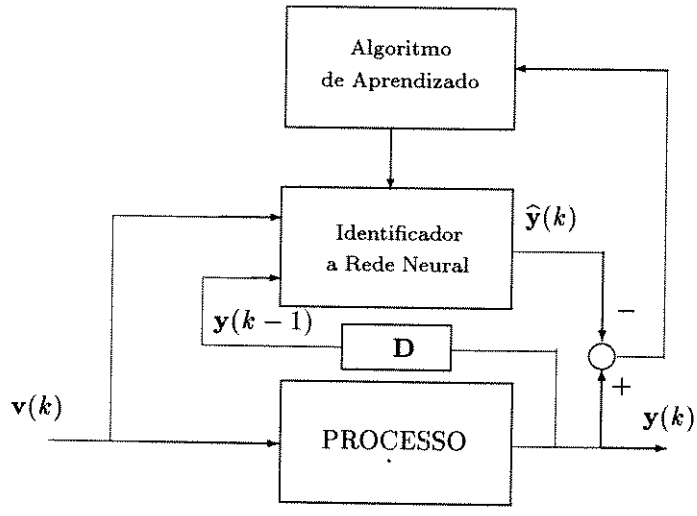

Figura 3.5: Modelo para identificação com **RNA's.**

3.4 Redes Neurais em Controladores

Narendra[48] sugere, e adotaremos, a configuração de controlador a redes neurais vista na figura 3.6. As especificações de desempenho para o sistema como um *todo unificado* são alocadas no *modelo referência.* A equação 3.1 aplicada ao modelo da figura 3.6 passa a ter a forma apresentada pela equação 3.8, onde a função $\mathcal{F}(\cdot)$ é a representação de uma rede neural.

$$\mathbf{u}(k+1) = \begin{cases} \mathbf{u}_0 & \text{se } k = 0 \\ \mathcal{F}(\mathbf{r}(k), \mathbf{y}(k), \mathbf{u}(k-1)) & \text{se } k > 0 \end{cases} \tag{3.8}$$

Assumiremos que o controlador atua em função do erro $\mathbf{e}$ entre $\mathbf{y}_r$, valor esperado para a saída da planta, e $\mathbf{y}$, valor real da saída da planta, como mostrado na figura 3.7. Assim equação 3.8 pode ter a forma simplificada e descrita em forma incremental pela equação 3.9.

$$\mathbf{u}(k+1) = \begin{cases} \mathbf{u}_0 & \text{se } k = 0 \\ \mathcal{F}(\mathbf{e}(k), \mathbf{u}(k-1)) - \mathcal{F}(\mathbf{e}(k-1), \mathbf{u}(k-2)) + \mathbf{u}(k) & \text{se } k > 0 \end{cases} \tag{3.9}$$

Com base na figura 3.7, o valor esperado para a saída da planta é $\mathbf{y}_r$, sendo $\mathbf{y}$ o sinal na saída. Assim, podemos definir uma função de custo para o erro na saída da planta

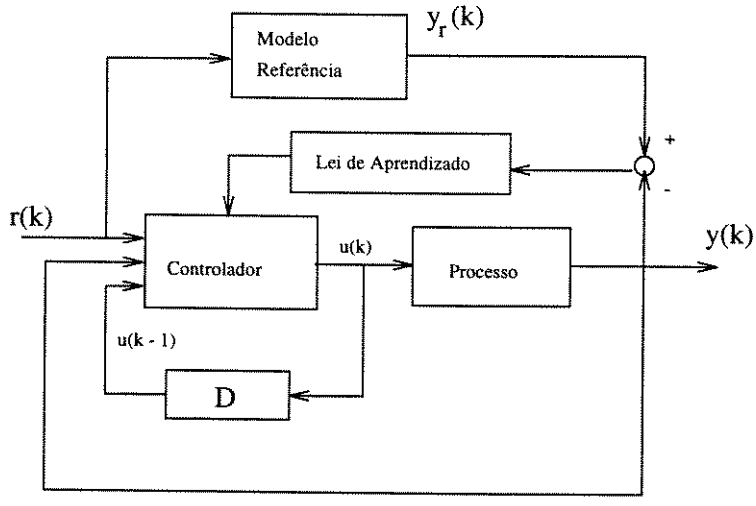

Figura 3.6: Modelo para controlador com **RNA's**.

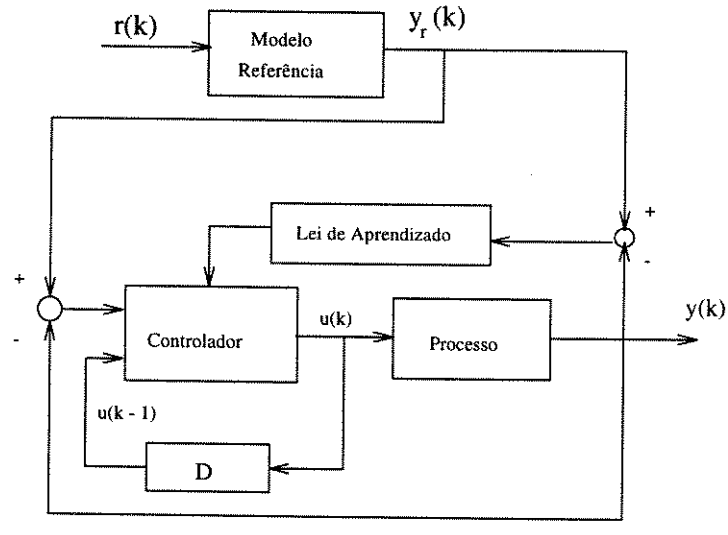

Figura 3.7: Modelo para controlador com **RNA's** baseado no erro.

na forma:

$$J(\mathbf{w},(k)) = \frac{1}{2}\,\mathbf{e}(k)'\,\mathbf{e}(k) \tag{3.10}$$

Em contraste com a aplicação em identificação, aqui não podemos comparar diretamente equação 3.10 com equação 1.2 visto que $\mathbf{y}$ não é a saída direta da rede neural. A saída do controlador serve de entrada para a planta, que por sua vez possui uma função $\mathcal{G}(\cdot)$ (veja a equação 3.5). Assim, há a necessidade de calcularmos o gradiente de equação 3.10 para a sua utilização no algoritmo de aprendizado. Faremos aqui uma substituição de variáveis para a saída de uma rede neural. A saída da rede passa a ser referenciada por um vetor $\mathbf{u}$.

O gradiente da equação 3.10 em relação ao vetor de pesos da rede neural é dado por:

$$\begin{aligned}
\frac{\partial J(\mathbf{w},(k))}{\partial \mathbf{w}} &= -\left[\frac{\partial \mathbf{y}(k)}{\partial \mathbf{w}}\right]' \mathbf{e}(k) \\
&= -\left[\frac{\partial \mathbf{u}(k)}{\partial \mathbf{w}}\right]' \left[\frac{\partial \mathbf{y}(k)}{\partial \mathbf{u}(k)}\right]' \mathbf{e}(k) \\
&= -\left[\frac{\partial \mathbf{u}(k)}{\partial \mathbf{w}}\right]' \mathbf{y}_u'(k)\,\mathbf{y}(k)
\end{aligned} \tag{3.11}$$

Surge aqui um novo fator na expressão do gradiente, $\mathbf{y}_u$, com $\mathbf{y}_u \in \mathbb{R}^{m\times n}$, que é chamado de *matriz jacobiana* ou de *sensibilidade* do processo em relação às saídas do controlador, sendo definida por:

$$\mathbf{y}_u(k) = \frac{\partial \mathbf{y}(k)}{\partial \mathbf{u}(k)} \tag{3.12}$$

Uma forma de se estimar $\mathbf{y}_u$ é através de um modelo identificado para o processo. Logo, no nosso caso, podemos obter um valor estimado para a *matriz de sensibilidade*, $\widehat{\mathbf{y}}_u$, através da equação 3.6 na forma:

$$\widehat{\mathbf{y}}_u(k) = \frac{\partial \widehat{\mathbf{y}}(k)}{\partial \mathbf{u}(k)} \tag{3.13}$$

Observe que a equação 3.13 pode ser obtida a partir da matriz de propagação, equação 1.18, como uma sub-matriz da matriz $\boldsymbol{\lambda}_0$, que propaga a saída até a entrada em uma **RNA** que estiver sendo utilizada na identificação.

Assim, a partir da equação 3.11 podemos adotar para o controlador os desenvolvimentos feitos nos capítulos 1 e 2.

3.5 Análise de Estabilidade para RNA's Aplicadas em Identificação e Controle de Processos

Como mostrado na seção 3.3, o algoritmo de aprendizado utilizado na identificação a redes neurais é o mesmo discutido no capítulo 2. Logo, a análise de estabilidade para este caso é a mesma discutida no capítulo 2.

A seção 3.4 nos mostrou que é necessário fazer-se uma alteração no algoritmo de aprendizado discutido no capítulo 1 para a sua utilização em controladores a redes neurais. A partir da equação 3.11 foi introduzido um termo referente à *sensibilidade* da planta em relação as variáveis de saída do controlador. Assim, é necessário nova análise de estabilidade do algoritmo de aprendizado para a situação presente.

A partir de equação 3.11 temos que:

$$\Delta \mathbf{w}(k) = \eta \left[\frac{\partial \mathbf{u}(k)}{\partial \mathbf{w}}\right]' \mathbf{y}'_u(k)\, \mathbf{e}(k) \tag{3.14}$$

e temos ainda que:

$$\frac{\partial \mathbf{e}(k)}{\partial \mathbf{w}} = -\,\mathbf{y}_u(k) \left[\frac{\partial \mathbf{u}(k)}{\partial \mathbf{w}}\right] \tag{3.15}$$

Fazendo-se $\mathbf{A} = \partial \mathbf{u}(k)/\partial \widehat{\mathbf{w}}$ e $\mathbf{B} = \mathbf{y}_u(k)$ nas expressões 3.14 e 3.15 e substituindo-as nas equações 2.8 e 2.10 teremos que:

$$\Delta \mathcal{V}(k) = -\,\eta\, \mathbf{e}'(k)\, \mathbf{B}\, \mathbf{A}\, \mathbf{A}'\, \mathbf{B}' \left\{2\, \mathbf{I}_m - \eta\, \mathbf{B}\, \mathbf{A}\, \mathbf{A}'\, \mathbf{B}'\right\} \mathbf{e}(k) \tag{3.16}$$

Com base no Segundo Método de Lyapunov para sistemas discretos o sistema é estável se $\Delta \mathcal{V} < 0$. Analisando a equação 3.16 podemos ver que isso impõe, para que o sistema seja estável, $\eta > 0$, satisfazendo o decréscimo do erro, e ainda que:

$$2\, \mathbf{I}_m - \eta\, \mathbf{B}\, \mathbf{A}\, \mathbf{A}'\, \mathbf{B}' \;>\; 0$$

$$2 - \eta\, \lambda_{\max}(\mathbf{B}\, \mathbf{A}\, \mathbf{A}'\, \mathbf{B}') \;>\; 0 \tag{3.17}$$

Na expressão 3.17 $\lambda_{\max}$ corresponde ao maior autovalor da matriz $(\mathbf{B}\, \mathbf{A}\, \mathbf{A}'\, \mathbf{B}')$ e $\mathbf{I}_m$ a uma matriz identidade com a dimensão m, número de saídas da planta. Logo, temos que:

$$0 \;<\; \eta < \frac{2}{\lambda_{\max}(\mathbf{B}\, \mathbf{A}\, \mathbf{A}'\, \mathbf{B}')} = \frac{2}{\left\| \mathbf{y}_u(k)\, \frac{\partial \mathbf{u}(k)}{\partial \mathbf{w}} \right\|^2}$$

$$0 < \eta < \frac{2}{\left\| \mathbf{y}_u(k) \frac{\partial \mathbf{u}(k)}{\partial \widehat{\mathbf{w}}} \right\|^2} \tag{3.18}$$

$$0 < \eta < \frac{2}{\|\mathbf{y}_u(k)\|^2 \left\| \frac{\partial \mathbf{u}(k)}{\partial \widehat{\mathbf{w}}} \right\|^2} \tag{3.19}$$

isto pois $\|\mathbf{B}\,\mathbf{A}\|^2 \leq \|\mathbf{B}\|^2 \, \|\mathbf{A}\|^2$.

Logo a estabilidade do algoritmo, no caso do controlador, é garantida no intervalo apresentado na equação 3.19. Além disso, esta equação impõe que $\| \mathbf{y}_u(k) \| < \infty$, $\forall k$. Para termos esta garantia, imporemos ao processo a condição do mesmo ser **BIBO** estável. Assim como foi feito na seção 2.2, os desenvolvimentos realizados nesta seção correspodem à uma convergência para um ponto de mínimo local.

3.6 O Algoritmo Adaptativo para Aprendizado Aplicado a Neuro-Controladores

Na seção 2.6 apresentamos um algoritmo adaptativo antecipativo à divergências para **RNA's** a *perceptrons*. Na seção anterior verificamos que nas aplicações de **RNA's** em controladores, o processo de aprendizado pode tornar-se instável não só devido às condições apresentadas no capítulo 2, mais também devido à natureza da planta na qual se está aplicando o neuro-controlador, o que é demonstrado pela inclusão da norma da matriz de sensibilidade da planta na expressão do critério de estabilidade, equação 3.19.

Em aplicações reais, não sabemos como se comporta a matriz de sensibilidade. Desde a seção 3.4 sabemos que esta informação pode ser obtida a partir da matriz $\boldsymbol{\lambda}_0$ de algum neuro-identificador que se utilize para a modelagem da planta.

A extensão do algoritmo para controladores é bastante simples, uma vez que é possível se obter a matriz de sensibilidade a partir de um neuro-identificador. A estrutura é a mesma já apresentada na seção 2.6, com a inclusão apenas de informações a respeito da matriz $\widehat{\mathbf{y}}_u$.

Caso a planta seja **BIBO** estável, a norma máxima de $\mathbf{y}_u$ pode ser expressa na forma:

$$\|\widehat{\mathbf{y}}_u\|_{\max} \leq m \, n \, \widehat{y}_{u_{\max}} \tag{3.20}$$

onde m é o número de saídas da planta e n é o número de saídas do controlador.

Assim o algoritmo da seção 2.6, além de agora apresentar os parâmetros $w^0_{2_{\max}}$, $w^{f^0}_{1_{\max}}$ e $x^0_{n_{\max}}$, apresentará também um novo parâmetro, $\hat{y}^0_{u_{\max}}$, que terá a mesma função dos anteriores, já apresentada na seção 2.6, e se adaptará de acordo com as necessidades de convergência do sistema. A estrutura básica do algoritmo é mostrada a seguir em linguagem **PDL**.

Algoritmo 3.1 *Estrutura básica do Algoritmo Adaptativo Antecipativo de Divergências em* **RNA***'s a perceptrons aplicado a neuro-controladores.*

procedure $learn(\mathbf{x} \in \mathbb{R}^{ne}, \mathbf{y}_d \in \mathbb{R}^{ns}, \mathbf{y} \in \mathbb{R}^{ns})$
begin
  def var: $x_{\max} \in \mathbb{R}$;
  $w_{2_{\max}} \in \mathbb{R}$;
  $w_{2_{\max}} \in \mathbb{R}$;
  $\hat{y}_{u_{\max}} \in \mathbb{R}$;
  $\hat{\mathbf{y}}_u \in \mathbb{R}^{m \times n}$;
  $\hat{\mathbf{y}}_u := NeuroIdentifier.sensibility()$;
  $x_{\max} := get\_max(\mathbf{x})$;
  $w_{2_{\max}} := get\_max(\mathbf{w}_2)$;
  $w^f_{1_{\max}} := get\_max(\mathbf{w}^f_1)$;
  $\hat{y}_{u_{\max}} := get\_max(\mathbf{y}_u)$;
  if $(x_{\max} \geq x^0_{\max})$ **or** $(w_{2_{\max}} \geq w^0_{2_{\max}})$
    or $(w^f_{1_{\max}} \geq w^{f^0}_{1_{\max}})$ **or** $(y_{u_{\max}} \geq y^0_{u_{\max}})$ **then**
    if $(x_{\max} \geq x^0_{\max})$ **then**
      $x^0_{\max} := \kappa \times x_{n_{\max}}$ **endif**
    if $(w_{2_{\max}} \geq w^0_{2_{\max}})$ **then**
      $w^0_{2_{\max}} := \kappa \times w_{2_{\max}}$ **endif**
    if $(w^f_{1_{\max}} \geq w^{f^0}_{1_{\max}})$ **then**
      $w^{f^0}_{1_{\max}} := \kappa \times w^f_{1_{\max}}$ **endif**
    if $(y_{u_{\max}} \geq y^0_{u_{\max}})$ **then**
      $y^0_{u_{\max}} := \kappa \times y_{u_{\max}}$ **endif**
    $get\_learn\_rate(x^0_{\max}, w^0_{2_{\max}}, w^{f^0}_{1_{\max}})$;
  endif
  $apply\_gradient\_descent(\cdot)$;
end

3.7 Sumário

Neste capítulo foi mostrado como o paradigma das **RNA**'s pode ser aplicado em sistemas de controle. Após fornecida uma visão geral de sistema de controle, foram desenvolvidos modelos e estruturas para identificação e controle de processos, seções 3.3 e 3.4, sendo feita a extensão da lei de aprendizado pelo gradiente para controladores.

No caso dos controladores, a lei de aprendizado necessita de informações a respeito da sensibilidade da saída do processo em relação à do controlador. Essa informação pode ser obtida a partir da matriz de propagação, como sub-matriz da matriz λ_0, que propaga a saída até à entrada em uma **RNA** que estiver sendo utilizada na identificação do processo. Na análise de estabilidade, verificou-se que para a obtenção do intervalo de admissibilidade da taxa de aprendizado é necessário conhecermos a norma da matriz de sensibilidade do processo em relação às saídas do controlador. Foi feita também uma extensão do algoritmo adaptativo para o aprendizado para o caso dos controladores.

O conteúdo das seções 3.3 e 3.4, bem como o algoritmo adaptativo para o aprendizado em controladores seram utilizados na parte experimental deste trabalho.

Capítulo 4

Descrição de um Sistema Robótico

Neste capítulo é feita uma descrição matemática de um sistema robótico, apresentada a análise cinemática e uma dinâmica com a inclusão do modelo de acionadores elétricos, seguido de um método para o projeto de controladores **PID's**.

4.1 Introdução

Uma sistema robótico ou manipulador robótico é constituido de um sistema mecânico composto de elos (*links*) conectados por juntas em uma cadeia aberta ou fechada, controlada por um sistema hierárquico programável em vários níveis. Esta cadeia de elementos mecânicos tem por objetivo final o posicionamento e orientação da extremidade do último elo em um ponto do espaço, com ou sem controle de trajetórias pré-estabelecido.

As juntas são tipicamente classificadas em rotacionais, que permitem movimentos de rotação entre dois elos, e prismáticas, que permitem movimentos de translação. Define-se como *variáveis de juntas*, a representação do deslocamento relativo entre dois elos adjacentes, denotadas por θ_i, *deslocamento angular*, para as juntas rotacionais e por d_i, *deslocamento linear*, para as juntas prismáticas. A posição e a orientação da extremidade do último elo são influenciadas por todas as variáveis de juntas.

As juntas são acionadas através da aplicação de torques ou forças externas por meio de atuadores, que podem ser hidraúlicos, que permitem torques ou forças mais elevadas; pneumáticos, que permitem movimentos de precisão, porém com baixa potência; ou elétricos, com características intermediárias entre os outros dois, mas com facilidade de implementação e custos mais baixos.

A maioria dos sensores utilizados estão no nível de controle de juntas e são

para as medidas das variáveis de juntas e para algumas variáveis dos atuadores como corrente e tensão elétrica. Porém, em algumas plantas são utilizados sensores de torque, de posicionamento espacial, de visão, de proximidade, de deslizamento, de força, etc.

Em geral, o número de juntas determina o número de graus de liberdade do mecanismo (*Degree-Of-Freedom*, **DOF**). Tipicamente, um robô deve possuir no mínimo seis **DOF** independentes, três para o posicionamento e três para a orientação. Com menos de seis **DOF** indepedentes não é possível atingir um posicionamento-orientação espacial completo no espaço de trabalho pré-definido. Aplicações onde o ambiente de trabalho possui obstáculos aos movimentos do robô podem requerer mais de seis **DOF**. Um robô com mais de seis **DOF** é dito *cinematicamente redundante.*

O estudo ou análise de manipuladores pode ser dividido em duas partes, a *análise cinemática* e a *análise dinâmica.* Na *análise cinemática* é feito o estudo das variáveis de movimento do sistema (posições, velocidades e acelerações) sem preocupação com suas causas. Já a *análise dinâmica* estuda os movimentos preocupando-se com as suas causas, os torques ou forças aplicados às juntas.

Para facilitar a escrita de muitas equações que se seguem, adotaremos para a primeira, segunda e terceira derivadas no tempo de alguma variável x, como por exemplo, posição cartesiana ou então variáveis de junta, as seguintes notações respectivamente $\dot{x}$, $\ddot{x}$ e $x^{(3)}$.

4.2 Análise da Cinemática de Manipuladores

Cada junta tem um **DOF**, podendo esta ser rotacional ou prismática. Para um manipulador com n juntas, numeradas de 1 até n, existem $n+1$ elos, numerados de 0 (*zero*) até n. O elo *zero*, geralmente fixo, é chamado de *elo base do sistema* e o elo n possui a extremidade final do sistema. Cada junta i, para $i = 1, 2, \ldots, n$, conecta os elos i e $i-1$.

Um elo pode ser considerado como um corpo rígido, podendo definir assim uma relação entre duas juntas vizinhas. Pode ser especificado através de dois parâmetros, comprimento (*link length*) e posição angular (*link twist*), que definem um posicionamento em dois eixos coordenados espaciais.

Uma junta também pode ser especificada por dois parâmetros, a distância de um elo ao próximo ao longo do eixo da junta (*link offset*), e pelo ângulo da junta que é a distância angular de um elo ao próximo sobre o eixo da junta (*joint angle*).

Para facilitar a localização de cada elo são fixados eixos de coordenadas (xyz)

nos mesmos, onde o eixo i esta fixado no elo i. Denavit e Hartenberg [19] propuseram um método matricial de alocação sistemática dos eixos coordenados para cada elo de uma cadeia articulada. No método, o eixo de cada junta i é alinhado com o eixo z do sistema de coordenadas $i-1$. O eixo x do sistema de coordenadas $i-1$ é direcionado ao longo da normal entre os eixos z_{i-1} e z_i, como visto na figura 4.1. Os parâmetros dos elos das juntas estão resumidos na tab. 4.1.

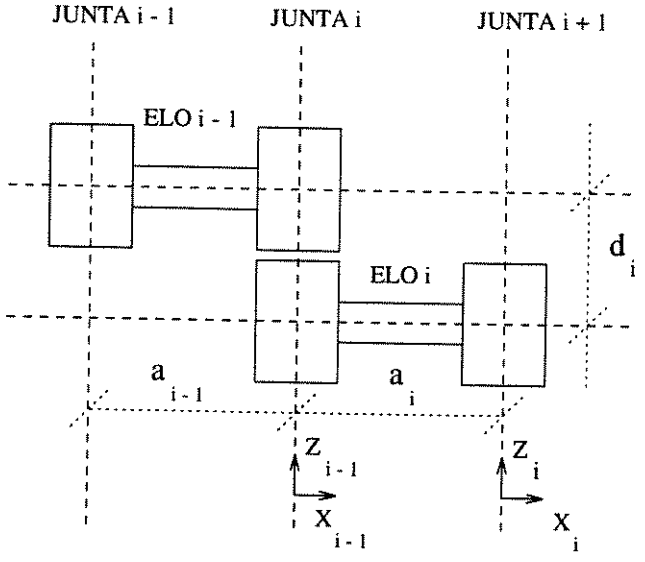

Figura 4.1: Localização dos eixos coordenados em uma cadeia articulada.

Para uma junta rotacional, θ_i é a variável de junta e d_i, *off-set*, é uma constante, enquanto que para uma junta prismatica, d_i é a variável de junta e θ_i uma constante. É habitual generalizar-se as variáveis de junta por q_i, ditas *coordenadas generalizadas.*

A representação de *Denavit-Hartenberg* (**DH**) resulta em uma matriz de transformação de coordenadas entre elos adjacentes na forma [66]:

$$A_i^{i-1} = \begin{bmatrix} \cos\theta_i & -\sin\theta_i \cos\alpha_i & \sin\theta_i \sin\alpha_i & a_i \cos\theta_i \\ \sin\theta_i & \cos\theta_i \sin\alpha_i & -\cos\theta_i \sin\alpha_i & a_i \sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

A representação das coordenadas de cada elo em relação ao sistema de coordenadas base ($i = 0$) é dado por:

$$T_i^0 = T_{i-1}^0 \, A_i^{i-1} \tag{4.1}$$

onde T_i^0 representa a transformação homogênea do sistema de coordenadas i em relação ao sistema de coordenadas base.

link lenght	a_i	distância entre os eixos z_{i-1} e z_i ao longo do eixo x_i
link twist	α_i	ângulo entre os eixos z_{i-1} e z_i sobre o eixo x_i
link offset	d_i	distância entre a origem do eixo coordenado $i-1$ e o eixo x_i ao longo do eixo z_{i-1}
joint angle	θ_i	ângulo entre os eixos x_{i-1} e x_i sobre o eixo z_{i-1}

Tabela 4.1: Parâmetros dos elos e das juntas

Para um manipulador com n elos considerados rígidos, a *análise cinemática direta* é a determinação da posição e orientação da extremidade do último elo em relação ao sistema de coordenadas base como função das variáveis de junta.

Este resultado é obtido através da equação 4.1 na forma:

$$T_n^0 = A_1^0 A_2^1 \cdots A_n^{n-1} = \mathcal{K}(\mathbf{q}) \tag{4.2}$$

que é o produto das matrizes de transformação de cada elo. A transformação T_n^0 será aqui denotada simplificadamente na forma T_n.

A *análise cinemática inversa* consiste na determinação das variáveis de juntas que permitam um determinado posicionamento e/ou orientação da extremidade do último elo, tendo a forma:

$$\mathbf{q} = \mathcal{K}^{-1}(T_n) \tag{4.3}$$

Esta solução não é única. Se não existir solução na equação 4.3 para uma determinada posição do manipulador esta é dita singular. As singularidades podem ocorrer devido a algum alinhamento de eixos de juntas, ou a reduzidos números de **DOF** ou ainda a um ponto fora do alcance do manipulador.

Define-se como *trajetória* a um conjunto de pontos no espaço pelos quais o manipulador deverá posicionar-se sequencialmente no tempo, contínua ou discretamente. Em muitos casos, a trajetória é especificada com inclusão de velocidades cartesianas, que podem ser obtidas através da diferenciação da equação 4.2, *equação cinemática direta*, na forma:

$$\dot{\mathbf{x}} = J(\mathbf{q}) \tag{4.4}$$

onde $J(q)$ é chamada de *matriz jacobiana* e relaciona *velocidades no espaço das juntas* com a *velocidade cartesiana* da extremidade do último elo do manipulador, sendo seus elementos derivadas parciais na forma:

$$\frac{\partial x_i}{\partial q_j} \quad , \forall\, i, j \tag{4.5}$$

Para posições de não singularidade em robôs com até seis **DOF** a matriz jacobiana é inversível, dando origem a chamada *matriz jacobiana inversa* na forma:

$$\dot{\mathbf{q}} = J^{-1}(\mathbf{q})\,\dot{\mathbf{x}} \tag{4.6}$$

que relaciona velocidades no espaço cartesiano com velocidades no espaço das juntas.

Assim, especificada uma trajetória a ser seguida pelo manipulador e estabelecendo-se restrições de velocidades cartesianas, $\dot{\mathbf{x}}$, pode-se especificar velocidades para o espaço de juntas de maneira a permitir ao manipulador sair de uma posição inicial a uma posição final, passando por todos os pontos da trajetória, de forma contínua. O planejamento de trajetória feito desta maneira, baseado na equação 4.6, é chamado de *Planejamento pelo Jacobiano Inverso*, o qual assume que a matriz jacobiana é inversível para cada ponto da trajetória especificada.

4.3 Análise da Dinâmica de Manipuladores

A análise dinâmica de manipuladores é feita baseada em equações de movimento dos manipuladores, sendo estes movimentos respostas a torques externos aplicados.Existem dois problemas básicos relacionados com a dinâmica de manipuladores:

Dinâmica Direta: Onde as equações de movimento são resolvidas para se encontrar as respostas a torques aplicados.

Dinâmica Inversa: Onde as equações de movimento são resolvidas para a determinação dos torques necessários para um dado movimento.

Spong e Vidyasagar [66] descrevem as equações de movimento, pela formulação de *Euler-Lagrange*, para um manipulador de n **DOF**, com $n \in \mathbb{I}$, na forma:

$$\tau_i \;=\; \sum_{j=1}^{n} d_{ij}\,\ddot{q}_j + \sum_{j=1}^{n}\sum_{k=1}^{n} c_{ijk}\,\dot{q}_j\,\dot{q}_k + g_i \tag{4.7}$$

para $i = 1, 2, \cdots, n$, onde:

- q_i, $\dot{q}_i$ e $\ddot{q}_i$ são respectivamente a posição, a velocidade e a aceleração no espaço das juntas para a i-ésima junta do manipulador;
- d_{ij} são os elementos da matriz de inércia do sistema, $D(q)$, com $D(q) \in \mathbb{R}^{n \times n}$;
- c_{ijk} são os elementos da matriz de torques ou forças centrípeta e de coriolis para a i-ésima junta, C_i, com $C_i \in \mathbb{R}^{n \times n}$;
- g_i são os torques ou forças gravitacionais visto pela i-ésima junta.

4.3.1 Dinâmica do Atuador Elétrico

Na maioria das aplicações, os manipuladores robóticos possuem suas juntas acionadas por motores elétricos de corrente contínua, **DC**. Na figura 4.2 temos o modelo de campo elétrico e na figura 4.3 o modelo do sistema mecânico de um motor elétrico sendo a equação dinâmica do sistema elétrico dada por:

$$R\, i(t) + L\, \frac{d\, i(t)}{d\, t} + e(t) = u(t) \tag{4.8}$$

onde:

- R é a resistência elétrica da armadura do motor;
- L é a indutância da armadura;
- $i(t)$ é a corrente elétrica do sistema;
- $e(t)$ é a força contra-eletromotriz;
- $u(t)$ é a voltagem elétrica aplicada na armadura.

Para motores **DC** a campo constante, $i_c = i_{c_0} = cte.$, a força contra-eletromotriz é proporcional à velocidade angular do eixo do motor na forma:

$$e(t) = K_e\, \dot{\theta}_m \tag{4.9}$$

onde θ_m é a posição angular do eixo do motor e K_e é uma constante de voltagem.

O torque produzido pelo motor é proporcional à corrente elétrica na forma:

$$\tau_m(t) = K_t\, i(t) \tag{4.10}$$

onde τ_m é o torque motor e K_t é a constante de torque.

No acionamento de uma carga por um motor elétrico muitas vezes são utilizados sistemas redutores de movimento, ou amplificadores de torque, para adequar o movimento

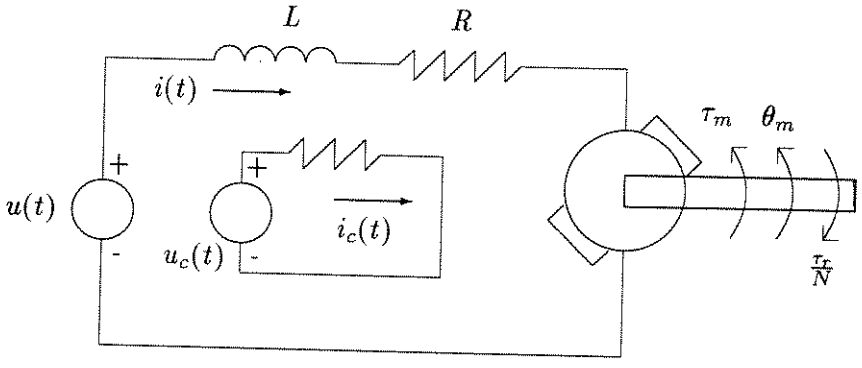

Figura 4.2: Modelo de campo elétrico do motor.

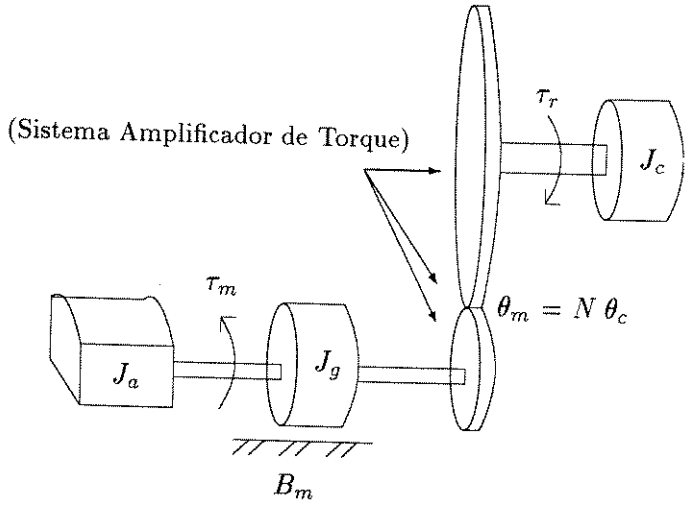

Figura 4.3: Modelo mecânico do motor elétrico.

ou torque do motor elétrico ao movimento ou torque requerido pela carga. Assim temos a seguinte relação entre os movimentos da carga e do motor elétrico:

$$N = \frac{\theta_m}{\theta_c} \tag{4.11}$$

onde N é o fator de redução e θ_c é a posição angular da carga.

A relação entre o torque motor e o torque resistente (veja figura 4.3) possui a forma:

$$\tau_m(t) - \frac{\tau_r(t)}{N} = J_m\,\ddot{\theta}_m + B_m\,\dot{\theta}_m \tag{4.12}$$

onde $J_m = J_a + J_g$ é o momento de inércia do motor e B_m é o coeficiente de atrito viscoso.

Substituindo as equações 4.8, 4.9 e 4.10 e na equação 4.12 teremos:

$$\frac{K_t}{R}\,u(t) = J_m\,\ddot{\theta}_m + \left(B_m + \frac{K_t\,K_e}{R}\right)\dot{\theta}_m + \frac{L\,K_t}{R}\,\frac{d\,i(t)}{d\,t} + \frac{1}{N}\,\tau_r \tag{4.13}$$

4.3.2 Dinâmica do Manipulador com Inclusão do Atuador Elétrico

Nesta seção veremos o modelo matemático da dinâmica do manipulador com inclusão da dinâmica do atuador elétrico (veja seção 3.2). A equação dinâmica 4.7 descreve, para cada junta i, o torque resistente produzido pelo manipulador. Na situação de equilíbrio temos que $\tau_m = (1/N)\tau_r$ e com base na equação 4.10 temos que para o acionador da i-ésima junta:

$$i_i(t) = \frac{1}{N_i\,K_{t_i}}\left(\sum_{j=1}^{n} d_{ij}\,\ddot{q}_j + \sum_{j=1}^{n}\sum_{k=1}^{n} c_{ijk}\,\dot{q}_j\,\dot{q}_k + g_i\right) \tag{4.14}$$

para $i = 1, 2, \cdots, n$.

Diferenciando-se a equação 4.14 teremos:

$$\frac{d\,i_i(t)}{d\,t} = \frac{1}{N_i\,K_{t_i}}\left\{\sum_{j=1}^{n}(d_{ij}\,q_j^{(3)} + \dot{d}_{ij}\,\ddot{q}_j)\right.$$

$$\left. + \sum_{j=1}^{n}\sum_{k=1}^{n}(\dot{c}_{ijk}\,\dot{q}_j\,\dot{q}_k + c_{ijk}\,\ddot{q}_j\,\dot{q}_k + c_{ijk}\,\dot{q}_j\,\ddot{q}_k) + \dot{g}_i\right\} \tag{4.15}$$

Ainda, substituindo θ_m por $N\,q = N\,\theta_c$ na equação 4.13, para cada junta i teremos:

$$\frac{K_{t_i}}{R_i}\,u_i(t) = N_i\,J_{m_i}\,\ddot{q}_i + N_i\left(B_{m_i} + \frac{K_{t_i}\,K_{e_i}}{R_i}\right)\dot{q}_i + \frac{L_i\,K_{t_i}}{R_i}\,\frac{d\,i_i(t)}{d\,t} + \frac{1}{N_i}\,\tau_r \tag{4.16}$$

para $i = 1, 2, \cdots, n$.

Substituindo as equações 4.14 e 4.15 na equação 4.16 teremos:

$$\begin{aligned}\frac{K_{t_i}}{R_i}\, u_i(t) \;=\; & \left(N_i\, J_{m_i}\, \ddot{q}_i + \frac{1}{N_i} \sum_{j=1}^{n} d_{ij}\, \ddot{q}_j \right) \\ & + \left(N_i\, B_{m_i}\, \dot{q}_i + \frac{N_i\, K_{t_i}\, K_{e_i}}{R_i}\, \dot{q}_i + \frac{1}{N_i} \sum_{j=1}^{n} \sum_{k=1}^{n} c_{ijk}\, \dot{q}_j\, \dot{q}_k \right) + \frac{1}{N_i}\, g_i \\ & + \frac{L_i}{R\, N_i} \left\{ \sum_{j=1}^{n} (d_{ij}\, q_j^{(3)} + \dot{d}_{ij}\, \ddot{q}_j) \right. \\ & \left. + \sum_{j=1}^{n} \sum_{k=1}^{n} (\dot{c}_{ijk}\, \dot{q}_j\, \dot{q}_k + c_{ijk}\, \ddot{q}_j\, \dot{q}_k + c_{ijk}\, \dot{q}_j\, \ddot{q}_k) + \dot{g}_i \right\} \end{aligned} \tag{4.17}$$

para $i = 1, 2, \cdots, n$.

Analisando a equação 4.17, podemos observar que a mesma é não linear de terceira ordem em relação à posição angular e de segunda ordem em relação à velocidade angular, tendo também forte acoplamento entre as variáveis de junta.

4.3.3 Controladores Clássicos para Manipuladores Robóticos

Nesta seção abordaremos os controladores clássicos utilizados em sistemas robóticos, controladores da classe **PID** (proporcional mais integral mais derivativo) ou variações do mesmo. Esta classe de controlador possui a seu favor fatos como o de ser de fácil implementação, de baixo custo, de boa estabilidade e relativamente fácil de se projetar. Utilizaremos a estratégia de controle de juntas independentes, sem entretanto utilizarmos algum método de desacoplamento entre as mesmas [66]. Inicialmente, iremos obter uma função de transferência aproximada para cada junta independente no domínio da *Transformada de Laplace*, $\mathcal{L}$.

O valor da *indutância*, L, em motores elétricos é na prática muito pequeno, assim desprezaremos os termos multiplicados por L na equação 4.17. O fato desses termos estarem também sendo divididos pelo *fator de redução*, N_i, que em geral é muito grande, apoia esta decisão.

Assim, podemos reescrever a equação 4.17 na seguinte forma:

$$\frac{K_{t_i}}{R_i} u_i(t) = \left(N_i J_{m_i} + \frac{1}{N_i} d_{ii} \right) \ddot{q}_i + \left(N_i B_{m_i} + \frac{N_i K_{t_i} K_{e_i}}{R} \right) \dot{q}_i$$

$$+ \frac{1}{N_i} \left\{ \sum_{j=1, j \neq i}^{n} d_{ij} \ddot{q}_j + \sum_{j=1}^{n} \sum_{k=1}^{n} c_{ijk} \dot{q}_j \dot{q}_k + g_i \right\} \tag{4.18}$$

para $i = 1, 2, \cdots, n$.

Se considerarmos N_i muito grande poderemos reescrever a equação 4.18 na forma:

$$J_{eft_i} \ddot{q}_i + B_{eft_i} \dot{q}_i = K_{eft_i} u(t) - \frac{1}{N_i} p_i(t) \tag{4.19}$$

onde J_{eft_i} é a inercia efetiva para a i-ésima junta, assim como B_{eft_i} é o atrito efetivo, K_{eft_i} é o ganho efetivo, e $p_i(t)$ é uma variável de perturbação, as quais assumem respectivamente as formas:

$$J_{eft_i} = N_i J_{m_i} + \frac{1}{N_i} d_{ii} \tag{4.20}$$

$$B_{eft_i} = N_i B_{m_i} + \frac{N_i K_{t_i} K_{e_i}}{R} \tag{4.21}$$

$$K_{eft_i} = \frac{K_{t_i}}{R_i} \tag{4.22}$$

$$p_i(t) = \sum_{j=1, j \neq i}^{n} d_{ij} \ddot{q}_j + \sum_{j=1}^{n} \sum_{k=1}^{n} c_{ijk} \dot{q}_j \dot{q}_k + g_i \tag{4.23}$$

para $i = 1, 2, \cdots, n$.

Podemos observar na equação 4.19 que a mesma é de segunda ordem e o *fator de redução*, N_i, quando muito grande, é de grande importância na redução da influência do acoplamento entre as variáveis do sistema (veja Spong e Vidyasagar [66]). Neste caso, $p_i(t)$ pode ser considerado uma variável externa, independente. Com isso o modelo expresso pela equação 4.19, de juntas independentes, torna-se linear. A Eletro-Craft Corporation [21] apresenta um método para o cálculo de um valor ótimo para N_i de modo a produzir máxima transferência de torque ou força.

A Transformada de Laplace aplicada à equação 4.19 é da forma:

$$Q_i(s) = \frac{K_{eft_i}}{s\,(J_{eft_i}\,s + B_{eft_i})}\,U_i(s) - \frac{1}{N_i}\,\frac{1}{s\,(J_{eft_i}\,s + B_{eft_i})}\,P_i(s) \tag{4.24}$$

para $i = 1, 2, \cdots, n$.

A equação 4.24 é para posição angular da i-ésima junta do manipulador, porém se definirmos $\omega_i = \dot{q}_i$, velocidade angular da i-ésima junta, teremos:

$$\Omega_i(s) = \frac{K_{eft_i}}{(J_{eft_i}\,s + B_{eft_i})}\,U_i(s) - \frac{1}{N_i}\,\frac{1}{(J_{eft_i}\,s + B_{eft_i})}\,P_i(s) \tag{4.25}$$

para $i = 1, 2, \cdots, n$.

As equações 4.24 e 4.25 podem ser reescritas como:

$$Q_i(s) = \frac{K_{x_i}}{s\,(\tau_{x_i}\,s + 1)}\,U_i(s) - \frac{\frac{1}{N_i}\,B_{eft_i}}{s\,(\tau_{x_i}\,s + 1)}\,P_i(s) \tag{4.26}$$

$$\Omega_i(s) = \frac{K_{x_i}}{(\tau_{x_i}\,s + 1)}\,U_i(s) - \frac{\frac{1}{N_i}\,B_{eft_i}}{(\tau_{x_i}\,s + 1)}\,P_i(s) \tag{4.27}$$

onde τ_{x_i} e K_{x_i} são respectivamente a *constante de tempo* e o *ganho* para a i -ésima junta na forma:

$$\tau_{x_i} = \frac{J_{eft_i}}{B_{eft_i}} \tag{4.28}$$

$$K_{x_i} = \frac{K_{eft_i}}{B_{eft_i}} \tag{4.29}$$

Analisando-se as equações 4.26 e 4.27 notamos que podemos tomar $U_i(s)$ como variável de atuação, ou a ser manipulada, para o controle da posição ou velocidade da i-ésima junta do manipulador e assim a tomaremos como sendo saída dos controladores que abordaremos a seguir. Ainda, nota-se que a equação 4.26, *função de transferência* para a *posição angular*, não é **BIBO** estável, pois possui um *pólo na origem* do plano cartesiano s. Já a equação 4.27, *função de transferência* para a *velocidade angular*, é **BIBO** estável, possuindo um único *pólo*, localizado no ponto $-1/\tau_{x_i}$ do eixo real do plano cartesiano s.

Na secão 4.2, observamos que é possível se fazer um posicionamento espacial de sistema robótico através da especificação de velocidades para o espaço de juntas, ω_i, permitindo ao manipulador sair de uma posição inicial a uma posição final, passando por todos os pontos da trajetória de forma contínua. Assim, visto que a função de transferência para a velocidade angular é **BIBO** estável, adotaremos para o nosso trabalho o posicionamento por controle de velociade angular, ω_i, definindo-se perfis de velocidade com aceleração, desde $\omega_i = 0$, e desaceleração, até $\omega_i = 0$.

Controladores tipo PID

A equação dinâmica que descreve o funcionamento de um controlador **PID**, com realimentação unitária, para o controle da velocidade angular da i-ésima junta pode ser do tipo:

$$U_i(s) = \frac{(K_{d_i}\, s^2 + K_{p_i}\, s + Ki_i)}{s} (\Omega_{d_i}(s) - \Omega_i(s)) \tag{4.30}$$

onde $\Omega_{d_i}(s)$, K_{i_i}, K_{d_i} e K_{p_i} são respectivamente a *velocidade angular esperada*, *ganho integral*, *ganho derivativo* e *ganho proporcional* para o controlador.

Substituindo a equação 4.30 na equação 4.27, teremos para o sistema em malha fechada a seguinte expressão:

$$\Omega_i(s) = \frac{K_{x_i}\,(K_{d_i}\, s^2 + K_{p_i}\, s + Ki_i)}{(\tau_{x_i} + K_{x_i}\, K_{d_i})\, s^2 + (1 + K_{x_i}\, K_{p_i})\, s + K_{x_i}\, K_{i_i}} \Omega_{d_i}(s)$$

$$+ \frac{\frac{1}{N_i\, B_{eft_i}}}{(\tau_{x_i} + K_{x_i}\, K_{d_i})\, s^2 + (1 + K_{x_i}\, K_{p_i})\, s + K_{x_i}\, K_{i_i}} P_i(s) \tag{4.31}$$

Observando o *polinômio característico* da equação 4.31, vemos que ele é de segunda ordem e assim podemos escolher valores para K_{p_i}, K_{d_i} e K_{i_i} de modo que a resposta do sistema em malha fechada, para a i-ésima junta, obedeça a igualdade:

$$s^2 + \frac{1 + K_{x_i}\, K_{p_i}}{\tau_{x_i} + K_{x_i}\, K_{d_i}}\, s + \frac{K_{x_i}\, K_{i_i}}{\tau_{x_i} + K_{x_i}\, K_{d_i}} = s^2 + 2\,\zeta_i\,\omega_{n_i} + \omega_{n_i}^2$$

onde ζ_i e ω_{n_i} são respectivamente a taxa de amortecimento e a frequência natural da resposta da i-ésima junta. Para manipuladores robóticos é hábito fazer $\zeta_i \in [0.707, 1) \subset \mathbb{R}$, produzindo uma resposta rápida, ficando o parâmetro ω_{n_i} responsável pelo tempo de estabilização do sistema.

Na síntese do controlador tipo **PID**, equação 4.32, temos um parâmetro de livre escolha, ou K_{p_i} ou K_{d_i} ou K_{i_i}. Uma escolha não adequada para algum desses parâmetros que se tome como livre pode provocar problemas de *sensibilidade* em relação às variações dos parâmetros τ_{x_i} e K_{x_i} do modelo da planta. Assim adotaremos o controlador tipo **PI** para o nosso trabalho, fazendo-se $K_{d_i} = 0$, pois, além de com isso deixarmos de ter um parâmetro livre para o sistemas, o controlador **PI** permite a redução do *erro de regime*, de grande importâcia quando se faz o *rastreamento assintótico*. No entanto, a ação *integral* faz com que o processo se torne menos estável. Assim, caso haja a verificação de instabilidades no sistemas, o bom senso recomenda a inclusão da ação *derivativa*.

Nessa situação os parâmetros K_{p_i} e K_{i_i} podem ser obtidos com as expressões:

$$K_{p_i} = \frac{2\,\zeta\,\omega_{n_i}\,\tau_{x_i} - 1}{K_{x_i}} \tag{4.32}$$

$$K_{i_i} = \frac{\omega_{n_i}^2\,\tau_{x_i}}{K_{x_i}} \tag{4.33}$$

Um controlador da classe **PID**, como demostrado, é bastante simples e de fácil síntese. Entretanto, ele requer, para ser eficiente e a atender aos critérios de funcionalidade pré-estabelecidos, um conhecimento profundo do sistema, tanto a nível de modelo quanto a nível de precisão de seus parâmetros.

4.4 Sumário

Neste capítulo foi feita uma descrição matemática de um sistema robótico, apresentando-se uma análise cinemática e uma dinâmica com a inclusão do modelo de acionadores elétricos. Foi feita uma linearização do modelo dinâmico do sistema e aplicado a Transformada de Laplace. Em seguida foi desenvolvido um método para projeto de controladores tipo **PID**.

O modelo dinâmico do sistema será utilizado para simulações na parte experimental do trabalho, bem como o método de projeto de controladores **PID**, que será utilizado para o ajuste de controladores tipo **PI**, com a finalidade de comparações com controladores baseados em **RNA**'s.

Capítulo 5

Aplicações de RNA's em Sistemas Robóticos: Identificação e Controle para o Espaço de Juntas

Neste capítulo são feitas a aplicações de **RNA's** no controle de sistemas robóticos. São propostos um neuro-identificador e um neuro-controlador para estes sitemas. O método de controle proposto é aplicado sobre o modelo simulado de um robô de 2 **DOF**. Três experimentos são realizados: identificação off-line; sintonia on-line do neuro-controlador; e comparação entre o neuro-controlador e o controlador **PI**.

5.1 Introdução

No capítulo 3 observamos que as **RNA**'s possuem um grande potencial para aplicações em identificação e controle de processos, tendo em vista serem ajustados por métodos de aprendizado, sem a necessidade de um profundo conhecimento do processo ao qual esteja sendo aplicado.

Já no capítulo 4 vimos que, em geral, apesar dos sistemas robóticos terem suas dinâmicas conhecidas, as mesmas apresentam um comportamento não linear, com fortes interações entre as diversas variáveis, e possuindo ainda uma grande quantidade de parâmetros de difícil estimação e, em alguns casos, variantes no tempo.

Esses fatos nos motivam à utilização de soluções a base de **RNA's** para problemas de sistemas robóticos. No nosso caso, faremos aplicações à modelagem dinâmica direta e ao controle para o espaço de juntas, visto que as soluções mais simples que existem, a

classe de controladores **PID**, dependem fortemente de um modelo analítico do sistema e possui desempenho muito pobre para casos muito não lineares.

Na seção 4.3.3 já abservamos que a *Função de Transferência* para a varíavel posição angular não é **BIBO** estável, pois possui um pólo na origem do sistema cartesiano, o que, a princípio, a descredencia para um controle direto com **RNA**'s. Já a *Função de Transferência* para a variável velocidade angular é **BIBO** estável, credenciando-a para um controle direto com **RNA's**. Na seção 4.2, verificamos que é possível fazer um posicionamento no espaço cartesiano, fornecendo-se ao sistema referências de velocidades para o espaço de juntas, através do *Planejamento pelo Jacobiano Inverso*. Com tudo isso, abordaremos aplicações de identificação e controle para velocidades angulares. Também faremos apenas o uso das redes **DRNN**.

Apresentaremos a proposta de um *neuro-identificador* e de um *neuro-controlador* para o espaço de juntas e faremos aplicações, atraves de simulações, a um sistema robótico planar de dois **DOF**, como mostrado na figura 5.1, comparando os resultados com os obtidos com a classe de controladores **PID**. O modelo do sistema robótico simulado e o método de simulação empregado podem ser encontrado em Tarn *et al* [67], Yu e Lloyd [73] e Ishiguro *et al* [29].

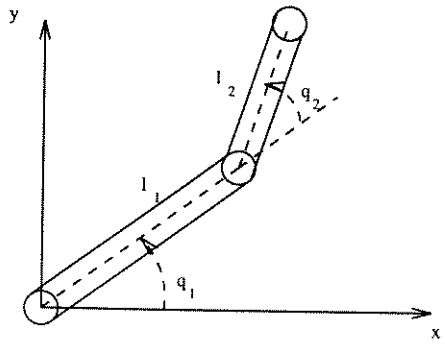

Figura 5.1: Sistema Robótico com dois **DOF**

O conjunto de equações dinâmicas que descrevem o sistema robótico é dado pela equação 4.7, que aplicado neste caso se torna:

$$\tau = \mathbf{d}(\mathbf{q})\,\ddot{\mathbf{q}} + \mathbf{c}(\mathbf{q},\dot{\mathbf{q}})\,\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q})$$

com $\mathbf{d}(\mathbf{q}) \in \mathbb{R}^{2\times 2}$, $\mathbf{c}(\mathbf{q},\dot{\mathbf{q}}) \in \mathbb{R}^{2\times 2}$ e $\mathbf{g}(\mathbf{q}) \in \mathbb{R}^{2}$, onde:

$$d_{11} = m_1\, l_{c_1}^2 + m_2\,(l_1^2 + l_1\, l_2\, \cos(q_1)) + I_1 + I_2$$

$$d_{12} = m_2\,(l_{c_2}^2 + l_1\, l_2\, \cos(q_2)) + I_2$$

$$d_{21} = m_2 (l_{c_2}^2 + l_1 l_{c_2} \cos(q_2)) + I_2$$

$$d_{22} = m_2 l_{c_2}^2 + I_2$$

$$c_{11} = -m_2 l_1 l_{c_2} \sin(q_2) \dot{q}_2$$

$$c_{12} = -m_2 l_1 l_{c_2} \sin(q_2) (\dot{q}_1 + \dot{q}_2)$$

$$c_{21} = m_2 l_1 l_{c_2} \sin(q_2) \dot{q}_1$$

$$c_{22} = 0$$

$$g_1 = (m_1 l_{c_1} + m_2 l_1) g \cos(q_1) + m_2 l_{c_2} \cos(q_1 + q_2)$$

$$g_2 = m_2 l_{c_2} g \cos(q_1 + q_2)$$

Detalhes sobre a obtenção dessas equações podem ser encontrados em Spong e Vidyasagar [66]. A tabela 5.1 mostra o significado de cada parâmetro e os respectivos valores utilizados para as simulações. I_1 e I_2 são os *momentos de inércia* em relação aos centros de massas dos elos 1 e 2.

Foram utilizados também motores elétricos **DC** na simulação do acionamento das duas juntas do sistema. Na tabela 5.2 apresenta-se os valores utilizados para os parâmetros dos motores.

Além disso, como drives de potência para os motores, foram utilizados amplificadores com características descritas na tabela 5.3. Para o sistema de medição de velocidade no espaço das juntas, utilizamos sensores com características descritas na tabela 5.4. Assim, o sistema como um *"todo unificado"* assume a forma apresentada na figura 5.2.

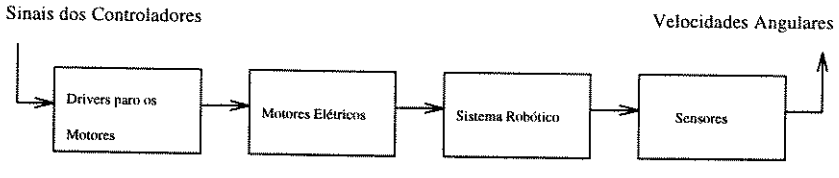

Figura 5.2: Sistema Robótico como um "todo unificado"

Comprimento do elo um	l_1	0.432 m
Centro de massa do elo um	l_{c_1}	0.216 m
Massa do elo um	m_1	15.91 kg
Momento de inércia do elo um	I_1	0.247 kgm^2
Fator de redução da junta um	N_1	62.55
Comprimento do elo dois	l_2	0.432 m
Centro de massa do elo dois	l_{c_2}	0.216 m
Massa do elo dois	m_2	11.36 kg
Momento de inércia do elo dois	I_2	0.177 kgm^2
Fator de redução da junta dois	N_2	62.55

Tabela 5.1: Parâmetros do Sistema Robótico de dois graus de liberdade (2 **DOF**)

Constante de voltagem	K_e	0.19 volts s/m
Constante de torque	K_t	0.35 N m/A
Resistência elétrica	R	1.6 Ω
Indutância	L	0.0048 H
Momento de inércia	J	0.007 kgm^2
Coeficiente de atrito viscoso	B	zero

Tabela 5.2: Parâmetros dos Motores Elétricos

Ganho de voltagem	G_v	15
Impedância de entrada	Z_i	infinita
Impedância de saída	Z_s	zero
Frequência de corte	f_{corte}	muito alta

Tabela 5.3: Parâmetros dos Amplificadores de Potência

Ganho	G	1 volts/ω
Frequência de corte	f_{corte}	muito alta

Tabela 5.4: Parâmetros dos sensores de velocidade

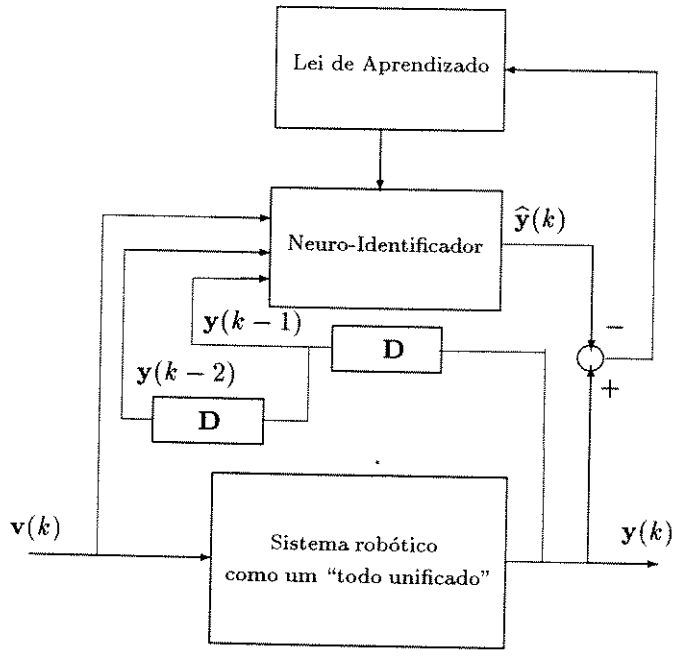

Figura 5.3: Neuro-Identificador para Sistemas Robóticos

5.2 Proposta do Neuro-Identificador

Com base na equação 3.6, e analisando-se a equação 4.17, notamos que a dimensão do vetor de entrada de um neuro-identifador para a modelagem dinâmica em relação à velocidade angular é $(2 \times m) + (1 \times l)$, como mostra o exemplo da figura 5.3.

Uma caracterítica importante deste neuro-identificador é que ele fornece a estimação da *Matriz de Sensibilidade*, como mostrado na seção 3.4.

No nosso caso, os atrasos de tempo existentes nos amplificadores de potência e nos sensores foram considerados desprezíveis. Caso isso não fosse possível, os atrasos teriam de ser levados em consideração na determinação da ordem do identificador.

5.3 Proposta do Neuro-Controlador

O controlador que adotaremos é o descrito pela equação 3.9. Para o aprendizado da sintonia é anexado um neuro-identificador, já previamente ajustado, para o fornecimento da matriz de sensibilidade como pode ser visto na figura 5.4.

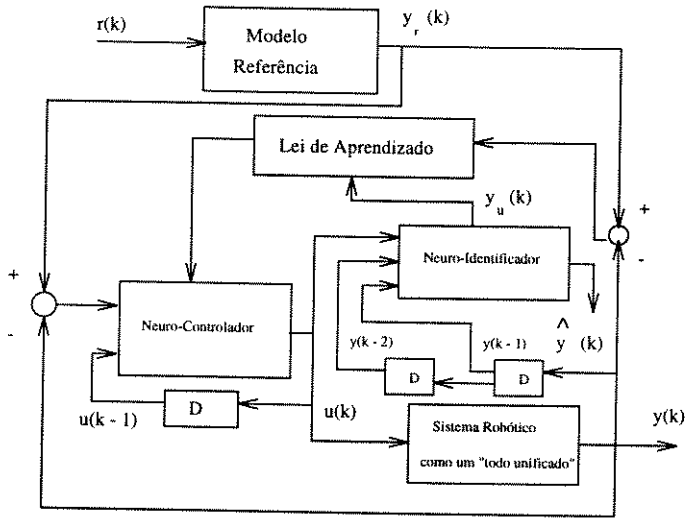

Figura 5.4: Neuro-Controlador para Sistemas Robóticos

5.4 Parte Experimental

Nesta seção abordaremos as simulações em computador digital e os resultados das mesmas. Os experimentos foram realizados em um computador tipo IBM-PC com processador Intel 486/66Mhz. As rotinas desenvolvidas foram feitas em *Borland* C++ para ambiemte *Windows*, ambos em versões 3.1.

5.4.1 Identificação Off-Line da Dinâmica Direta para o Espaço de Juntas

Este experimento foi desenvolvido conforme o exposto na seção 3.3. Foi feita a identificação de um modelo dinâmico multivariável para o sistema robótico. Os vetores $\mathbf{u}(k)$, $\mathbf{y}(k-1)$ e $\mathbf{y}(k-2)$, de entrada para o identificador da figura 5.3, pertencem ao $\mathbb{R}^2$. A primeira componente do vetor $\mathbf{u}$ contempla o sinal de excitação da primeira junta, e a segunda componente o sinal de excitação da segunda junta. No vetor $\mathbf{y}$, a primeira componente corresponde à velocidade angular da primeira junta, sendo a velocidade angular

da segunda junta alocada na segunda componente no vetor. Os sinais estão dispostos na entrada do identificador na sequência $[\mathbf{u}(k)\ \ \mathbf{y}(k-1)\ \ \mathbf{y}(k-2)]'$.

O experimento foi desenvolvido como um procedimento off-line onde o sistema foi excitado com sinais apropriados e colhidos os sinais de saída, obtendo-se assim os pares $(\mathbf{u}, \mathbf{y})$ a serem utilizados na identificação. Em seguida, esses dados foram utilizados no ajuste, por aprendizado, no neuro-identificador. Ao conjunto de pares $(\mathbf{u}, \mathbf{y})$ chamaremos de *perfil de treinamento.*

A **RNA** utilizada é do tipo **DRNN** com aprendizado pelo gradiente recursivo. Possui seis entradas, cinco neurônios na primeira camada, e dois na segunda, sendo que a saída do primeiro neurônio representa a saída identificada para a primeira junta, e a saída do segundo neurônio representa a saída identificada para a segunda junta. Os *bias* foram tomados constantes e iguais a 0.2. Em nossa rotina, desenvolvida com orientação a objeto, **OOD**, a rede é criada com as condições iniciais para os parâmetros $w^0_{2_{\max}}$, $w^{f^0}_{1_{\max}}$ e $x^0_{\max}$. Para o gradiente descendente dinâmico α é definido pelo usuário e $\widehat{\eta}$ automaticamente determinado através das equações 2.41 e 2.47. Os pesos das diversas camadas são inicializados com valores aleatoriamente distribuidos no intervalo $(-1; 1) \subset \mathbb{R}$. A tabela 5.5 apresenta um resumo desses valores (veja o algoritmo 2.1).

Na figura 5.5 podemos observar os sinais de excitação utilizados. Estes sinais são do tipo **RBS** (*Random Binary Signal*) e incorrelatos, conforme sugerido por Ljung [39]. Os sinais tipo **RBS** possibilitam a excitação da planta com uma faixa de frequência bastante ampla. Na figura 5.6 são apresentados os sinais de saída do sistema. O intervalo de amostragem foi tomado 10 milisegundos e o tempo de duração do experimento 10 segundos.

O ajuste do identificador foi realizado com a aplicação por dez vezes do perfil de treinamento sobre o identificador, sendo os resultados considerados satisfatórios. Na figura 5.7 temos os sinais de saída da planta e do identificador sobrepostos.

Na figura 5.8 são vistas as variações no tempo das componentes da matriz de sensibilidade das saídas do sistema em relação às entradas, obtidas a partir do identificador e do perfil de treinamento. Note que a da matriz $\boldsymbol{\lambda}_0$ da **RNA** utilizada no identificador pertence ao $\mathbb{R}^{2\times 6}$. A matriz de sensibilidade é obtida como sub-matriz de $\boldsymbol{\lambda}_0$ em relação ao vetor de entrada $\mathbf{u}$ da rede. Assim, a composição da matriz de sensibilidade em relação às componentes de $\boldsymbol{\lambda}_0$ é da forma:

$$\widehat{\mathbf{y}}_u = \begin{bmatrix} \lambda_{0_{11}} & \lambda_{0_{12}} \\ \lambda_{0_{21}} & \lambda_{0_{22}} \end{bmatrix}$$

Na figura 5.9 temos o comportamento médio da função objetivo da saída do identificador após cada passagem pelo perfil de treinamento. Os valores foram obtidos a

Parâmetro	*Valor*	*Característica*
N	6	Fixo e definido pelo usuário
M_1	5	Fixo e definido pelo usuário
M_2	2	Fixo e definido pelo usuário
x_0	0.2	Fixo e pré-definido internamente
y_{1_0}	0.2	Fixo e pré-definido internamente
α	0.75	Fixo e definido pelo usuário
$w^0_{2_{\max}}$	1	Adaptativo e pré-definido internamente
$w^{f^0}_{1_{\max}}$	1	Adaptativo e pré-definido internamente
$x^0_{\max}$	1	Adaptativo e pré-definido internamente
$\widehat{\eta}$		Adaptativo e determinado internamente

Tabela 5.5: Parâmetros da rede **DRNN** utilizada no experimento de identificação.

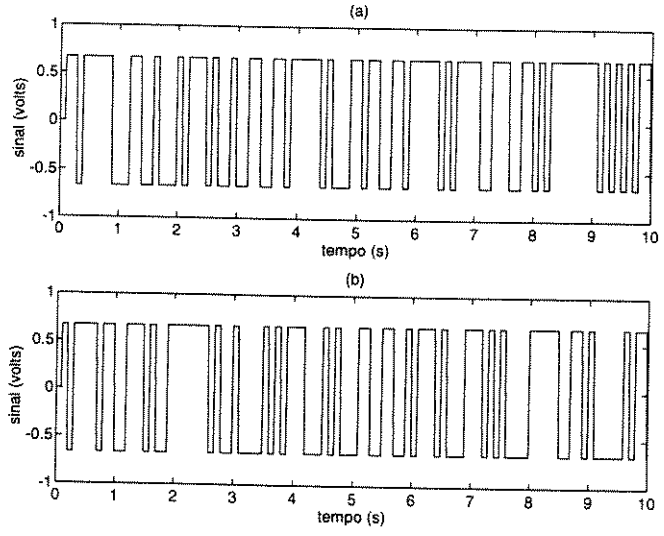

Figura 5.5: Sinais de excitação para identificação: a) sinal para junta um; b) sinal para junta dois.

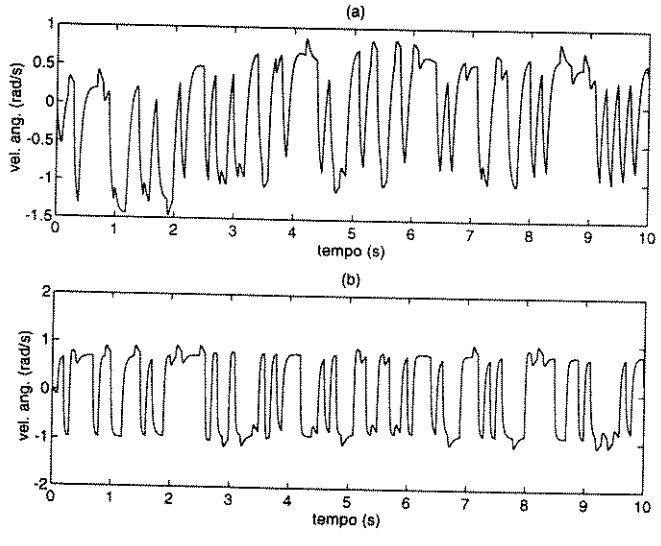

Figura 5.6: Sinais de saída de sistema robótico durante procedimento de identificação: a) saída da junta um; b) saída da junta dois.

partir das expressões:

$$J_{1_m} = \frac{1}{N} \sum_{n=0}^{N-1} J_1(k-n)$$

$$J_{2_m} = \frac{1}{N} \sum_{n=0}^{N-1} J_2(k-n)$$

$$J_m = \frac{1}{N} \sum_{n=0}^{N-1} J(k-n)$$

onde N é o número de amostras postas a treinamento.

Na figura 5.10 temos o comportamento das variáveis internas da **RNA** utilizada no identificador, $\mathbf{w}^0_{2_{max}}$, $\mathbf{w}^{f^0}_{1_{max}}$, $\mathbf{x}^0_{n_{max}}$ e $\widehat{\eta}$, após cada passagem pelo perfil de treinamento.

As matrizes $\mathbf{w}_2$, $\mathbf{w}_1$ e $\mathbf{w}^f_1$ tiveram a seguinte composição final:

$$\mathbf{w}_2 = \begin{bmatrix} -0.5312 & -1.5329 & 0.9955 & 0.5448 & -0.3717 & 0.4525 \\ 0.0659 & 0.1801 & 0.4062 & -0.1725 & 1.2349 & -1.5000 \end{bmatrix}$$

$$\mathbf{w}_1 = \begin{bmatrix} -0.2390 & -0.7016 & 0.4247 & -0.7558 & -0.0899 & -0.1831 & -0.0917 \\ 0.1957 & -0.0748 & 0.1781 & 0.6113 & -0.3347 & 0.7220 & 0.3424 \\ 0.3456 & 0.5356 & 0.0241 & 0.3349 & -0.1087 & 0.2687 & -0.0073 \\ -0.4573 & -0.1075 & 0.9727 & -0.2188 & 0.7851 & -0.0908 & -0.0279 \\ -0.0231 & 0.1715 & -0.9324 & -0.4284 & -0.8506 & 0.0600 & -0.1198 \end{bmatrix}$$

$$\mathbf{w}^f_1 = \begin{bmatrix} 0.0755 & 0 & 0 & 0 & 0 \\ 0 & 0.0068 & 0 & 0 & 0 \\ 0 & 0 & 0.0261 & 0 & 0 \\ 0 & 0 & 0 & 0.1375 & 0 \\ 0 & 0 & 0 & 0 & 0.1439 \end{bmatrix}$$

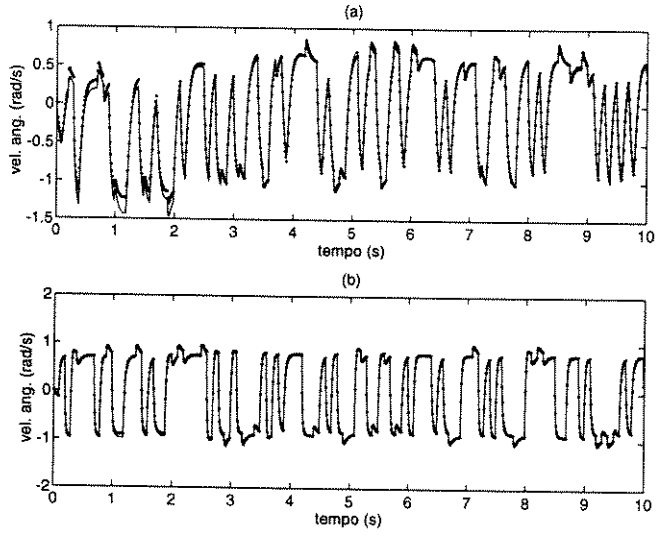

Figura 5.7: Sinais de saída de sistema robótico (linhas contínuas) e do identificador (linhas pontilhadas) sobrepostos como resultado do procedimento de identificação: a) sinais da junta um; b) sinais da junta dois.

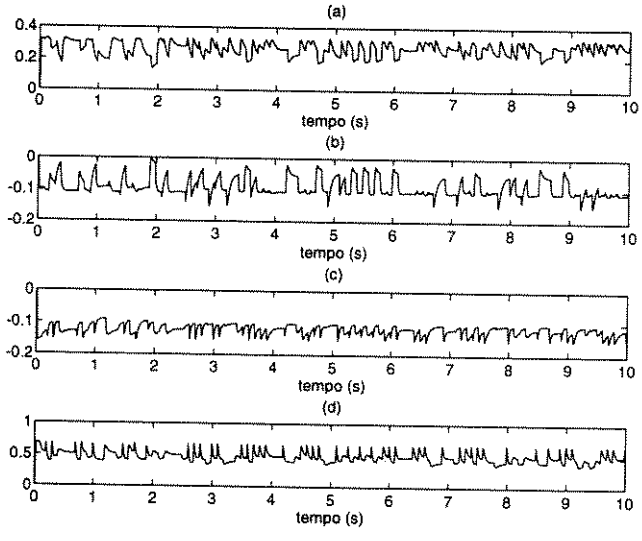

Figura 5.8: Componentes da matriz de sensibilidade (rad s^{-1}/volts): a) $\widehat{y}_{u_{11}}$; b) $\widehat{y}_{u_{12}}$; c) $\widehat{y}_{u_{21}}$; d) $\widehat{y}_{u_{22}}$.

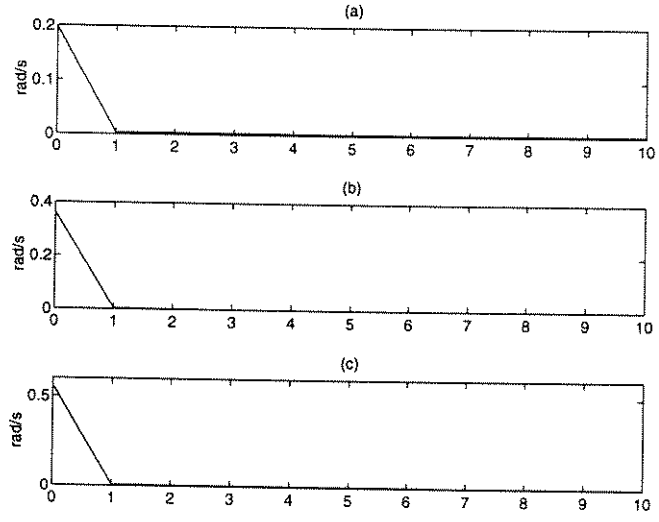

Figura 5.9: Comportamento médio da função objetivo durante o processo de treinamento do identificador: a) função J_{1_m}; b) função J_{2_m} c) função J_m.

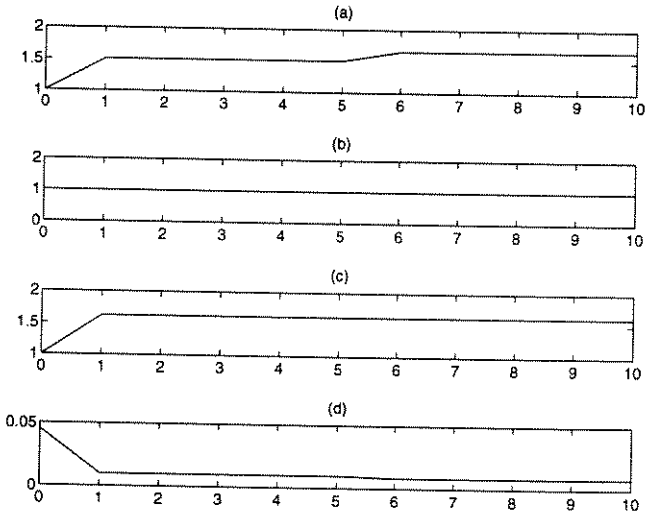

Figura 5.10: Comportamento dos parâmetros internos da **RNA** do identificador: a) $w^0_{2_{\max}}$; b) $w^{f^0}_{1_{\max}}$; c) $x^0_{\max}$; d) $\widehat{\eta}$.

Análise dos Resultados

Observando as figuras 5.7 e 5.9, vemos que o neuro-identificador proposto conseguiu incorporar a dinâmica do sistema de maneira bastante rápida, pois já após a primeira passagem pelo perfil de treinamento a função objetivo era desprezível em relação ao valor inicial.

Na figura 5.8 temos as componentes da matriz de sensibilidade da planta, obtidas a partir do neuro-identificador, como sub-matriz da matriz de propagação λ_0. Observe as componentes $\widehat{y}_{u_{12}}$ e $\widehat{y}_{u_{21}}$, que representam os acoplamentos existentes entre as juntas e possuem correspondência com os sinais $p_i(t)$ (equação 4.23). Essas componentes possuem valores negativos, estando assim de acordo com a equação 4.27.

A tabela 5.6 mostra os valores médios da matriz de sensibilidade da planta para os sinais gerados durante o processo de identificação e a figura 5.11 fornece uma interpretação gráfica desses dados. A interferência da entrada u_2 da junta dois sobre a junta um é de 33% de atuação resistiva ao movimento definido por u_1 e de 25% a da entrada u_1 da junta um em relação à junta dois. Observamos assim que o sistema é significativamente interativo.

Informações como essas, de sensibilidade, podem ser úteis para a tomada de decisões a níveis de projetos e para modifições de sistemas já existentes. Por exemplo, esses dados podem ser usados para decisões de escolha de estruturas de controle, seleção de atuadores, etc.

No caso de um sistemas robóticos, por exemplo, pode-se usar essas informações para auxílio à seleção de motores e à especificão de um fator de redução ótimo. Na seção 4.3.3 verificou-se que um valor elevado para o fator de redução de cada junta, N_i, é de grande importância na redução do acoplamento entre as variáveis do sistemas, permitindo assim uma linearização e o tratamento por juntas independentes. No entanto, o valor desse fator de redução não deve ser grande a ponto de reduzir significativamente a transferência de torque ou força, o que resultaria em desperdício de energia. Assim, através de substituições sucessivas de fatores de redução em sistemas robóticos reais e da avaliação da sensibilidade dos ramos cruzados entre as variaveis de junta, é possível a escolha de um fator de redução que mantenha compromisso entre redução de acoplamento e máxima transferência de torque.

Na figura 5.10 podemos observar o funcionamento do algoritmo adaptativo antecipativo às divergências para **RNA's**, cuja estrutura básica é apresentada no algoritmo 2.1. Podemos ver, através desta figura, que a taxa de aprendizado foi modificada algumas vezes, principalmente devido às variações de $w^0_{2_{max}}$.

Ramo direto $u_1 \rightarrow \widehat{y}_1$	$\widehat{y}_{u_{11}}$	0.27 rad s^{-1}/volts
Ramo cruzado $u_2 \rightarrow \widehat{y}_1$	$\widehat{y}_{u_{12}}$	- 0.09 rad s^{-1}/volts
Ramo cruzado $u_1 \rightarrow \widehat{y}_2$	$\widehat{y}_{u_{21}}$	- 0.12 rad s^{-1}/volts
Ramo direto $u_2 \rightarrow \widehat{y}_2$	$\widehat{y}_{u_{22}}$	0.48 rad s^{-1}/volts

Tabela 5.6: Valores médios da matriz de sensibilidade do sistema robótico

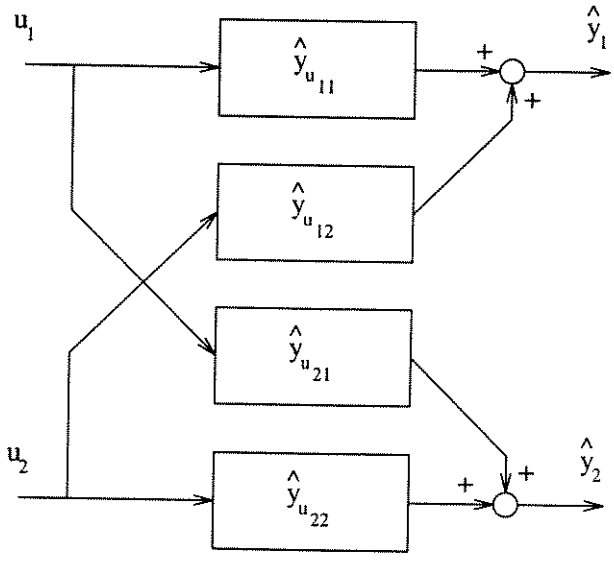

Figura 5.11: Interpretação gráfica da matriz de sensibilidade

5.4.2 Sintonia On-Line do Neuro-Controlador para o Espaço de Juntas

Este experimento foi desenvolvido com a utilização do controlador apresentado na seção 5.3. Foi utilizado um controlador independente, **SISO**, para cada junta. Foram realizados novos experimentos de identificação para a obtenção de **SISO** para cada junta e anexação aos controladores. A sintonia ou ajuste do controlador foi feito como procedimento on-line em uma planta robótica real.

As **RNA's** utilizadas são do tipo **DRNN** com aprendizado pelo gradiente recursivo. Possuem duas entradas, cinco neurônios na primeira camada e um na segunda, que representa o incremento na saída do controlador. Os *bias* foram tomados constantes e iguais a 0.2. As nossa rotina, desenvolvida com metodologia **OOD** criam as mesmas condições que apresentamos no procedimento de identificação, e incluindo também o parâmetro $y^0_{u_{max}}$ do algoritmo 3.1, que no caso foi tomada inicialmente igual a 0.5. A tabela 5.7 apresenta um resumo desses valores (veja o algoritmo 3.1).

O modelo referência adotado foi de segunda ordem, na forma:

$$Y_i(s) = \frac{\omega_{n_i}^2}{s^2 + 2\,\zeta_i\,\omega_{n_i}\,s + \omega_{n_i}^2}\,R_i(s)$$

onde foram tomados $\zeta_i = 0.707$ e $\omega_{n_i} = 8$ Hz (veja na tabela 5.8 as constantes de tempo do sistema).

O experimento foi realizado por um período de 30 segundos com tempo de amostragem de 10 milisegundos. Foram aplicados aos controladores sinais de referência na forma:

$$\begin{cases} r_1(t) = 0.3\,\sin(2\,t) + 0.3\,\sin(3\,t) + 0.4\,\sin(4\,t) \\ r_2(t) = -\,exp(-\,t) + 0.3\,\cos(t) + 0.3\,\cos(3\,t) + 0.4\,\cos(4\,t) \end{cases}$$

Os sinais de referência foram aplicados simultâneamente aos controladores e após cada ação de controle eram feitos os treinamentos dos controladores. As figuras 5.12 e 5.13 mostram, respectivamente para as juntas um e dois, o comportamento do sistema durante os 30 segundos.

Nas figuras 5.14 e 5.15 temos o comportamento dos parâmetros $w^0_{2_{max}}$, $w^{f^0}_{1_{max}}$, x^0_{max} e $y^0_{u_{max}}$, internos às redes. Na figura 5.16 observa-se as variações da taxa de aprendizado, $\widehat{\eta}$. Na figura 5.17 temos o comportamento das funções objetivo dos dois controladores.

As matrizes $\mathbf{w}_2$, $\mathbf{w}_1$ e $\mathbf{w}_1^f$ para o controlador da junta um tiveram a seguinte composição final:

$$\mathbf{w}_2 = \begin{bmatrix} 0.0301 & 2.0551 & 0.0508 & -0.8425 & -0.2602 & 1.2799 \end{bmatrix}$$

Parâmetro	*Valor*	*Característica*
N	2	Fixo e definido pelo usuário
M_1	5	Fixo e definido pelo usuário
M_2	1	Fixo e definido pelo usuário
x_0	0.2	Fixo e pré-definido internamente
y_{1_0}	0.2	Fixo e pré-definido internamente
α	0.75	Fixo e definido pelo usuário
$w^0_{2_{\max}}$	1	Adaptativo e pré-definido internamente
$w^{f^0}_{1_{\max}}$	1	Adaptativo e pré-definido internamente
$x^0_{\max}$	1	Adaptativo e pré-definido internamente
$\widehat{y}^0_{u_{\max}}$	0.67	Adaptativo e pré-definido internamente
$\widehat{\eta}$		Adaptativo e determinado internamente

Tabela 5.7: Parâmetros da rede **DRNN** utilizada no experimento de sintonia do neuro-controlador.

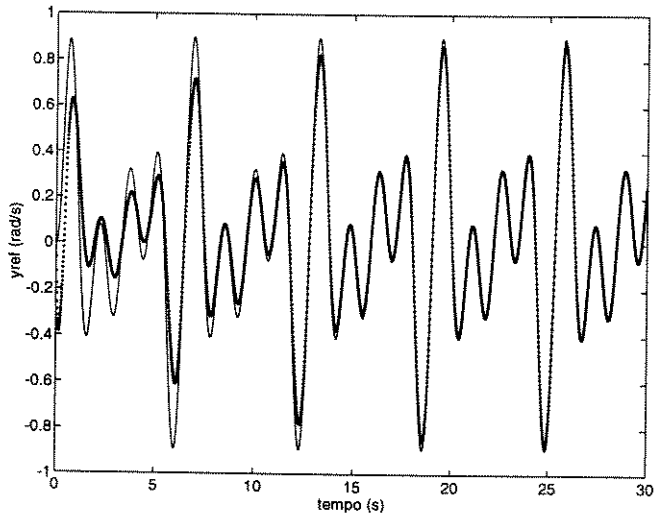

Figura 5.12: Sinal esperado para a saída da junta 1, y_r (linha contínua), e sinal real na saída, y (linha pontilhada), sobrepostos como resultado do procedimento de sintonia do controlador.

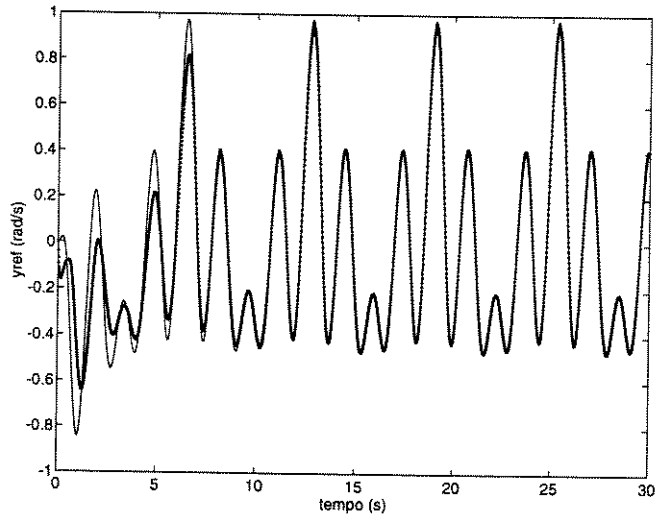

Figura 5.13: Sinal esperado para a saída da junta 2, y_r (linha contínua), e sinal real na saída, y (linha pontilhada), sobrepostos como resultado do procedimento de sintonia do controlador.

$$\mathbf{w}_1 = \begin{bmatrix} 0.0491 & 1.6820 & 1.0726 \\ -0.4944 & -0.1533 & 0.1711 \\ 0.3355 & -0.8117 & -0.4138 \\ -0.5676 & -0.0022 & -0.5267 \\ -0.0129 & 1.0264 & 0.7917 \end{bmatrix}$$

$$\mathbf{w}_1^f = \begin{bmatrix} 0.4586 & 0 & 0 & 0 & 0 \\ 0 & 0.0003 & 0 & 0 & 0 \\ 0 & 0 & 0.0843 & 0 & 0 \\ 0 & 0 & 0 & 0.0053 & 0 \\ 0 & 0 & 0 & 0 & 0.1881 \end{bmatrix}$$

As matrizes $\mathbf{w}_2$, $\mathbf{w}_1$ e $\mathbf{w}_1^f$ para o controlador da junta dois tiveram a seguinte composição final:

$$\mathbf{w}_2 = \begin{bmatrix} -0.3058 & 1.9831 & 0.4715 & -0.9745 & 0.0506 & 1.2510 \end{bmatrix}$$

$$\mathbf{w}_1 = \begin{bmatrix} 0.5257 & 1.5623 & 1.0669 \\ -0.4996 & 0.3330 & 0.4073 \\ -0.0145 & -0.9856 & -0.4650 \\ -0.4791 & 0.4064 & -0.3626 \\ 0.2964 & 0.9755 & 0.7791 \end{bmatrix}$$

$$\mathbf{w}_1^f = \begin{bmatrix} 0.4497 & 0 & 0 & 0 & 0 \\ 0 & 0.0215 & 0 & 0 & 0 \\ 0 & 0 & 0.1144 & 0 & 0 \\ 0 & 0 & 0 & 0.0001 & 0 \\ 0 & 0 & 0 & 0 & 0.1855 \end{bmatrix}$$

Análise dos Resultados

Analisando-se as figuras 5.12 e 5.13 podemos observar que o algoritmo de aprendizado é capaz de realizar a sintonia dos controladores de maneira rápida, visto que em aproximadamente 15 segundos os mesmos já estam bem sintonizados.

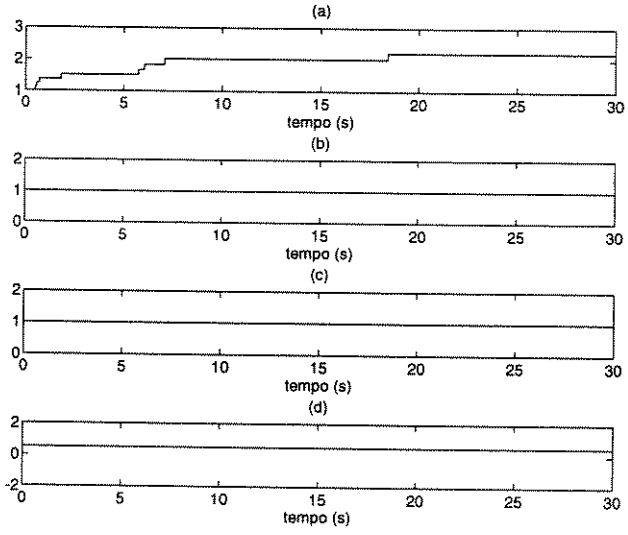

Figura 5.14: Comportamento dos parâmetros internos da **RNA** do controlador da junta 1: a) $w^0_{2_{\max}}$; b) $w^{f^0}_{1_{\max}}$; c) $x^0_{\max}$; d) $\widehat{y}^0_{u_{\max}}$.

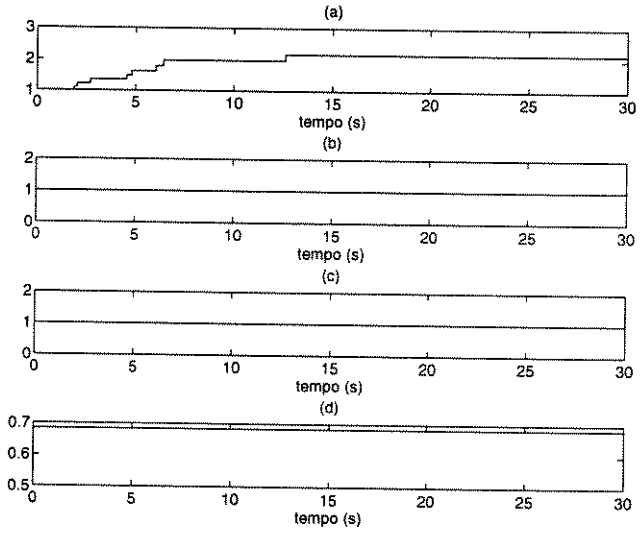

Figura 5.15: Comportamento dos parâmetros internos da **RNA** do controlador da junta 2: a) $w^0_{2_{\max}}$; b) $w^{f^0}_{1_{\max}}$; c) $x^0_{\max}$; d) $\widehat{y}^0_{u_{\max}}$.

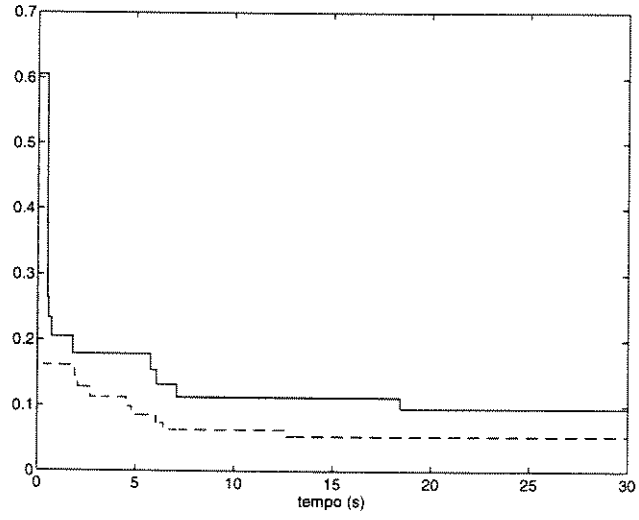

Figura 5.16: Comportamento das taxas de aprendizado dos controladores das juntas 1 (linha contínua) e 2 (linha tracejada).

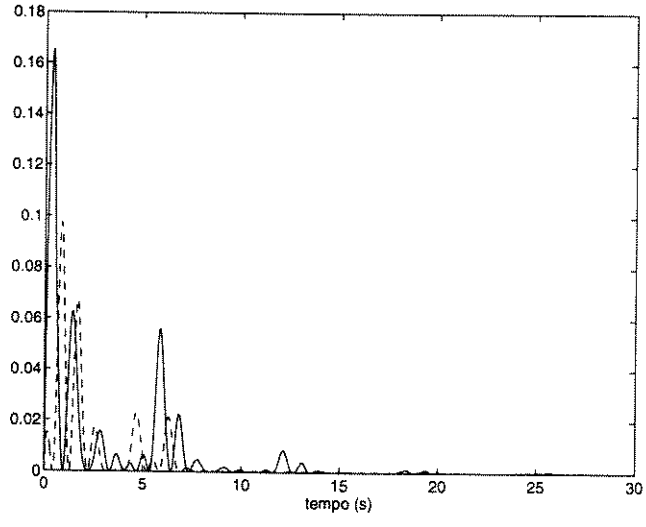

Figura 5.17: Comportamento das funções objetivos dos controladores das juntas 1 (linha contínua) e 2 (linha tracejada).

Nas figuras 5.14, 5.15 e 5.16 observamos a eficiência do algoritmo adaptativo antecipativo às divergências aplicado aos controladores. A taxas de aprendizado, $\widehat{\eta}_i$, foram modificadas várias vezes, principalmente devido às variações do parâmetro $w^0_{2_{max}}$.

Na figura 5.17 observamos o comportamento das funções objetivos dos dois controladores. As mesmas foram rapidamente minimizadas.

5.4.3 Comparação entre o Neuro-Controlador e o Controlador PI

Este experimento foi desevolvido utilizando-se os neuro-controladores sintonizados na seção anterior e o modelo de controlador **PI** apresentado na seção 4.3.3.

Foram utilizados dois procedimentos diferentes. No primeiro a planta foi posta sob o controle dos neuro-controladores e no segundo sob controle dos controladores **PI**. Em ambos os casos o sistema foi excitado com os mesmos sinais de referência da seção anterior durante um período de 10 segundos com intervalos de amostragens de 10 milisegundos.

Os controladores **PI** foram sintonizados pelo método descrito na seção 4.3.3. Os parâmetros da planta, obtidos a partir dos dados e equações apresentados nas seções 5.1 e 4.3.3 (note a inclusão de amplificadores de potência como drives para os motores elétricos), em relação aos pontos $\theta_1 = \theta_2 = 0$, são apresentados na tabela 5.8. O modelo referência para os controladores **PI** foi o mesmo da seção anterior. Os parâmetros dos controladores são listados na tabela 5.9.

Nas figuras 5.18 e 5.19 podemos observar o comportamento dos dois tipos de controladores sobrepostos, respectivamente para as juntas um e dois.

constante de tempo da junta um	τ_{x_1}	0.205 s
ganho da junta um	K_{x_1}	1.26 rad s^{-1} / volts
constante de tempo da junta dois	τ_{x_2}	0.173 s
ganho da junta dois	K_{x_2}	1.26 rad s^{-1} / volts

Tabela 5.8: Parâmetros do sistema robótico utilizados para o projeto dos controladores **PI**.

ganho proporcional para junta um	K_{p_1}	1.05 volts/(rad s^{-1})
ganho integral para junta um	K_{i_1}	10.4 volts/rad
ganho proporcional para junta dois	K_{p_2}	0.76 volts/(rad s^{-1})
ganho integral para junta dois	K_{i_2}	8.8 volts/rad

Tabela 5.9: Parâmetros dos controladores **PI** do sistema robótico.

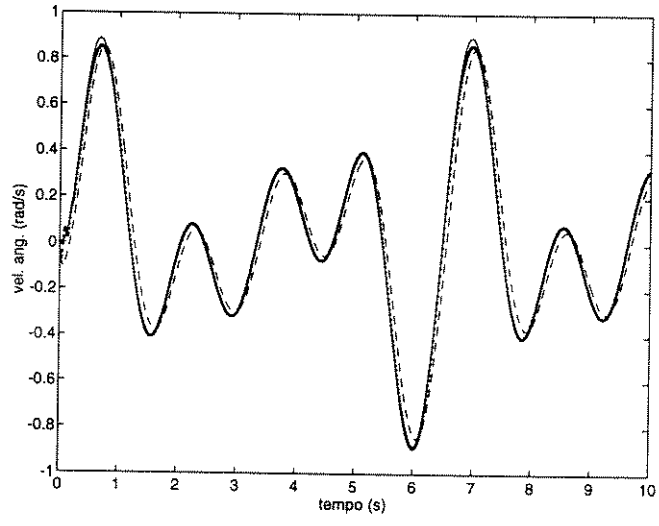

Figura 5.18: Sinal esperado para a saída da junta 1, y_r (linha contínua), e sinal de saída do neuro-controlador (linha pontilhada) e sinal de saída do controlador **PI** (linha tracejada) sobrepostos.

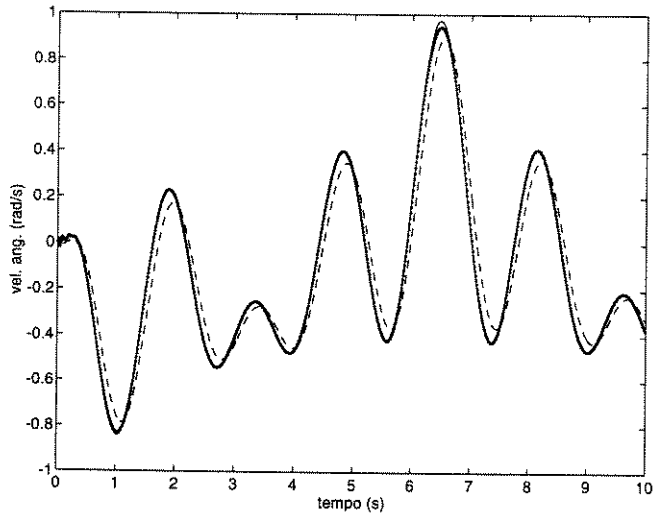

Figura 5.19: Sinal esperado para a saída da junta 2, y_r (linha contínua), e sinal de saída do neuro-controlador (linha pontilhada) e sinal de saída do controlador **PI** (linha tracejada) sobrepostos.

Análise dos Resultados

As figuras 5.18 e 5.19 mostram que para ambas as juntas o neuro-controlador conseguiu fazer um rastreamento assintótico mais eficiente que o controlador **PI**, que possui defasagem de rastreamento perceptível.

Os parâmetros do controlador **PI** são obtidos a partir de um modelo com parâmetros abstraído do sistema, tornando-se assim dependente e sensível à precisão do modelo e de seus parâmetros.

Os neuro-controladores tiveram suas sintonias feitas por aprendizado, sem a dependência direta a modelos abstraídos do sistema e de seus parâmetros, utilizando informações mínimas a respeito da ordem do sistema.

Ainda, tanto o neuro-controlador como o neuro-identificador podem ser adaptados a novas situações de funcionamento da planta sem a necessidade de modelagens específicas para essas situações.

5.5 Sumário

Neste capítulo foi feita as aplicações de **RNA**'s no controle de sistemas robóticos. Foram propostos um neuro-identificador e um neuro-controlador para este tipo de sistema. O método proposto foi aplicado sobre o modelo simulado de um robô de 2 **DOF**. Foram realizados três experimentos: identificação off-line; sintônia on-line do neuro-controlador; e comparação entre o neuro-controlador e o controlador **PI**.

O neuro-identificador foi capaz de incorporar rapidamente o modelo do sistema. Além disso, o mesmo permitiu a estimação da matriz de sensibilidade do processo como sub-matriz da matriz de propagação da **RNA**, λ_0.

O algoritmo de aprendizado para controladores também foi capaz de fazer uma sintônia rápida para o mesmo, com as figuras 5.12 a 5.17 mostrando o comportamento do sistema durante o procedimento de sintônia dos controladores. Na comparação entre os neuros controladores e os controladores tipo **PI** observamos que os neuro-controladores conseguiram fazer um rastreamento assintótico mais eficiente. Os parâmetros dos controladores **PI** foram obtidos a partir de um modelo abstraído do sistema, enquanto os neuro-controladores tiveram suas sintônias feitas por aprendizado, sem dependência direta à modelos abstraídos para o sistema.

Capítulo 6

Considerações Finais e Perspectivas Futuras

Com base nos objetivos delineados no prefácio e pelo exposto durante o texto podemos fazer as seguintes considerações:

1. Os resultados das simulações apresentadas no capítulo 5 mostraram que a solução proposta para o problema de controle de movimento de robôs é factível. A estrutura de controle sugerida, contendo identificador e controlador a base de aprendizado e de **RNA's**, foi capaz de realizar um controle eficiente, atendendo inclusive à especificações de desempenhos pré-estabelecidas. Além disso, a estrutura é de fácil utilização, requerendo do usuário pouco conhecimento teórico sobre seu funcionamento interno. A análise de estabilidade local, juntamente com o algoritmo adaptativo para o processo de aprendizado delegaram relativa confiabilidade ao sistema.
2. Os desenvolvimentos teóricos feitos no decorrer do texto com certeza fornecem os subsídios básicos necessários para a continuidade de trabalhos de pesquisas em sistemas por aprendizados e a **RNA's**, em robótica e em outras áreas. Foram tratados aspectos como modelagem, análise de estabilidade e aplicações.
3. O trabalho possibilitou o desenvolvimento de rotinas computacionais em linguagem C++, que permitem o desenvolvimento de pesquisas e aplicações de **RNA's**, de fácil utilização e confiáveis.

Como principais contribuições deste trabalho podemos apontar:

1. A elaboração de uma nova formulação, vetorial, para a classe *perceptrons* de **RNA's**, que permite facilmente a compreensão de como ocorre o processo de aprendizado, a

obtenção de informações a respeito de sensibilidade paramétrica da rede, e a verificação de que todo o processo de aprendizado é regido por uma lei de aprendizado baseada no gradiente descendente, fato que não pode ser observado com nitidez na formulação pelo delta generalizado (veja o apêndice B).

2. O estabelecimento de um critério para a análise de estabilidade local no aprendizado pelo gradiente e sua aplicação às **RNA's**.
3. Desenvolvimento de um algoritmo adaptativo para o treinamento de **RNA's** com base em um critério de estabilidade local, fornecendo assim relativa confiabiliade às suas aplicações, visto que o algoritmo modifica a taxa de aprendizado para evitar divergências no processo de treinamento das **RNA's**.
4. Proposta de uma estrutura de controle por aprendizado, utilizando **RNA's**, de fácil utilização e com potencialidade para ser aceito como um paradigma para soluções de vários problemas de controle em robôs, com possibilidade de extensão a outras áreas.

Como sugestões para trabalhos futuros podemos enumerar:

1. Comprovação em uma planta robótica real da funcionalidade da estrutura de controle e da metodologia proposta, em vista os elementos não modelados do sistema, e que existem em uma planta real. O plano é a implementação na planta robótica **JECA II**, existente e desenvolvida no *Laboratório de Sistemas Modulares Robóticos* da UNICAMP. Estuda-se ainda a possibilidade de elaboração de um sistema de gerenciamento e tomada de decisão para a ativação dos processos de adaptação dos neuro-controladores e neuro-identificadores.
2. Extensão da estrutura de controle proposta para a malha de controle de posição em um sistema robótico, visto esta não ser **BIBO** estável em malha aberta, onde pode ser tentado uma estabilização por aprendizado, utilizando-se **RNA's** [36, 37].
3. Estudo mais aprofundado da lei de aprendizado pelo gradiente descendente, ou até mesmo de outras leis, que levem em conta, por exemplo, não somente o erro mas também a energia consumida, com o objetivo de garantir a convergência a um ponto de mínimo global.
4. Estudo do comportamento caótico das redes neurais, tendo em vista que os modelos discretos não lineares e de ordem elevada apresentam conportamentos caóticos dependendo da região de operação selecionada [5, 6, 10].

Apêndice A

Incapacidade de RNA's de Única Camada em Representar a Função Lógica XOR

Este apêndice tem o objetivo de mostrar que uma rede de uma única camada não tem a capacidade de mapear a função lógica **XOR**. Isto foi provado originalmente por Minsky e Papert [44] em relação à rede de única camada de Rosenblatt [61].

A rede de Rosenbratt é baseada no modelo de neurônio de McCulloch e Pitts [41]. Assim, consideremos inicialmente o neurônio de McCulloch e Pitts da figura 1.1, com duas entradas, x_1 e x_2, e o *bias*, μ. A função de ativação é do tipo threshold. As entradas x_1 e x_2 possuem um comportamento binário. A tabela A.1 mostra-nos o comportamento entrada saída que deve ser mapeado pelo sistema (não := -1 e sim := 1).

x_1	x_2	y
-1	-1	-1
-1	1	1
1	-1	1
1	1	-1

Tabela A.1: Tabela verdade da função **XOR**

O modelo matematico do sistema, para $\mathbf{w} = [w_1 \quad w_2]'$ e $x = [x_1 \quad x_2]'$ é:

$$y(k) = g\left(\mathbf{w}'\,\mathbf{x}(k) - \mu\right)$$

onde:

$$g(k) = \begin{cases} 1 & \text{se } (\mathbf{w}'\,\mathbf{x}(k) - \mu) > 0 \\ -1 & \text{se } (\mathbf{w}'\,\mathbf{x}(k) - \mu) < 0 \end{cases}$$

Assim, com aplicação da tabela verdade ao sistema teremos:

$$w_1 + w_2 < \mu \tag{A.1}$$

$$-w_1 - w_2 < \mu \tag{A.2}$$

$$w_1 - w_2 > \mu \tag{A.3}$$

$$-w_1 + w_2 > \mu \tag{A.4}$$

Combinando as equações A.1 e A.4 nós obteremos $w_1 > 0$, enquanto que combinando A.2 e A.3 obteremos o oposto, $w_1 < 0$, o que é um absurvo.

Mais detalhes sobre essa prova pode ser encontrado tambem em Hertz *et. all.* [24] e Anderson e Rosenfeld [8].

Apêndice B

RNA's: Formulação pela Regra dos Deltas Generalizados

Este apêndice tem por objetivo apresentar a formulação para a classe *perceptrons* de **RNA's** com base na regra do delta generalizado proposta originalmente por Rumelhart *et. all.* [64]. Faremos a apresentação, sem perda de generalidade, para uma rede de duas camadas.

Consideremos inicialmente uma rede com duas camadas como visto na figura 1.2, com dois neurônios na camada de saída, a segunda, três neurônios na primeira camada e quatro sinais de entrada. Supondo uma rede com L camadas, N entradas e cada camada possuindo M_l neurônios, com $l \in \mathbb{N}$, $i \in \mathbb{N}$ e $j \in \mathbb{N}$, será adotada a seguinte convenção:

- $y_{l,i}(k) \in \mathbb{R}$ é a saída do neurônio i pertencente à camada l, onde o índice i é a posição relativa do neurônio na camada e o índice l é a camada a que o neurônio pertence para $l = 1, 2, \ldots, L$ e $i = 1, 2, \ldots, M$;
- $x_j(k) \in \mathbb{R}$ é uma entrada da rede, onde o índice j é a posição relativa de uma dada entrada para $j = 1, 2, \ldots, N$;
- $w_{l,ij} \in \mathbb{R}$ é o peso que relaciona a entrada j com o neurônio i em uma camada l para $j = 1, 2, \ldots, N$, $i = 1, 2, \ldots, M$ e $l = 1, 2, \ldots, L$;
- $g_{l,i}(\cdot) \in \mathbb{R}$ é a função de ativação do neurônio i da camada l $i = 1, 2, \ldots, M$ e $l = 1, 2, \ldots, L$;
- $h_{l,i}(\cdot) \in \mathbb{R}$ é o sinal ponderado total na entrada da função de ativação do neurônio i da camada l para $i = 1, 2, \ldots, M$ e $l = 1, 2, \ldots, L$.

Supondo uma rede de duas camadas, $L = 2$, com N entradas, M_1 neurônios na primeira camada e M_2 neurônios na segunda camada, que também é a camada de saída, o sinal na saída do j-ésimo neurônio da primeira camada é dado por:

$$y_{1,j}(k) = g_{1,j}(h_{1,j}(k)) = g_{1,j}\left(\sum_{j=0}^{N} w_{1,jn}\, x_n(k)\right)$$

para $j = 1, 2, \ldots, M_1$.

Os sinais de saída dos neurônios da primeira camada são entradas para os neurônios da segunda camada. Assim, o sinal de saída para o i-ésimo neurônio da segunda camada é dado por:

$$y_{2,i}(k) = g_{2,i}(h_{2,i}(k)) = g_{2,i}\left(\sum_{j=1}^{M_1} w_{2,ij}\, y_{1,j}(k)\right)$$

$$= g_{2,i}\left(\sum_{j=1}^{M_1} w_{2,ij}\, \left(g_{1,j}\left(\sum_{n=0}^{N} w_{1,jn}\, x_n(k)\right)\right)\right)$$

para $i = 1, 2, \ldots, M_2$.

Ainda, definiremos um vetor $\mathbf{y}_l = [y_{l,1} \quad y_{l,2} \quad \cdots \quad y_{l,M_2}]$, com $\mathbf{Y}_l \in \mathbb{R}^M$, onde $y_{l,i}$ corresponde à saída individual do i-ésimo neurônio da camada l e um vetor $\mathbf{x} = [x_1 \quad x_2 \quad \cdots \quad x_N]$, com $\mathbf{X} \in \mathbb{R}^N$, onde x_n correspode a n-ézima entrada individual da rede.

Podemos então definir uma função de custo para o erro na saída da rede na forma:

$$J(k) = \frac{1}{2} \sum_{i=1}^{M_L} e_i(k) \tag{B.1}$$

onde $e_i = (y_i - y_{d_i})$ é o erro na saída do i-ésimo neurônio da camada de saída da rede durante o processo de aprendizado.

A função de custo do erro medida na saída do i-ésimo neurônio da segunda camada é portanto da forma:

$$J_i(W, k) = \frac{1}{2}\,(yd_i(k) - y_{2,i}(k))^2$$

$$= \frac{1}{2}\left(yd_i(k) - g_{2,i}\left(\sum_{j=1}^{M_1} w_{2,ij}\, \left(g_{1,j}\left(\sum_{n=0}^{N} w_{1,jn}\, x_n(k)\right)\right)\right)\right)^2$$

Deve ser observado que a minimização da equação B.2 depende apenas dos parâmetros $w_{l,ij}$. Agora será utilizado o algoritmo do gradiente para a atualização dos pesos da segunda camada. Temos que:

$$\begin{aligned}\Delta w_{2,ij}(k) &= -\eta \frac{\partial J_i(k)}{\partial w_{2,ij}} \\ &= \eta \left(yd_i(k) - y_{2,i}(k)\right) \frac{\partial g_{2,i}(k)}{\partial h_{2,i}} y_{1,j}(k)\end{aligned}$$

para $i = 1, 2, \ldots, M_2$ e $j = 1, 2, \ldots, M_1$.

Definindo-se uma variável *delta generalizado* índice i para a camada dois, na forma:

$$\delta_{2,i}(k) = \left(yd_i(k) - y_{2,i}(k)\right) \frac{\partial g_{2,i}(k)}{\partial h_{2,i}(k)}$$

teremos:

$$\Delta w_{2,ij}(k) = \eta\, \delta_{2,i}(k)\, y_{1,j}(k)$$

para $i = 1, 2, \ldots, M_2$ e $j = 1, 2, \ldots, M_1$.

Agora, para a aplicação do gradiente aos pesos da primeira camada será necessário a utilização da regra da cadeia. Assim, temos que:

$$\begin{aligned}\Delta w_{1,jn}(k) &= -\eta \sum_{i=1}^{M_2} \left(\frac{\partial J_i(k)}{\partial w_{1,jn}} \right) \\ &= \eta \sum_{i=1}^{M_2} \left(\frac{\partial J_i(k)}{\partial y_{1,j}} \frac{\partial y_{1,j}(k)}{\partial w_{1,jn}} \right)\end{aligned}$$

$$\Delta w_{1,jn}(k) = -\eta \left(\sum_{i=1}^{M_2} w_{2,ij} \frac{\partial g_{2,i}(k)}{\partial h_{2,i}} \left(yd_i(k) - y_{2,i}(k)\right) \right) \frac{\partial g_{1,j}(k)}{\partial h_{1,j}} x_n(k)$$

$$\Delta w_{1,jn}(k) = \eta\, \delta_{1,j}(k)\, x_n(k)$$

para $j = 1, 2, \ldots, M_1$ e $n = 0, 1, 2, \ldots, N$.

O *delta generalizado* para a primeira camada fica então definido como:

$$\delta_{1,j}(k) = \frac{\partial g_{1,j}(k)}{\partial h_{1,j}(k)} \left(\sum_{i=1}^{M_2} w_{2,ij}\, \delta_{2,i}(k) \right)$$

Deve-se notar que equação B.2 e equação B.2 possuem a mesma forma, porém, com diferentes definições para os *deltas generalizados* ($\delta_{1,j}$).

A expressão geral para atualização dos pesos é então definida como:

$$w_{l,ij}(k) = w_{l,ij}(k) + \Delta w_{l,ij}(k) \tag{B.2}$$

Assim, podemos resumir de maneira genérica, passo a passo, a construção de uma rede neural, *perceptron*, considerando uma rede com L camadas e usando $y_{l,i}$ como saída do i-ésimo neurônio da l-ésima camada.

1. Definem-se as dimensões da entrada, do número de camadas, bem como da quantidade de neurônios em cada camada. Inicializam-se os pesos com valores pequenos e aleatórios;
2. Fazendo-se aqui uma mudança de variável, de x_j para $y_{0,j}$ ($l = 0$), temos que a expressão geral para a propagação do sinal através da rede é:

$$y_{l,i}(k) = g_{l,i}(h_{l,i}(k)) = g_{l,i}\left(\sum_j w_{l,ij}\, y_{(l-1),j}(k)\right) \quad l = L, L-1, \ldots, 1. \tag{B.3}$$

3. A computação dos *deltas generalizados* para a camada de saída é dada por:

$$\delta_{L,i}(k) = (yd_i(k) - y_{L,i}(k))\, \frac{\partial g_{L,i}(k)}{\partial h_{L,i}(k)} \tag{B.4}$$

4. A computação dos *deltas generalizados* para as outras camadas é dada por:

$$\delta_{l,i}(k) = \frac{\partial g_{l,i}(k)}{\partial h_{l,i}(k)} \left(\sum_j w_{(l+1),ji}\, \delta_{(l+1),j}(k)\right) \quad l = L-1, \ldots, 1. \tag{B.5}$$

5. As quantidades de alterações dos pesos são dadas por:

$$\Delta w_{l,ij}(k) = \eta\, \delta_{l,i}\, y_{(l-1),j}(k) \quad l = L, L-1, \ldots, 1. \tag{B.6}$$

6. A atualização dos pesos é dada por:

$$w_{l,ij}(k+1) = w_{l,ij}(k) + \Delta w_{l,ij}(k) \quad l = L, L-1, \ldots, 1. \tag{B.7}$$

Observe que esta formulação, pelo *delta generalizado* não facilita a observação de sensibilidade em relação aos parâmetros ajustáveis da redes, necessários a aplicação do critério de estabilidade expresso pela equação 2.12. A computação da variável *delta generalizado*, equações B.4 e B.5, embutem sempre o erro na saída da rede, fazendo com que o processo de aprendizado aparente não ser baseado em uma lei pelo gradiente, equação B.6.

BIBLIOGRAFIA

[1] J. S. Albus. Data storage in the celabellar model articulation controller (cmac). *Trasactions of the ASME, Jornal of Dynamic Systems, Measurement, and Control*, pages 228–233, September 1975.

[2] J. S. Albus. A new approach to manipulator control: The celabellar model articulation controller (cmac). *Trasactions of the ASME, Jornal of Dynamic Systems, Measurement, and Control*, pages 220–227, September 1975.

[3] L. B. Almeida. A learning rule for asynchronous perceptrons with feed-back in a combinatorial environment. In M. Caudill and C. Butler, editors, *In IEEE First International Conference on Neural Networks (San Diego 1987)*, volume 2, pages 609–618. New York. IEEE, 1987.

[4] L. B. Almeida. Backpropagation in perceptrons with feedback. In R. Eckmiller and Ch. von der Malsburg, editors, *In Neural Computers (Neuss 1987)*, pages 199–208. Berlin: Spreinger Verlag, 1988.

[5] A. I. M. Álvarez. *Estabilidade e Caos em Sistemas Dinâmicos não Lineares: Aplicação no Sistema* **PLL**-*Dual.* PhD thesis, Faculdade de Engenharia Elétrica, Universidade Estadual de Campinas, SP, Brasil, 1994.

[6] A. I. M. Álvarez and A. G. B. Palhares. Chua's circuit with a discontinuos nonlinearity. *Journal of Circuits, Systems and Computer*, 3:231–237, 1993.

[7] J. A. Anderson. A memory model using spatial correlation functions. *Kybernetik*, (5):113–119, 1968.

[8] J. A. Anderson and E. Rosenfeld. Neurocomputing: Foundations of research. The MIT Press, Cambridge, Massachusetts, 1988.

[9] K. J. Åström and B. Wittenmark. Adaptive control. Addison-Wesley, New York, 1989.

[10] A. Barton. Two-dimensional moviment controlled by a caotic neural networks. *Automatica*, 31(8):1149–1155, 1995.

[11] M. G. Bello. Enhanced training algoritms, and integrated training/architecture selection for multilayer perceptron networks. *IEEE Transaction on Neural Networks*, 3(6):864–875, 1992.

[12] G. E. P. Box and G. M. Jenkens. Times series analysis, forecasting and control. Holden-Day, San Francisco, 1970.

[13] R. P. Brent. Fast training algorithms for multilayer neural nets. *IEEE Transaction on Neural Networks*, 2(3):346–354, 1991.

[14] S. Bryson and Y. C. Ho. Applied optimal control. New York: Blaisdell, 1969.

[15] T. Campos. Connectionist modeling for arm kinematics using visual information. *IEEE Transaction on Systems, Man, and Cybernetics - Part B:Cybernetics*, 26(1):89–99, 1996.

[16] C. Chen. Linear system theory and desing. Holt, Rinehart and Winston, Inc, 1984.

[17] X. Cui and K. G. Shin. Direct control and coordination using neural networks. *IEEE Transaction on Systems, Man, and Cybernetics*, 23(3):686–697, 1993.

[18] D. DeMers and K. Kreutz-Delgado. Canonical parameterization of excess motor degrees with self-organizing maps. *IEEE Transaction on Neural Networks*, 7(1):43–55, 1996.

[19] J. Denavit and R. S. Hartenberg. A kinematic notacion for lower pair mechanisms based on matrices. *Jornal of Applied Mechanics*, 77:215–221, June 1955.

[20] S. Dubowsky and D. T. DesForges. The application of model-referenced adasptive control to robotic manipulators. *Journal of Dynamic Systems, Measurement, and Control*, 101:193–200, 1979.

[21] Eletro-Craft Corporation. *DC Motors Speed Controls Servo System*, secund edition, 1973.

[22] M. Fukimi and S. Omatu. A new back-propagation algorithm with coupled neuron. *IEEE Transaction on Neural Networks*, 2(5):535–538, 1991.

[23] S. Grossberg. Nonlinear difference-differential equations in perdiction and learning theory. *Proceedings of the National Academy of Sciences, USA*, 59:368–372, 1967.

[24] J. Hertz, A. Krogh, and R. G. Palmer. Introduduction to the theory of neural computation. Addison-Wesley Publishing Company, 1991.

[25] K. J. Hunt, D. Sbarbaro, R. Żbikowski, and P. J. Gawthrop. Neural networks for control systems - a survey. *Automatica*, 28(6):1083–1112, 1992.

[26] W. T. Miller III. Real-time neural network control of a biped walking robot. *IEEE Control Systems Mag.*, pages 41–48, 1994.

[27] W. T. Miller III, R. P. Hewes, F. H. Glanz, and G. Kraft III. Real-time dynamic control of an industrial manipulator using a neural-networks-based learning controller. *IEEE Transaction on Robotics and Automation*, 6(1):1–9, 1990.

[28] R. Isermann. Digital control systems. Springer-Verlag Berlin, 1989.

[29] A. Ishiguro, T. Furuhashi, S. Okuma, and Y. Uchikawa. A neural network compensator for uncertainties of robotics manipulator. *IEEE Transaction on Industrial Electronics*, 39(6):565–569, 1993.

[30] R. A. Jacobs. Incleased rates of convergence through learning rates adaptation. *Neural Networks*, 1:295–307, 1988.

[31] M. Khalid, S. Omatu, and R. Yusof. Temperature regulation with neural networks and alternative control schemes. *IEEE Transactions on Neural Networks*, 6(3):572–582, 1995.

[32] T. Koehonen. An adaptive associative memory principle. *IEEE Transaction on Computers C*, 23:444–445, 1974.

[33] E. B. Kosmatopoulos, M. M. Polycarpou, M. A. Christodoulou, and P. A. Ioannou. High-order neural network structures for identification of dynamics systems. *IEEE Transaction on Neural Networks*, 6(2):422–431, 1995.

[34] C. C. Ku and K. Y. Lee. Diagonal recurrent neural networks for dynamic systems control. *IEEE Transaction on Neural Networks*, 6(1):144–156, 1995.

[35] C. S. G. Lee and C. L. Chen. Efficient mapping algorithms for sheduling robot inverse dynamics computation on a multiprocessor system. *IEEE Transaction on Systems, Man, and Cybernetics*, 20(3):582–595, 1990.

[36] A. U. Levin and K. S. Narendra. Control of nonlinear dynamics systems using neural networks: Controllability and stabilization. *IEEE Transaction on Neural Networks*, 4(2):192–206, 1993.

[37] A. U. Levin and K. S. Narendra. Control of nonlinear dynamics systems using neural networks - part ii: Observability, identification, and control. *IEEE Transaction on Neural Networks*, 7(1):30–42, 1996.

[38] F. L. Lewis, K. Liu, and A. Yesilderek. Neural net robot controller with guaranteed tracking performance. *IEEE Transaction on Neural Networks*, 6(3):703–715, 1995.

[39] L. Ljung. System indentification - theory for the user. Prentice Hall, Englewood Cliffs, NJ, 1987.

[40] L. Ljung and T. Södertrom. Theory and practice of recursive identification. Cambridge. MA: M.I.T. Press, 1985.

[41] J. L. McCulloch and W. Pitts. A logical calculus of ideas immanent in nervons activity. *Bulletin of Mathematical Biophysics*, (5):115–133, 1943.

[42] W. Messner, R. Horowitz, and W.-W. Kao. Exponential convergence of a lerning controller for robot manipulators. *IEEE Transaction on Automatic Control*, 36(2):188–197, 1991.

[43] W. Messner, R. Horowitz, and W.-W. Kao. A new adaptive learning rule. *IEEE Transaction on Automatic Control*, 36(7):890–894, 1991.

[44] M. L. Minsky and S. A. Papert. Perceptrons. Cambridge: MIT Press, 1969.

[45] K. S. Narendra. Adaptive and learning systems. Ed. New York: Plenum., 1986.

[46] K. S. Narendra and A. M. Annaswamy. Stable adaptive systems. Englewood Cliffs, NJ: Prentice-Hall, 1989.

[47] K. S. Narendra, J. Balakrishnam, and M. K. Ciliz. Adaptive and learning using multiple models, switching, and tuning. *IEEE Control Systems*, pages 37–51, June 1995.

[48] K. S. Narendra and K. Parthasarathy. Identification and control of dynamical systems using neural networks. *IEEE Transactions on Neural Networks*, 1(1):4–27, 1990.

[49] R. B. Newell and P. L. Lee. Applied process control - a case study. Prentice Hall, 1989.

[50] R. Ortega and M. W. Spong. Adaptive motion control of rigid robots: a tutorial. *Automatica*, 25(6), 1989.

[51] J. H. S. Osman and P. D. Roberts. A class of descentralizad tracking controller for robot manipulators. *Proc. Instn. Mech. Engrs. Part I*, 205:141–150, 1991.

[52] J. H. S. Osman and P. D. Roberts. Descentralizad and hierarchical control of a class of robot manipulators. *Proc. Instn. Mech. Engrs. Part I*, 208:139–148, 1994.

[53] D. C. Park, M. A. El-Sharkawi, and R. Marks II. An adaptively trained neural network. *IEEE Transaction on Neural Networks*, 2(3), 1991.

[54] D. B. Parker. Learnig logic. Technical Report TR-47, Center for Computacional Research in Economiics and Management Science, Massachusetts Institute of Technology, Cambridge, MA, 1985.

[55] F. J. Pineda. Generalization of back-propagation to recurrent neural networks. *Physical Review Letter*, 59:2229–2232, 1987.

[56] F. J. Pineda. Dynamics and architecture for neural computation. *Journal of Complexity*, 4:216–245, 1988.

[57] M. M. Polycarpou. Stable adaptive neural control scheme for nolinear systems. *IEEE Transaction on Automatic Control*, 41(3):447–451, 1996.

[58] M. M. Polycarpou and P. A. Ioannou. Identification and control of nolinear systems using neural networks models: Design and stability analysis. Technical Report 91-09-01, Dept. Elect. Eng. Systems, Univ. of Southern Cal., Los Angeles, 1991.

[59] M. M. Polycarpou and P. A. Ioannou. Learning and convergence analysis of neural-type structured networks. *IEEE Transaction on Neural Networks*, 3(1):39–50, 1992.

[60] R. S. Pressman. Software engineering: A practitioner's approach. Third Edition, McGraw-Hill, Inc., 1992.

[61] F. Rosenblatt. Principles of neurodynamics. New York: Spartan, 1962.

[62] G. A. Rovithakis and M. A. Christodoulou. Direct adaptive regulation of unknown nolinear dynamical systems via dynamic neural networks. *IEEE Transaction on Systems, Man, and Cybernetics*, 24(3):400–412, 1994.

[63] G. A. Rovithakis and M. A. Christodoulou. Direct adaptive regulation of unknown nolinear dynamical systems via dynamic neural networks. *IEEE Transaction on Systems, Man, and Cybernetics*, 25(12):1578–1594, 1995.

[64] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by back-propagating errors. *Nature*, (323):533–536, 1985.

[65] H. J. Sira-Ramírez and S. H. Zak. The adaptation of perceptrons with applications to inverse dynamics identification of unknown dynamic systems. *IEEE Transaction on Systems, Man, and Cybernetics*, 21(3):634–642, 1991.

[66] M. W. Spong and M. Vidyasagar. Robot dynamics and control. John Willy & Sons, 1989.

[67] T. J. Tarn, A. K. Bejczy, X. Yun, and Z. Li. Effect of motor dynammics on nonlinear feedback robot arm control. *IEEE Transaction on Robotcs and Automation*, 7(1):114–121, 1991.

[68] W. K. Taylor. Electrical simulation of some nervous system functional activities. In C. Cherry, editor, *In Information Theory (london 1985)*, pages 314–328. London: Butterworths, 1956.

[69] G.-J. Wang and C.-C. Chen. A fast multlayer neural-networks training algorithm based on the layer-by-layer optimizimg procedures. *IEEE Transaction on Neural Networks*, 7(3), 1996.

[70] P. Werbos. *Beyond regression: new tools for prediction and analysis in the behavionral sciences.* PhD thesis, Havard Universit, 1974.

[71] D. J. Willshaw, O. P. Buneman, and H. C. Longuet-Higgins. Non-holographic associative memory. *Nature*, (222):960–962, 1969.

[72] B. Windrow and M. E. Hoff. Adaptive switching circuits. In *IRE WESCON Convention Record*, pages 96–104. New York: IRE, 1960.

[73] H. Yu and S. Lloyd. Adaptive cantrol of robot manipulators including motor dynamics. *Proc. Instn. Mech. Engrs*, 209:207–217, 1995.

[74] S. M. Ziauddin and A. M. S. Zalzala. Model-based compensation of neural network controller for uncertainties of robotc arms. *IEE Proc. Control Theory Appl.*, 142(5):501–507, 1995.