

UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
COPPE

USO DE REDES NEURAIS NA ANÁLISE DE RESPOSTA
DINÂMICA DE ESTRUTURAS

ROSIMAR GUARIZE

USO DE REDES NEURAIS NA ANÁLISE DE RESPOSTA DINÂMICA DE
ESTRUTURAS

Rosimar Guarize

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS
PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA
UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS
REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE
EM CIÊNCIAS EM ENGENHARIA CIVIL.

Aprovada por:

Prof. Edison Castro Prates de Lima, D.Sc.

Prof. Luis Volnei Sudati Sagrilo, D.Sc.

Prof. Marcos Queija de Siqueira, D.Sc.

Dr. Isaias Quaresma Masetti, D.Sc.

Prof. Gilberto Bruno Ellwanger, D.Sc.

RIO DE JANEIRO,RJ - BRASIL
MARÇO DE 2004

GUARIZE, ROSIMAR

Uso de Redes Neurais na Análise
Dinâmica de Estruturas [Rio de Janeiro]
2004

XII, 81 p. 29,7 cm (COPPE/UFRJ,
M.Sc., Engenharia Civil, 2004)

Tese - Universidade Federal do Rio
de Janeiro, COPPE

1. Backpropagation
2. Gradiente descendente
3. Gradiente conjugado

I. COPPE/UFRJ II. Título (série)

Dedico aos meus pais,
Deusdete e Luzia,
e ao meu irmão
Rômulo.

Agradecimentos

É extremamente difícil colocar em palavras o meu sincero agradecimento a todas as pessoas que contribuíram para a realização dessa tese de mestrado.

Agradeço inicialmente a Deus, por te me dado coragem para seguir em frente, mesmo diante de tantas situações adversas.

Agradeço aos meus pais que inúmeras vezes abriram mão dos próprios sonhos para que eu pudesse realizar os meus.

Agradeço ao Professor Sagrilo, pelos alicerces deste trabalho, pela paciência, pelo apoio e por ter acreditado em mim, mesmo mediante a todas as dificuldades encontradas durante os créditos e no transcorrer desta dissertação.

Agradeço ao Professor Edison pelos indispensáveis ensinamentos.

Agradeço ao Sr. Luiz Eduardo, a Raquel e a Ana Maria, por terem me acolhido com carinho e viabilizado mais este passo em minha vida.

Agradeço a Rosilene pela amizade, paciência, e pelas profícuas discussões relacionadas ou não à nossa carreira acadêmica e profissional.

Agradeço ao Klaus Ole e a Mônica pelo apoio, auxílio e amizade em momentos cruciais.

Agradeço ao Fernando e a Clícia pela ajuda carinhosamente dispensada em momentos de pânico.

Agradeço a todas as pessoas que contribuíram direta ou indiretamente para a realização desta tese.

Muito obrigada!

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.).

USO DE REDES NEURAIIS NA ANÁLISE DE RESPOSTA DINÂMICA DE ESTRUTURAS

Rosimar Guarize

Março/2004

Orientador: Edison Castro Prates de Lima

Programa: Engenharia Civil

Este trabalho apresenta uma investigação sobre a utilização de redes neurais artificiais, especificamente os *Perceptrons* de Múltiplas Camadas, (MLP - *Multilayer Perceptron*), na identificação de respostas dinâmicas de estruturas. Através dos exemplos investigados verifica-se que há um grande potencial das redes neurais serem empregadas na análise de resposta de estruturas não-lineares. Um dos fatores mais importantes a ser considerado na modelagem é o tamanho dos atrasos das excitações.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfilment of the requirements for the degree of Mater of Science (M.Sc.).

THE USE OF NEURAL NETWORK IN THE ANALYSIS OF DYNAMIC
RESPONSE OF STRUCTURES

Rosimar Guarize

March/2004

Advisor: Edison Castro Prates de Lima

Department: Civil Engineering

This work presents an approach of the usage of artificial neural network, specifically the MLP - Multilayer Perceptron, in the identification of dynamic reponse of structures. Throughout the examples assessed it is possible to verify a large potential for neural networks to be applied in the analysis of response of non-linear structures. An important factor to be considered in the modelling is the size of the delays of the excitations.

Sumário

1	Introdução	1
2	Modelagem Estatística	5
2.1	Modelo NARMAX	10
3	Redes Neurais Artificiais	12
3.1	Introdução	12
3.2	Histórico	14
3.3	Definição de uma rede neural	17
3.3.1	Funções de ativação ou funções sigmóides	19
3.4	Tipos de Redes Neurais	21
3.5	Benefícios na Utilização de Redes Neurais	24
3.6	Aplicação de Redes Neurais	26
3.7	Aprendizado	27
3.7.1	Superfície de erro	28
3.7.2	Métodos de minimização dos erros	30
3.7.3	Mínimo local	35
3.8	Derivadas do Erro	36
3.8.1	Derivadas dos pesos sinápticos ligados a um nó de saída	37
3.8.2	Derivada dos pesos sinápticos em relação a um nó oculto	38
3.9	Generalização	40
3.10	Validação Cruzada	45
3.10.1	Limitando o número de nós ocultos	46

3.10.2	Interrompendo o treinamento	47
4	Aplicações	50
4.1	Modelo com um grau de liberdade	51
4.2	Análise de uma linha de ancoragem	66
5	Considerações Finais	80
5.1	Sugestões para próximos trabalhos	81
	Referências Bibliográficas	82

Lista de Figuras

2.1	Erro do modelo de regressão.	7
3.1	Neurônio.	13
3.2	Arquitetura de uma rede neural	17
3.3	Modelo matemático de um neurônio	18
3.4	Funções de ativação mais comuns em redes neurais.	20
3.5	Gráfico de uma rede <i>Feedforward</i>	22
3.6	Gráfico de uma rede <i>Feedback</i>	22
3.7	Representação esquemática da rede neural usada neste trabalho . . .	27
3.8	Superfície de erro.	29
3.9	Mínimo local e global.	36
3.10	Grafo de fluxo de sinal em uma rede neural.	37
3.11	Grafo de fluxo de sinal dos pesos sinápticos de um nó intermediário. .	39
3.12	Nível ótimo de generalização.	41
3.13	Dados ajustados adequadamente (boa generalização).	42
3.14	Dados ajustados adequadamente em excesso (generalização pobre). .	43
4.1	Sistema massa-mola avaliado.	51
4.2	Caso 1 - 25 nós ocultos e 50s de atraso. Erro igual a 1.68E-04.	54
4.3	Caso 1 - 25 nós ocultos e 15s de atraso. Erro igual a 6.61E-04.	54
4.4	Caso 2 - 25 nós ocultos e 40s de atraso. Erro igual a 1.57E-04	55
4.5	Caso 2 - 25 nós ocultos e 100s de atraso com erro igual a 6.96E-04 . .	55
4.6	Caso 3 - 25 nós ocultos e 50s de atraso. Erro igual a 1.14E-05.	56
4.7	Caso 3 - 25 nós ocultos e 20s de atraso com erro igual a 1.67E-04. . .	56

4.8	Caso 4 - 25 nós ocultos e 50s de atraso. Erro igual a 6.20E-04.	57
4.9	Caso 4 - 25 nós ocultos e 20s de atraso. Erro igual a 2.19E-03.	57
4.10	Erros encontrados para o Caso 1 variando-se o tempo de atraso (25 nós ocultos).	58
4.11	Erros encontrados para o Caso 2 variando-se o tempo de atraso (25 nós ocultos).	59
4.12	Erros encontrados para o Caso 3 variando-se o tempo de atraso (25 nós ocultos).	60
4.13	Erros encontrados para o Caso 4 variando-se o tempo de atraso (25 nós ocultos).	61
4.14	Erros mínimos obtidos para o Caso 1.	61
4.15	Erros mínimos obtidos para o Caso 2.	62
4.16	Erros mínimos obtidos para o Caso 3.	62
4.17	Erros mínimos obtidos para o Caso 4.	63
4.18	Caso 1 - Predição da resposta para uma realização das forças de excitação diferente da usada no treinamento.	64
4.19	Caso 2 - Predição da resposta para uma realização das forças de excitação diferente da usada no treinamento.	64
4.20	Caso 3 - Predição da resposta para uma realização das forças de excitação diferente da usada no treinamento.	65
4.21	Caso 4 - Predição da resposta para uma realização das forças de excitação diferente da usada no treinamento.	65
4.22	Séries de movimento de uma das realizações utilizadas - <i>surge</i> (m). . .	67
4.23	Séries de movimento de uma das realizações utilizadas - <i>sway</i> (m). . .	68
4.24	Séries de movimento de uma das realizações utilizadas - <i>heave</i> (m). . .	69
4.25	Janela da Serie de Movimentos de Baixa Freqüencia de uma das Re- alizacoes Utilizadas - <i>surge</i> e <i>sway</i> (m).	70
4.26	Janela da Serie de Movimentos de Alta Freqüencia de uma das Rea- lizacoes Utilizadas - <i>heave</i> (m).	71

4.27	Ajuste encontrado considerando 25 nós na camada oculta, 10s de atraso para <i>surge</i> e <i>sway</i> e 20s para <i>heave</i> . Erro igual a 1.00E-04. . .	72
4.28	Ajuste encontrado considerando 25 nós na camada oculta, 10s de atraso para <i>surge</i> e <i>sway</i> e 20s para <i>heave</i> . Erro igual a 1.00E-04. . .	72
4.29	Ajuste encontrado considerando 25 nós na camada oculta, 10s de atraso para <i>surge</i> e <i>sway</i> e 50s para <i>heave</i> . Erro igual a 6.00E-04. . . .	73
4.30	Ajuste encontrado considerando 25 nós na camada oculta, 10s de atraso para <i>surge</i> e <i>sway</i> e 50s para <i>heave</i> . Erro igual a 6.00E-04. . . .	73
4.31	Variação do erro na rede com base no tamanho dos atrasos.	74
4.32	Erros para a rede com atraso de 5s para <i>surge</i> e <i>sway</i>	74
4.33	Erros para a rede com atraso de 10s para <i>surge</i> e <i>sway</i>	75
4.34	Saídas da rede neural para a realização 2.	75
4.35	Saídas da rede neural para a realização 2.	76
4.36	Saídas da rede neural para a realização 3.	76
4.37	Saídas da rede neural para a realização 3.	77
4.38	Comportamento da rede para um conjunto de treinamento variável. .	77
4.39	Comportamento da rede para um conjunto de treinamento variável. .	78
4.40	Comportamento da rede para um conjunto de treinamento variável. .	78
4.41	Comportamento da rede para um conjunto de treinamento variável. .	79

Lista de Tabelas

4.1	Casos analisados para modelo com um grau de liberdade	52
4.2	Estatística das séries de movimento e resposta	67

Capítulo 1

Introdução

A análise dinâmica é uma ferramenta imprescindível no projeto de muitas estruturas encontradas em engenharia naval, *offshore*, aeronáutica, aeroespacial, etc... Em muitos destes casos, como por exemplo na engenharia *offshore*, além do comportamento estrutural ser dinâmico as excitações são de caráter aleatório.

A modelagem matemática para representar o comportamento físico de um problema estrutural dinâmico resulta em equações diferenciais (lineares ou não-lineares) dependentes do tempo. Na prática de engenharia estrutural a maneira mais eficiente para a solução de problemas de dinâmica tem sido o emprego do método dos elementos finitos, que resulta em resolver um sistema de equações diferenciais dependentes do tempo, CLOUGH & PIENZIEN (1975) [5]. No caso de sistemas de estruturas geométrica e fisicamente lineares a solução pode ser obtida de forma eficiente no domínio da frequência. Em muitos casos onde as não-linearidades envolvidas são significativa, a solução dos sistemas de equações diferenciais devem ser realizadas no domínio do tempo.

Em estruturas de engenharia *offshore*, tais como *risers* e linhas de ancoragem, os modelos matemáticos para a análise dinâmica tornam-se mais complexos com o aumento da lâmina d'água. Neste caso o tamanho da estrutura aumenta e, para manter uma discretização por elementos finitos adequada (ou um modelo

estrutural adequado) para a análise dinâmica, necessariamente o tamanho do sistema de equações diferenciais torna-se também maior. No caso de carregamentos aleatórios são necessárias análises dinâmicas com durações (tempo de integração) mais longas que os períodos típicos das fontes de excitação para se obter uma estabilidade estatística dos parâmetros de resposta da estrutura. Estes aspectos aliados à busca atual de petróleo em águas cada vez mais profundas tem tornado a análise dinâmica aleatória de estruturas *offshore*, tais como *risers* e linhas de ancoragem, cada vez mais cara do ponto de vista computacional. Em outras áreas da engenharia, tais como controle e automação, devido, entre outras, à dificuldade de se descrever o problema físico dinâmico por um modelo matemático mais rigoroso, se recorre a modelos matemáticos mais simples para descrever a resposta do sistema com comportamento dinâmico linear e não-linear. Esta linha de análise é comumente conhecida como Identificação de Sistemas, CASSINE & AGUIRRE [2].

A modelagem através de técnicas de identificação é uma abordagem experimental, que consiste no desenvolvimento de técnicas para estimação de modelos de sistemas dinâmicos através de dados experimentais, não exigindo conhecimento prévio do processo. A resposta dinâmica atual de um sistema é representada por um modelo matemático simples em função dos parâmetros de entrada dinâmicos do sistema, valores prévios da própria resposta e possíveis erros de medição presentes nos dados. Em identificação de sistemas é sempre necessário se obter um conjunto de entradas com as respectivas respostas de interesse para o desenvolvimento de um modelo matemático.

Na definição de um modelo matemático de um sistema físico com características dinâmicas deve ser considerado que o valor de suas saídas (respostas) no instante seguinte depende não somente das entradas atuais, mas também das entradas e estados passados. As não-linearidades presentes no problema contribuem para respostas dinâmicas mais complexas. O comportamento dinâmico não-linear de

sistemas dificulta a formulação de uma teoria sistemática e geral, aplicável a projetos de identificação e controle de sistemas. Entretanto, ultimamente algumas ferramentas tem sido propostas para identificação de sistemas, dentre as quais se destacam as redes neurais artificiais, [11] e o modelo NARMAX (*Nonlinear AutoRegressive Moving Average with eXogenous inputs*) [2].

Em sistemas dinâmicos, redes neurais artificiais e o modelo NARMAX podem ser vistos como dois casos de modelagem estatística que realizam um mapeamento não-linear das entradas e saídas passadas para a saída atual, [25]. O que os diferencia são as equações de aproximação, chamadas de regressores, e a forma com que as constantes (ou coeficientes) do modelo são obtidas.

O uso de esquema híbrido, obtenção de uma resposta dinâmica curta da estrutura através da análise dinâmica por elementos finitos aliado a um procedimento de identificação de sistemas para predição da resposta futura, pode ser uma alternativa interessante para a redução do custo computacional de análises dinâmicas de sistemas estruturais complexos. Dando início na busca deste sistema híbrido, este trabalho tem como principal objetivo fazer uma investigação sobre a possibilidade de se empregar redes neurais artificiais na modelagem da resposta de sistemas estruturais. A idéia é utilizar uma pequena simulação inicial da resposta, obtida por integração numérica, para treinar a rede neural e depois usar esta para predizer a resposta para um tempo maior.

No Capítulo II são apresentados alguns conceitos de modelagem estatística. No Capítulo III são apresentados os principais aspectos teóricos e práticos com relação as redes neurais artificiais. No Capítulo IV são apresentados duas aplicações de redes neurais em sistemas dinâmicos. O primeiro exemplo constitui-se de modelo não-linear simples com um grau de liberdade para entender melhor os conceitos envolvidos e o segundo constitui-se da resposta dinâmica, numa condição ambiental

extrema, de uma linha de ancoragem de um FPSO *Floating Production Storage and Offloading* instalado em águas profundas. No Capítulo V são apresentados as principais conclusões obtidas e sugestões de trabalhos futuros nesta linha de pesquisa.

Capítulo 2

Modelagem Estatística

O objetivo principal da modelagem estatística é encontrar uma equação que expresse apuradamente o modelo de um relacionamento.

Um exemplo protótipo de uma técnica de modelagem estatística é a regressão linear, cuja equação é:

$$y = a_0 + \sum_{i=1}^I a_i x_i \quad (2.1)$$

onde y é a variável dependente que se deseja estimar, I é o número de variáveis independentes x_i , a constante a_0 e os coeficientes a_i são parâmetros cujos valores são encontrados pela regressão.

A equação produzida por um método de modelagem pode ser entendida como um mapeamento, pois ela permite mapear qualquer ponto no espaço das variáveis independentes para um ponto no espaço das variáveis dependentes. A função de mapeamento definida pela equação 2.1 tem a forma linear. Caso existam duas variáveis independentes, então a equação de regressão define um plano em um espaço tridimensional (duas variáveis independentes e uma variável dependente). A estimativa da variável dependente para um dado exemplo é a coordenada y do ponto no plano que corresponde às coordenadas x_1 e x_2 do exemplo.

Quando existem mais de duas variáveis independentes não é possível visualizar a função de mapeamento geometricamente, mas é possível, entretanto, defini-la em termos geométricos. Se uma equação de regressão tem I variáveis independentes, a função de mapeamento define um hiperplano I -dimensional e os valores das variáveis independentes I terão pontos nesta superfície. O valor estimado da variável dependente é a altura deste ponto sobre o espaço das variáveis independentes em uma dimensão correspondente à variável dependente.

A variável dependente da função de mapeamento pode não ser igual ao resultado atual para alguns exemplos, o que significa que a função de mapeamento produz erros. Estes erros podem ser mensurados de várias formas. Uma forma simples para medir o erro é a média quadrática da diferença entre o valor estimado e o correto, para todos os exemplos. Em termos práticos, esta medida de erro concede maior importância aos exemplos para os quais o erro é maior. A equação para esta medida de erro é:

$$E = \frac{\frac{1}{2} \sum_{i=1}^N (y_i - t_i)^2}{N} \quad (2.2)$$

onde E é o erro, N é o número de exemplos para os quais o erro é medido, y_n é a predição da equação para o exemplo n , e t_n é o resultado real para o exemplo n . O fator 2 é simplesmente usado para facilitar na dedução das equações e não altera os resultados.

Um exemplo de como esse erro pode ser interpretado geometricamente é a apresentado na Figura 2.1. Nesta figura cada um dos N exemplos é um ponto definido pelos valores das variáveis independentes e dependentes para aquele exemplo. Para cada um desses pontos, o erro é o quadrado da distância do ponto até a superfície definida pela função de mapeamento, seja ela uma linha, plano, ou hiperplano I -

dimensional. Esta distância é medida ao longo de uma linha paralela ao eixo das variáveis dependentes (y), isso ocorre porque é mais importante conhecer o erro na estimativa da variável independente do que conhecer quanto uma variável independente deve ser mudada para eliminar o erro.

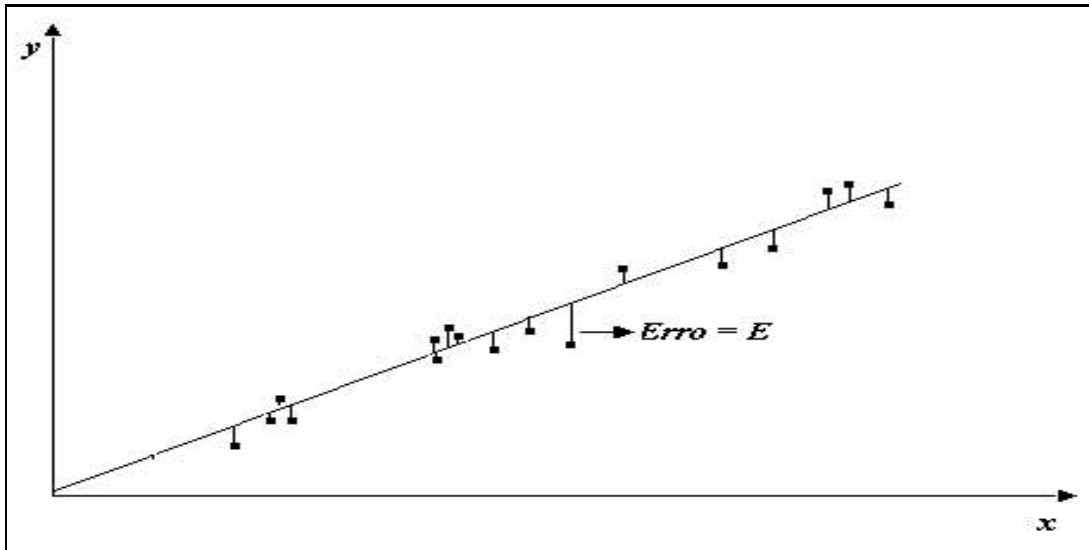


Figura 2.1: Erro do modelo de regressão.

Em aplicações práticas são encontradas duas fontes de erro. A primeira fonte é o ruído, que pode ser interpretado como falta de precisão nos dados, introduzida por instrumentos de medição, ou pelo fato das variáveis independentes não conterem todas as informações necessárias para determinar a variável dependente, e outros fatores ausentes no modelo e que desempenham um papel importante. O ruído não é um fator randômico inerente ou ausente de casualidades, e sim o efeito da falta de informações, ou de informações imprecisas.

A segunda fonte de erro refere-se ao fato da função de mapeamento não ter a mesma forma da função objetivo. O nome função objetivo (*target function*) significa uma função idealizada e desconhecida que expressa o verdadeiro relacionamento entre as variáveis independentes e dependentes.

No caso de uma regressão linear I -dimensional os coeficientes $a^T = (a_0, a_1, \dots, a_I)$ constituem a solução que minimiza o erro médio quadrático expresso pela equação 2.2. Matematicamente esta solução é obtida derivando-se esta equação com relação a cada um dos coeficientes e igualando-se a zero. Isto resulta no seguinte sistema de $I + 1$ equações.

$$\begin{aligned} \frac{\partial E}{\partial a_0} &= 0 \\ \frac{\partial E}{\partial a_1} &= 0 \\ &\vdots \\ \frac{\partial E}{\partial a_I} &= 0 \end{aligned} \tag{2.3}$$

Após algumas manipulações algébricas este sistema de equações acima pode ser re-escrito como:

$$\vec{A}\vec{a} = \vec{b} \tag{2.4}$$

onde os termos das matrizes \vec{A} e \vec{B} são dados por

$$\vec{A}_{ij} = \frac{\sum_{k=1}^N \vec{x}_i^k \vec{x}_j^k}{N}, i = 0, 1, \dots, I, j = 0, 1, \dots, I \tag{2.5}$$

$$\vec{B}_j = \frac{\sum_{k=1}^N \vec{x}_j^k \vec{y}^k}{N}, j = 0, 1, \dots, I \tag{2.6}$$

onde $\vec{x}_0^k = 1$, $\vec{x}_{1=0,1,\dots,I}^k$ e \vec{y}^k são os valores das variáveis independentes e dependente, respectivamente, do k -ésimo exemplo da amostra de dados. Desta forma, então, tem-se que:

$$\vec{a} = \vec{A}^{-1}\vec{b} \tag{2.7}$$

O fato de uma regressão linear definir linearmente a função de mapeamento pode limitar severamente sua precisão. Visando obter um modelo mais representativo, muitas vezes é necessário transformar as variáveis. Desta forma obtém-se o seguinte modelo de regressão:

$$y = a_0 + \sum_{I=1}^L a_I f_I(x_1, x_2, \dots, x_I) \quad (2.8)$$

onde L é o número de funções de transformação $f_I(x_1, x_2, \dots, x_I)$ empregadas. A equação 2.8 pode ser redefinida como:

$$y = a_0 + \sum_{i=1}^L a_i z_i \quad (2.9)$$

onde

$$z_i = f_i(x_1, x_2, \dots, x_I) \quad (2.10)$$

As equações 2.4 e 2.7 podem ser então utilizados para definir os coeficientes definidos pelas equações 2.8 ou 2.9.

A transformação de variáveis visando a linearização representada na equação 2.10, conhecida na literatura como regressão não-linear, teoricamente habilita a técnica de regressão a ser tão precisa quanto qualquer outro modelo estatístico. Não é uma tarefa simples automatizar o processo de decisão sobre qual forma a função de mapeamento deve assumir em função dos dados que estão sendo processados.

Em geral, a técnica de regressão tem um grande atrativo no que se refere à forma em que os coeficientes da equação de mapeamento são determinados, i.e., através da inversão de uma matriz, embora na maioria das vezes esta seja mal-condicionada [2].

Existem outros mapeamentos possíveis para representar a relação entre as variáveis dependentes e independentes, como por exemplo, mapeamentos baseados em funções logísticas [8]. Neste tipo de mapeamento, o conjunto de coeficientes, variáveis indeterminadas do mapeamento, é normalmente obtido através de um algoritmo numérico de otimização que busque a minimização do erro médio quadrático do mapeamento.

As regressões polinomiais NARMAX e as redes neurais artificiais, citadas no Capítulo I, são exemplos que se baseiam, respectivamente, em modelos de regressão não-linear e mapeamento baseado em funções logísticas. O mapeamento através de redes neurais artificiais, empregado neste trabalho, será comentado em maiores detalhes no Capítulo III. A seguir, a título de ilustração, será brevemente comentado o modelo polinomial NARMAX.

2.1 Modelo NARMAX

Os modelos NARMAX (*Nonlinear Auto Regressive Moving Average with eXogenous inputs*) foram desenvolvidos para modelagem de sistemas dinâmicos não-lineares discretos que e por sua vez são uma extensão do modelo ARMAX para o caso não-linear conforme pode ser visto LEONTARITIS & BILLINGS, (1985) [15], CHEN & BILLINGS, (1989),[3] e SJÖBERG (1995) [24].

Estes modelos realizam um mapeamento não-linear das entradas e saídas (respostas) passadas para a saída (resposta) atual e são capazes de representar uma ampla classe de sistemas não-lineares [1],[4].

Seja um sistema dinâmico com duas variáveis de entrada $u_1(t)$ e $u_2(t)$ e uma variável de saída $y(t)$. O modelo NARMAX representa a resposta y num instante de tempo t como:

$$y(t) = f(y(t - \Delta t), \dots, y(t - n_y \Delta t) u_1(t - \Delta t), \dots, \quad (2.11)$$

$$u_1(t - n_{u1} \Delta t), u_2(t - \Delta t), \dots,$$

$$u_2(t - n_{u2} \Delta t)) + \varepsilon(t)$$

onde Δt é o intervalo discreto de tempo, n_y , n_{u1} e n_{u2} representam, respectivamente, o tempo total dos atrasos considerados para a resposta $y(t)$; as entradas são $u_1(t)$ e $u_2(t)$; $f(\cdot)$ é a relação funcional do mapeamento e $\varepsilon(t)$ é o erro da modelagem. No modelo NARMAX polinomial, a função $f(\cdot)$ é representada pela expansão polinomial das variáveis entre parênteses. Representado os n termos entre parênteses por $x_{i,i=1,\dots,n}$ a equação 2.12 na forma polinomial é representada por:

$$y(t) = a_0 + \sum_{i=1}^n a_i x_i + \sum_{i=1}^n \sum_{j=1}^n a_{i,j} x_i x_j + \sum_{i=1}^n \sum_{j=1}^n \sum_{k=j}^n a_{i,j,k} x_i x_j x_k + \dots + \varepsilon(t) \quad (2.12)$$

Desprezando-se o termo do erro, os coeficientes desta equação podem ser então calculados pelo método de regressão não-linear, exposto anteriormente, a partir de um conjunto de dados de entradas e suas correspondentes saídas (respostas). Um modelo ideal seria aquele em que os atrasos considerados assim como a ordem da expansão polinomial conduzissem a um sinal de erro $\varepsilon(t)$ que representasse um ruído branco. Maiores detalhes acerca dos modelos NARMAX podem ser encontrados em CASSINE & AGUIRRE (1999) [2].

Capítulo 3

Redes Neurais Artificiais

3.1 Introdução

Redes Neurais Artificiais, RNA, são uma representação computacional de um sistema de processamento de informação com desempenho e características semelhantes às redes neurais biológicas, usando um grande número de elementos básicos interconectados, chamados neurônios artificiais.

Um computador é muitas vezes associado ao cérebro, e visando justificar tal associação surgiram na Inteligência Artificial dois paradigmas.

O primeiro paradigma é denominado simbolista, sendo fortemente influenciado pelos estudos realizados em psicologia. Tais estudos tiveram início em trabalhos pioneiros de MINSKY (1961) [19], MCCARTHY (1963) [17] e NEWELL & SIMON (1972) [21]. A abordagem simbolista coloca que a solução de problemas pode ser obtida através de um processo algorítmico. Estas idéias geraram, por exemplo, linguagens de programação simbólicas voltadas para aplicações em Inteligência Artificial.

O segundo paradigma, chamado conexionista, defende que é basicamente impossível reduzir a uma seqüência de passos lógicos e aritméticos, as inúmeras tarefas que o cérebro executa com facilidade e rapidez.

O conexionismo diz que a solução de tarefas complexas está fundamentada no modo de operação do sistema nervoso. Suas características básicas são a capacidade de aprendizado, generalização e adaptação. O princípio básico da aprendizagem está no ajuste dos pesos das sinápses.

O modelo de neurônio artificial baseia-se nos modelos biológicos. Ele é uma célula formada em sua maior parte por citoplasma e núcleo, dendritos e axônio.

Os dendritos são responsáveis por receber estímulos nervosos vindos de outros neurônios através de neuro-transmissores especiais.

O axônio é responsável por transmitir o estímulo gerado pelo neurônio aos demais que se relacionam com ele.

A Figura 3.3 ilustra um neurônio biológico.

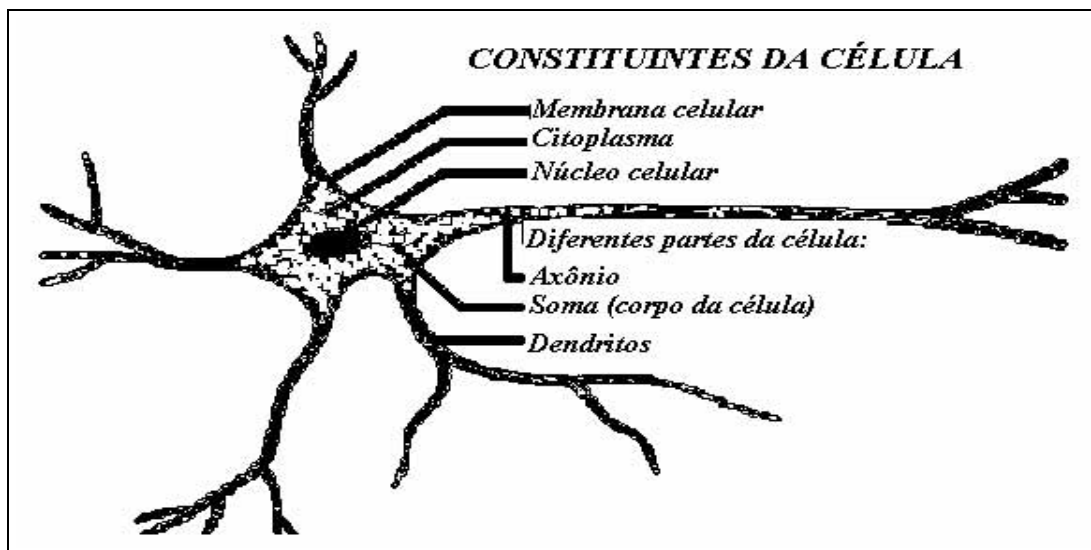


Figura 3.1: Neurônio.

Analogamente às redes biológicas, unidades de processamento das redes neurais

artificiais podem ser definidas para computação de determinadas funções matemáticas, geralmente não-lineares.

O formato da rede assemelha-se a um grafo direcionado, onde cada nó representa um neurônio e cada aresta possui pesos representando uma sinápsse. As sinápses são unidades estruturais elementares que medeiam as interações entre neurônios.

Cada neurônio recebe um vetor de valores de entrada (estímulo) multiplicado pelos respectivos pesos sinápticos, que é processado através de uma função de ativação do neurônio. O resultado é transmitido na saída, onde há sinápses com os outros neurônios.

Através de uma fase de aprendizagem, um conjunto de determinadas entradas é apresentado à rede, que extrai informações relacionadas ao ajuste dos pesos das sinápses, necessárias para generalizar o problema e gerar respostas coerentes para uma nova entrada futura.

3.2 Histórico

O primeiro modelo artificial de um neurônio biológico foi fruto do trabalho pioneiro de MCCULLOCH & PITTS (1943) [18]. Eles apresentaram uma discussão sofisticada de redes lógicas de nodos e novas idéias sobre máquinas de estados finitos, elementos de decisão de limiar lineares e representações lógicas de várias formas de comportamento e memória. Este modelo foi aprimorado posteriormente por ROSENBLATT (1960) [22] e WIDROW & HOFF (1960) [28]. Eles chamaram os elementos adaptativos resultantes de *perceptron* e *adaline*, respectivamente, cuja maior contribuição diz respeito à implementação de uma regra de aprendizado baseada na minimização de uma aproximação do erro quadrático médio, definido na saída do neurônio artificial, em relação a algum comportamento desejado de

entrada-saída; esta regra de aprendizado recebeu o nome de regra delta, que por sua vez é baseada no método do gradiente.

Demonstrou-se que com o novo modelo, o *perceptron*, que se fossem acrescentadas de sinápses ajustáveis, as redes neurais poderiam ser treinadas para classificar certos tipos de padrões. O *perceptron* simples possui três camadas: a primeira recebe as entradas do exterior e possui conexões fixas (retina); a segunda recebe impulsos da primeira através de conexões cuja eficiência de transmissão (peso) é ajustável e, por sua vez, envia saídas para a terceira camada (resposta). Este tipo de *perceptron* comporta-se como um classificador de padrões, e somente é capaz de classificar padrões que sejam linearmente separáveis.

Um grande arrefecimento no interesse pelo estudo de redes neurais artificiais ocorreu alguns anos mais tarde, quando MINSKY & PAPERT (1969) [20] demonstraram a capacidade limitada deste elemento adaptativo - o neurônio artificial - considerado isoladamente no mapeamento de operações lógicas. O *perceptron* não consegue detectar paridade, conectividade e simetria, que são problemas não-linearmente separáveis. O principal argumento de MINSKY & PAPERT (1969) [20] era de que o problema do crescimento explosivo do espaço ocupado e do tempo para solução de problemas complexos afetaria as redes neurais, inclusive os *perceptrons*, e que o algoritmo de aprendizado não garantia convergência para uma rede *perceptron* com mais de uma camada.

Este quadro começou a se reverter a partir do início dos anos 80, quando neurofisiologistas e biofísicos começaram a defender a idéia de que o desenvolvimento mais avançado da inteligência artificial demandaria o uso de estruturas de processamento paralelo e distribuído, capacidade de aprendizado e adaptação.

Entretanto, o ressurgimento efetivo das redes neurais artificiais é geralmente

associado à publicação do trabalho sobre a relação de redes neurais totalmente recorrentes auto-associativas e sistemas físicos, desenvolvido por HOPFIELD (1992) [9], e à apresentação do modelo de mapas auto-organizáveis, desenvolvido por KOHONEN (1982) [12].

Um forte impulso para o desenvolvimento de modelos conexionistas foi dado quando RUMELHART & MCCLELLAND (1986) [23] e LECUN (1987) [14] apresentaram o algoritmo de aprendizado de retropropagação (*Backpropagation*), indispensável para a viabilização do treinamento de redes neurais multicamadas, superando assim as limitações anteriormente apontadas por MINSKY & PAPERT (1969) [20].

Vale salientar que os resultados fundamentais empregados no desenvolvimento do algoritmo de retropropagação já haviam sido obtidos por WERBOS (1974) [27], em um outro contexto (fato que retardou seu aproveitamento junto a área de redes neurais artificiais).

Até o início dos anos 90, em muitos campos científicos, vários pesquisadores estavam entusiasmados com as possibilidades de aplicação das redes neurais artificiais, especialmente considerando o *perceptron* multicamadas (MLP - *Multilayer Perceptron*). A justificativa para este entusiasmo pode ser atribuída à obtenção de demonstrações da capacidade de aproximação universal do MLP.

Para um número adequado de neurônios na composição da camada intermediária da rede (aquela que não apresenta conexões diretas nem com as entradas nem com as saídas), com funções de ativação respeitando certas propriedades de suavidade, CYBENKO (1989b) [6] e HORNIK (1993) [10], dentre outros, demonstraram ser possível aproximar qualquer mapeamento contínuo definido em uma região compacta do espaço de aproximação.

Dependendo da complexidade do problema, os algoritmos de treinamento tradicionais para redes neurais artificiais apresentam-se computacionalmente ineficientes. Para contornar este problema, várias ferramentas de engenharia visando prover um aumento da eficiência dos algoritmos de treinamento destas redes têm sido propostas, como algoritmos genéticos e outras técnicas de computação evolutiva, e estratégias não-paramétricas de aproximação.

3.3 Definição de uma rede neural

Uma rede neural é um aproximador universal de funções que opera de acordo com o aspecto apresentado na Figura 3.2.

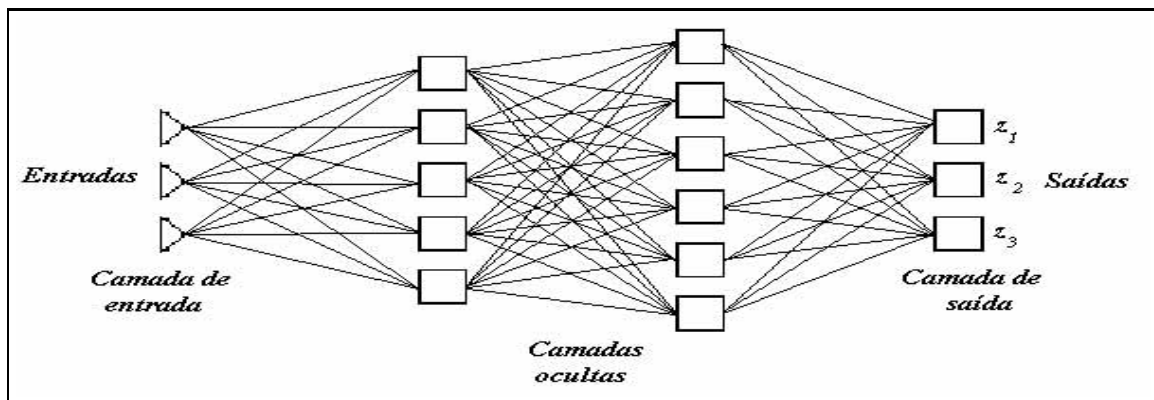


Figura 3.2: Arquitetura de uma rede neural

A camada de entrada é geralmente formada pelas n variáveis independentes do problema mais uma variável correspondente a uma perturbação ou *bias* (geralmente igual a 1.0). Todos os elementos da camada de entrada se comunicam com todos os neurônios nas camadas intermediárias, que por sua vez se comunicam com os neurônios da camada de saída. Os resultados da camada de saída são os resultados fornecidos pela rede.

As camadas intermediárias e a camada de saída são formadas por neurônios que operam como mostra a Figura 3.3.

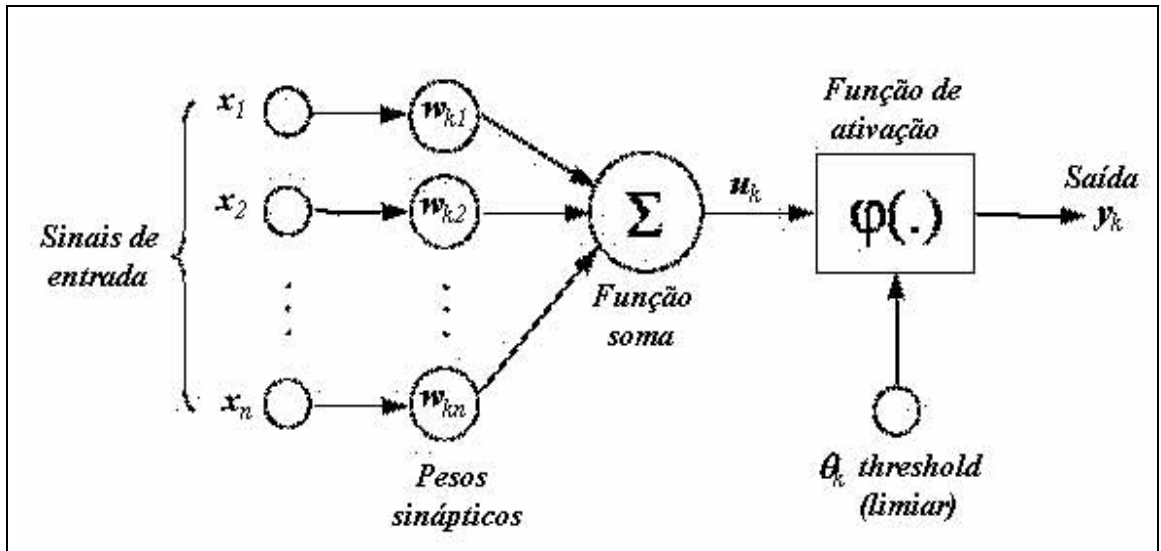


Figura 3.3: Modelo matemático de um neurônio

Um neurônio recebe informações x_i da camada de entrada ou da camada intermediária, dependendo da sua posição, multiplica-se estes neurônios por pesos sinápticos $w_{j,i}$, faz-se o somatório de todas as informações como apresentado na equação 3.1 a seguir e calcula o valor de saída y_j como sendo o valor da função de ativação $\varphi(.)$ avaliada em v_j .

$$v_j = \sum_{i=1}^{w+1} w_{j,i} x_j \quad (3.1)$$

Desta forma, considerando uma rede neural com uma camada de entrada com N variáveis independentes, uma camada intermediária com M neurônios e apenas uma saída Z , tem-se que:

$$z = \varphi\left(\sum_{j=1}^M w_j y_j\right) = \varphi\left(\sum_{j=1}^M w_j^{HO} \left(\varphi\left(\sum_{i=1}^{N+1} w_{j,i}^{IH} x_i\right)\right)\right) \quad (3.2)$$

onde w_j^{HO} são os pesos sinápticos da camada intermediária para a camada de saída e $w_{j,i}^{IH}$ são os pesos sinápticos entre as ligações da camada de entrada com a camada de saída.

A Equação 3.1 mostra qual é o mapeamento empregado pelas redes neurais para representar a função:

$$z = f(x_1, \dots, x_N) \quad (3.3)$$

Neste mapeamento, as variáveis desconhecidas são o número de camadas intermediárias, o número de neurônios em cada camada e os pesos sinápticos.

Demonstra-se matematicamente [8] que uma rede neural pode representar qualquer tipo de função matemática contínua.

3.3.1 Funções de ativação ou funções sigmóides

A função sigmóide demonstra a capacidade de uma rede neural de mostrar como a função que ela calcula pode assumir qualquer forma particular. As funções sigmóides são blocos básicos de construção com os quais a função de mapeamento é formada.

Para que uma função $s(u)$ seja sigmóide ela tem necessariamente que possuir certas características, como ser limitada, o que significa que seus valores $s(u)$ nunca excederão um determinado limite superior e outro inferior, independente do valor de u . O valor de uma função sigmóide sempre cresce quando o valor de u cresce e vice-versa, isto é, deve ser monotonicamente crescente. Uma função sigmóide é contínua e suave, entretanto possui curvaturas definidas em todos os pontos, e é

totalmente diferenciável.

Um número de diferentes funções tem essas características e, portanto, qualificam-se como funções sigmóide.

As funções mais usadas no mapeamento de funções contínuas são a tangente hiperbólica e a função logística mostradas na Figura 3.4.

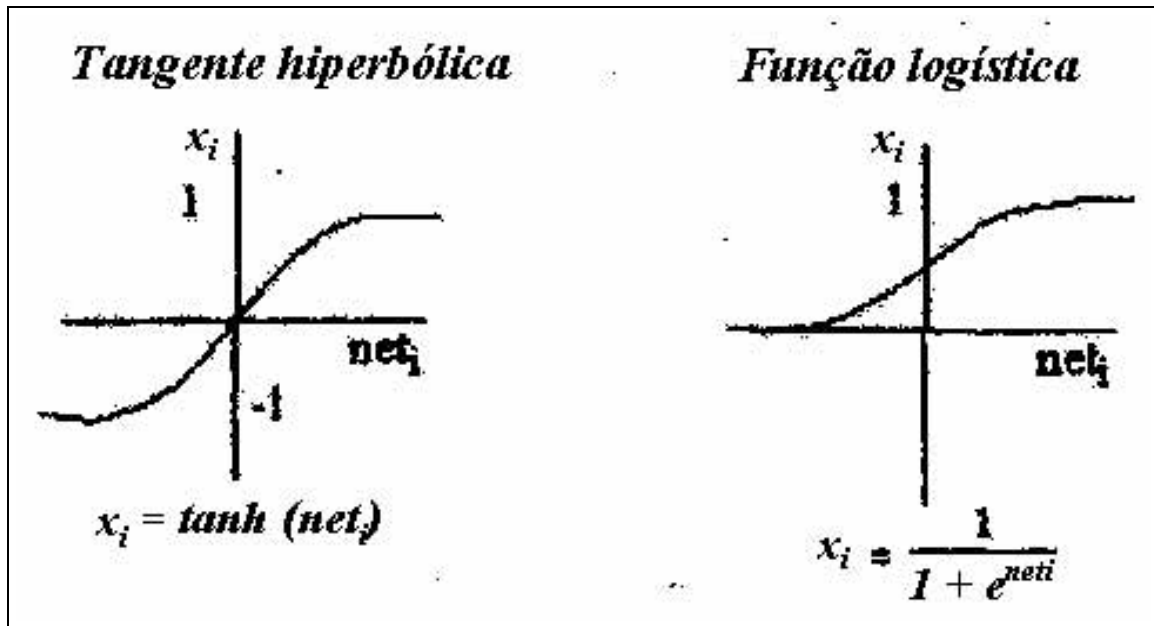


Figura 3.4: Funções de ativação mais comuns em redes neurais.

A função logística é definida por:

$$g(u) = \frac{1}{1 + \frac{1}{e^u}} = \frac{1}{1 + e^{-u}} \quad (3.4)$$

e é definida no intervalo $[0,1]$. A derivada desta função é dada por:

$$g'(u) = g(u)(1 - g(u)) \quad (3.5)$$

Uma função de ativação muito utilizada em redes neurais artificiais é a função tangente hiperbólica, que na sua forma mais geral é definida por:

$$g(u) = \tanh(u) \tag{3.6}$$

e é definida no intervalo $[-1,1]$. A derivada desta função é dada por:

$$g'(u) = 1 - g(u)^2 \tag{3.7}$$

3.4 Tipos de Redes Neurais

Existem diversos tipos de redes neurais, que podem ser categorizadas de acordo com sua orientação, estrutura, algoritmo de treinamento ou suas aplicações.

A orientação define como o grafo está orientado e pode ser:

- *Feedforward* ou acíclica: a saída de um neurônio não pode ser usada como entrada de outro neurônio em nenhuma camada anterior, um exemplo é mostrado na Figura 3.5.
- *Feedback* ou cíclica: a saída de um neurônio pode ser entrada de outros neurônios em camadas anteriores. Um exemplo é mostrado na Figura 3.6.

A estrutura define quantas camadas formam a rede, que podem ser:

- Redes de uma camada: possui apenas uma camada de neurônios entre a entrada e a saída da rede;
- Redes de múltiplas camadas: possui mais de uma camada de neurônios entre a entrada e saída.

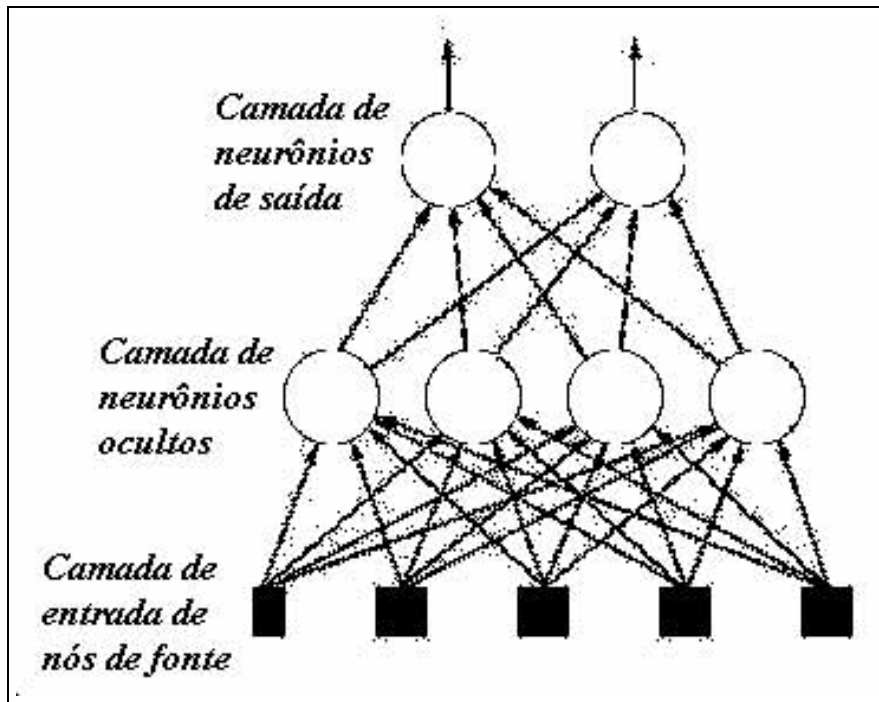


Figura 3.5: Gráfico de uma rede *Feedforward*.

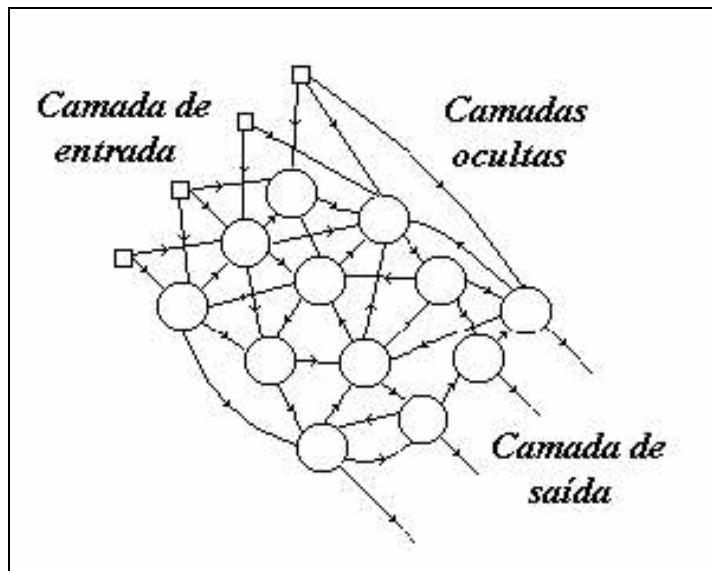


Figura 3.6: Gráfico de uma rede *Feedback*.

Neste trabalho será somente utilizado redes do tipo *feedforward* com uma única camada intermediária e um único nó de saída.

O algoritmo de treinamento representa o conjunto de procedimentos através dos quais os parâmetros da rede mudam de valor, ou seja, a maneira pela qual o ambiente influencia no comportamento da rede. Podem ser:

- **Aprendizado supervisionado:** as entradas e saídas são fornecidas e controladas por um agente externo. A forma de ajuste da rede é realizada através da correção de erros a partir da saída obtida na rede para uma dada entrada em comparação com a saída desejada; na verdade, a resposta desejada representa a ação ótima a ser adotada pela rede neural. Os parâmetros da rede são ajustados sob influência combinada do vetor de entrada e o sinal de erro. Com o aprendizado supervisionado, o conhecimento disponível ao agente externo é indiretamente transferido à rede neural. Os principais algoritmos de aprendizado supervisionado são a regra delta e o algoritmo de retropropagação.
- **Aprendizado por reforço,** SUTTON & BARTO, 1998 [26]: o comportamento da rede é avaliado simplesmente com base em algum critério numérico, fornecido em instantes espaçados de tempo. Este procedimento procura maximizar um índice escalar de desempenho chamado sinal de reforço, onde ocorre a avaliação indireta do comportamento, não fornecendo indicações se o melhoramento é possível, ou de como o sistema deverá proceder para melhor atender os objetivos.
- **Aprendizado não-supervisionado,** KOHONEN, 1997 [13]: apenas os padrões de entrada são fornecidos. Não há um exemplo específico da função a ser aprendida pela rede. A rede deve ser capaz de extrair informações e aprender a classificar genericamente os padrões recebidos. Uma vez que a rede conclui um processo de auto-organização baseado nos dados de entrada, ela explora a representação interna gerada para discriminar características presen-

tes na entrada. Os principais algoritmos de aprendizado não-supervisionado são aprendizado Hebbiano e aprendizado por competição.

3.5 Benefícios na Utilização de Redes Neurais

A utilização de redes neurais oferece as seguintes propriedades e capacidades:

- Não-linearidade: um neurônio artificial pode ser linear ou não-linear. Numa rede neural, constituída por conexões de neurônios não-lineares é ela mesma não-linear, a não-linearidade é distribuída por toda rede. A não-linearidade é extremamente importante, particularmente se o sinal de entrada for não-linear.
- Mapeamento de entrada-saída: um paradigma popular de aprendizagem chamado aprendizagem supervisionada, como apresentado acima, envolve a modificação dos pesos sinápticos de uma rede neural pela aplicação de um conjunto de amostras de treinamento rotuladas ou exemplos de tarefa. Cada exemplo consiste de um sinal de entrada único e de uma resposta desejada correspondente. Apresenta-se para a rede um exemplo escolhido aleatoriamente do conjunto, e os pesos sinápticos da rede são modificados para minimizar as diferenças entre a resposta desejada e a resposta real da rede, produzida pelo sinal de entrada, de acordo com um critério estatístico apropriado. O treinamento da rede é repetido para muitos exemplos do conjunto amostral até que a rede alcance estabilidade, onde não haja mais modificações significativas nos pesos sinápticos. Os exemplos de treinamento previamente aplicados podem ser reaplicados durante a sessão de treinamento, mas em uma ordem diferente. Assim, a rede aprende dos exemplos ao construir um mapeamento de entrada-saída para o problema considerado.
- Adaptabilidade: as redes neurais têm a capacidade inata de adaptar seus pesos sinápticos a modificações do meio ambiente. Em particular, uma rede neural treinada para operar em um ambiente específico pode ser facilmente retreinada para lidar com pequenas modificações nas condições operativas do

ambiente. Além disso, quando está operando em um ambiente onde as estatísticas mudam com o tempo, uma rede neural pode ser projetada para modificar seus pesos sinápticos em tempo real. A arquitetura natural de uma rede neural para classificação de padrões, processamento de sinais e aplicações de controle, aliada à capacidade de adaptação da rede, a torna uma ferramenta muito útil para classificação adaptativa de padrões, processamento adaptativo de sinais e controle adaptativo. Deve-se ressaltar que nem sempre a adaptabilidade resulta em robustez; um sistema adaptativo com constantes de tempo pequenas, por exemplo, pode se modificar rapidamente e assim responder a perturbações espúrias, causando uma drástica degradação no desempenho do sistema. Para aproveitar todos os benefícios da adaptabilidade, as constantes de tempo principais devem ser grandes o suficiente para que o sistema ignore perturbações espúrias, e suficientemente pequenas para responder a mudanças significativas no ambiente; o problema aqui descrito é referido como dilema da estabilidade-plasticidade , GROSSBERG, (1988b) [7].

- Resposta a evidências: no contexto de classificação de padrões, uma rede neural pode ser projetada para fornecer informações não somente sobre qual padrão particular selecionar, mas também sobre a confiança na decisão tomada, visando rejeitar padrões ambíguos, e com isso melhorar o desempenho de classificação da rede.
- Informação contextual: o conhecimento é representado pela própria estrutura e estado de ativação de rede neural. Cada neurônio na rede é potencialmente afetado pela atividade de todos os outros neurônios na rede.
- Tolerância a falhas: uma rede neural, implementada na forma física (*hardware*) tem o potencial de ser inerentemente tolerante a falhas, ou capaz de realizar computação robusta, no sentido que seu desempenho se degrada suavemente sob condições de operação adversas.
- Uniformidade de análise de projeto: as redes neurais desfrutam de universa-

lidade como processadores de informação. Os neurônios, de modo geral, representam um ingrediente comum a todas as redes neurais; essa uniformidade torna possível compartilhar teorias e algoritmos de aprendizagem em diferentes aplicações de redes neurais. Redes modulares podem ser construídas através de uma integração homogênea de módulos.

- Analogia neurobiológica: o projeto de uma rede neural é motivado pela analogia com o cérebro, que é uma prova viva de que o processamento paralelo tolerante a falhas não só é possível fisicamente como também rápido e poderoso.

3.6 Aplicação de Redes Neurais

A utilização de redes neurais é multidisciplinar e algumas áreas de grande aplicação são:

- Reconhecimento de padrões;
- Processamento de sinais;
- Processamento de imagens;
- Otimização combinatória;
- Processamento de dados imprecisos;
- Previsões financeiras;
- Controle de processos, etc.

No presente trabalho pretende-se utilizar as redes neurais para representar a resposta dinâmica de estruturas. Desta forma, o parâmetro de saída da rede será a resposta considerada num dado instante t e os parâmetros de entrada serão as fontes dinâmicas de excitação, tais como forças ou movimentos prescritos.

Como em dinâmica a resposta em um tempo t qualquer depende do passado (efeito de memória), um outro aspecto deve ser considerado na análise: o tempo de atraso da excitação.

Supondo genericamente as fontes de excitação como $x(t)$ e a resposta como $z(t)$ dadas por séries discretas, a representação esquemática das redes neurais utilizadas neste trabalho é mostrada na Figura 3.7.

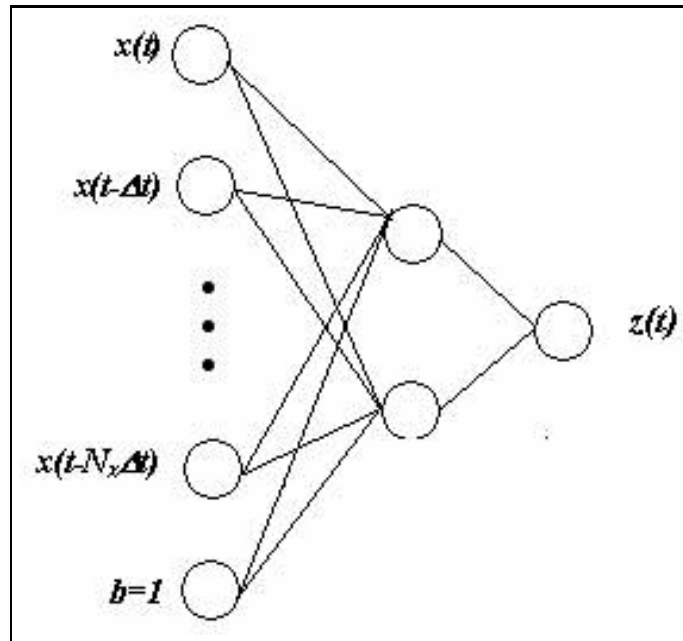


Figura 3.7: Representação esquemática da rede neural usada neste trabalho

Nesta figura $\Delta(t)$ é o intervalo de tempo discreto e N_x é o número de intervalos de tempo que representa o atraso da excitação que está influenciando no valor presente da resposta $z(t)$.

3.7 Aprendizado

O processo de aprendizado é desenvolvido visando encontrar os coeficientes ou pesos que propiciem o melhor ajuste entre a função de mapeamento e os dados,

que consistem dos exemplos da função desejada. O número de dados de entrada e amostras, o número de nós nas camadas intermediárias, o número de camadas intermediárias e a complexidade do problema em questão interferem no aprendizado da rede. Normalmente para abordagens utilizando redes neurais artificiais usa-se somente uma camada oculta.

O cálculo dos pesos sinápticos é um processo matemático que recebe o nome de treinamento ou aprendizado.

Na verdade, como será visto mais adiante, o cálculo dos pesos sinápticos pode ser visto como a solução de um problema de otimização onde se deseja minimizar o erro médio quadrático entre a resposta da rede neural e o valor correto da mesma para um dado conjunto de exemplos constituintes de uma amostra de treinamento.

3.7.1 Superfície de erro

Como em uma regressão, o erro médio quadrático é a forma convencional para medir o ajuste dos dados. O erro médio quadrático para acomodar a saída de uma rede com uma única saída pode ser definido como:

$$E = \frac{1}{2N} \sum_{i=1}^N (z_n - t_n)^2 \quad (3.8)$$

onde N é o número de exemplos do conjunto de treinamento, t_n é a saída desejada para o n -ésimo exemplo, e z_n é a n -ésima saída atual para o n -ésimo exemplo.

É útil interpretar essa medida de erro geometricamente. Considerando um modelo linear $y = a_0 + a_1x_1$, que possui dois parâmetros que correspondem aos pesos a_0 e a_1 , estes dois pesos formam um espaço-peso bidimensional. A terceira dimensão é a altura, que representa o erro, conforme a Figura 3.8:

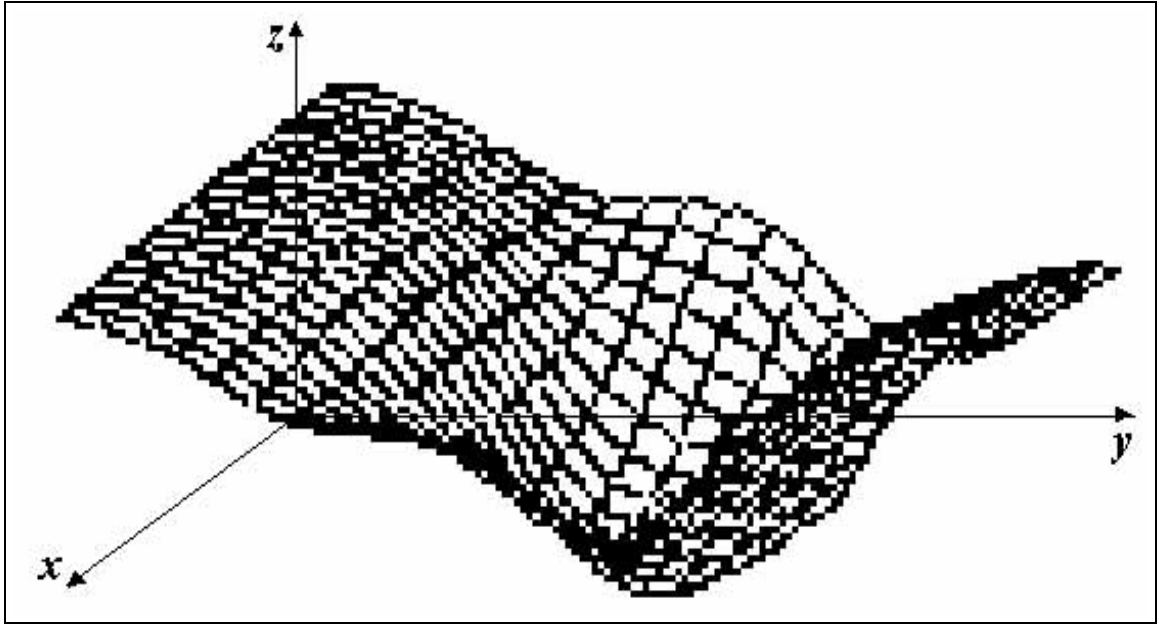


Figura 3.8: Superfície de erro.

Neste espaço tridimensional encontra-se a superfície de erro. A altura de cada ponto nesta superfície de erro representa o erro em um certo grupo de exemplos de um modelo linear em particular, cujos parâmetros correspondem aos pontos das coordenadas a_0 e a_1 . Os pesos ideais são os pares que correspondem aos pontos mais baixos na superfície de erro. O melhor modelo é aquele que produz o menor erro.

Observa-se que a forma da superfície de erro e a localização do ponto no espaço de pesos sobre o qual ele é o mais baixo depende do exemplo em particular no qual o erro está sendo medido. Se o exemplo mudar o erro também muda.

Quando o modelo tem mais de dois parâmetros, não é possível visualizar a superfície de erro, mas é possível defini-la geometricamente. A superfície de erro tem um ponto correspondente a cada ponto no espaço de pesos. A altura sobre o espaço de pesos de cada ponto na superfície representa o erro de um modelo com o peso correspondente.

Um dos aspectos mais importantes em redes neurais artificiais é encontrar o conjunto ótimo \vec{w}_0 de pesos sinápticos que minimizem o erro médio quadrático para a aplicação em questão.

3.7.2 Métodos de minimização dos erros

Regressão linear é uma técnica para calcular a série de coeficientes de uma função linear que produz o erro quadrático mínimo em um conjunto específico de dados: o ponto no espaço de pesos sobre o qual a superfície de erro é mínima.

Nos casos de funções não lineares mais complexas computadas por redes neurais, não há um caminho prático e direto para calcular os valores ótimos para os pesos.

Uma forma de encontrar os melhores pesos seria a força bruta, calculando o erro para alguns grupos de valores para todos os pesos. Todavia o número de computações de erro necessárias seria igual ao número de valores testados para cada peso, elevado à potência do número de pesos. Uma aproximação por força bruta é, contudo computacionalmente impossível.

Desta forma, vários algoritmos de otimização têm sido propostos para a busca do conjunto de pesos que minimizem o erro da rede. Nesta dissertação estão comentados apenas dois deles:

- Gradiente descendente,
- Gradiente conjugado,

porém, outros podem ser encontrados na literatura [8].

Gradiente descendente

O gradiente descendente é um dos métodos mais simples empregados em redes neurais artificiais. Neste método, selecionam-se pontos arbitrários no espaço de

pesos, computa-se a inclinação da superfície de erro nesse ponto, modifica-se os pesos na direção em que a superfície de erro desce mais acentuadamente.

Matematicamente

$$\vec{w}_{i+1} = \vec{w}_i - \eta \Delta \vec{w}_i \quad (3.9)$$

onde η é chamado de taxa de aprendizagem e $\Delta \vec{w}_i = \frac{\partial E}{\partial \vec{w}_i}$ é a derivada do erro médio quadrático com relação aos valores atuais dos pesos sinápticos.

Utilizando esses novos valores de pesos, a idéia do método é repetir o processo iterativamente, até o que os valores dos pesos se aproximem do ponto mais baixo na superfície de erro.

Na literatura de redes neurais esse método é comumente conhecido como algoritmo de retropropagação ou *backpropagation*.

Gradiente conjugado

O método do gradiente conjugado pertence à classe dos métodos de otimização de segunda ordem, conhecidos coletivamente como método de direção conjugada [8].

Para exemplificar a discussão do método, considera-se a minimização da função quadrática.

$$f(\vec{x}) = \frac{1}{2} \vec{x}^T A \vec{x} - \vec{b}^T \vec{x} + \vec{c} \quad (3.10)$$

onde:

- \vec{x} é um vetor de parâmetros $W \times 1$,
- \vec{A} é uma matriz $W \times W$ simétrica, positivamente definida,

- \vec{b} é um vetor $W \times 1$
- c é um escalar.

A minimização da função quadrática $f(\vec{x})$ é alcançada atribuindo a x um valor único:

$$\vec{x}^* = \vec{A}^{-1}\vec{b} \quad (3.11)$$

Com isso, há uma equivalência entre minimizar $f(\vec{x})$ e resolver o sistema de equações lineares $\vec{A}\vec{x}^* = \vec{b}$.

Dada a matriz \vec{A} , um vetor contendo vetores não-nulos $\vec{s}_0, \vec{s}_1, \dots, \vec{s}_{W-1}$ é um conjugado de \vec{A} , isto é, não interferem entre si no contexto da matriz \vec{s}_0 , se a seguinte condição for satisfeita:

$$\vec{s}_i^T \vec{A} \vec{s}_j = 0, \quad (3.12)$$

para todo i e j , tal que $i \neq j$.

Se \vec{A} for igual a matriz identidade, a conjugação é equivalente a noção usual de ortogonalidade. Uma importante propriedade dos vetores conjugados de \vec{A} é que eles são linearmente independentes, portanto formam uma base que cobre todo o espaço vetorial de \vec{w} .

Para um dado conjunto de vetores conjugados de \vec{A} $\vec{s}_0, \vec{s}_1, \dots, \vec{s}_{W-1}$, o método da direção conjugada correspondente para minimização irrestrita da função erro quadrática $f(\vec{x})$ é definida como:

$$\vec{x}_{i+1} = \vec{x}_i + \eta_i \vec{s}_i, i = 0, 1, \dots, W - 1 \quad (3.13)$$

onde \vec{x}_0 é um vetor inicial arbitrário e η_i é um escalar definido por:

$$f(\vec{x}_i + \eta_i \vec{s}_i) = \min_{\eta} (f(\vec{x}_i) + \eta \vec{s}_i) \quad (3.14)$$

O procedimento para a escolha de η de forma a minimizar a função $f(\vec{x}_i + \vec{s}_i)$ para um i fixo é referido como uma busca em linha que cobre o espaço vetorial \vec{x} . Um algoritmo de busca em linha é um procedimento iterativo que gera uma seqüência de estimativas η_i para cada iteração do algoritmo do gradiente conjugado. Deve-se realizar uma busca em linha ao longo de cada direção de busca.

Assim o parâmetro para a taxa de aprendizagem por definição é:

$$\eta_i = \frac{\vec{s}_i^T \vec{A} \vec{\varepsilon}_i}{\vec{s}_i^T \vec{A} \vec{s}_i}, i = 0, 1, \dots, w - 1 \quad (3.15)$$

onde $\vec{\varepsilon}_i$ é o vetor erro definido por:

$$\vec{\varepsilon}_i = \vec{x}_i - \vec{x}^* \quad (3.16)$$

Começando de um ponto arbitrário \vec{x}_0 , o método da direção conjugada garante encontrar a solução ótima \vec{x}^* da equação quadrática $f(\vec{x}) = 0$ em W iterações.

Para o método da direção conjugada funcionar, é necessária a disponibilidade de um conjunto de vetores conjugados de \vec{A} $\vec{s}_0, \vec{s}_1, \dots, \vec{s}_{W-1}$. No método do gradiente conjugado, os vetores de direção sucessivos são gerados como versões conjugadas de \vec{A} dos vetores de gradiente sucessivos da função quadrática $f(\vec{x})$, conforme o

método avança.

Exceto para $i = 0$, o conjunto de vetores de direção \vec{s}_i não é especificado previamente, sendo determinado de forma seqüencial nos passos sucessivos do método.

O residual dado pela direção descendente mais íngreme:

$$\vec{r}_i = \vec{b} - \vec{A}x_i \quad (3.17)$$

Usando uma combinação linear de \vec{r}_i e \vec{s}_{i-1} :

$$\vec{s}_i = \vec{r}_i + \beta_i \vec{s}_{i-1}, i = 1, 2, \dots, W - 1 \quad (3.18)$$

onde β_i é um fator de escala definido como:

$$\beta_i = \frac{\vec{s}_{i-1}^T \vec{A} \vec{r}_i}{\vec{s}_{i-1}^T \vec{A} \vec{s}_{i-1}} \quad (3.19)$$

Quando a matriz \vec{A} não é previamente conhecida, pode-se calcular β usando a fórmula de Polak-Ribière, definida por:

$$\beta_i^{PR} = \frac{\vec{r}_i^T (\vec{r}_i - \vec{r}_{i-1})}{\vec{r}_{i-1}^T \vec{r}_{i-1}} \quad (3.20)$$

Para assegurar a convergência do método de Polak-Ribière deve escolher [8]:

$$\beta_i = \max(\beta_i^{PR}, 0) \quad (3.21)$$

Para se utilizar o método do gradiente conjugado para minimizar o erro médio quadrático relativo ao treinamento supervisionado de uma rede neural, faz-se:

- A aproximação da função $E(\vec{w})$, equação 3.8 por uma função quadrática, usando a série de Taylor. Os termos de ordem mais elevada são ignorados, propiciando a operação próxima a um mínimo local na superfície de erro.
- Formula-se os coeficientes β e η no algoritmo do gradiente conjugado de modo a necessitar apenas da informação do gradiente, utilizando por exemplo a fórmula de Polak-Ribière.

3.7.3 Mínimo local

Na busca do conjunto ótimo de pesos sinápticos (mínimo global), no processo de otimização podem ser encontrados mínimos locais na superfície de erro. Em um mínimo local, pequenas variações nos pesos sinápticos causam um aumento considerável da função de custo. Todavia, há um outro conjunto de pesos sinápticos para o qual a função de custo é menor que o mínimo local no qual a rede se encontra presa.

Para um mínimo local existir, todos os pesos devem estar simultaneamente em uma posição na qual mudanças em quaisquer direções aumentam o erro. Entretanto, a probabilidade de um mínimo local existir diminui quando o número de pesos da rede aumenta, assim, caso haja um mínimo local, a adição de mais nós intermediários tende a eliminá-los.

A Figura 3.9 a seguir mostra um exemplo de mínimo local e mínimo global.

A discussão sobre mínimos locais ainda é um tema em discussão na literatura sobre redes neurais artificiais [8].

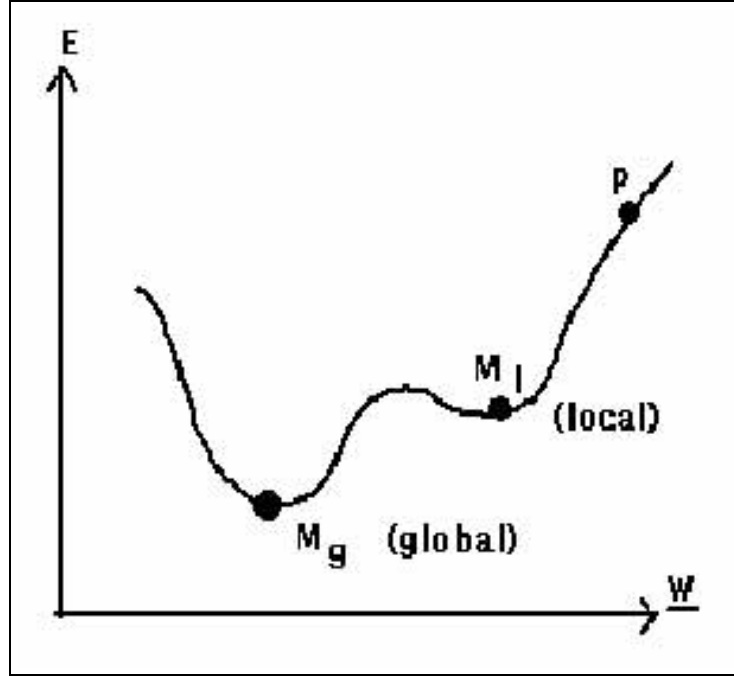


Figura 3.9: Mínimo local e global.

3.8 Derivadas do Erro

Como visto anteriormente, na minimização da função do erro médio quadrático em uma rede neural, é necessário conhecer as derivadas parciais dos mesmos em relação aos seus pesos. As derivadas do erro médio quadrático em relação a um dado peso sináptico w_{ij} para todo o conjunto de treinamento é simplesmente a soma das derivadas em cada exemplo deste conjunto, i.e.,

$$\frac{\partial E(\vec{w})}{\partial w_{ij}} = \frac{\partial}{\partial w_{ij}} \frac{1}{2N} \sum_{k=1}^N (z_k^{\vec{w}} - t_k)^2 = \frac{1}{N} \sum_{k=1}^N \frac{\partial E_k}{\partial w_{ij}} \quad (3.22)$$

A derivada parcial do erro médio em relação aos pesos sinápticos representa um fator de sensibilidade, determinando a direção de busca no espaço de pesos, para o peso sináptico w_{ji} , onde $E_k = \frac{1}{2}(z_k(\vec{w}) - t_k)^2$ e N é o número de exemplos da amostra de treinamento.

3.8.1 Derivadas dos pesos sinápticos ligados a um nó de saída

Conforme a Figura 3.10, para um nó de saída não-linear da rede tem-se:

$$z_k = \varphi(v) = \varphi\left(\sum w_j y_j\right) \quad (3.23)$$

onde $\varphi(\cdot)$ é a função de ativação, y_i são as saídas dos nós intermediários e w_j são os respectivos pesos sinápticos.

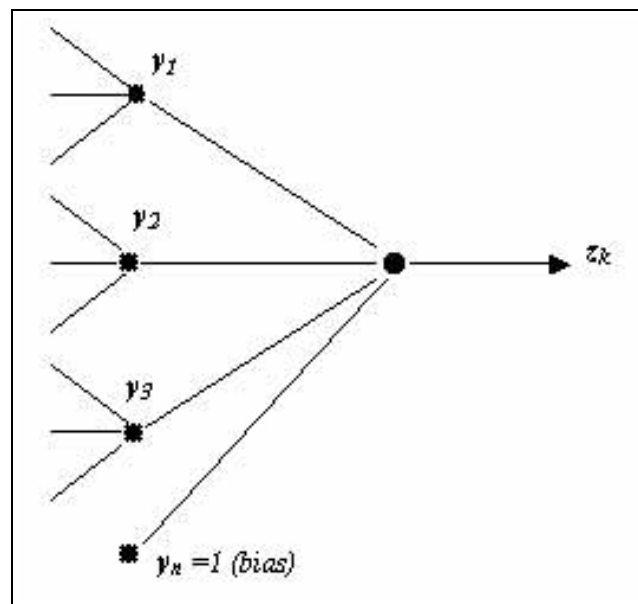


Figura 3.10: Grafo de fluxo de sinal em uma rede neural.

Um nó específico de saída não exerce nenhum efeito sobre o erro gerado por outro nó de saída; ele é responsável única e exclusivamente pelos seus próprios erros. O erro quadrático médio de um nó de saída em um exemplo é:

$$E_k = \frac{1}{2}(z_k - t_k)^2 \quad (3.24)$$

onde z_k é a saída atual e t_k é a saída correta para o exemplo considerado.

Observando-se as equações 3.23 e 3.24 verifica-se que a derivada do erro E_k em relação a um peso w_i pode ser escrita como:

$$\frac{\partial E_k}{\partial w_j} = \frac{\partial E_k}{\partial z_k} \frac{\partial z_k}{\partial v} \frac{\partial v}{\partial w_j} \quad (3.25)$$

Através da Equação 3.25, observa-se que a derivada do erro em relação ao peso das conexões ligadas a um nó de saída é o produto de três termos:

- Derivada do erro em relação a saída daquele nó, $\frac{\partial E}{\partial z_k} = z_k - t_k$
- Derivada da saída deste nó em relação a sua soma ponderada de entradas, $\frac{\partial z_k}{\partial v} = \varphi'(v)$
- Derivada daquela soma em relação ao peso em questão, $\frac{\partial v}{\partial w_j} = y_i$

Quando a saída da rede for linear observa-se que o termo $\frac{\partial z_k}{\partial v}$ é igual a unidade.

3.8.2 Derivada dos pesos sinápticos em relação a um nó oculto

Conforme mostra a Figura 3.11, um peso sináptico w_{ij} associado a uma camada oculta está diretamente ligado ao valor de saída y_j que por sua vez interfere diretamente na saída final da rede z_k , ou na sua resposta.

Em função do que foi exposto anteriormente, observa-se que a derivada de erro E_k para uma amostra k qualquer do conjunto de treinamento em função de um peso sináptico w_{ij} pode ser escrito como:

$$\frac{\partial E_k}{\partial w_{ij}} = \frac{\partial E_k}{\partial z_k} \frac{\partial z_k}{\partial v} \frac{\partial v}{\partial y_j} \frac{\partial y_j}{\partial w_{ij}} \quad (3.26)$$

onde

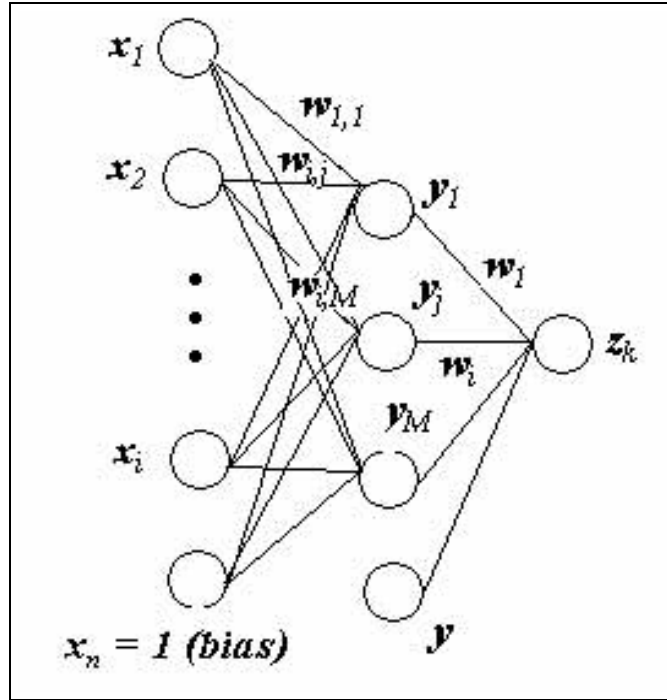


Figura 3.11: Grafo de fluxo de sinal dos pesos sinápticos de um nó intermediário.

$$\frac{\partial E}{\partial z_k} = (z_k - t_k) \quad (3.27)$$

$$\frac{\partial z_k}{\partial v} = \varphi'(v) = \varphi'(\sum w_{ij}y_i) \quad (3.28)$$

$$\frac{\partial v}{\partial y_j} = w_j \quad (3.29)$$

$$\frac{\partial y_i}{\partial w_{ij}} = \frac{\partial(\sum w_{ij}x_i)}{\partial w_{ij}} = \varphi'(\sum w_{ij}x_i)x_i \quad (3.30)$$

sendo x_i a saída do i -ésimo neurônio da rede. Quando a rede possui somente uma camada oculta, x_i é o próprio valor da i -ésima variável de entrada para o k -ésimo exemplo da amostra de treinamento.

3.9 Generalização

Este capítulo trata a forma de como manter a rede em sintonia com os exemplos de treinamento. Para a rede é importante aprender com o passado para generalizar o futuro.

Há uma concepção errônea relativa ao treinamento iterativo. Adota-se a idéia de que se deve treinar a rede neural para reduzir mais e mais o erro junto ao conjunto de treinamento de modo a aumentar o desempenho da rede neural. Embora, parcialmente verdadeira, esta iniciativa pode conduzir à especialização excessiva junto ao conjunto de treinamento, o qual pode conter amostras sujeitas a ruído, que passariam a ser incorporados à solução, e certamente representa um conjunto finito de informação.

Diz-se que uma rede generaliza bem quando o mapeamento de entrada-saída computado pela rede for correto (ou aproximadamente correto) para dados de teste não-utilizados para criação ou treinamento da rede.

O processo de aprendizagem da rede (i.e. treinamento de rede neural) pode ser visto como um problema de "ajuste de curva". A própria rede pode ser considerada simplesmente como um mapeamento não-linear de entrada-saída. A rede realiza boa interpolação fundamentalmente porque *perceptrons* de múltiplas camadas com funções de ativação contínuas produzem funções de saída que também são contínuas.

A Figura 3.12 apresenta um esboço do erro de uma rede neural para o conjunto de treinamento, e para um conjunto de teste ou validações (ambos gerados a partir do mesmo conjunto de dados, mas independentes entre si). Respostas com esta conformação geralmente ocorrem em treinamento de redes neurais. Como desejado, o erro para o conjunto de treinamento decresce continuamente (com exceção de pequenas oscilações ocasionais). Para o conjunto de teste, o erro decresce no

princípio, embora o mesmo se mantenha superior ao do conjunto de treinamento. Continuando o treinamento, o erro referente ao conjunto de teste começa a crescer. Uma vez que o conjunto de teste (ou validação) é presumivelmente representativo do problema para o qual a rede deverá representar a solução, isto significa que a rede está perdendo capacidade de generalização. Assim, o ideal é acompanhar o erro de treinamento, mas conforme o número de iterações cresce, também se deve procurar acompanhar o erro de teste, que avalia o desempenho da rede (um tipo de validação cruzada). Deve-se parar o treinamento quando o mínimo do erro frente ao conjunto de teste foi atingido ou esteja dentro de um valor pré-estabelecido, de forma a garantir uma melhor capacidade de generalização.

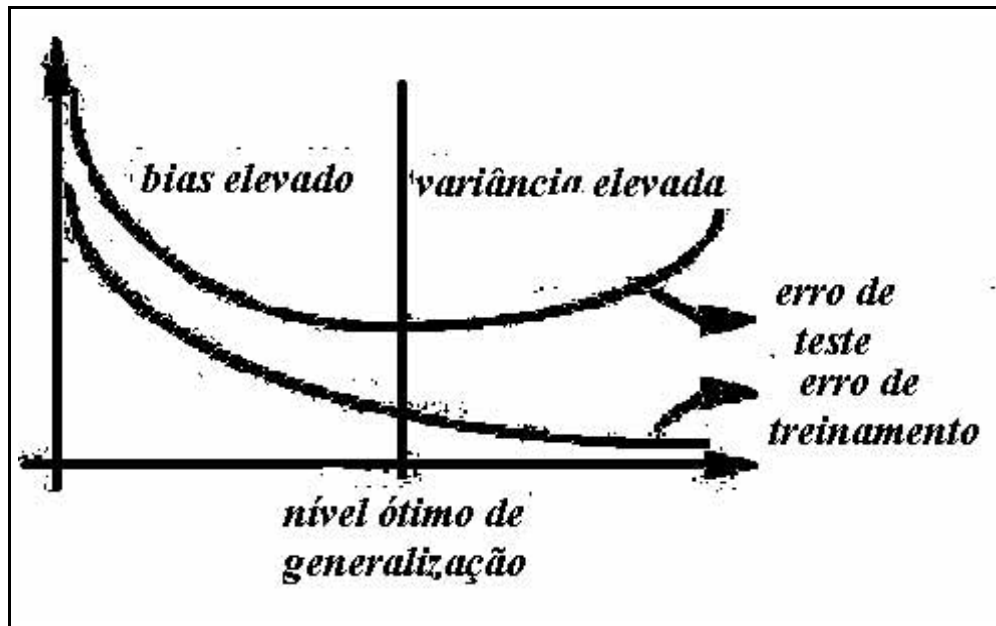


Figura 3.12: Nível ótimo de generalização.

A Figura 3.13 ilustra como a generalização pode ocorrer em uma rede hipotética. O mapeamento não-linear de entrada-saída representado pela curva mostrada nessa figura é computado pela rede como resultado da aprendizagem dos pontos rotulados como "dados de treinamento". O ponto marcado sobre a curva como generalização é visto assim como o resultado da interpolação realizada pela rede.

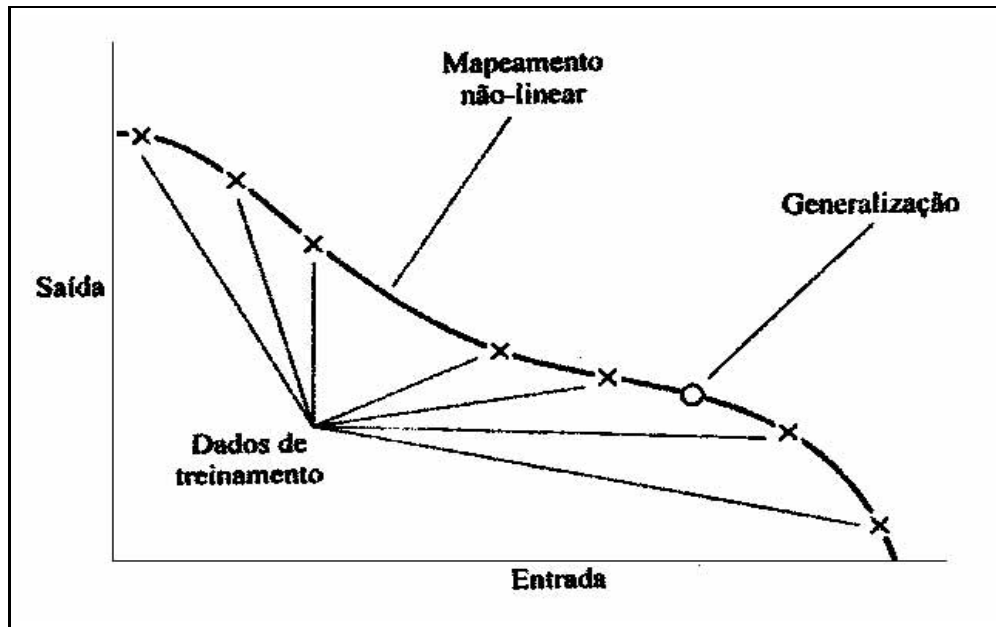


Figura 3.13: Dados ajustados adequadamente (boa generalização).

Uma rede neural, que é projetada para generalizar bem, produzirá um mapeamento de entrada-saída correto, mesmo quando a entrada for um pouco diferente dos exemplos usados para treinar a rede, como ilustrado na Figura 3.13. Entretanto, quando a rede neural aprende um número excessivo de exemplos de entrada-saída, a rede pode acabar memorizando os dados de treinamento. Ela pode fazer isso encontrando uma característica que está presente nos dados de treinamento, mas não na função subjacente que deve ser modelada. Este fenômeno é conhecido como excesso de ajuste ou excesso de treinamento. Quando a rede é treinada em excesso, ela perde a habilidade de generalizar entre padrões de entrada-saída similares. Um exemplo de como pode ocorrer generalização pobre devido à memorização em uma rede neural está ilustrado na Figura 3.14 para os mesmos dados apresentados na Figura 3.13.

Em linhas gerais a generalização é influenciada por três fatores:

- O tamanho do conjunto de treinamento, e quão representativo do ambiente de

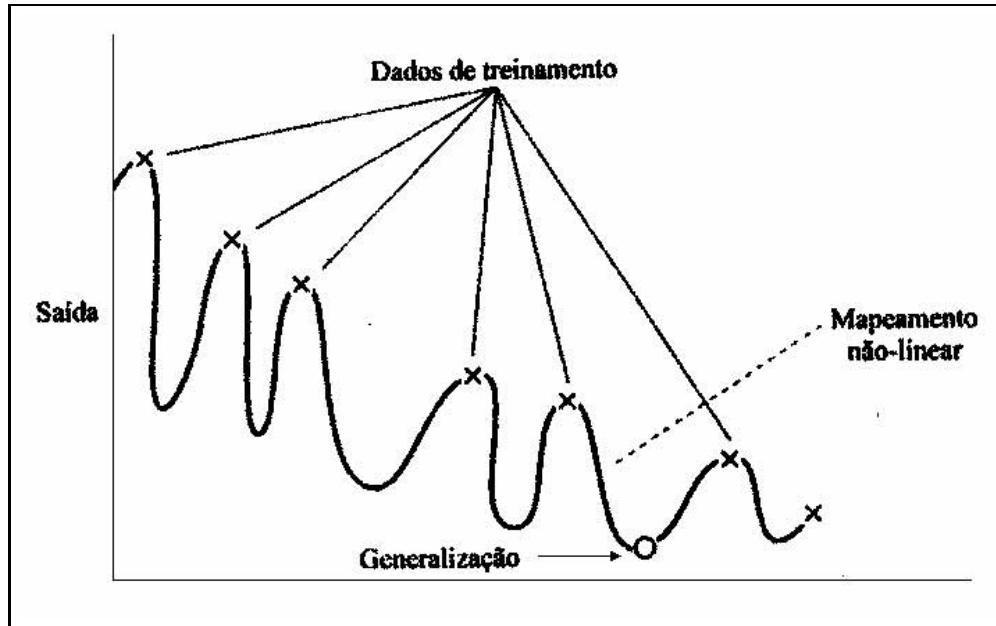


Figura 3.14: Dados ajustados adequadamente em excesso (generalização pobre).

interesse ele é;

- A arquitetura da rede neural;
- A complexidade física do problema em questão, evidentemente não há controle sobre este fator.

Na definição da topologia de uma rede neural, normalmente, o número de camadas e o número de nós em cada camada são definidos em função de uma inspeção prévia nos dados e na complexidade do problema. Uma vez definida a topologia inicial, a estrutura final mais adequada para a modelagem é geralmente obtida através de refinamentos sucessivos, que podem levar a um tempo de dimensionamento alto, já que este tem um grande componente empírico.

O objetivo desta etapa de ajuste é a obtenção de uma topologia de rede que modele com precisão os dados do conjunto de treinamento, mas que também resulte em uma aproximação com boa capacidade de generalização.

Como na maioria dos casos o conjunto de treinamento de uma rede neural é composto de dados experimentais, este contém implicitamente erros inerentes aos processos de amostragem. Desta forma, a aproximação através de redes neurais deve ser feita visando à obtenção de uma rede de estrutura que seja capaz de modelar os dados sem modelar ruído contido neles.

Este é um problema no projeto de redes neurais conhecido na literatura como *bias and variance dilemma*, que envolve a obtenção de um modelo que não seja muito rígido a ponto de não modelar fielmente os dados, mas que também não seja excessivamente flexível a ponto de modelar também o ruído.

O equilíbrio entre a rigidez e a flexibilidade da rede é obtido por meio de seu dimensionamento. Quanto maior sua estrutura, maior o número de parâmetros livres ajustáveis e, conseqüentemente, maior a sua flexibilidade. Porém, quando os dados são apresentados à rede, não se tem real conhecimento de sua complexidade, daí a dificuldade do problema de dimensionamento.

Uma forma de se evitar o *overfitting* é estimar o erro de generalização durante o processo de treinamento. Para isso, o conjunto de dados é dividido em um conjunto de treinamento e um conjunto de validação. O conjunto de treinamento é utilizado na modificação dos pesos e o conjunto de validação é utilizado para estimar a capacidade de generalização da rede durante o processo de aprendizagem.

O treinamento deve ser interrompido quando o erro do conjunto de validação começar a subir, ou seja, quando a rede começar a incorporar o ruído presente nos dados, o que causa degradação na sua capacidade de generalização. Embora essa alternativa possa se mostrar eficiente em algumas situações, sua utilização é limitada para os casos em que um conjunto de treinamento muito grande esteja disponível, já que os dados do conjunto de validação não podem ser utilizados para

treinamento. Este tema será mais detalhado a seguir.

3.10 Validação Cruzada

A essência da aprendizagem por retropropagação é codificar um mapeamento de entrada-saída (representado por um conjunto de exemplos rotulados) nos pesos sinápticos e limiares de um *perceptron* de múltiplas camadas. A rede deve ser bem treinada de modo que aprenda o suficiente sobre o passado para generalizar o futuro. Desta perspectiva, o processo de aprendizagem se transforma em uma escolha de parametrização da rede para este conjunto de dados.

Para demonstrar que o caminho para determinar quão bem uma rede pode capturar a natureza de uma função é validar a rede em exemplos adicionais que não foram utilizados no treinamento da rede.

Este método é baseado numa técnica estatística denominada validação cruzada. Não há garantias de que a validação cruzada vá produzir o melhor e mais eficiente modelo. Quão menor a amostra de validação e maior o nível de ruído, maior as chances da validação cruzada falhar em atingir seu ideal.

Há, entretanto, uma possibilidade considerável de que o modelo assim selecionado, com os valores de parâmetros de melhor desempenho, possa acabar ajustando excessivamente o subconjunto de validação. Apesar deste risco teórico, a validação cruzada vem sendo utilizada a um longo tempo em estatística e é uma ferramenta prática valiosa.

Pode-se limitar o poder da rede de três formas:

- Limitando o número de nós ocultos

- Desencorajando a rede a usar pesos com valores grandes
- Limitando o número de iterações do treinamento
- Medir o desempenho de generalização do modelo selecionado sobre o conjunto de teste, que é diferente do subconjunto de validação.

Embora teoricamente as três situações funcionem de forma satisfatória, limitar o treinamento é o método mais indicado, porque ele requer muito menos esforço computacional que as outras aproximações.

O uso de validação cruzada é atrativo particularmente quando temos que projetar uma rede neural grande cujo objetivo seja uma boa generalização.

3.10.1 Limitando o número de nós ocultos

O objetivo da rede é encontrar a forma da função objetivo, e isto pode ser obtido limitando o poder da rede de alguma forma. O método discutido neste item é a limitação do número de nós ocultos.

Cada peso na rede é um parâmetro que adiciona capacidade a rede. O número de pesos determina o grau de liberdade com o qual a rede pode se adequar aos dados. Para limitar a capacidade da rede devemos limitar o número de pesos.

O número de pesos na rede é uma função do número de nós que ela tem. Desde que o número de nós de saída numa rede seja geralmente determinado pela natureza do problema, é possível controlar o número de pesos controlando o número de nós de entrada e nós ocultos.

Para identificar o número ótimo de nós ocultos para um dado problema é necessário que:

- A amostra seja dividida em sub-amostras de treinamento e validação;

- Treinamento da rede com diferentes números de nós ocultos por todo o processo de convergência na amostra de treinamento, medindo suas performances na amostra de validação;
- E escolhendo a rede que melhor se adequa a amostra de validação. Esta técnica funciona, porém requer um grande esforço computacional. Várias redes com diferentes números de nós devem ser treinadas. Sobretudo todas elas devem ser treinadas até a convergência, até que a derivada dos erros em relação a cada peso seja igual a 0 e os pesos realmente parem de mudar.

3.10.2 Interrompendo o treinamento

Conforme uma rede é treinada, ela passa de uma função de mapeamento relativamente simples para uma mais complexa. Este aumento na complexidade não é constante no tempo, mas ocorre em bateladas. Cada batelada adiciona uma característica à função de mapeamento. Como regra geral, cada batelada corresponde a alterações notáveis nos pesos de um único nó, e estas transições são separadas por longos períodos de sintonia fina onde nada realmente notável acontece.

Durante o processo de treinamento, a rede passa por sucessivos estágios nos quais o número de nós ocultos que estão efetivamente fazendo alguma coisa cresce um a um. Conseqüentemente, a rede passa por estágios durante os quais suas saídas são similares as saídas de redes convergidas com vários números de nós ocultos. O processo de desenvolvimento passa por estágios que aproximadamente assemelha-se completamente às redes desenvolvidas com diferentes níveis de complexidade.

Conforme a rede é treinada e sua função de mapeamento torna-se mais complexa, ela passa em algum ponto por uma configuração que fornece a melhor generalização; depois deste ponto, o que a rede aprende acrescenta-se ao *overfitting*. Se for possível determinar esse ponto, pode-se parar o treinamento antes que o *overfitting* ocorra e utilizar a configuração da rede que melhor generaliza.

Não é possível determinar quando parar o treinamento simplesmente observando o erro na amostra de treinamento. O erro na amostra de treinamento sempre decresce, às vezes, exaustivamente devagar, e então mais rapidamente quando o um nó oculto encontra alguma característica importante nos dados de treinamento. Em algum momento, um dos nós ocultos encontra uma característica que está presente na amostra de treinamento, mas não é verdadeiro e representativo para a população em geral, neste ponto começa o *overfitting*. Mas o erro na amostra de treinamento decresce, portanto não é possível identificar quando acontece o *overfitting* analisando o erro no treinamento.

É possível, entretanto, identificar o *overfitting* verificando o erro em uma amostra de validação. Dividindo-se a amostra em treinamento e sub-amostras de validação. Treina-se a rede na amostra de treinamento, mas periodicamente interrompe-se o treinamento e mede-se o erro na amostra de validação. Para efetuar este procedimento, é necessário executar somente a passagem *forward* através da rede e calcular o erro para cada exemplo. Cada vez que o treinamento for interrompido para se medir o erro na amostra de validação, salva-se os pesos. Quando o erro na amostra de validação crescer, indica que se iniciou o *overfitting*. Conseqüentemente interrompe-se o treinamento, retorna-se aos pesos que produziram o menor erro na amostra de validação, então utiliza-se esse peso para o modelo.

Em geral o erro na amostra de validação é maior que o erro na amostra de treinamento. Isto ocorre porque a rede está trabalhando para reduzir os erros subseqüentes, não os anteriores.

Se um erro de validação nunca cresce, isto indica que a rede não tem nós suficientes para ter *overfitting*. Todavia, isto também indica que a rede não tem nós suficientes para obter sua melhor performance. Neste caso, se a rede nunca sofre

overfitting, a solução seria utilizar mais nós ocultos.

A validação cruzada através da interrupção do treinamento antes que o *overfitting* ocorra requer muito menos esforço computacional que a validação cruzada para encontrar o número correto de nós ocultos. Para a validação cruzada, com limitação do número de nós ocultos, seria necessário fazer várias tentativas de redes com diferentes números de nós ocultos até encontrar a convergência.

Capítulo 4

Aplicações

Neste trabalho foi implementado em FORTRAN um modelo de rede neural constituído de apenas uma camada oculta e uma saída linear. Este programa dispõe dos métodos do gradiente conjugado e gradiente descendente para a avaliação dos pesos sinápticos. Os dados de análise devem ser separados em um conjunto de treinamento e um conjunto de validação. Todos os dados de entrada e saída dos conjuntos de treinamento e validação são normalizados dentro do intervalo $[-0.95,0.95]$.

Uma vez que os pesos sinápticos tenham sido definidos, ou a rede tenha sido treinada, um conjunto de dados de entrada pode ser fornecido para que seja processada a predição da correspondente saída (resposta).

A seguir, serão apresentados dois exemplos de aplicação. O primeiro caso constitui-se um sistema dinâmico simples com um grau de liberdade com restauração não-linear. Este modelo foi utilizado para verificar a aplicação prática das redes neurais assim como investigar a sensibilidade das mesmas em relação aos seus parâmetros variáveis perante mudanças no grau de não-linearidade do sistema, o carregamento aleatório imposto, assim como o atraso temporal considerado para a resposta.

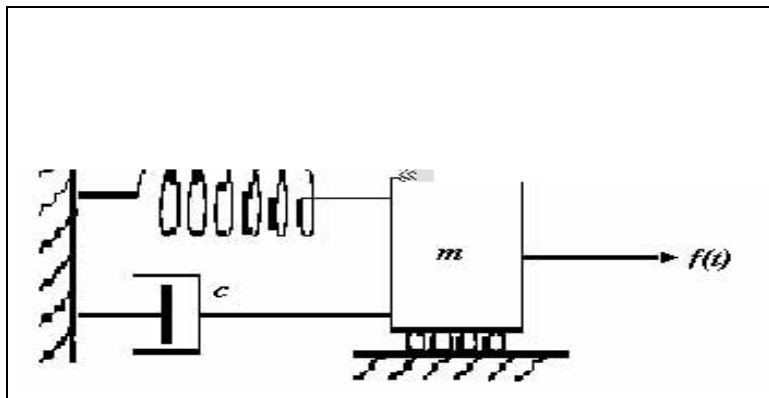


Figura 4.1: Sistema massa-mola avaliado.

O segundo caso considerado constitui-se de uma linha de ancoragem de um FPSO (*Floating Production Storage and Offloading*) com ancoragem do tipo DICAS (*Differentiated Compliance Anchoring System*), MASETTI (1995) [16] numa lâmina d'água de 800m. Neste sistema as entradas são os movimentos prescritos de *surge*, *sway* e *heave* da unidade flutuante no topo da linha e a resposta considerada é a tração dinâmica no topo da mesma.

4.1 Modelo com um grau de liberdade

Como o primeiro exemplo deste trabalho, considera-se o sistema massa-mola-amortecedor apresentado na Figura 4.1. Neste sistema o carregamento $f(t)$ utilizado é dado por:

$$f(t) = a_1 p(t) + a_2 p(t)^2 + a_3 p(t)^3 \quad (4.1)$$

onde $p(t)$ é um carregamento aleatório gaussiano com intensidade numérica igual a um espectro de Pierson-Moscovitz gerado para um estado de mar com altura significativa $H_s=7.0\text{m}$ e período de cruzamento zero $T_z=11.0\text{s}$. Os parâmetros a_1 ,

Tabela 4.1: Casos analisados para modelo com um grau de liberdade

Caso	Massa	Amortecimento	k1	k2	k3	a1	a2	a3
1	10.00	2.00	5.00	0.50	0	1.00	0	0
2	10.00	2.00	5.00	0.50	0	1.00	0.50	0
3	10.00	2.00	5.00	0.50	0	1.00	0	0
4	10.00	2.00	5.00	0.50	0.02	1.00	0.50	0.20

a_2 e a_3 podem ser variados para modificar o grau de não-linearidade da excitação $f(t)$.

A equação de equilíbrio do sistema representada na Figura 4.1 é dada por:

$$m \frac{\partial^2 x(t)}{\partial t^2} + c \frac{\partial x(t)}{\partial t} + k_1 x(t) + k_2 x(t)^2 + k_3 x(t)^3 = f(t) \quad (4.2)$$

onde m é a massa do sistema, c o amortecimento, k_1 , k_2 e k_3 são parâmetros que representam a restauração do sistema e $x(t)$ é o deslocamento no tempo. Variações nos parâmetros k_1 , k_2 e k_3 interferem no grau de não-linearidade da restauração do sistema. A força de restauração do sistema é dada por:

$$f_r(t) = k_1 x(t) + k_2 x(t)^2 + k_3 x(t)^3 \quad (4.3)$$

Foram analisados quatro casos com diferentes graus de não-linearidade conforme as propriedades do sistema apresentadas na Tabela 4.1. Note que o caso 3 representa um sistema linear tanto no carregamento quanto na restauração.

A resposta correta do sistema para cada um dos casos considerados foi obtida

empregando-se o algoritmo de Runge-Kutta quarta ordem para um tempo de simulação de 1150s. Na investigação se as redes neurais são capazes de representar o comportamento do sistema nos diferentes casos analisados foram considerados os seguintes aspectos:

- Entrada do sistema: força de excitação $f(t)$
- Saída (resposta do sistema): força de restauração $fr(t)$ (vide Equação 4.3)

No treinamento da rede considerou-se inicialmente como conjunto de treinamento os primeiros 200s das séries, depois de desprezar 50s de transiente. Foi utilizado o método do gradiente conjugado para obter os pesos. Como conjunto de validação, considerou-se toda a série excluindo-se o transiente.

Os parâmetros variáveis da rede neural são os números de neurônios da camada oculta e o tempo de atraso ou o tempo que a excitação continua atuando no sistema. Nas Figuras 4.2 a 4.9 são mostrados dois resultados obtidos para cada um dos casos considerados, como pode ser observado, em função dos parâmetros utilizados a rede neural consegue ou não representar adequadamente a resposta do sistema.

As Figuras 4.10 a 4.13 ilustram o comportamento do erro para o conjunto de validação em alguns casos analisados ao longo do processo de busca dos pesos sinápticos. Nas Figuras 4.14 a 4.17 apresentam-se as curvas de erro mínimo obtido para o conjunto de validação em função do tempo de atraso e o número de neurônios da camada oculta.

Dentre os casos analisados, e estruturas de rede analisadas, o Caso 3 é onde a rede melhor representa a resposta, e o Caso 4, o mais não-linear, o ajuste é bom, porém nos picos os valores calculados pela rede apresentam uma pequena diferença.

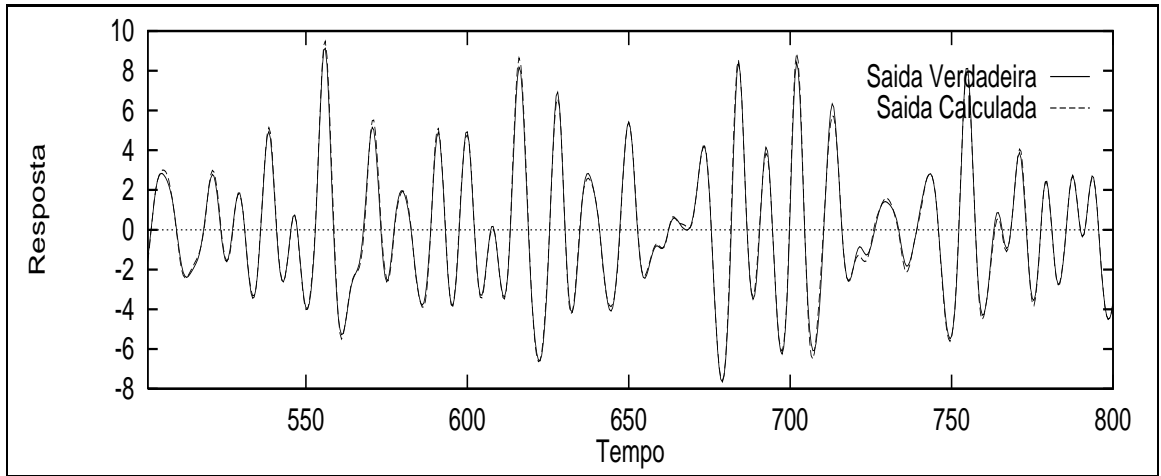


Figura 4.2: Caso 1 - 25 nós ocultos e 50s de atraso. Erro igual a $1.68E-04$.

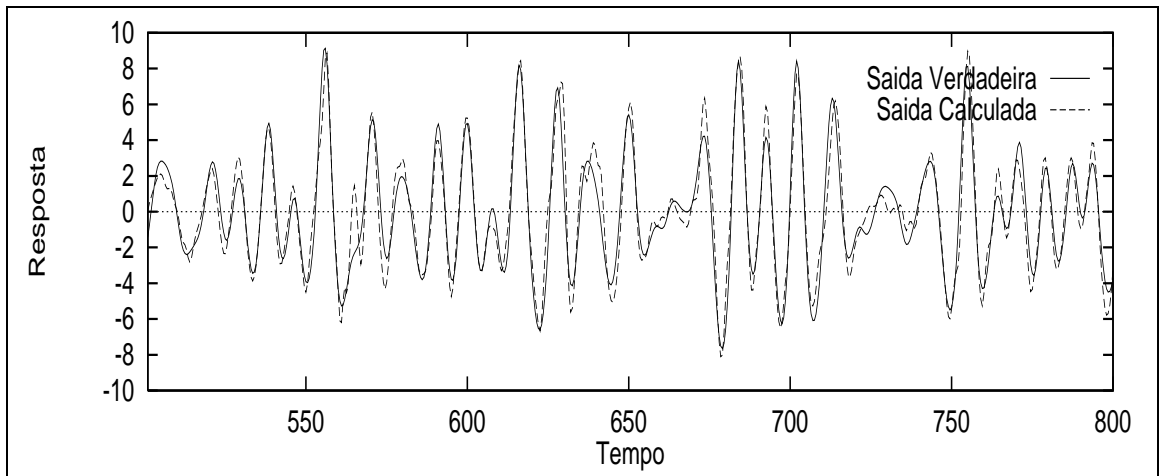


Figura 4.3: Caso 1 - 25 nós ocultos e 15s de atraso. Erro igual a $6.61E-04$.

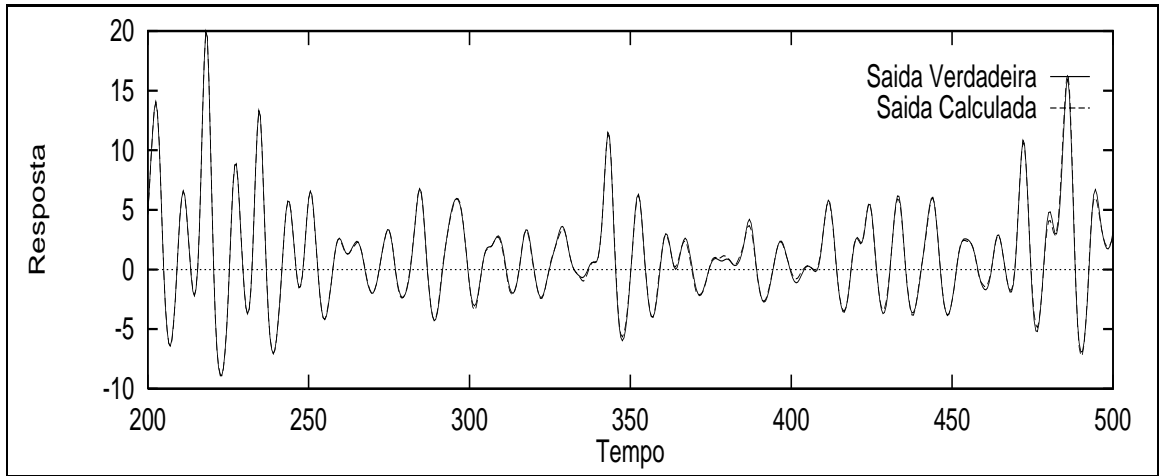


Figura 4.4: Caso 2 - 25 nós ocultos e 40s de atraso. Erro igual a $1.57E-04$.

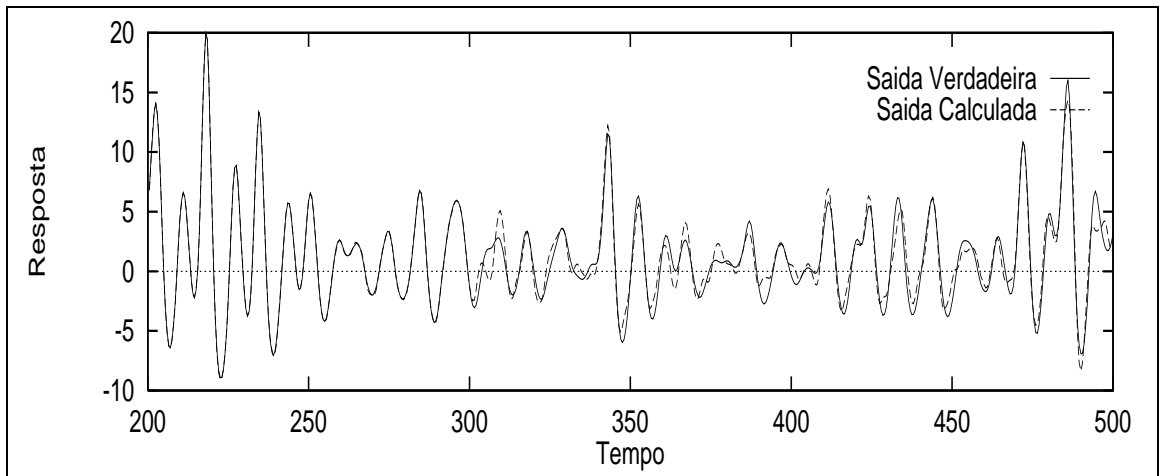


Figura 4.5: Caso 2 - 25 nós ocultos e 100s de atraso com erro igual a $6.96E-04$

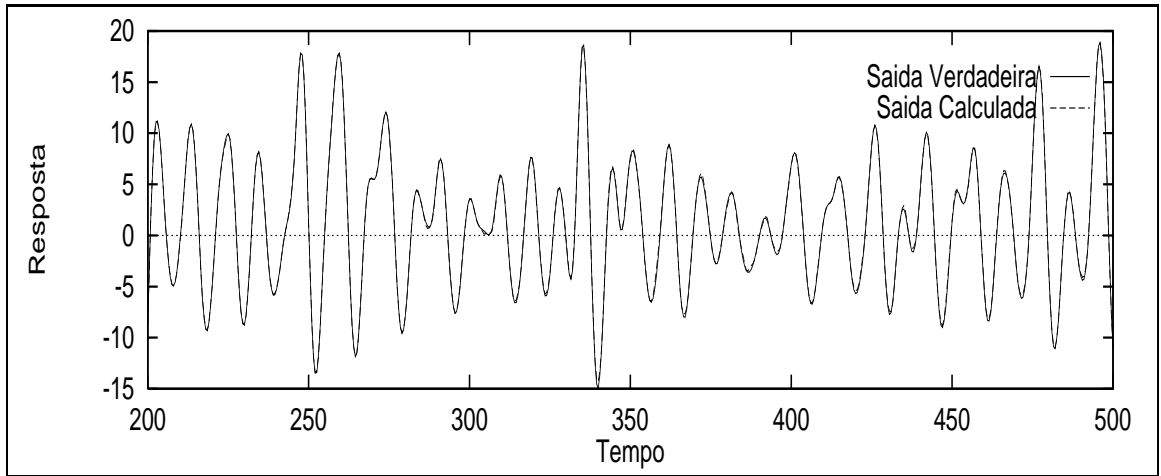


Figura 4.6: Caso 3 - 25 nós ocultos e 50s de atraso. Erro igual a $1.14E-05$.

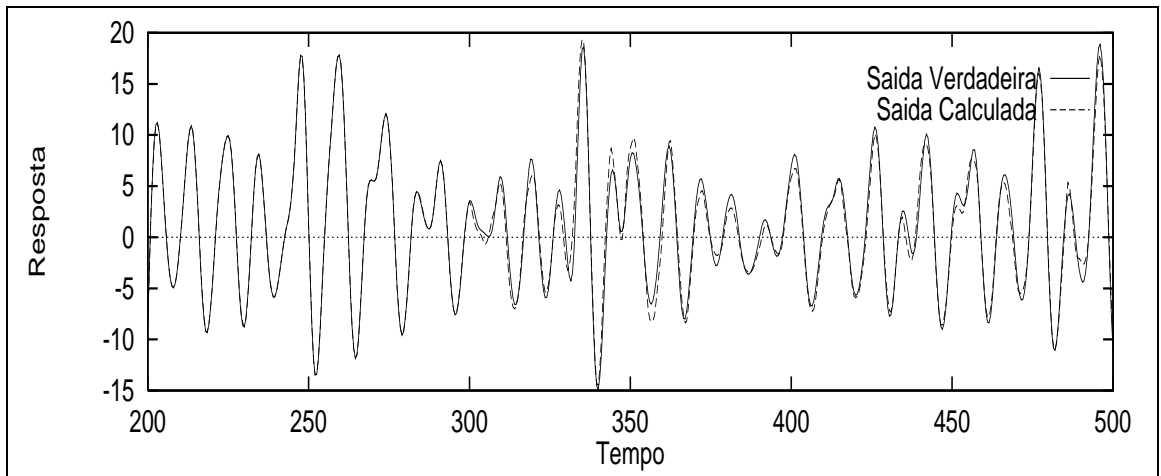


Figura 4.7: Caso 3 - 25 nós ocultos e 20s de atraso com erro igual a $1.67E-04$.

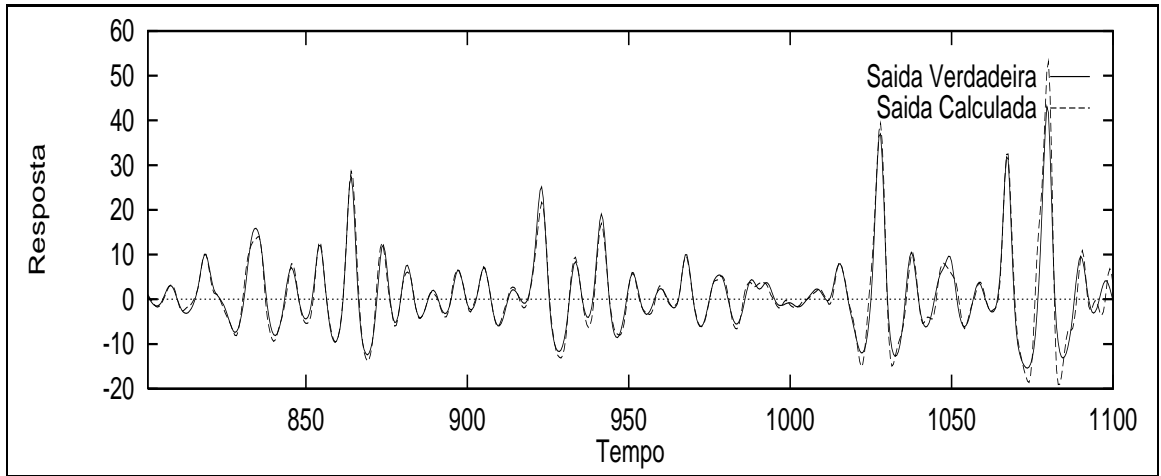


Figura 4.8: Caso 4 - 25 nós ocultos e 50s de atraso. Erro igual a 6.20E-04.

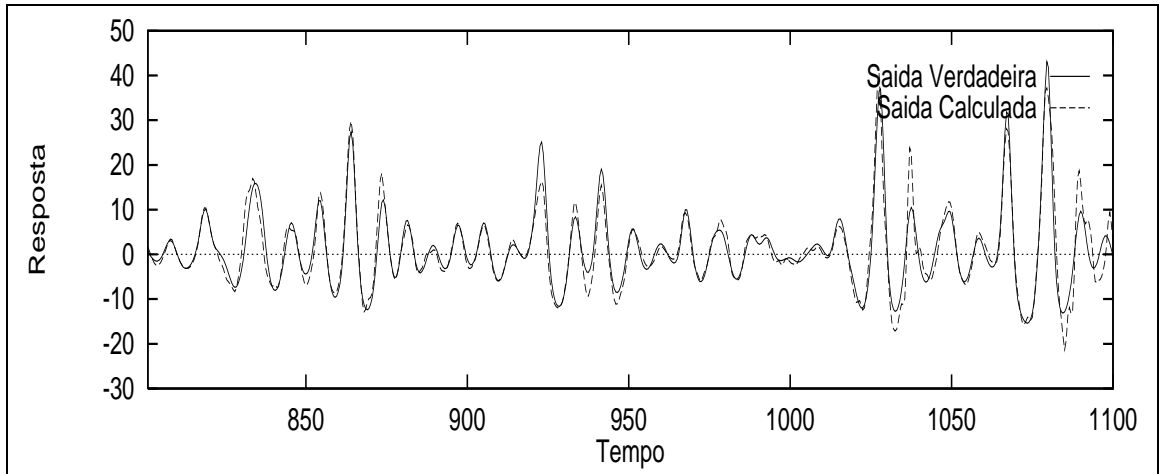


Figura 4.9: Caso 4 - 25 nós ocultos e 20s de atraso. Erro igual a 2.19E-03.

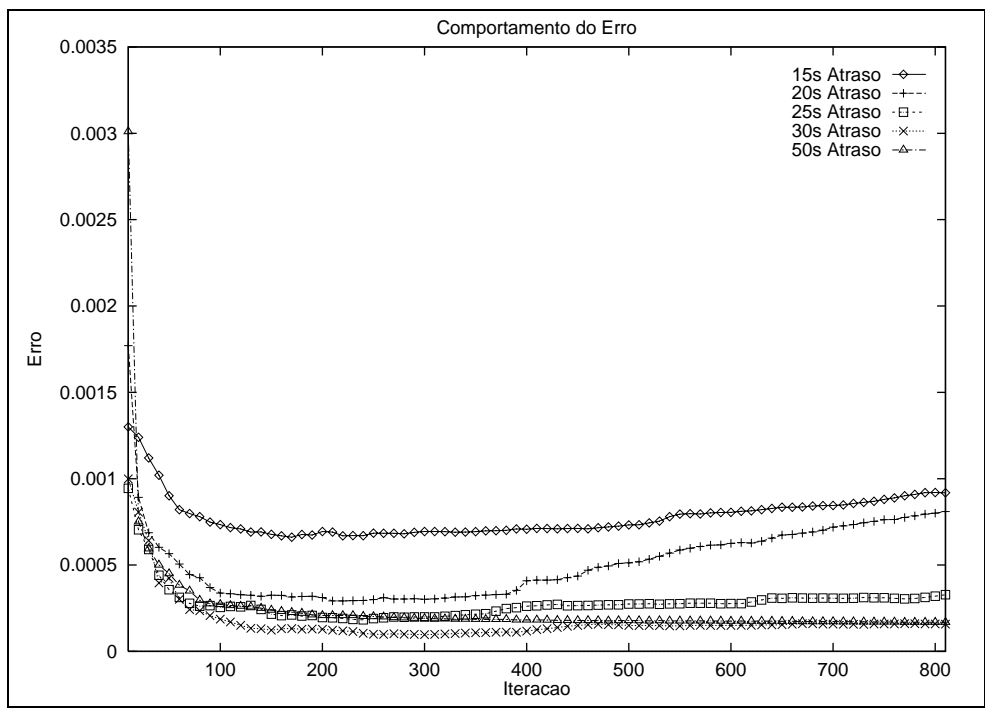


Figura 4.10: Erros encontrados para o Caso 1 variando-se o tempo de atraso (25 nós ocultos).

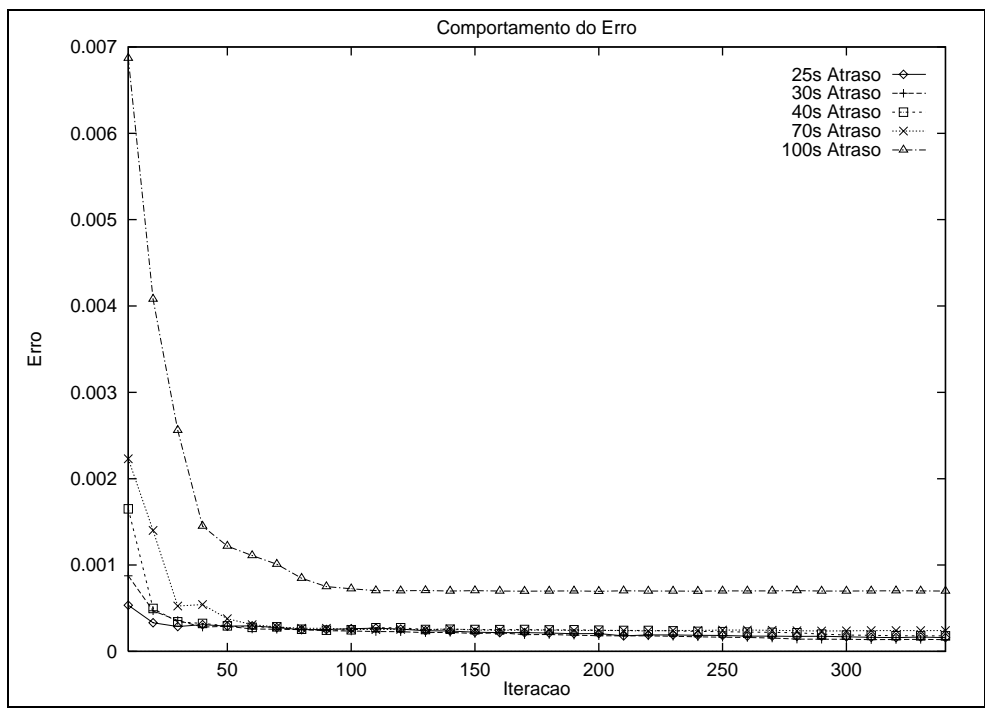


Figura 4.11: Erros encontrados para o Caso 2 variando-se o tempo de atraso (25 nós ocultos).

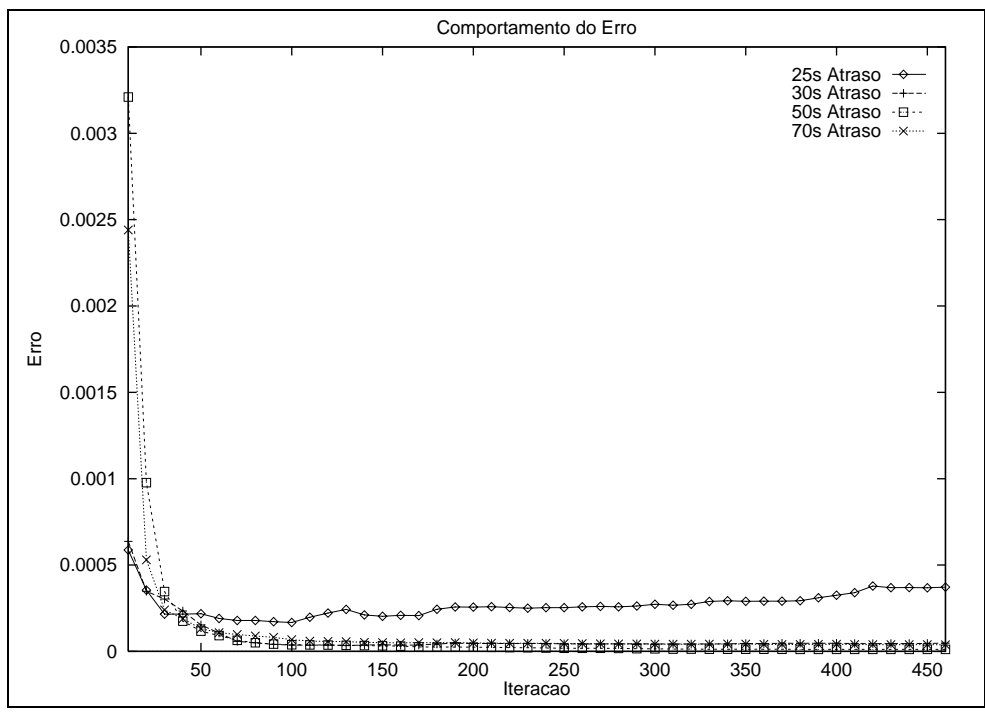


Figura 4.12: Erros encontrados para o Caso 3 variando-se o tempo de atraso (25 nós ocultos).

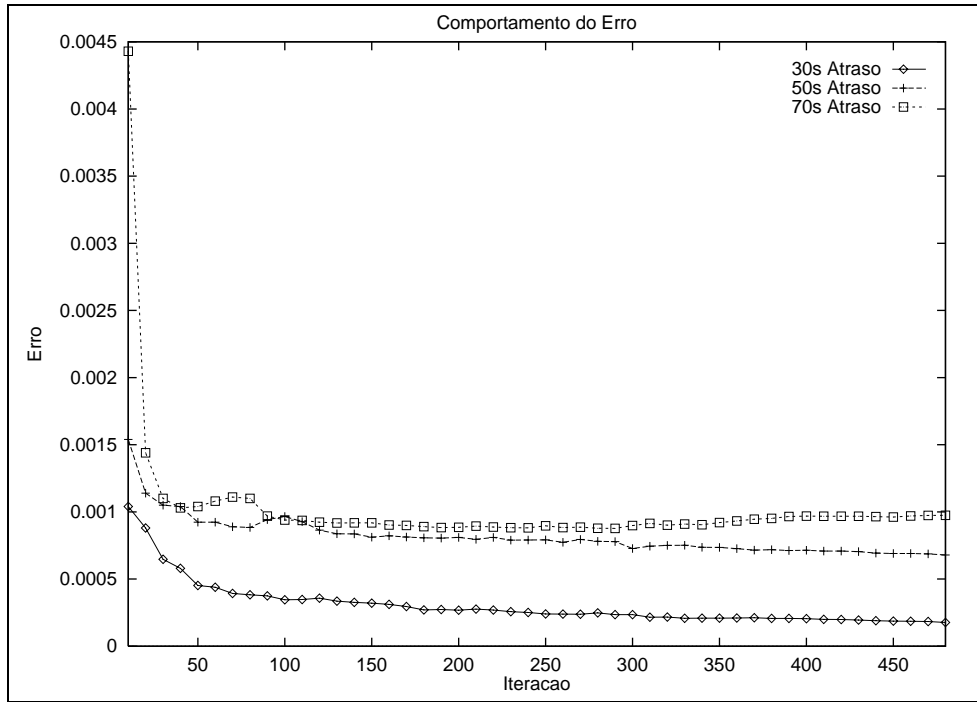


Figura 4.13: Erros encontrados para o Caso 4 variando-se o tempo de atraso (25 nós ocultos).

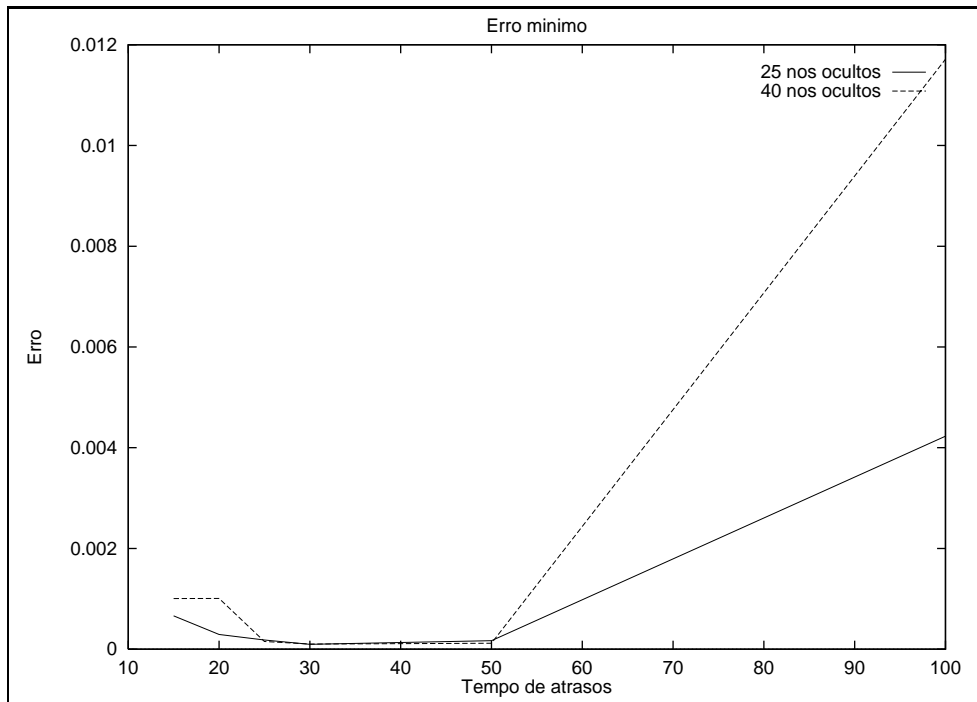


Figura 4.14: Erros mínimos obtidos para o Caso 1.

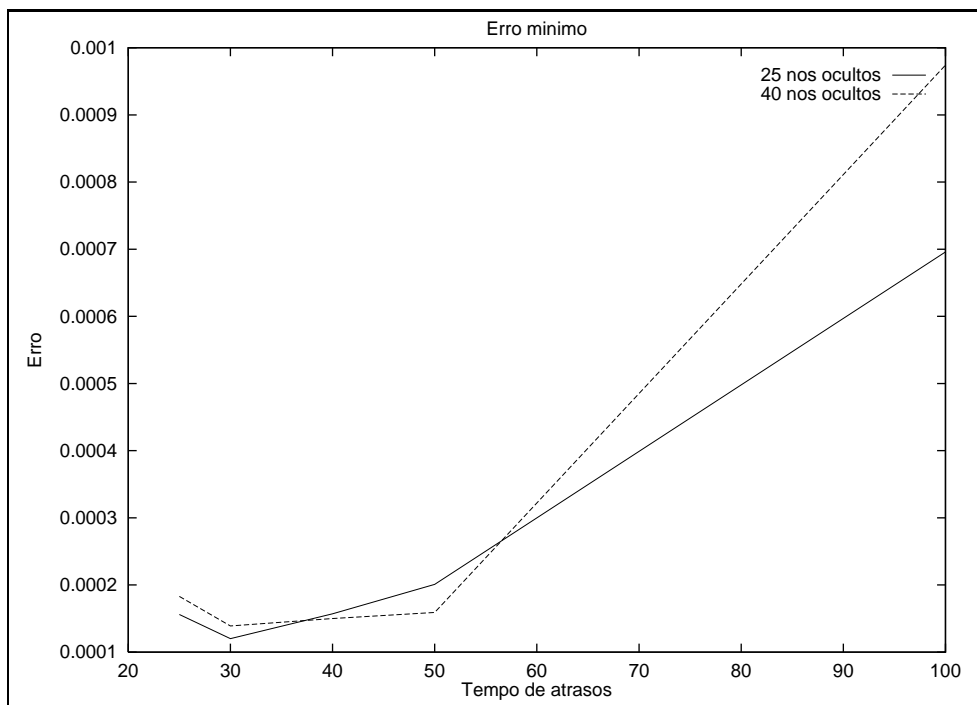


Figura 4.15: Erros mínimos obtidos para o Caso 2.

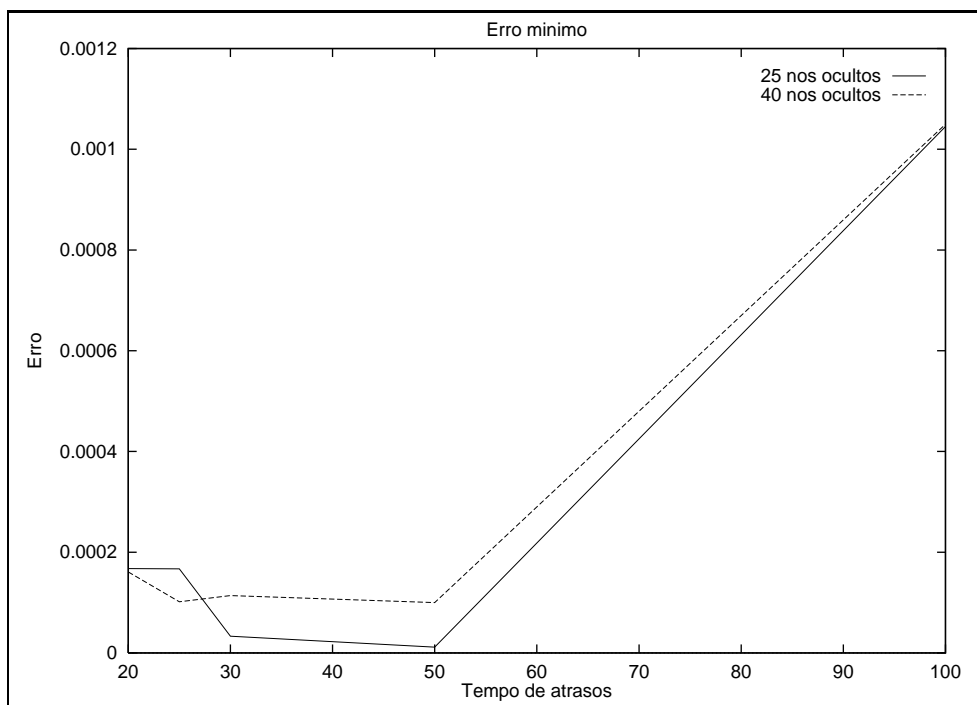


Figura 4.16: Erros mínimos obtidos para o Caso 3.

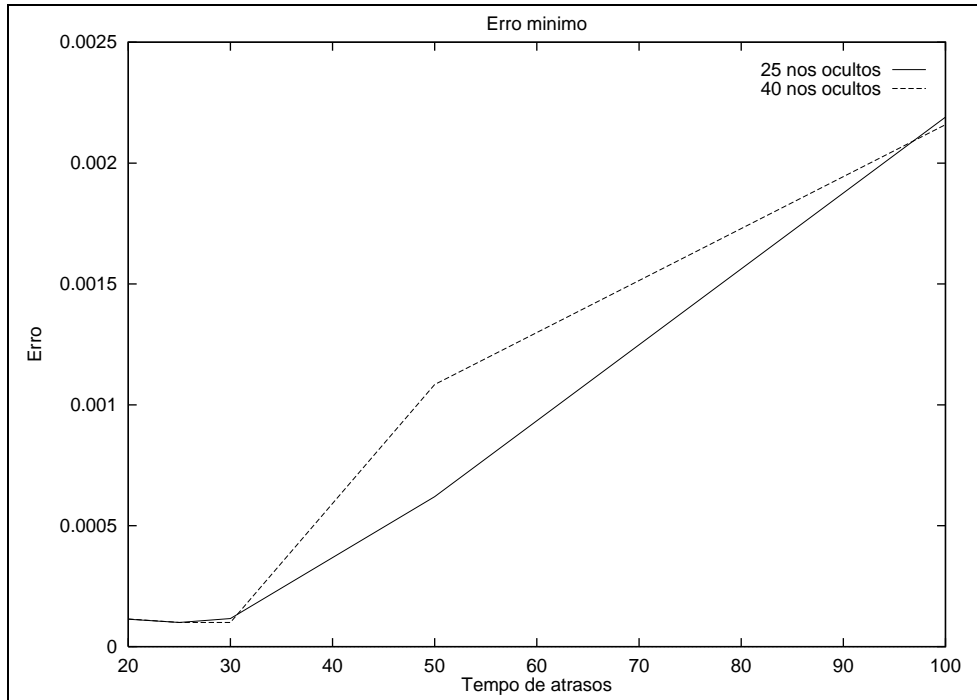


Figura 4.17: Erros mínimos obtidos para o Caso 4.

Em função dos resultados previamente apresentados, é possível identificar que uma rede neural pode representar muito bem o comportamento do sistema independentemente do grau de não-linearidade considerado. Entretanto, para que isto aconteça, o tamanho do atraso da excitação exerce um papel muito importante. Um tempo de atraso pequeno faz com que a rede não consiga generalizar para responder em situações futuras. Atrasos muitos longos "saturam" a rede e fazem com que a mesma não consiga generalizar. Observa-se também que o número de neurônios na camada oculta não tem grande influência no comportamento da rede.

Nas Figuras 4.18 a 4.21 são apresentadas predições da força de restauração dos sistema para uma outra realização obtida a partir de forças de excitação diferentes daquela usada no treinamento. Observa-se que uma vez que a rede esteja bem treinada, ela pode responder adequadamente para realizações diferentes da excitação.

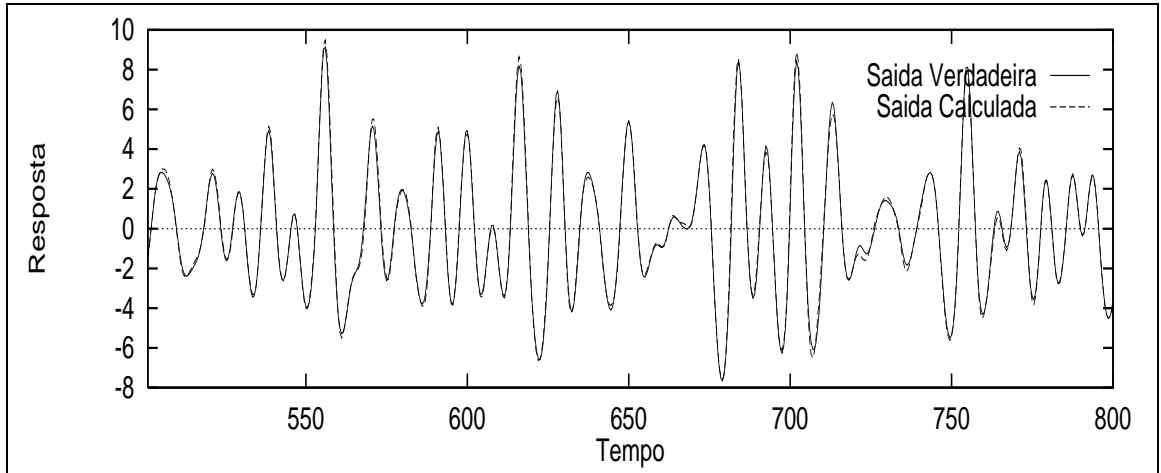


Figura 4.18: Caso 1 - Predição da resposta para uma realização das forças de excitação diferente da usada no treinamento.

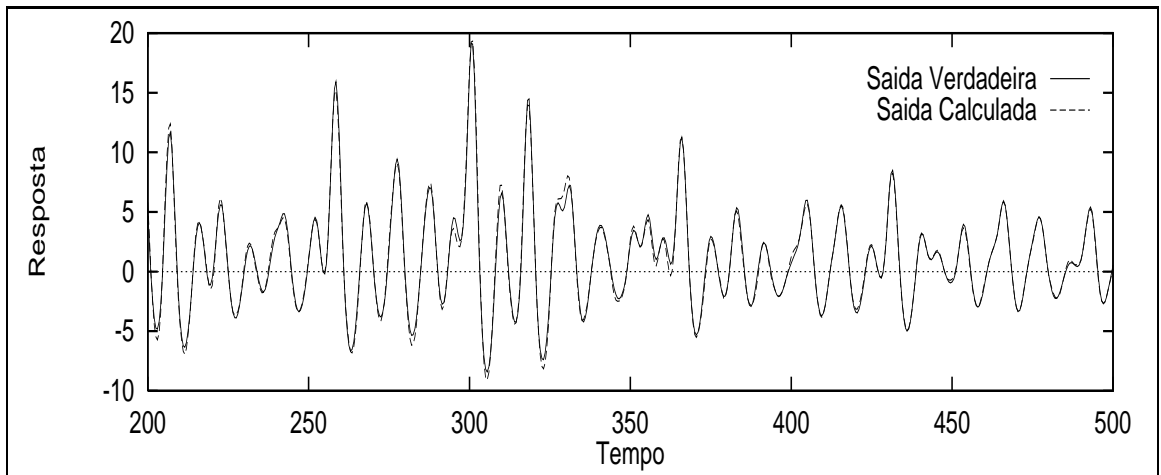


Figura 4.19: Caso 2 - Predição da resposta para uma realização das forças de excitação diferente da usada no treinamento.

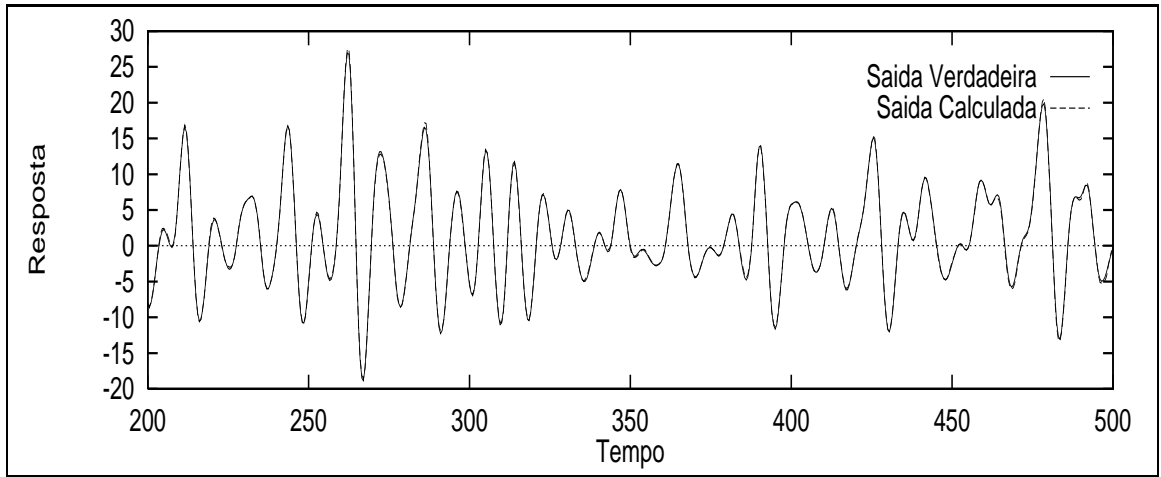


Figura 4.20: Caso 3 - Predição da resposta para uma realização das forças de excitação diferente da usada no treinamento.

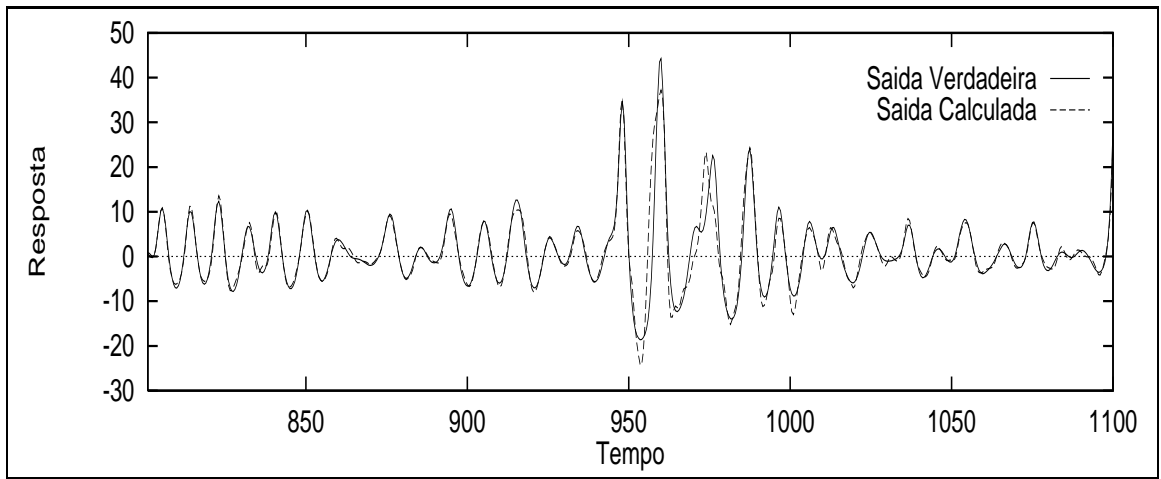


Figura 4.21: Caso 4 - Predição da resposta para uma realização das forças de excitação diferente da usada no treinamento.

4.2 Análise de uma linha de ancoragem

Este exemplo investiga a aplicabilidade de redes neurais na modelagem da dinâmica de um Caso real de uma linha de ancoragem, composta por trechos de amarras de cabo de poliéster, pertencente a um FPSO com sistema de ancoragem do tipo DICAS, projetado para uma lâmina d'água de 800m na Bacia de Campos. A linha analisada corresponde àquela mais solicitada numa condição ambiental extrema de projeto.

Os parâmetros de entrada do sistema (ou excitação) são movimentos do FPSO no topo da linha nas direções x (*surge*), y (*sway*) e z (*heave*) e a resposta dinâmica é a tração no topo. As séries temporais aleatórias consideradas, tanto de resposta como dos movimentos das unidades, correspondentes a duas realizações distintas, foram obtidas no Laboratório de Confiabilidade e Risco e Análise Aleatória (LCRAA) do Programa de Engenharia Civil da COPPE através de programas dedicados para este tipo de situação. As séries fornecidas são de 10800s de duração, já livres do transiente. Na Tabela 4.2 é apresentado um resumo estatístico das séries de movimentos e da resposta. Nas Figuras 4.22 a 4.24 observa-se uma janela das séries de movimento de uma das realizações utilizadas. Nestas figuras observa-se nitidamente a predominância de baixa frequência nos movimentos do plano horizontal (*surge* e *sway*) e movimentos de frequência mais alta no plano vertical (*heave*).

As Figuras 4.25 e 4.26 mostram uma janela das respostas de baixa e alta frequência para uma das realizações utilizadas.

No treinamento da rede neural utilizou-se os primeiros 200s das séries da realização 1 como conjunto de treinamento e o restante como conjunto de validação. Foi utilizado o método do gradiente conjugado para a obtenção dos pesos sinápticos. O método do gradiente descendente, quando converge, é muito lento.

Tabela 4.2: Estatística das séries de movimento e resposta

Par	Média	Desvio Padrão	Frequência de Cruzamento Zero (Hz)
x(m)	91.44	2.351	0.0130
y(m)	78.30	3.513	0.0097
z(m)	0	1.315	0.098
Ttopo(kN)	4194.00	707.00	0.0477

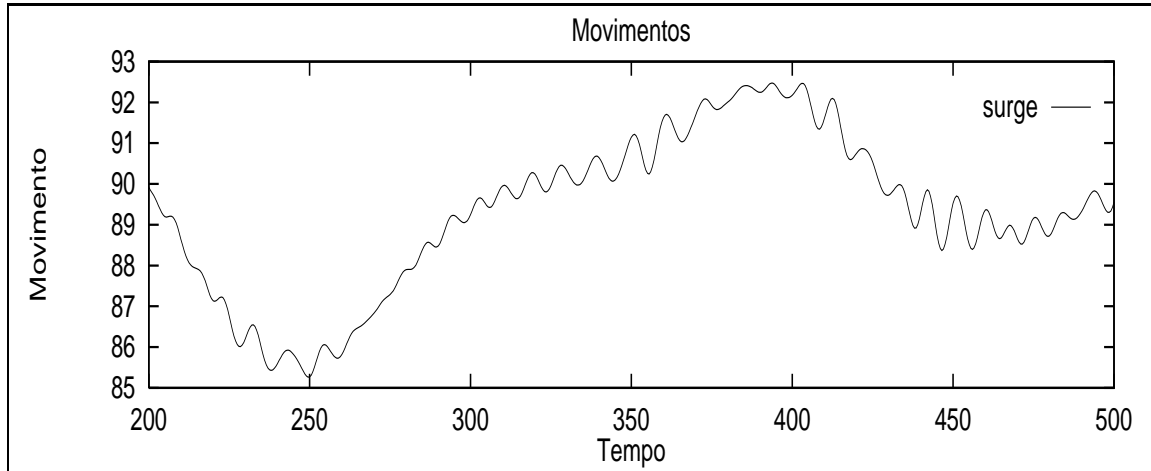


Figura 4.22: Séries de movimento de uma das realizações utilizadas - *surge*(m).

As Figuras 4.27 e 4.30 são mostrados alguns trechos da série temporal da resposta da realização 1 obtidos utilizando-se 25 neurônios na camada oculta e duas combinações de tamanhos de atrasos:

- 10s para *surge* e *sway* e 20s para *heave*
- 10s para *surge* e *sway* e 50s para *heave*

Devido a baixa frequência no plano horizontal é possível utilizar um tempo de atraso inferior ao atraso necessário para os movimentos de frequência mais alta.

Na Figura 4.31 apresenta-se a variação de erro da rede ao longo do processo de busca dos pesos sinápticos ótimos para alguns casos analisados. As Figuras 4.32 a

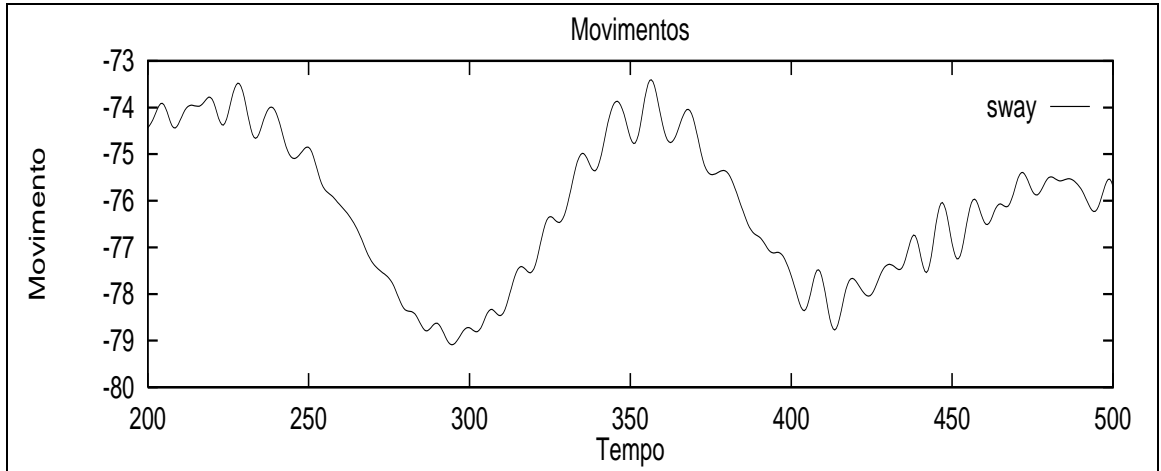


Figura 4.23: Séries de movimento de uma das realizações utilizadas - $sway(m)$.

4.33 apresentam o menor erro obtido para o conjunto de validação em função dos atrasos e do número de neurônios na camada oculta.

Conforme pode-se observar nos resultados apresentados nos gráficos, dependendo principalmente do tamanho considerado para os atrasos nas entradas, (movimentos prescritos), a rede consegue representar de forma satisfatória. O número de neurônios na camada oculta não é muito relevante, os erros de previsão, conforme esperado são mais significativos fora do intervalo de tempo utilizado para o treinamento da rede. Os maiores erros estão acontecendo justamente onde a resposta correta alcança os maiores picos.

Nas Figuras 4.34 e 4.37 são apresentadas partes de duas das quatro diferentes realizações, onde os valores das respostas foram obtidos pela rede neural treinada com a realização 1. Mais uma vez observa-se a capacidade da rede de representar adequadamente a resposta em uma realização diferente daquela usada no treinamento.

Nas Figuras 4.38 e 4.41 é mostrado o comportamento da rede utilizando apenas

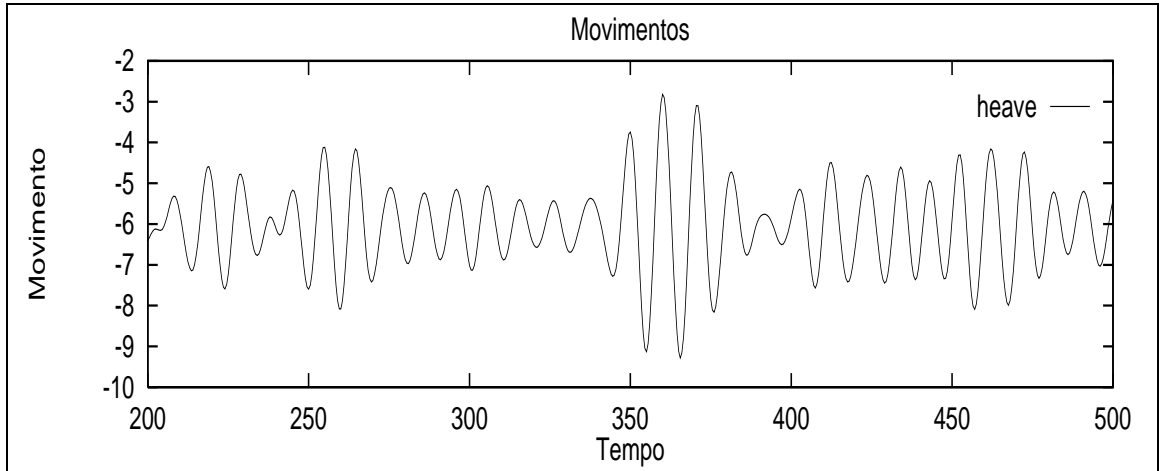


Figura 4.24: Séries de movimento de uma das realizações utilizadas - *heave*(m).

um conjunto de validação de 400s, 25 nós na camada oculta 10s de atrasos em *surge* e *sway* e 20s de atrasos em *heave* com variação no tamanho do conjunto de treinamento. Como podemos observar, o treinamento da rede após 200s não propicia um melhor aprendizado à rede, assim, para as avaliações realizadas, verificou-se que um tempo de treinamento de 200s e cerca de 300s de validação demonstrou representar adequadamente a modelagem da dinâmica de uma linha de ancoragem.

Em linhas gerais, observa-se que uma rede neural obtida com uma série "original" muito curta (cerca de 300s) é capaz de representar com uma precisão razoável a dinâmica da tração de topo num período de tempo muito longo.

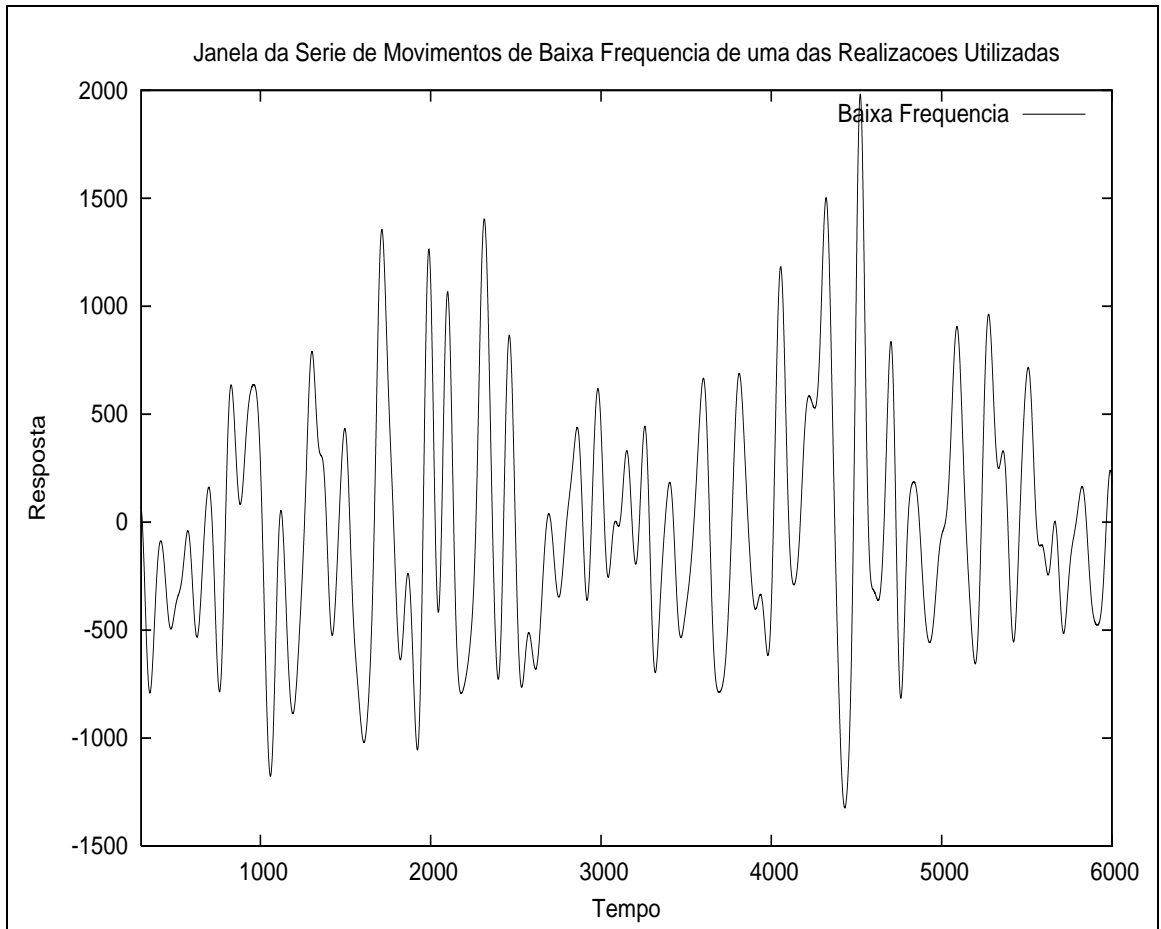


Figura 4.25: Janela da Serie de Movimentos de Baixa Frequncia de uma das Realizacoes Utilizadas - *surge* e *sway*(m).

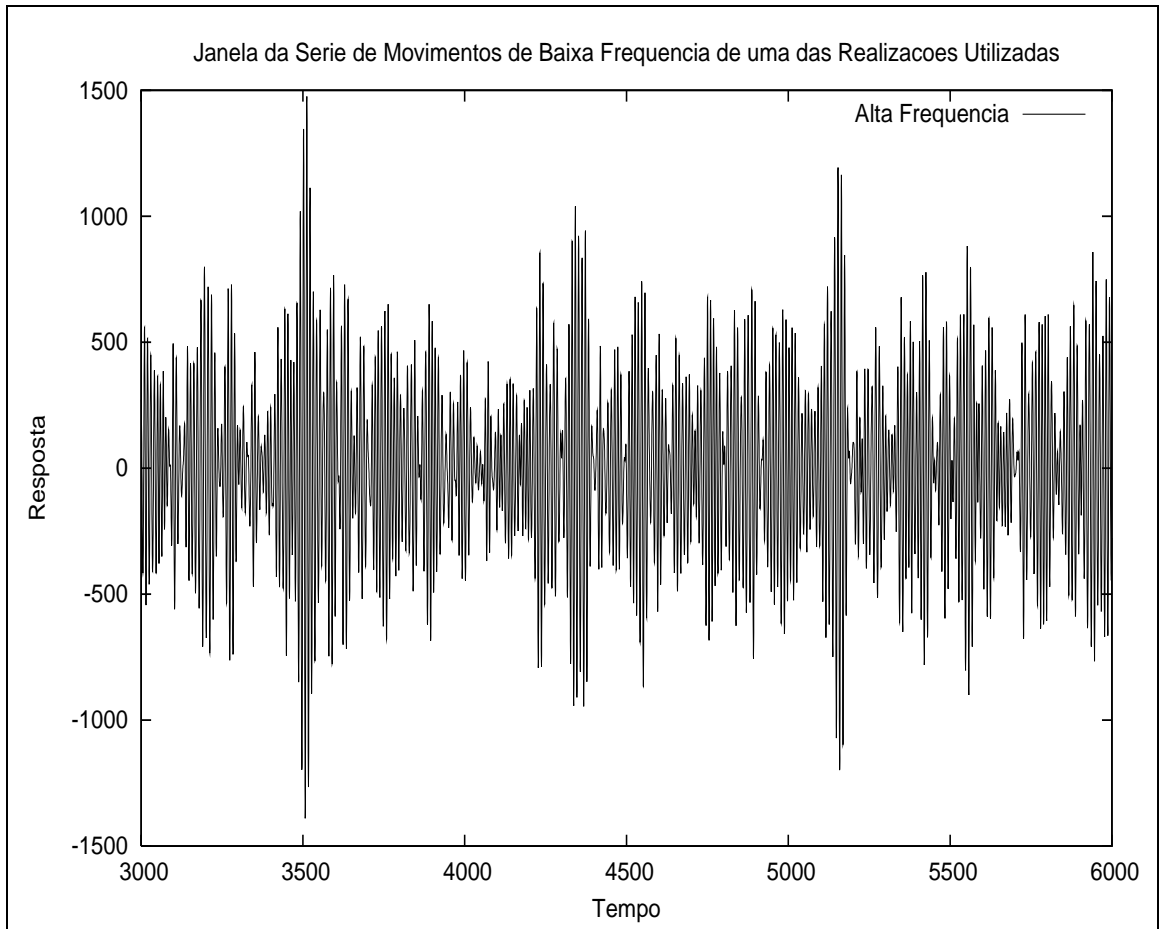


Figura 4.26: Janela da Serie de Movimentos de Alta Frequencia de uma das Realizacoes Utilizadas - *heave*(m).

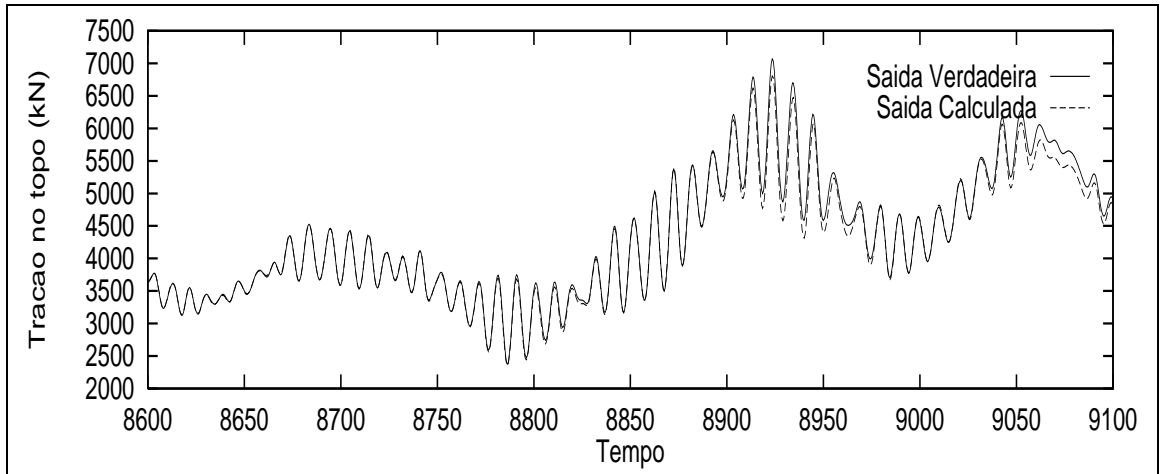


Figura 4.27: Ajuste encontrado considerando 25 nós na camada oculta, 10s de atraso para *surge* e *sway* e 20s para *heave*. Erro igual a 1.00E-04.

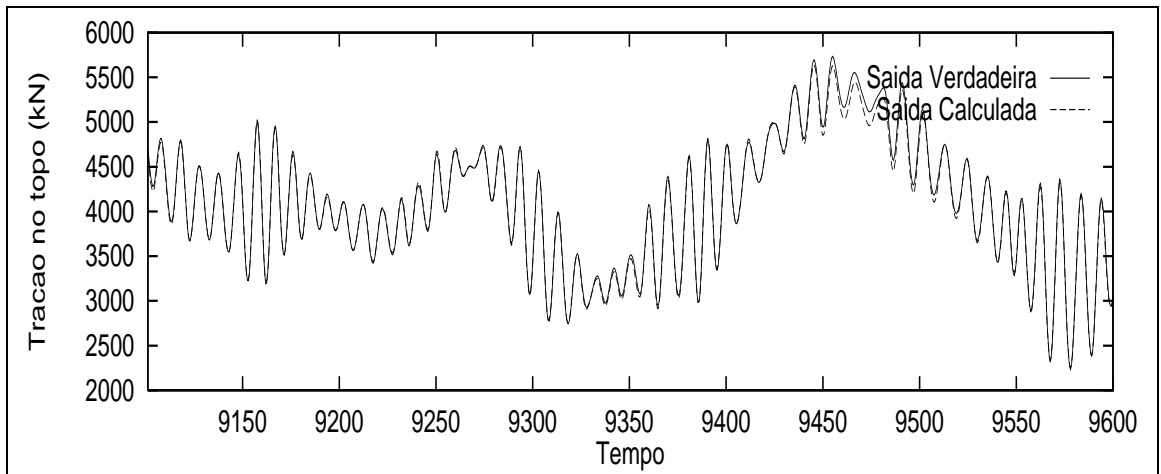


Figura 4.28: Ajuste encontrado considerando 25 nós na camada oculta, 10s de atraso para *surge* e *sway* e 20s para *heave*. Erro igual a 1.00E-04.

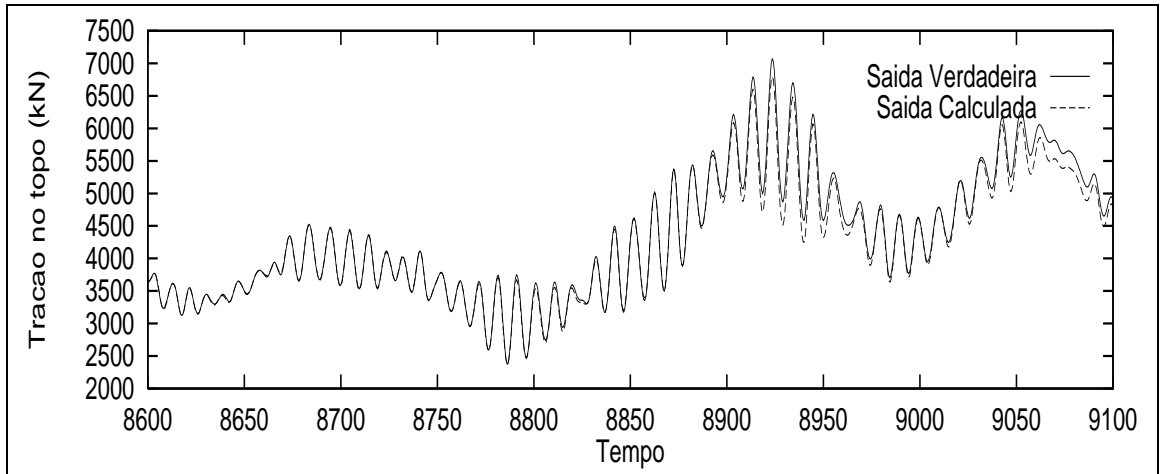


Figura 4.29: Ajuste encontrado considerando 25 nós na camada oculta, 10s de atraso para *surge* e *sway* e 50s para *heave*. Erro igual a 6.00E-04.

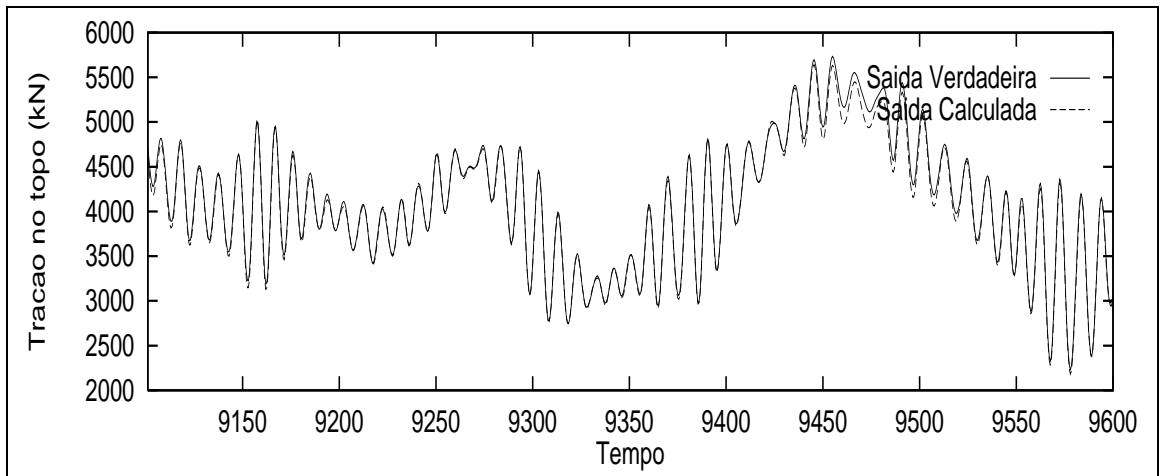


Figura 4.30: Ajuste encontrado considerando 25 nós na camada oculta, 10s de atraso para *surge* e *sway* e 50s para *heave*. Erro igual a 6.00E-04.

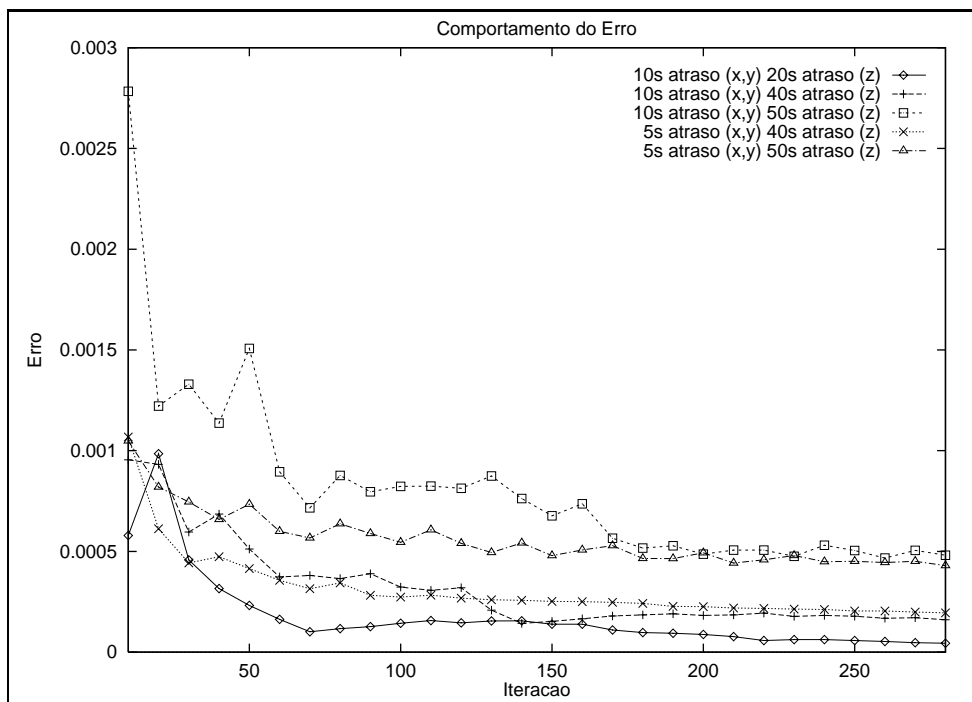


Figura 4.31: Variação do erro na rede com base no tamanho dos atrasos.

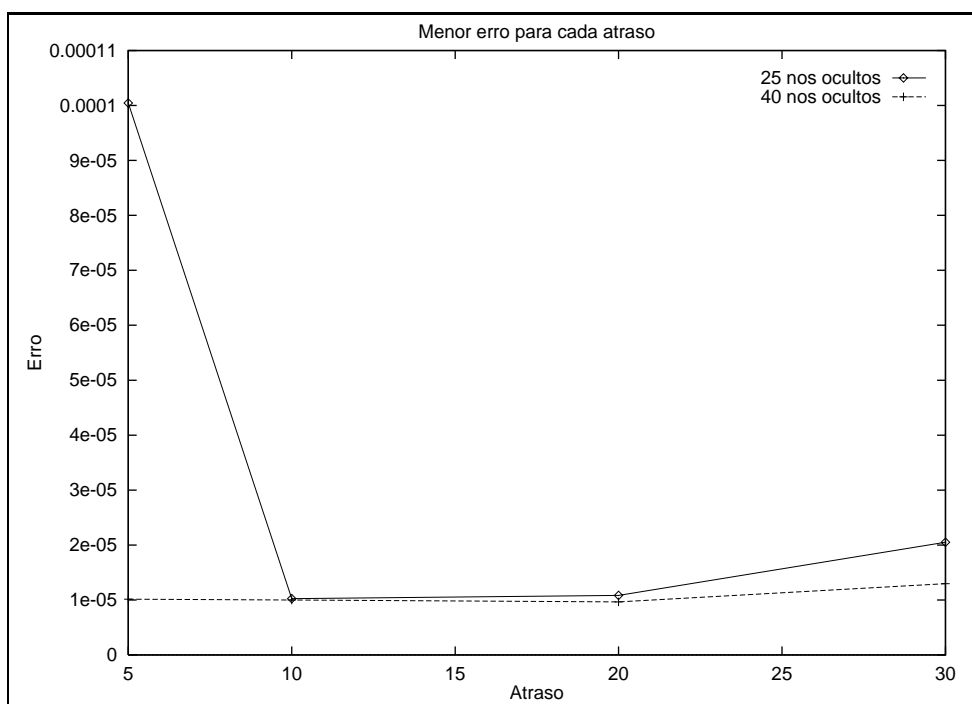


Figura 4.32: Erros para a rede com atraso de 5s para *surge* e *sway*.

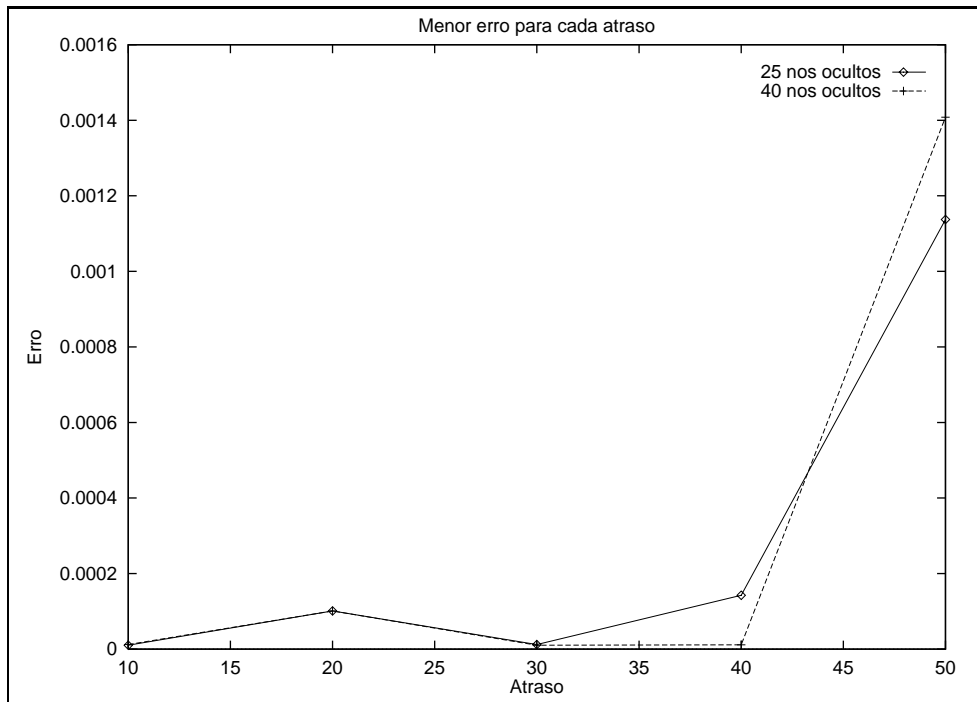


Figura 4.33: Erros para a rede com atraso de 10s para *surge* e *sway*.

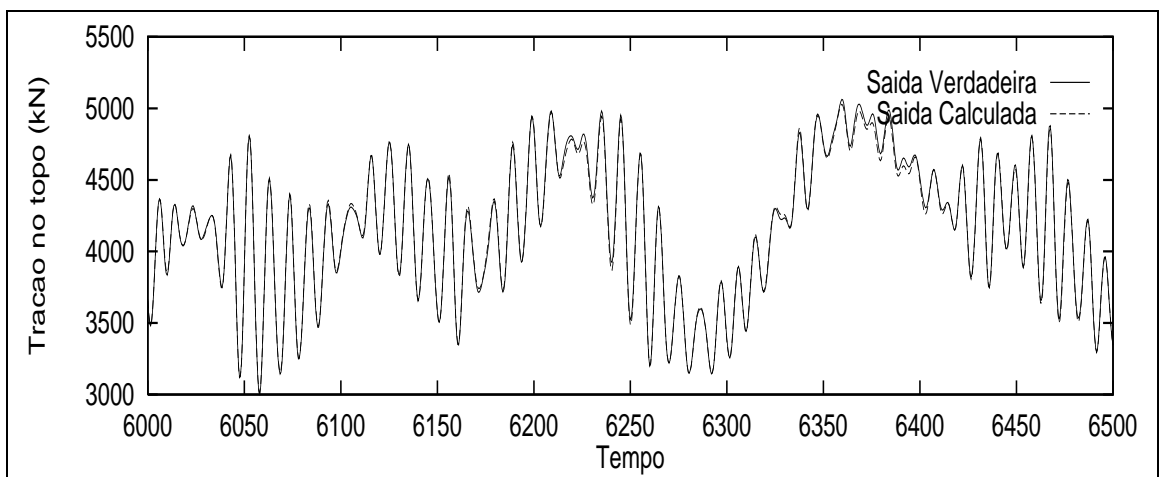


Figura 4.34: Saídas da rede neural para a realização 2.

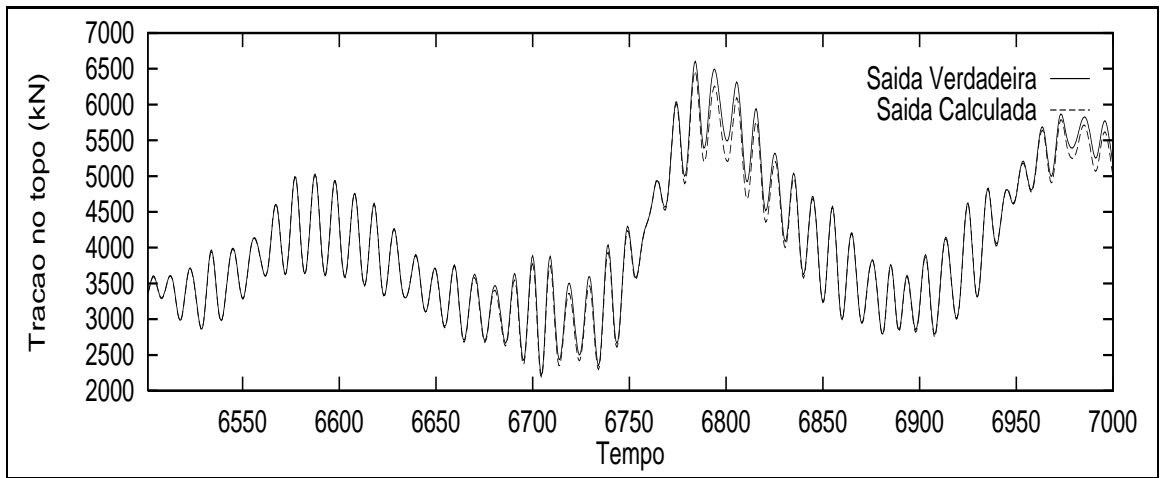


Figura 4.35: Saídas da rede neural para a relização 2.

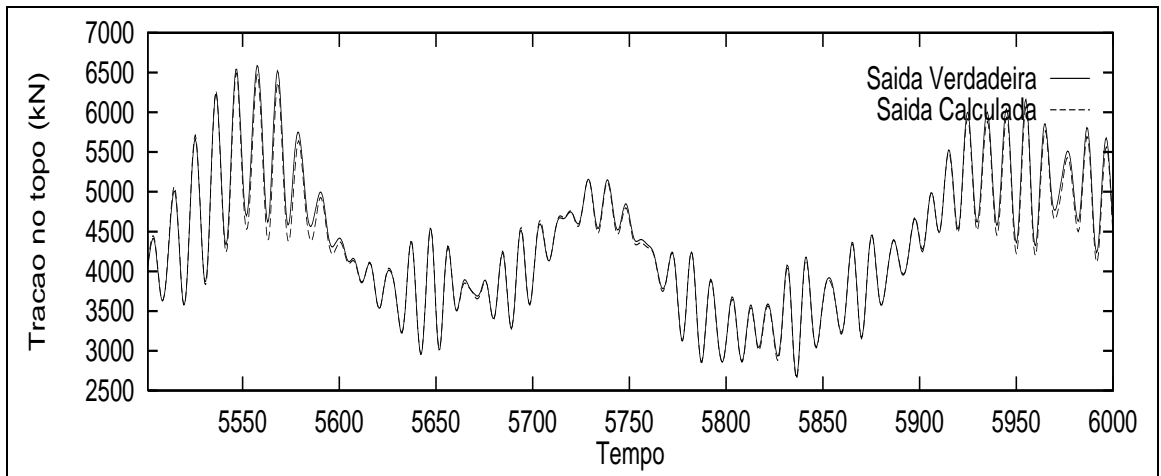


Figura 4.36: Saídas da rede neural para a relização 3.

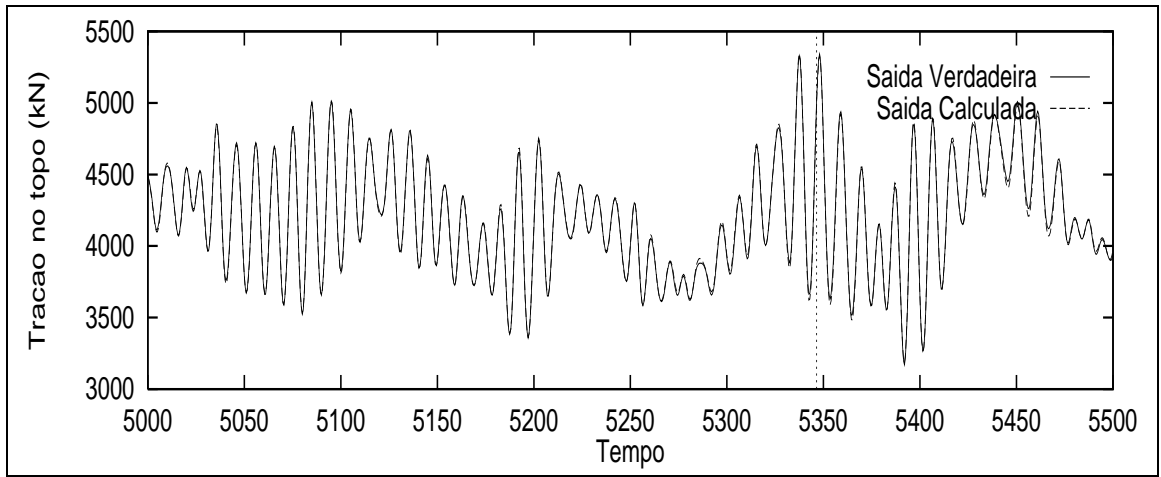


Figura 4.37: Saídas da rede neural para a realização 3.

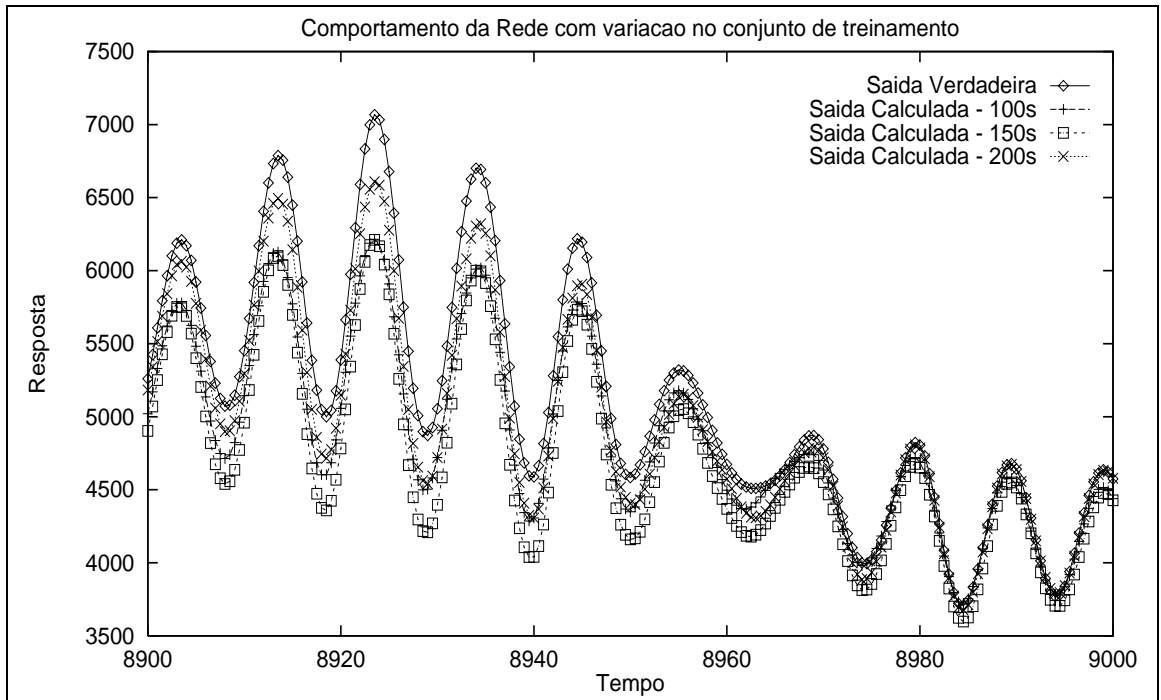


Figura 4.38: Comportamento da rede para um conjunto de treinamento variável.

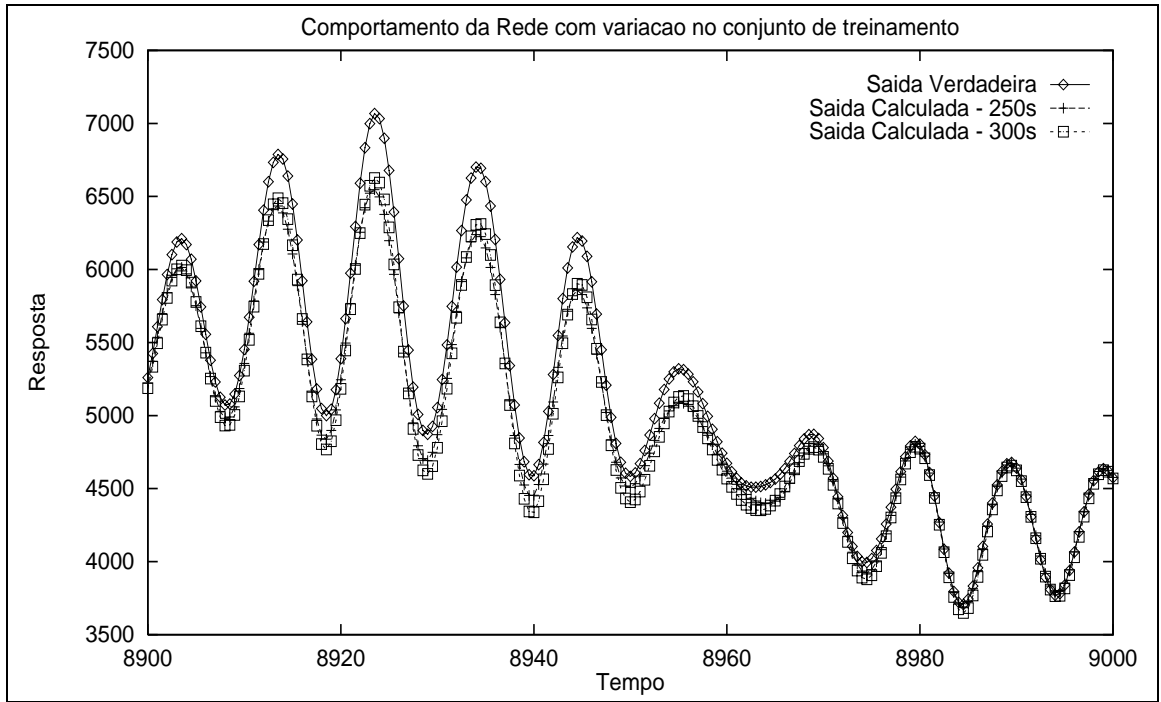


Figura 4.39: Comportamento da rede para um conjunto de treinamento variável.

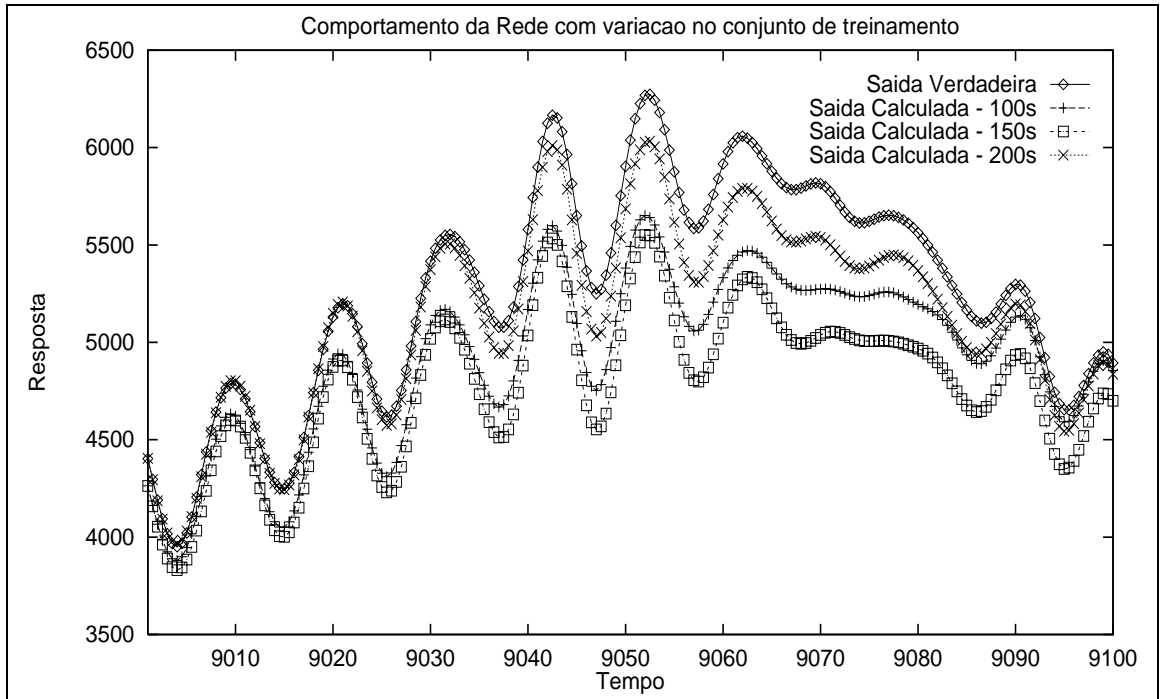


Figura 4.40: Comportamento da rede para um conjunto de treinamento variável.

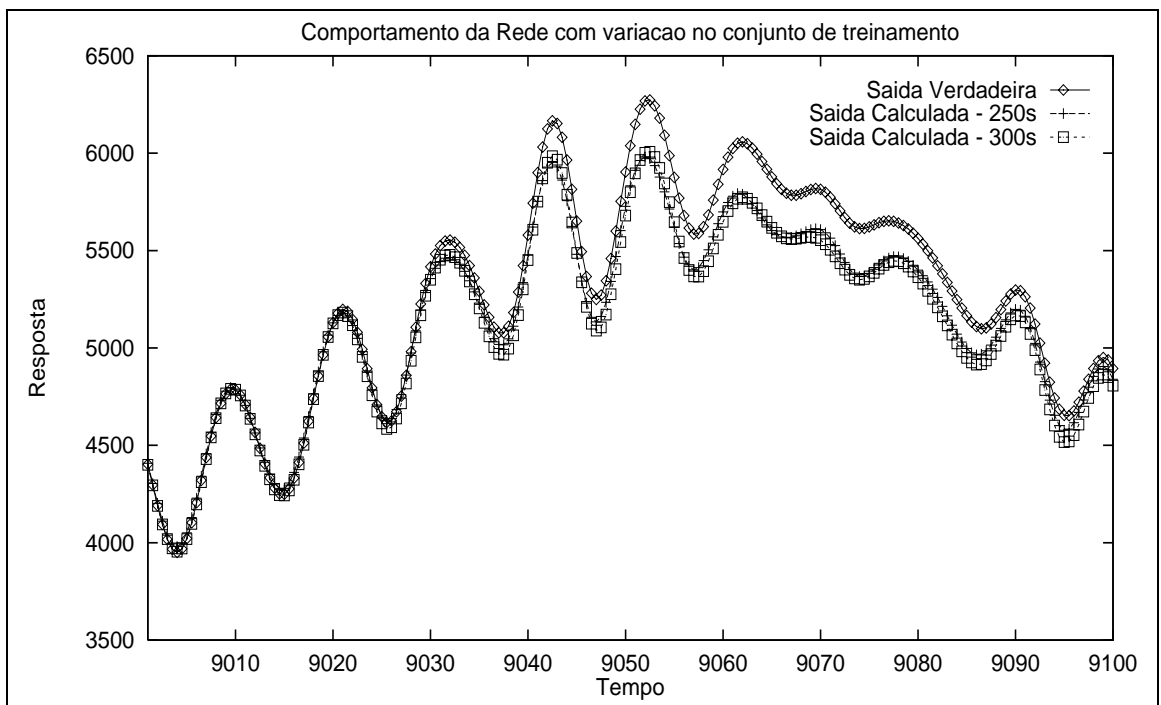


Figura 4.41: Comportamento da rede para um conjunto de treinamento variável.

Capítulo 5

Considerações Finais

Do ponto de vista matemático, uma rede neural artificial faz um mapeamento não-linear entre as entradas (excitação) e saídas (reposta) de um sistema dinâmico, onde as variáveis desconhecidas (incógnitas) são os pesos sinápticos das suas conexões. Os pesos sinápticos são definidos e através da solução de um problema de otimização onde minimiza-se o erro entre as saídas da rede e os valores corretos das mesmas para um dado conjunto de treinamento. Para solução de problemas de otimização, métodos como o gradiente conjugado e gradiente descendente são comumente usados, sendo que o primeiro apresenta maior robustas.

Através das análises realizadas neste trabalho, uma rede neural artificial, com uma camada oculta e uma saída linear, demonstrou ser uma ferramenta eficiente de indentificação de um sistema estrutural dinâmico não-linear.

Uma rede treinada com séries temporais curtas (200s e 300s) pode eficientemente prever respostas mais longas.

Um dos principais aspectos a ser considerado no treinamento da rede é o tamanho dos atrasos a serem considerados para as fontes de excitação. Por um lado, atrasos muito curtos podem ser insuficientes para representar adequadamente a solução e por outro, atrasos muito longos podem saturar a rede, conseqüentemente

prejudicando a solução. O número de neurônios na camada oculta tem uma influência menor sobre o comportamento global da rede.

5.1 Sugestões para próximos trabalhos

De uma forma geral, os resultados encontrados neste trabalho são motivadores para novos desenvolvimentos, visando a se chegar a um procedimento misto, computacionalmente mais baratos para análise dinâmica de estruturas complexas.

Neste sentido, para trabalhos futuros, a sugestão seria implementar um algoritmo, que nos permitisse através de refinamentos sucessivos encontrar o número ótimo de unidades intermediárias e o ideal número de atrasos, eliminando assim a subjetividade e os componentes empíricos inerentes as redes neurais. Este passo envolve também a atribuição de um tratamento estatístico do erro resultante da aproximação.

A solução do problema de otimização inerente ao erro por redes neurais é normalmente feita através de algoritmos baseados em gradientes que são de forma geral de convergência lenta. Desta forma, outros métodos de otimização baseados em algoritmos genéticos poderiam ser investigados.

Referências Bibliográficas

- [1] S.A. BILLINGS. Identification of non-linear systems a survey. In *IEEE Proceedings Pt.D*.
- [2] C.C.S. CASSINI and L.A. AGUIRRE. Uma introdução à identificação de sistemas não lineares. Master's thesis, Universidade Federal de Minas Gerais, 1999.
- [3] S. CHEN and S.A. BILLINGS. Representation of non-linear systems: the nar-max model. *International J. Control*, 1989.
- [4] S.A. CHEN, S.and BILLINGS. Recursive prediction error parameter estimation for non-linear models. *International Journal Control*, 1989.
- [5] R.W. CLOUGH and J. PIENZIEN. *Dynamics of structures*. McGraw-Hill, 1975.
- [6] G. CYBENKO. Approximation by superpositions of a sigmoidal function. *Mathematical Control, Signals, and Systems*.
- [7] S. GROSSBERG. *Neural networks and natural intelligence*. MIT Press, 1988.
- [8] S. HAYKIN. *Neural networks - A comprehensive foundation*. Prentice Hall, 2001.
- [9] J.J HOPFIELD. Neural networks and physical system with emergent collective properties. In *Proc. Nat. Acad. Sci.*, pages –, 1992.

- [10] K. HORNIK. Some new results on neural network approximation. *Neural Networks*, 6(8):1069–1072, 1993.
- [11] K.J. HUNT, D. SBARBARO, R. ZBIKOWSKI, and P.J. GAWTHROP. Neural networks for control systems – a survey. *Automatica*, 28(6):1083–1112, 1992.
- [12] T. KOHONEN. Self-organised information of topologically correct feature maps. *Biological Cybernetics*, pages –, 1982.
- [13] T. KOHONEN. *Self-Organizing Maps – Second Edition*. Springer Verlag, 1997.
- [14] I. LECUN. *Modeles connexionnistes de l'apprentissage*. PhD thesis, Universite de Paris VI, France, 1987.
- [15] I.J. LEONTARITIS and S.A. BILLINGS. Input-output parametric models for non-linear systems. part i: Deterministic non-linear systems. part ii: Stochastic non-linear systems. *International Journal Control*, 41(2):304–344, 1985.
- [16] I. MASETTI. *Analise dinamica de navios ancorados em complacencia diferenciavel*. PhD thesis, COPPE, 1995.
- [17] J. MCCARTHY. A basis for mathematical theory of computation. In *Computer Programming and Formal Systems*, pages 33 – 70, 1963.
- [18] W.S. MCCULLOCH and W. PITTS. A logical calculus of the ideas immanent in nervous activities. *Bulletin of Mathematical Biophysics*, 5:115–133, 1943.
- [19] M.L. MINSKY. Step towards artificial intelligence. In *Proceedings for the Institute of Radio Engineers*, pages 08 – 30, 1961.
- [20] M.L. MINSKY and S.A. PAPER. *Perceptrons*. MIT Press, 1969.
- [21] A. NEWELL and H.A. SIMON. *Human Problem Solving*. Prentice-Hall, 1972.
- [22] F. ROSENBLATT. Perceptron simulation experiments. In *Proceedings of the Institute of Radio Engineers*, pages 301 – 309, 1960.

- [23] D.E RUMELHART and J.L. MCCLELLAND. *Parallel Distributed Processing, Vol1:Foundations*. MIT Press, 1986.
- [24] J. SJÖBERG. *Non-linear systems identification with neural networks*. PhD thesis, Department of Electrical Engineering, Linköping University, Sweden, 1995.
- [25] J. SJÖBERG, Q. ZHANG, L. LJUNG, A. BENVENISTE, B. DEYLON, P. GLORENNEC, H. HJALMARSSON, and A. JUDITSKY. Nonlinear black-box modeling in system identification: a unified overview. Technical report, 1995.
- [26] R.S. SUTTON and A.G. BARTO. *Reinforcement Learning: An Introduction (Adaptive Computation and Machine Learning)*. MIT Press, 1998.
- [27] P.J. WERBOS. *Beyond regression: New tools for prediction and analysis in the behavior science*. PhD thesis, Harvard University, Committee on Applied Mathematics, 1974.
- [28] B. WIDROW and M.E. HOFF. Adaptive switching circuits. In *IRE WESCON Convention Record*, pages 96 – 104, 1960.