2
ANÁLISE DE SÉRIES TEMPORAIS

2.1
Introdução

Este capítulo trata de uma revisão teórica sobre séries temporais, métodos estatísticos baseados nos modelos, redes neurais artificiais (RNA´s), redes neurais temporais (RNT´s) e métodos geoestatísticos, com enfoque nos conceitos teóricos utilizados neste trabalho.

2.2
Séries Temporais

Série temporal é um conjunto de observações ordenadas no tempo, não obrigatoriamente igualmente espaçadas, que apresentam dependência entre instantes de tempo. Uma série temporal pode ser decomposta nas componentes de **tendência**, **ciclo** e **sazonalidade** (Gutiérrez, 2003).

A **tendência** indica o comportamento "de longo prazo" da série, ou seja, se a série aumenta, diminui ou permanece estável, e qual a velocidade destas mudanças.

Os **ciclos** indicam as oscilações de subida e de queda nas séries, de forma suave e repetida, ao longo da componente de tendência, como por exemplo, os ciclos meteorológicos.

A **sazonalidade** corresponde às oscilações de subida e de queda que ocorrem sempre em um mesmo período.

O estudo da série temporal pode considerar:

a) investigação do mecanismo gerador da série temporal;

b) descrição do comportamento da série;

c) busca de periodicidades relevantes nos dados;

d) previsões de valores futuros da série.

A previsão de valores futuros da série temporal é amplamente utilizado nas áreas de economia, planejamento de produção, controle e otimização de processos industriais, planejamento comerciais entre outros.

Como exemplo de metodologia de previsão de valores futuros da série temporal temos: uma série temporal discreta e com os valores espaçados igualmente no tempo, com intervalo semanal. Utilizando-se o valor da série s_t da semana atual t e os valores das semanas anteriores $s_{t-1}, s_{t-2}, s_{t-3}, ...$ para previsão do valores futuros $\hat{s}_{t+1}, \hat{s}_{t+2}, \hat{s}_{t+3}, ..., \hat{s}_{t+8}$ que representam os passos à frente $l = 1,2,3,...,8$. Nosso objetivo é obter um modelo da série temporal que apresente o menor erro possível para cada passo previsto. Além da melhor previsão, podem-se calcular as incertezas associadas à previsão associando os valores previstos a limites de probabilidade (Box, Jenkins & Reinsel, 1994).

A figura 2.1 mostra uma série temporal composta de 28 valores e as previsões feitas a partir da origem t com 8 passos à frente, bem como o intervalo de confiança com probabilidade de 95%.

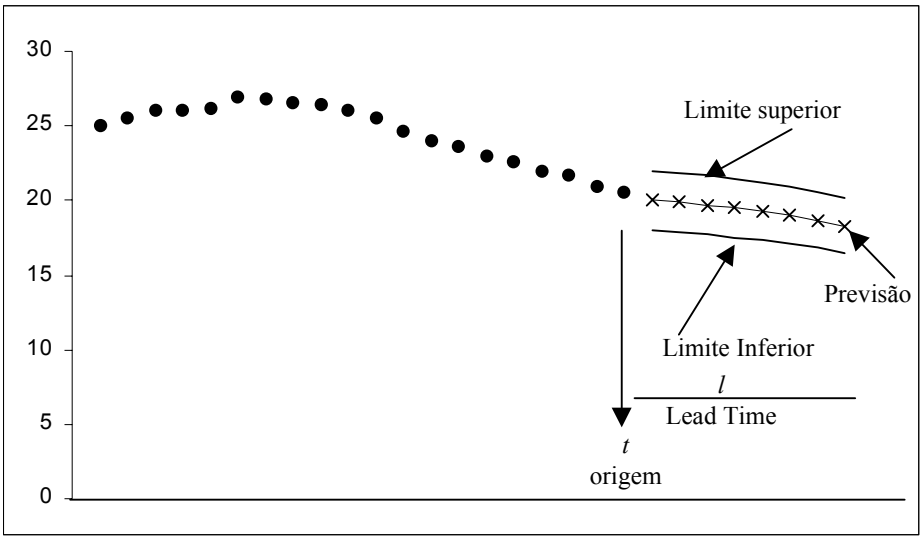

Figura 2.1 Valores de série temporal com previsão e intervalo de confiança com probabilidade de 95% (Box, Jenkins & Reinsel, 1994).

2.3
Modelos Arima de Box & Jenkins

Os modelos ARIMA são modelos estatísticos lineares para análise de séries temporais. A abreviação ARIMA em inglês significa “Auto-Regressive Integrated

Moving Average model", ou seja, auto-regressivo, integrado e médias móveis. Os termos auto-regressivos correspondem a defasagens da série e as médias móveis são as defasagens dos erros aleatórios. Integrado é o processo de diferenciação da série original para torná-la estacionária.

Na série temporal estacionária não existe tendência de crescimento ou decrescimento dos dados, ou seja, os dados oscilam sobre uma média constante e com a variância das flutuações constante. Quando a série é estacionária, a distribuição de probabilidade conjunta associada com n observações s_{t1}, s_{t2}, s_{t3},..., s_{tn} observadas em qualquer conjunto de tempo t_1, t_2, t_3,..., t_n é a mesma que a associada com outras n observações s_{t1+k}, s_{t2+k}, s_{t3+k},..., s_{tn+k} observadas nos tempos $t_1 + k$, $t_2 + k$, $t_3 + k$,..., $t_n + k$. A distribuição de probabilidade conjunta de qualquer conjunto de amostras não pode ser alterada pela modificação de todas observações da série temporal para frente ou para trás por qualquer valor inteiro k (Box, Jenkins & Reinsel, 1994).

No processo de diferenciação, a série com comportamento não-estacionário é transformada em uma nova série estacionária. Já que o modelos ARIMA assumem estacionariedade da série. A equação 2.1 indica a primeira diferença.

$$s_t^{'} = s_t - s_{t-1} \tag{2.1}$$

onde s_t é a série temporal não-estacionária com n valores

$s_t^{'}$ é a série nova estacionária que terá n-1 valores.

No modelo ARIMA, a série temporal é gerada por um processo estocástico cuja natureza pode ser representada através de um modelo. A notação utilizada para designar o modelo ARIMA é ARIMA (p, d, q) onde p é o número de termos auto-regressivos, d o número de diferenciações para que a série torne-se estacionária e q o número de termos de médias móveis. Os termos p, d e q são todos inteiros maiores ou iguais a zero.

São casos particulares: o modelo ARMA(p,q), o modelo auto-regressivo AR(p) e o modelo de médias móveis MA(q), sendo os três modelos utilizados para séries temporais estacionárias (d=0).

O modelo auto-regressivo AR(p) é definido por,

$$S(t) = \sum_{i=1}^{p} \alpha_i S(t-i) + \varepsilon(t) = \phi_L(S(t-1),...,S(t-p)) + \varepsilon(t) \quad (2.2)$$

onde a estimativa da série *S* no instante t depende de uma combinação linear de p termos da série, incluindo o termo aleatório ε(t) de ruído branco. Os parâmetros α_i ponderam os valores da série temporal do instante anterior *t-1* até o instante *t-p*. Estes parâmetros são determinados através de técnicas de minimização do erro.

O modelo de médias móveis MA(q) assume que a série modelada é gerada através de uma combinação linear de q sinais de ruídos $\varepsilon(t-i)$, aleatórios e independentes entre si,

$$S(t) = -\sum_{i=1}^{q} \theta_i \varepsilon(t-i) + \varepsilon(t) = \phi_L(\varepsilon(t-1),...,\varepsilon(t-q)) + \varepsilon(t) \quad (2.3)$$

O modelo ARMA(p,q) é uma combinação dos modelos AR(p) e MA(q), no qual

$$S(t) = \sum_{i=1}^{p} \alpha_i S(t-i) + \sum_{i=1}^{q} \theta_i \varepsilon(t-i) + \varepsilon(t) \quad (2.4)$$

ou

$$S(t) = \phi_L(S(t-1),...,S(t-p),\varepsilon(t-1),...,\varepsilon(t-q)) + \varepsilon(t) \quad (2.5)$$

2.4 Redes Neurais

Redes neurais artificiais (RNA´s) são sistemas paralelos distribuídos formados por unidades de processamento simples (neurônios) que realizam funções matemáticas, geralmente não-linear. As RNA´s tiveram sua origem na tentativa de gerar um modelo artificial que simule a estrutura do cérebro humano, mas atualmente podem ser definidas como uma metodologia estatística capaz de resolver com sucesso vários problemas de engenharia e de outras áreas (Dyminski , 2000).

Uma rede neural artificial é formada por diversos neurônios artificiais. A figura 2.2 representa um neurônio esquematicamente. Em um modelo de neurônio, podem ser identificados alguns elementos básicos:

a) sinapses que são responsáveis pelas conexões, cada uma caracterizada por um peso sináptico w_{kj}, onde o índice k corresponde ao número do neurônio e j ao estímulo de entrada;

b) um somatório dos sinais de entrada já multiplicados por seus respectivos pesos (combinação linear), resultando no valor u_k;

c) uma função de ativação φ que visa limitar os sinais de entrada a um determinado intervalo, normalmente entre 0 e 1 ou -1 e 1;

d) o limiar ou *polarizador*, θ_k, que possui um efeito de diminuir ou aumentar o valor da combinação linear das entradas (u_k) na função de ativação.

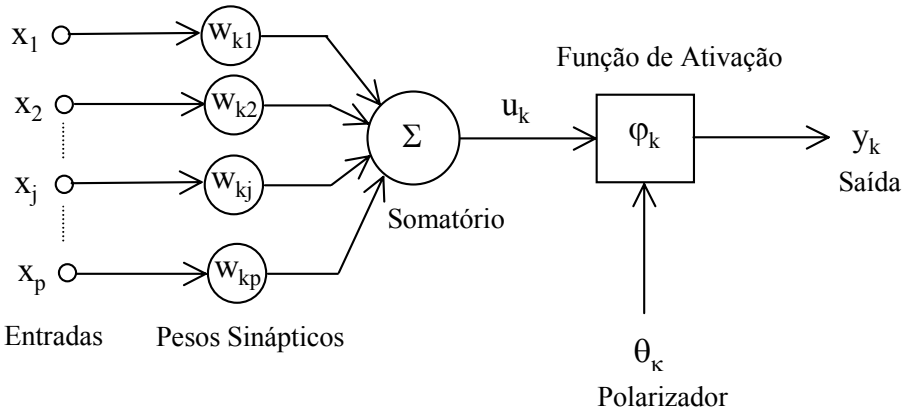

Figura 2.2 Modelo de um neurônio artificial. (Haykin, 1999)

O neurônio pode ser descrito através das seguintes equações:

$$u_k = \sum_{j=1}^{p} w_{kj} x_j \tag{2.6}$$

$$y_k = \varphi(u_k - \theta_k) \tag{2.7}$$

Ou

$$y_k = \varphi(v_k) \tag{2.8}$$

A função de ativação é capaz de assumir diversas formas, podendo ser em degrau, linear ou sigmoidal (logística ou tangente hiperbólica), conforme mostra a figura 2.3.

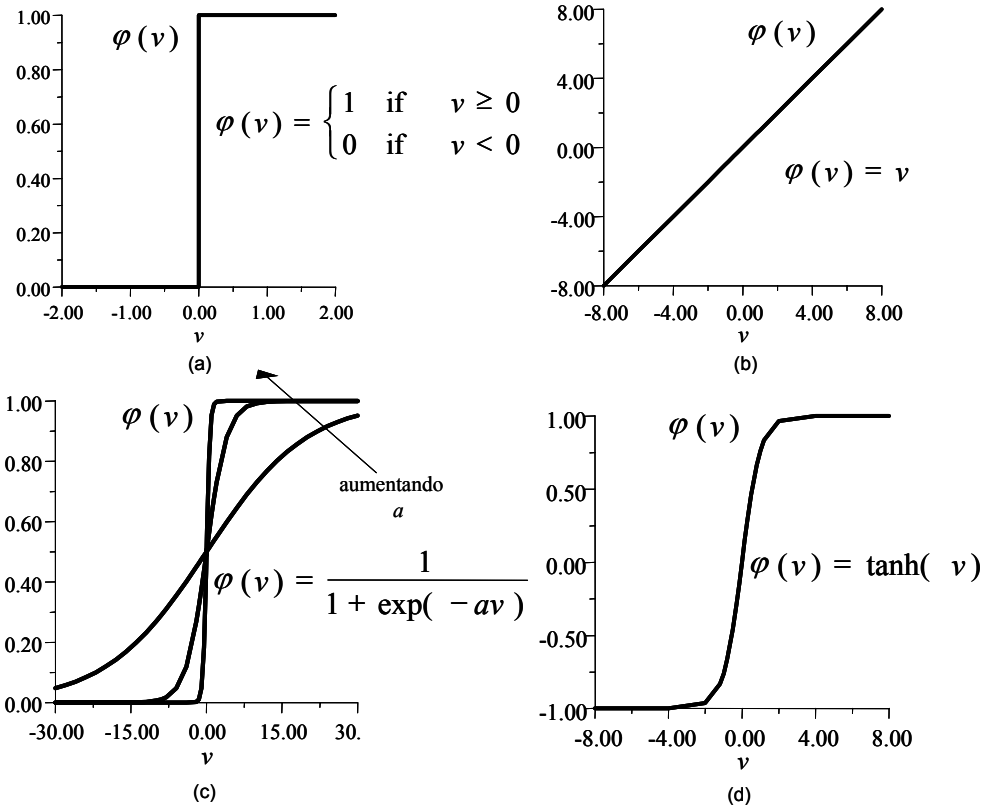

Figura 2.3 Funções de ativação (a) degrau, (b) linear, (c) sigmoidal logística e (d) tangente hiperbólica. (Haykin, 1999)

2.4.1
O uso das redes neurais em engenharia geotécnica

Redes neurais artificiais apresentam diversas aplicações na área de engenharia geotécnica. Encontram-se aplicações de RNA's em caracterização e classificação de solos e rochas, potencial de liquefação de depósitos de areias, modelos constitutivos com RNA's, fundações, escavações, estabilidade de taludes, reforço de solos, água subterrânea, mineração e outras. São apresentados a seguir alguns relevantes trabalhos sucintamente descritos.

Toll (1996) apresenta uma revisão de sistemas de inteligência artificial (IA) que foram desenvolvidas para aplicações geotécnicas. A revisão abrange sistemas especialistas e redes neurais. A revisão trata de trabalhos que envolvem uma grande quantidade de sistemas desenvolvidos para caracterização de sítios, classificação de solos e rochas, fundações, estruturas de retenção de terra, taludes, túneis e aberturas de escavações, minas, liquefação, geotêxteis entre outras aplicações. As técnicas de IA tem sido eficientes para resolver diversos problemas de engenharias. Ghaboussi (1992) apresenta uma revisão sobre a técnica de rede neural, sobre algoritmos de treinamento, sobre as técnicas de aprendizado e

apresenta uma revisão sobre a utilização de rede neurais em aplicações geotécnicas.

Veiga et al. (2002) investiga a distribuição espacial dos valores do número de golpes N dos ensaios SPT no subsolo da usina nuclear de Angra II através de redes neurais e métodos geoestatísticos. Na comparação das duas técnicas, os métodos geoestatísticos apresentaram resultados ligeiramente melhores, mas ambas as técnicas apresentaram resultados satisfatórios.

Gangopadhyay et al. (1999) associaram RNA's e GIS (Sistema de Informação Geográfica) para desenvolver um método para caracterização de subsuperfície. Dados de monitoramento da distribuição de materiais do aqüífero e logs litológicos são utilizados para treinar uma rede perceptron multicamada. A RNA faz a previsão dos materiais de formação da subsuperfície para cada ponto de um *grid* discretizado da área modelo. GIS é então usado para desenvolver perfis da subsuperfície dos dados gerados pela rede neural. Esses perfis de subsuperfície são então comparados com seções geológicas avaliadas para verificar a eficiência dos perfis gerados RNA-GIS. Esta metodologia é aplicada para determinar aqüíferos extensos e calcular parâmetros dos aqüíferos para entrada de modelos solo-água para um sistema multi-aqüífero existente na cidade de Bangkok, Tailândia. A metodologia RNA-GIS mostrou-se uma poderosa ferramenta para caracterização de geometria de aqüífero complexa, e para calcular parâmetros de aqüíferos para modelagem de fluxo solo-água.

Juang & Jiang (2001) utilizaram ensaios CPT de um depósito de areia como parâmetros para geração de um modelo neural do sítio analisado. No artigo é utilizado rede neural de regressão generalizada (GRNN). Esta rede é composta de três camadas: camada de entrada, camada oculta e camada de saída. Na camada oculta é realizado o mapeamento local não linear. Esta camada é compota de neurônios de base radial que utilizam função de transferência gaussiana. O autor utiliza as GRNN para dois tipo de caracterização do sítio: a análise unidimensional (caracterização do sítio somente na direção vertical) e a análise bidimensional (caracterização do sítio em que a coordenada z é fixada para analisar um plano xy).

Oulapour & Dadfar (2002) utilizaram resultados de ensaios geotécnicos (profundidade da amostra, índice de vazios, peso específico, grau de saturação, percentagem de solos finos que passam na peneira #200, limite de liquidez, limite

de plasticidade e umidade) para gerar um modelo neural que forneça parâmetros de resistência cisalhamento consolidado drenado, resistência cisalhamento consolidado não-drenado, resistência cisalhamento não-consolidado não-drenado e coeficiente de permeabilidade. A rede neural multi-camada perceptron é utilizada. Quatro algoritmos de treinamento são testados: *Quasi-Newton backpropagation (BFG), resilient backpropagation (RP), Levenberg-Marquardt backpropagation (LM) e Gradient Descent with momentum and adaptatitve learning rule (GDX).* Os resultados obtidos indicam que as melhores previsões dos parâmetros de resistência são obtidas utilizando o algoritmo *RP* e as melhores previsões do coeficiente de permeabilidade são obtidas pelo algoritmo *GDX*.

Potencial de liquefação de depósitos de areia foram analisados por Goh (1994) utilizando resultados de SPT e parâmetros do solo (teor de finos, dimensão do grão média (D_{50}), tensão cisalhante dinâmica equivalente ($\tau_{av}/\sigma_0^{'}$), tensão total (σ_0), tensão efetiva ($\sigma_0^{'}$), magnitude de terremoto (M) e aceleração horizontal máxima na superfície do solo). . Uma rede neural *feedforward* é utilizada associada ao algoritmo de treinamento *backpropagation*. Os autores observaram através de um estudo paramétrico que o teor de finos e SPT são os parâmetros de entrada mais importantes. Goh (1996) prevê o potencial de liquefação de depósitos de areia com o mesmo procedimento de Goh (1994), porém utilizando resultados de ensaios CPT.

Ghaboussi et al. (1991) discutem a utilização de RNA's na modelagem de materiais e aplicam na modelagem de concreto. Este artigo indica que a principal vantagem das RNA´s na geração de modelos constitutivos é que todo o comportamento pode ser representado dentro de um ambiente unificado de uma rede neural e que a rede é construída diretamente de dados experimentais que usam a capacidade auto-organizadora da rede neural, a rede é apresentada aos dados experimentais e "aprende" as relações entre tensões e deformações. Uma rede neural multi-camada perceptron é utilizada associada ao algoritmo de treinamento *backpropagation* para modelar o comportamento do concreto submetido a um estado plano de tensões sob um carregamento biaxial monotônico e submetido a um carregamento cíclico uniaxial compressivo.

Ellis et al. (1995) aplicaram RNA´s na modelagem das relações tensões - deformações da areia com variação da distribuição de tamanho dos grãos e da

história de tensões. Os autores constataram que RNA´s com *feedback* são mais eficientes que RNA´s sem *feedback* para modelagem das relações tensões – deformações. O artigo também demonstra a habilidade da rede neural para simular ciclos de carregamento e descarregamento das características tensões-deformações do solo.

Najjar & Ali (1999) utilizaram uma rede neural recorrente (R –CNN) para modelar o comportamento da areia de Nevada submetida ao estado monotônico triaxial consolidado não-drenado e modelar a resposta de um solo fino submetido ao estado cíclico uniaxial.

Na área de modelos constitutivos existem diversos trabalhos que utilizam redes neurais artificiais associados a método dos elementos finitos (MEF), como por exemplo, Ghaboussi et al. (1994) associaram redes neurais artificiais e MEF na geração de modelos constitutivos; Sidarta & Ghaboussi (1997) modelaram materiais submetidos a ensaios com distribuição não-uniforme de deformação dentro da amostra; Ghaboussi & Sidarta (1997) desenvolveram uma nova metodologia de modelagem de materiais utilizando redes neurais artificiais; Pande & Shin (2002) analisam problemas de valor de contorno com o auxilio de MEF e RNA's para prever o modelo constitutivo; Wu (1997) utilizou um algoritmo híbrido elementos finitos-redes neurais na modelagem de materiais e Shin & Pande (2002) utilizaram redes neurais e MEF para modelar geomateriais.

Chow et al. (1995) analisaram provas de carga dinâmica para fazer a previsão da capacidade de carga com auxilio de RNA. Para formulação do problema adotaram um modelo de equação de onda desenvolvida por Chow et al. (1988) que utiliza os seguintes parâmetros: número de elementos da estaca (i), força aplicada na cabeça da estaca F(t), velocidade aplicada na cabeça da estaca V(t), densidade da estaca (ρ_p), módulo de Young da estaca (E_p), comprimento da estaca(L), seção transversal da estaca (A_p), resistência estática do solo associada ao *i* elemento (R_u^i), coeficiente de rigidez por unidade de comprimento associado ao *i* elemento (k_s^i), coeficiente de amortecimento por unidade de comprimento associado ao *i* elemento (c_s^i), coeficientes do efeito da velocidade no eixo da estaca (J_s^i, N_s^i), resistência estática do solo na ponta da estaca (R_u^t), coeficiente de rigidez na ponta da estaca (k_s^t), coeficiente de amortecimento na ponta da estaca

(c_s^t) e coeficientes do efeito da velocidade na ponta da estaca (J^t, N^t). RNA's *feedforward* são utilizadas associada ao algoritmo de treinamento *backpropagation* para previsão da capacidade de carga de testes dinâmicos de estacas escavadas e para previsão da capacidade de carga de testes dinâmicos de estacas de aço perfil H cravadas. Apresentado bons resultados para ambas as previsões quando comparados com métodos tradicionais.

Teh et al. (1997) utilizam modelos RNA's multi-camada perceptron associadas ao algoritmo de treinamento *backpropagation* para previsão da capacidade de carga de teste dinâmicos de estacas de concreto reforçado (RC). Análises distintas são realizadas com três modelos de rede: a primeira rede faz a previsão da capacidade estática, a segunda rede prevê a resistência distribuída e a terceira prevê resistência, amortecimento e *quake*. Os melhores resultados são obtidos com a terceira análise, apresentado as duas primeiras análises resultados razoáveis . Um teste de generalização com conjunto de teste com estacas não RC é realizado indicando a capacidade de generalização da rede. As previsões são comparadas com soluções CAPWAP que utilizam como parâmetros: resistência última do solo(R_u), fator de quake(Q), fator de amortecimento(J), força medida na cabeça da estaca (F) e velocidade medida na cabeça da estaca(V).

Liu et al. (1997) utilizam os resultados obtidos do método Case para alimentar um RNA's multi-camada perceptron associadas ao algoritmo de treinamento *backpropagation* para fazer a previsão da capacidade de carga da estaca. O método Case é um sistema econômico e portátil que calcula a capacidade de carga da estaca de medições de força aplicada na estaca e velocidade de propagação de onda durante a cravação da estaca.

Kiefa (1998) utiliza redes neurais de regressão geral (GRNN) para previsão de capacidade de carga de estacas cravadas em solos de baixa coesão. São utilizadas três redes neurais GRNN para previsão: a primeira prevê a capacidade de carga total, a segunda prevê a capacidade de carga do topo e a terceira a capacidade de carga lateral. Os parâmetros de entrada das redes são: tangente do ângulo de atrito médio do solo ao longo da lateral da estaca ($\tan(\phi_L)$), tangente do ângulo de atrito do solo na ponta da estaca ($\tan(\phi_B)$), tensão efetiva vertical do solo na ponta da estaca, SPT na lateral do solo ($\sigma'_{V(B)}$), comprimento da estaca (L) , seção transversal da estaca (A) e diâmetro da estaca (D). Os resultados obtidos

pelas redes são comparados com quatro diferentes métodos empiricos e com valores medidos. O modelos da GRNN foram melhores que os métodos empiricos quando comparados com os valores medidos.

Musso et al. (2002) utilizaram um sistema de inferência adaptativo baseado em lógica fuzzy e redes neurais (ANFIS) para prever a evolução do deslocamento de sapatas. Testes de sapatas submetidas a cargas excêntricas verticais são analisados. A rede é treinada com resultados experimentais de uma série de testes de carga em condições normais e de gravidade incrementada. Os resultados obtidos pelas sistema são comparadas favoravelmente com os testes de laboratórios correspondentes. Os autores sugerem que novas curvas de carga-recalque podem ser simulados para obter diagramas de interação de colapso que são frequentemente utilizados em modelos matemáticos para previsão da evolução do deslocamento de uma sapata rasa.

Rahman & Wang (2002) utilizaram um sistema de inferência adaptativo baseado em lógica fuzzy e redes neurais (ANFIS) para analisar provas de carga dinâmica e a capacidade de arrancamento de caixões succionados. Os resultados obtidos indicam previsões bastante razoáveis e que a metodologia pode ser utilizada com sucesso em outros problemas geotécnicos.

Dyminski (2000) apresenta a análise de três diferentes problemas em geotecnia utilizando RNA's *feedforward* treinadas com o algoritmo de Levenberg-Marquardt (LM). A primeira aplicação simula resultados de provas de carga dinâmica, analisadas pelo método CAPWAP, através de RNA, sendo assim viabilizada a realização de uma pré-análise do comportamento da estaca ainda em campo, o que geralmente não acontece quando se trata da análise CAPWAP tradicional. A segunda aplicação análise o comportamento mecânico de dois tipos de solo: a areia de Ipanema e o solo residual gnáissico do Rio de Janeiro. Nesta modelagem, foram utilizados resultados de ensaios de cisalhamento direto, submersos e não submersos, e ensaios de compressão triaxial, drenados e não drenados. A terceira aplicação simula as características do subsolo do sítio da Usina Nuclear Angra 2, localizada no litoral do estado do Rio de Janeiro. Foram utilizadas para a modelagem, as informações disponíveis em boletins de sondagens do tipo SPT. Foram realizadas simulações envolvendo a disposição das camadas dos diferentes tipos de solo que poderiam existir no local, o nível de água

subterrâneo, a resistência à penetração do solo e a topografia do terreno. O trabalho obteve resultados bastante satisfatórios nos três diferentes problemas.

King & Signer (1994) avaliaram características mecânicas e movimentos de paredes de minas de carvão com o auxílio de redes neurais artificiais. O sistema desenvolvido utiliza RNA com treinamento não-supervisionado para coletar resultados de instrumentação e identificar feições geológicas com os padrões de razão de penetração, empuxo, torque, posição e razão de rotação. Os resultados obtidos da RNA com aprendizado não supervisionado foram utilizados para treinar duas RNA`s supervisionadas utilizando o algoritmo de treinamento *backpropagation*. Os resultados obtidos mostram que o sistema apresenta um grande potencial para identificação das características das camadas. O artigo sugere ainda que os resultados podem ser utilizados como entrada de um sistema especialista para localizar regiões de risco e recomendar atitudes preventivas que evitem a ruptura das paredes.

Millar & Calderbank (1995) analisaram a deformabilidade de rochas utilizando RNA's. Uma RNA perceptron multicamada é introduzida em um sistema denominado FLAC. A RNA é responsável no sistema por calcular as relações tensões-deformações. Os resultados obtidos demonstram sucesso na utilização de RNA como ferramenta de projetos de engenharia em rocha.

Zettler et al. (1997) analisaram o controle do processo de injeção considerando análise de pressão transiente (TPA) com o auxilio de RNA e lógica fuzzy. O processo de controle é chamado Pressão sensível de injeção (PSG) que envolve a pressão de injeção, pressão residual, razão pressão-volume de injeção e o comportamento dependente do tempo. O sistema combina conhecimento especialista, experiências físicas e investigações numéricas e ajusta regras para dados de treinamento para controlar o processo de injeção. A RNA é aplicada para encontrar as melhores regras esperadas para o conjunto de dados entrada-saída conhecido para o algoritmo de controle lógica fuzzy.

Tan & Hui (2001) aplicaram RNA's para gerar um modelo neural para maciços de rocha. Uma rede neural que utiliza como função de transferência a função de base radial. O algoritmo de treinamento é o *backpropagation.* Os dados utilizados para testar o modelo neural são curvas tensão-deformação obtidas de ensaios com mármore. Os resultados obtidos indcam que o método fornece curvas

tensões-deformações com precisão e que o método é valioso para mecânica das rochas computacional.

Bandyopadhyay et al. (1995) utilizaram RNA's no estudo da durabilidade da rocha calcária que compõe a famosa esfinge do Cairo. Distribuição do tamanho dos poros determinadas experimentalmente e fatores de durabilidade (DF) determinados com base no estado com o passar do tempo da Esfinge do Cairo são usados como padrões de entrada e saída respectivamente para uma RNA perceptron multicamada treinada com o *backpropagation.* Uma parte da camada da rocha calcária da Esfinge é utilizada para gerar os dados utilizados no treinamento. O modelo neural gerado é usado no cálculo do DF da camada restante da Esfinge. Os resultados obtidos combinam excelentemente com as condições observadas das camadas e com a durabilidade indicada pela constituição litológica.

Parâmetros de compactação do solo são previstos utilizando RNA's por Basheer & Najjar (1995). O teor de umidade ótimo e a densidade seca máxima são dois parâmetros de compactação que caracterizam e ajudam a controlar a construção e qualidade de uma grande quantidade de projetos de trabalhos com solo. No artigo, estes dois parâmetros são previstos utilizando RNA multicamada perceptron associada ao algoritmo de treinamento *backpropagation.* São gerados dois modelos neurais: o primeiro modelo tem como entradas limite de liquidez, limite de plasticidade e densidade específica do solo e a segunda rede tem apenas como entrada limite de liquidez. Os resultados obtidos são comparados com o valores experimentais e com valores obtidos com outros métodos de previsão. Apresentando o modelo neural valores similares aos obtidos experimentalmente.

Previsões de deslocamento em sistemas de suporte de escavação são analisadas com o auxilio de RNA's por Gao et al. (2001). Uma rede neural feedforward treinada com o algoritmo backpropagation é utilizado para previsão do deslocamento horizontal. Os deslocamento horizontal é medido ao longo do tempo gerando séries temporais. O modelo neural utiliza como entradas cinco medições sequenciais de deslocamento e a saída é o valor da previsão do deslocamento. A RNA consegue prever com sucesso os deslocamentos horizontais.

2.4.2
Arquitetura de rede

A arquitetura de rede define como as unidades de processamento simples são dispostas, ou seja, como os diversos neurônios são conectados entre si.

O tipo de rede neural artificial mais utilizado em aplicações em engenharia é chamado de "multi-camadas *feedforward"*, caracterizando-se por apresentar neurônios organizados em camadas. O neurônio pode estar total ou parcialmente conectado por sinapses (pesos). As entradas alimentam a rede e são processadas até alcançarem a camada de saída, sem realimentação. Por isso são denominadas como *feedforward* (alimentação e propagação para frente).

A figura 2.4 mostra uma rede neural totalmente conectada, com 4 entradas e 2 saídas e que possui uma camada oculta com 5 neurônios, localizada entre os nós de entrada e a camada de saída. Redes neurais artificiais, de uma ou duas camadas ocultas apenas, conseguem resolver a maior parte dos problemas de engenharia (Haykin, 1999).

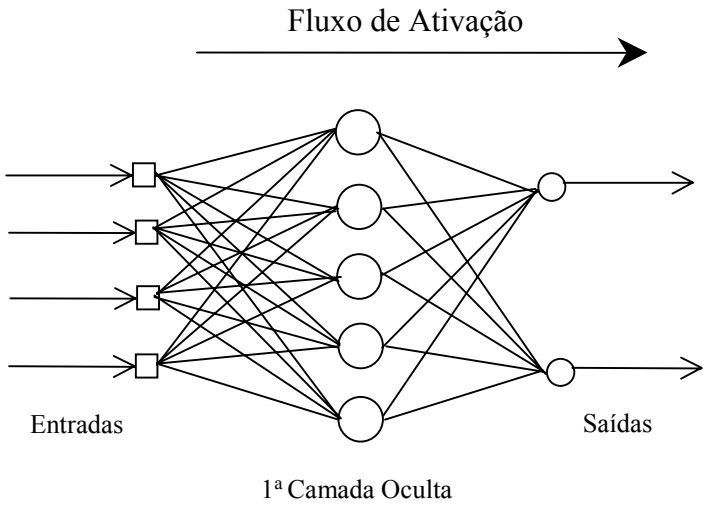

Figura 2.4 Exemplo de rede neural multi-camadas *feedforward*, totalmente conectada.

A complexidade do modelo pretendido irá definir a quantidade de pesos (parâmetros) e de camadas da arquitetura da rede.O número de neurônios não deve ser muito pequeno se o problema a ser resolvido apresentar grande complexidade e nem grande demais, a ponto de se prejudicar a capacidade de generalização da rede, gerando o problema conhecido como *overfitting* (ajuste excessivo) (Dyminski , 2000).

2.4.3
O conjunto de dados

Como em qualquer método estatístico o conjunto de dados deverá cobrir todo o domínio de interesse do problema a ser solucionado. O conjunto deverá ter amostras suficientes para que a rede aprenda sobre o problema. A figura 2.5 ilustra uma relação entre métodos, volume de dados e grau de conhecimento da realidade.

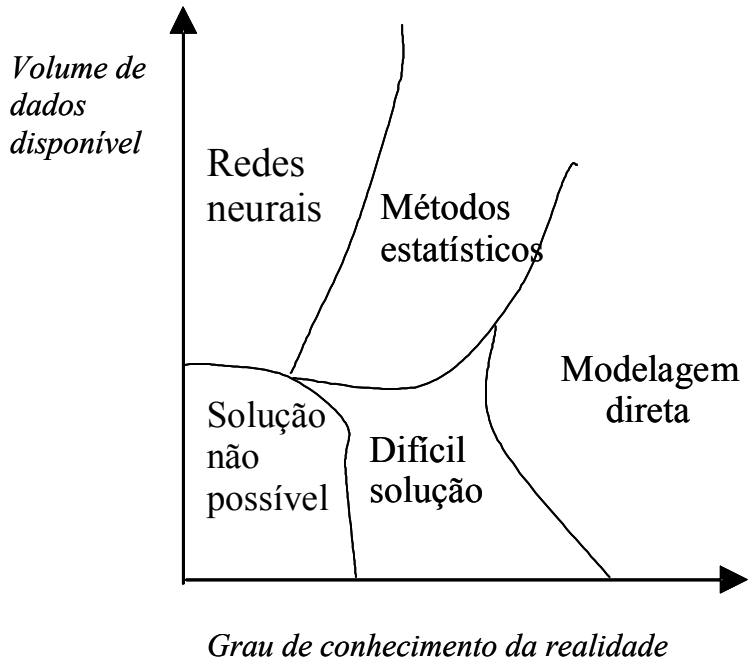

Figura 2.5 Métodos adequados em função da quantidade de dados e complexidade do problema. (Pedreira, 2001)

2.4.4
Treinamento e testes

Para iniciar o treinamento devem estar definidos:

a) o tipo e a arquitetura da rede;

b) o conjunto de amostras referentes ao fenômeno a ser modelado;

c) os parâmetros que servirão de entrada à rede;

d) as saídas que se quer obter.

Um método simples de treinamento consiste em dividir o conjunto de amostras em três subconjuntos: conjunto de treinamento, conjunto de teste 1 e conjunto de teste 2.

O conjunto de treinamento será composto de exemplos que serão utilizados na fase de treinamento da rede neural artificial. O conjunto de teste 1 será

composto de exemplos que serão utilizados na tarefa de se testar a capacidade de generalização da rede neural. O conjunto de teste 2 serve para comparar modelos diferentes.Cabe salientar que os exemplos de um subconjunto não deverão ter sido utilizados no outros subconjuntos, ou seja, os subconjuntos são formados por exemplos distintos entre si.

O treinamento consiste na apresentação das amostras do conjunto de treinamento à rede que processará os parâmetros de entrada relativos a estas amostras através da multiplicação dos mesmos pelos pesos sinápticos e da posterior aplicação destes valores às funções de ativação dos neurônios, fornecendo então as respostas (saídas da rede) a este estímulo. Estas saídas da RNA deverão ser comparadas com os valores reais dos parâmetros de saída correspondentes aos exemplos do conjunto de treinamento, e desta comparação será obtido um valor de erro da fase de treinamento. Procura-se então ajustar os valores dos pesos sinápticos, através de um algoritmo matemático, visando a diminuição do erro de treinamento. Durante a etapa de treinamento o erro do conjunto de teste 1 é monitorado para garantir a capacidade de generalização da rede. Quando o erro do conjunto de teste 1 começa a aumentar o processo de treinamento termina. Este método chama-se "*early stopping"*. A figura 2.6 mostra o monitoramento dos erros de treinamento e de generalização.

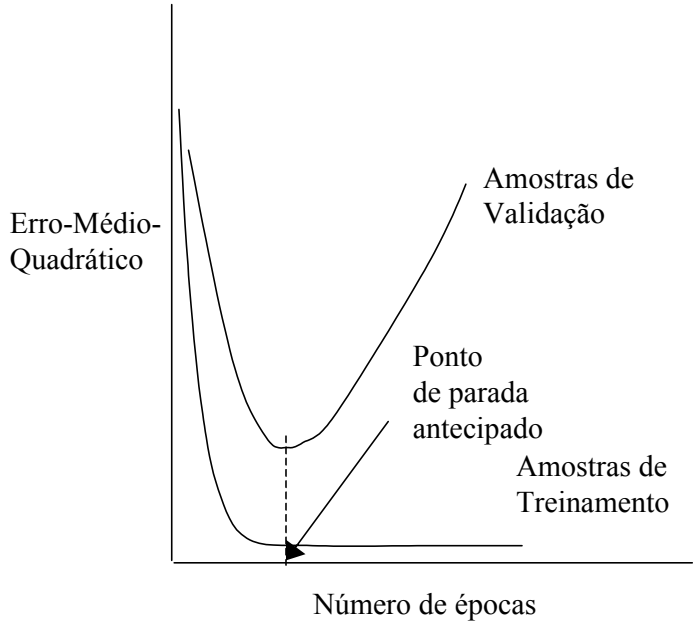

Figura 2.6 Ilustração do monitoramento do erro do método "*early stopping*". (Haykin, 1999).

Quando a quantidade de amostras for pequena não se deve dividir o conjunto de amostras em subconjuntos. O teste pode ser feito retirando uma amostra aleatoriamente para verificar a generalização e utilizando o restante das amostras para o conjunto de treinamento. Este procedimento deve ser repetido até a média do erro ficar constante. Este método é denominado "*leave-one-out*".

Assim, tem-se como objetivo da fase de treinamento ajustar da melhor maneira possível os valores dos parâmetros da rede, procurando fazer com que as saídas fornecidas pela RNA estejam bastante próximas dos valores reais de saída correspondentes a cada um das amostras apresentadas, sem que se perca sua capacidade de generalização.

Existem diversos algoritmos para se treinar RNA's. A seguir serão citados os algoritmos: retro-propagação do erro; Levenberg-Marquardt e regularização bayesiana.

2.4.5
O algoritmo de retro-propagação do erro

O algoritmo de retro-propagação do erro consiste dos seguintes passos (Dyminski , 2000):

a) as entradas das amostras do conjunto de treinamento são apresentadas à rede neural;

b) estes dados são processados pela rede: em cada camada, através da multiplicação pelos pesos sinápticos, posterior somatório e aplicação da função de ativação, são fornecidas as saídas relacionadas às entradas apresentadas;

c) as saídas fornecidas pela rede são comparadas com os valores reais obtidos dos experimentos e, desta variação, é calculado o erro;

d) com o valor do erro, o ajuste para os pesos da última camada é calculado (conforme desenvolvimento a seguir). O erro é então "retro-propagado" na rede, corrigindo os pesos sinápticos das camadas ocultas, visando um melhor ajuste da RNA ao fenômeno a ser modelado.

Este processo iterativo deverá acontecer até que o erro seja aceitável, ou seja, que a rede tenha aprendido a tarefa a ser realizada.

O erro de um neurônio de saída j na interação n, pode ser calculado por:

$$e_j(n) = d_j(n) - y_j(n) \tag{2.9}$$

onde $d_j(n)$ é o valor real de saída do neurônio *j* e $y_j(n)$ é o valor estimado pela rede na interação *n.*

O valor instantâneo do erro quadrático para o neurônio *j* é definido como $\frac{1}{2}e_j^2(n)$. O somatório $\frac{1}{2}e_j^2(n)$ de todos neurônios da camada de saída define o valor instantâneo E(n) como na equação 2.10.

$$E(n) = \frac{1}{2}\sum_{j \in C} e_j^2(n) \tag{2.10}$$

C é o conjunto de neurônios da camada de saída. O erro médio quadrático é obtido somando-se os *E(n)* é dividindo-se por *N*, como mostra a equação 2.11, onde *N* é a quantidade de amostras do conjunto de treinamento.

$$E_{av} = \frac{1}{N}\sum_{n=1}^{N} E(n) \tag{2.11}$$

O erro instantâneo *E(n)* e o erro médio quadrático E_{av} são funções dos parâmetros livres (pesos sinápticos e bias) da rede. O erro médio quadrático E_{av} é a função de custo da etapa de treinamento. O objetivo da etapa de treinamento é ajustar os parâmetros livres para minimizar E_{av}.

A figura 2.7 mostra, o neurônio *j* sendo alimentado, pelas saídas da camada a sua esquerda. O campo local induzido $v_j(n)$ produzido na entrada da função de ativação do neurônio *j* é mostrada na equação 2.12.

$$v_j(n) = \sum_{i=0}^{m} w_{ji}(n) y_i(n) \tag{2.12}$$

m é o número total de saídas (excluindo a bias) aplicada ao neurônio *j*. O peso sináptico w_{j0} (com entrada $y_0 = +1$) é igual ao polarizador aplicado ao neurônio *j*. A função $y_j(n)$ na saída do neurônio *j* na interação n, é dada pela equação 2.13.

$$y_j(n) = \varphi_j(v_j(n)) \tag{2.13}$$

O algoritmo de retro-propagação do erro aplicará uma correção $\Delta w_{ji}(n)$ para o peso sináptico $w_{ji}(n)$, que será proporcional à derivada parcial $\partial E(n)/\partial w_{ji}(n)$. Aplicando a regra da cadeia, a derivada parcial pode ser expressa pela equação 2.14.

$$\frac{\partial E(n)}{\partial w_{ji}(n)} = \frac{\partial E(n)}{\partial e_j(n)} \frac{\partial e_j(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial v_j(n)} \frac{\partial v_j(n)}{\partial w_{ji}(n)} \tag{2.14}$$

A equação 2.14 é o fator de sensitividade, determinando a direção de busca no espaço dos pesos para o peso sináptico w_{ji}.

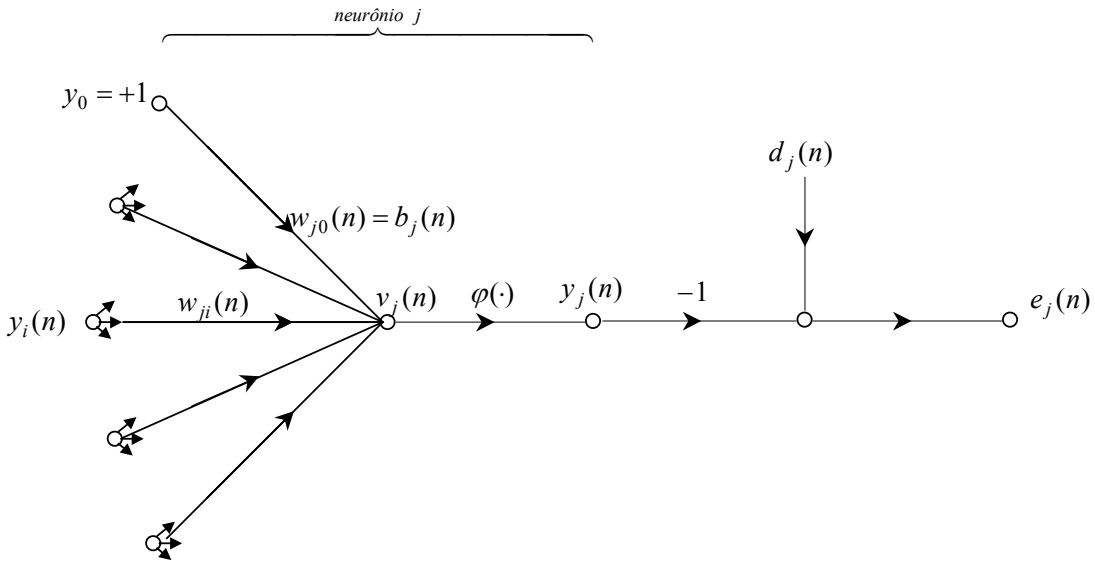

Figura 2.7 Fluxo de sinais dentro do neurônio de saída *j* (Haykin, 1999).

Diferenciando a equação 2.10 em relação $e_j(n)$, tem-se a equação 2.15.

$$\frac{\partial E(n)}{\partial e_j(n)} = e_j(n) \tag{2.15}$$

Diferenciando a equação 2.9 em relação $y_j(n)$, tem-se a equação 2.16.

$$\frac{\partial e_j(n)}{\partial y_j(n)} = -1 \tag{2.16}$$

Diferenciando a equação 2.13 em relação $v_j(n)$, tem-se a equação 2.17.

$$\frac{\partial y_j(n)}{\partial v_j(n)} = \varphi_j'(v_j(n)) \tag{2.17}$$

A "linha" do lado direito da equação significa diferenciação em relação ao argumento.

Diferenciando a equação 2.12 em relação $w_{ji}(n)$, tem-se a equação 2.18.

$$\frac{\partial v_j(n)}{\partial w_{ji}(n)} = y_i(n) \tag{2.18}$$

Substituindo as equações 2.15 à 2.18 na equação 2.14, tem-se a equação 2.19.

$$\frac{\partial E(n)}{\partial w_{ji}(n)} = -e_j(n)\varphi_j'(v_j(n))y_i(n) \tag{2.19}$$

A correção de $\Delta w_{ji}(n)$ aplicada em $w_{ji}(n)$ é denominada "regra delta" como mostra a equação 2.20.

$$\Delta w_{ji}(n) = -\eta \frac{\partial E(n)}{\partial w_{ji}(n)} \tag{2.20}$$

onde η é a taxa de aprendizado do algoritmo de retro-propagação do erro. O sinal negativo indica gradiente descendente no espaço dos pesos. Substituindo a equação 2.19 na equação 2.20.

$$\Delta w_{ji}(n) = -\eta \delta_j(n) y_i(n) \tag{2.21}$$

onde o gradiente local $\delta_j(n)$ é definido pela equação 2.22.

$$\delta_j(n) = -\frac{\partial E(n)}{\partial v_j(n)} = -\frac{\partial E(n)}{\partial e_j(n)} \frac{\partial e_j(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial v_j(n)} = e_j(n) \varphi_j'(v_j(n)) \tag{2.22}$$

A taxa de aprendizado η regulará a velocidade de aprendizado da rede.

Quando o neurônio *j*, figura 2.8, está situado em uma camada oculta o procedimento de ajuste seguirá a metodologia abaixo:

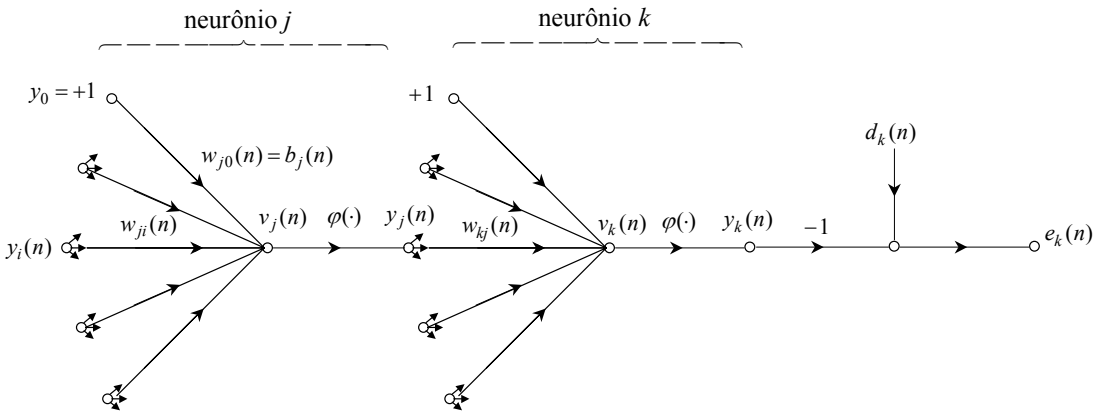

Figura 2.8 Fluxo de sinais dentro do neurônio de camada oculta *j* conectada ao neurônio de saída *k*. (Haykin, 1999).

Definindo o gradiente local $\delta_j(n)$ para o neurônio *j*, tem-se a equação 2.23.

$$\delta_j(n) = -\frac{\partial E(n)}{\partial v_j(n)} = -\frac{\partial E(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial v_j(n)} \tag{2.23}$$

Diferenciando a equação 2.10 em relação a $y_j(n)$, tem-se a equação 2.24, onde *k* representa o neurônio na camada de saída.

$$\frac{\partial E(n)}{\partial y_j(n)} = \sum_k e_k \frac{\partial e_k(n)}{\partial y_j(n)} \tag{2.24}$$

Aplicando a regra da cadeia na derivada parcial $\partial e_k(n) / \partial y_j(n)$, tem-se a equação 2.25.

$$\frac{\partial E(n)}{\partial y_j(n)} = \sum_k e_k(n) \frac{\partial e_k(n)}{\partial v_k(n)} \frac{\partial v_k(n)}{\partial y_j(n)} \tag{2.25}$$

Escrevendo as equações 2.9 e 2.12 utilizando o índice *k* para o neurônio de saída, têm-se as equações 2.26 e 2.27.

$$e_k(n) = d_k(n) - y_k(n) = d_k(n) - \varphi_k(v_k(n)) \tag{2.26}$$

$$v_k(n) = \sum_{i=0}^{m} w_{kj}(n) y_j(n) \tag{2.27}$$

Diferenciando a equação 2.26 em relação $v_k(n)$, tem-se a equação 2.28.

$$\frac{\partial e_k(n)}{\partial v_k(n)} = -\varphi_k'(v_k(n)) \tag{2.28}$$

Diferenciando a equação 2.27 em relação $y_j(n)$, tem-se a equação 2.29.

$$\frac{\partial v_k(n)}{\partial y_j(n)} = w_{kj}(n) \tag{2.29}$$

Diferenciando a equação 2.13 em relação $v_j(n)$, tem-se a equação 2.30

$$\frac{\partial y_j(n)}{\partial v_j(n)} = \varphi_j^{'}(v_j(n)) \tag{2.30}$$

Substituindo as equações 2.28 e 2.29 na equação 2.25, tem-se a equação 2.31.

$$\frac{\partial E(n)}{\partial y_j(n)} = \sum_k e_k(n)\varphi_k^{'}(v_k(n))w_{kj}(n) \tag{2.31}$$

Substituindo as equações 2.30 e 2.31 na equação 2.23, tem-se o gradiente local dado pela equação 2.32.

$$\delta_j(n) = \varphi_j^{'}(v_j(n))\sum_k e_k(n)\varphi_k^{'}(v_k(n))w_{kj}(n) \tag{2.32}$$

A correção de $\Delta w_{ji}(n)$ aplicada em $w_{ji}(n)$ é definida pela equação 2.33.

$$\Delta w_{ji}(n) = -\eta\varphi_j^{'}(v_j(n))\sum_k e_k(n)\varphi_k^{'}(v_k(n))w_{kj}(n)y_i(n) \tag{2.33}$$

Resumindo o algoritmo de retro-propagação pode ser descrito pelas equações 2.34 a 2.36.

$$w_{ji}(n+1) = w_{ji}(n) + \Delta w_{ji}(n) \tag{2.34}$$

$$\Delta w_{ji}^{l}(n) = -\eta\delta_j^{l}(n)\cdot a_i^{l-1}(n) \tag{2.35}$$

Onde o sobrescrito l indica a camada que o neurônio pertence e $a_i^{l-1}(n)$ é a entrada que o neurônio recebe.

$$\delta_j^{l}(n) = \begin{cases} e_j(n)\varphi_j^{'}(v_j(n)) & l = L \\ \varphi_j^{'}(v_j(n))\sum_k \delta_k^{l+1}(n)w_{kj}(n) & 1 \le l \le L-1 \end{cases} \tag{2.36}$$

L é o símbolo adotado para a camada de saída

2.4.6
O algoritmo de Levenberg-Marquardt

O algoritmo de retro-propagação do erro descrito no item 2.4.5 foi baseado no método de otimização do gradiente descendente que utiliza apenas as primeiras derivadas para minimizar o erro. Entretanto existem outras técnicas de otimização que tornam a convergência mais rápida, por exemplo, algoritmos de otimização que utilizam as segundas derivadas. Esses algoritmos são do tipo "quase Newton" sendo mais eficientes, porém precisam armazenar a matriz Hessiana (matriz com as segundas derivadas).

O método Levenberg-Marquardt (LM) é do tipo "quase Newton" utilizado para mínimos quadrados não lineares é foi incorporado ao algoritmo de retro-propagação do erro para resolver problemas de otimização que aparecem no treinamento de redes multi-camadas. O processo do método LM é descrito em Bishop (1995).

A função objetivo para o algoritmo LM é:

$$E = \frac{1}{2}\left\|\varepsilon(w_{velho}) + Z(w_{novo} - w_{velho})\right\|^2 + \lambda\left\|w_{novo} - w_{velho}\right\|^2 \qquad (2.37)$$

Onde $\varepsilon(w_{velho})$ é o vetor com os erros do conjunto de treinamento, w_{velho} é o peso sináptico utilizado no passo atual, w_{novo} é o peso sináptico calculado e λ é um parâmetro que governa o tamanho do passo.

A atualização dos pesos utiliza a equação 2.38.

$$w_{novo} = w_{velho} - (H + \lambda I)^{-1} Z^T \varepsilon(w_{velho}) \qquad (2.38)$$

Onde

$$(Z)_{ni} = \frac{\partial \varepsilon^n}{\partial w_i} \tag{2.39}$$

$$H = Z^T Z \tag{2.40}$$

2.4.7
O algoritmo de regularização bayseana

O algoritmo de treinamento utilizando regularização bayseana é uma técnica mais recente para treinar o modelo neural. Nesta técnica, a quantidade de neurônios da camada oculta não será estimada por um processo de tentativa e erro. Na regularização bayseana o processo de treinamento faz com que os parâmetros (pesos sinápticos) dos neurônios ocultos desnecessários fiquem com valor próximo a zero. O objetivo principal é encontrar um equilíbrio entre a quantidade de parâmetros e a complexidade do modelo neural exigida pelo problema. A função objetivo utilizada no algoritmo de retropropagação do erro, equação (2.41), é modificada adicionando-se um termo de regularização ($E_W = \frac{1}{2}\sum_i w_i^2$) e parâmetros de regularização ($\beta, \alpha > 0$) modificando a função objetivo para a equação (2.42).

$$E_D(w) = \frac{1}{N}\frac{1}{2}\sum_n \sum_j \left(d_j^{(n)} - y_j^{(n)}(x^{(n)}; w)\right)^2 \tag{2.41}$$

$$M(w) = \beta \cdot E_D + \alpha \cdot E_W \tag{2.42}$$

Os valores dos parâmetros da função objetivo devem ser calculados. Para encontrar valores ótimos para esses parâmetros. Pode-se utilizar a técnica "*bayesian framework*", onde os parâmetros da rede neural são assumidos como sendo variáveis randômicas com distribuição conhecida. Os parâmetros da função objetivo são relacionados com variáveis desconhecidas associadas com essas distribuições e são estimados com técnicas estatísticas. Medeiros & Pedreira (2001) descrevem o processo de otimização dos parâmetros de regularização aplicando a regra de Bayes e o algoritmo proposto por Foresee e Hagan em 1997

para otimização Bayseana dos parâmetros de regularização, com aproximação por Gauss-Newton para a matriz Hessiana em combinação com o algoritmo LM.

O algoritmo proposto por Foresee e Hagan apresenta os seguintes passos:

1. Inicializar β, α e os parâmetros da rede neural. Depois do primeiro passo de estimação, os parâmetros da função objetivo recuperarão o cenário inicial.
2. Levar um passo do algoritmo de LM para minimizar a função objetivo $M(w) = \beta \cdot E_D + \alpha \cdot E_W$.
3. Calcular o número efetivo de parâmetros $\gamma = N - 2\alpha \; trace(H)^{-1}$ utilizando a aproximação Gauss-Newton para a matriz Hessiana avaliada pelo algoritmo de otimização de LM $H = \nabla^2 M(w) \approx 2\beta J^T J + 2\alpha I_N$, onde J é a matriz Jacobiana do conjunto de erros estimados.
4. Calcular os novos valores para os parâmetros da função objetivo.
5. Repetir do passo 1 até o passo 3 até convergir.

O algoritmo de treinamento regularização bayseana utilizado para treinamento das RNA's deste trabalho já está implementado no pacote de redes neurais do programa MATLAB.

2.4.8
Redes neurais temporais (RNT´s)

As redes neurais temporais são redes dinâmicas que possuem propriedades de memória. Estas propriedades permitem-nas realizar tarefas de caráter temporal. A maneira como a memória é representada determina o tipo de rede neural temporal (Gutiérrez, 2003). A figura 2.9 mostra duas maneiras de se incluir habilidades de memória na rede neural:

a) Rede considerando entradas atrasadas no tempo;
b) Rede com laços de realimentação.

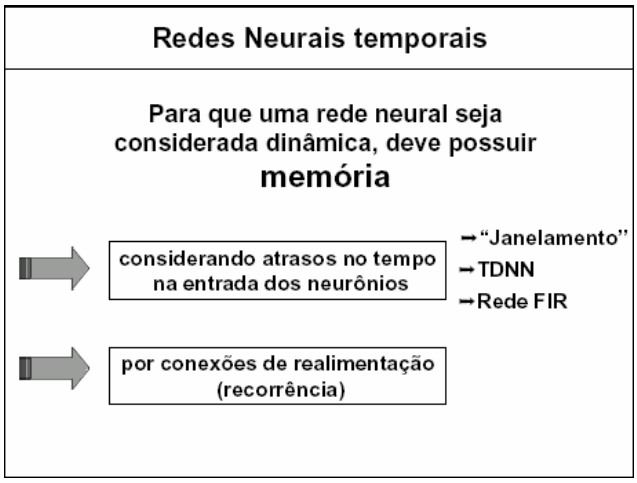

Figura 2.9 Classificação das redes neurais temporais (Gutiérrez, 2003).

2.4.8.1 Redes neurais com atrasos no tempo

Durante muito tempo, modelos de redes neurais com atraso no tempo foram os mais utilizados em problemas que envolviam processamento temporal (Soto, 1999). A memória era introduzida na rede, proporcionando aos neurônios os valores de entradas atuais e valores temporalmente anteriores. No método de "janelamento" somente os neurônios da primeira camada possuem memória. A Figura 2.10 mostra o método de "janelamento", que fornece à rede uma memória de ordem 2 na primeira camada oculta. Já que além da entrada atual $x_{(k)}$, estes neurônios recebem também como entrada dois valores anteriores $x_{(k-1)}$ e $x_{(k-2)}$ criando, portanto, sinapses novas.

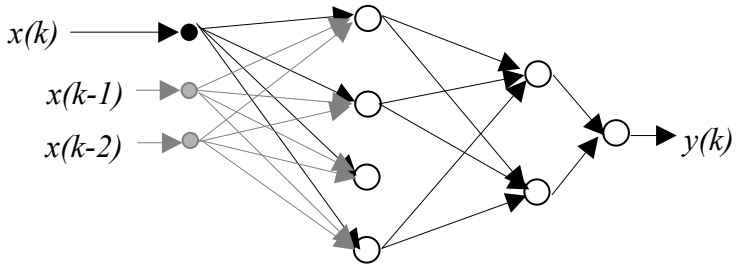

Figura 2.10 Método de "janelamento" para processamento temporal (Soto, 1999).

A Figura 2.11 mostra uma rede que fornece memória a todos os neurônios das camadas ocultas e da camada de saída. Esta rede é denominada rede TDNN (*Time Delay Neural Network*).

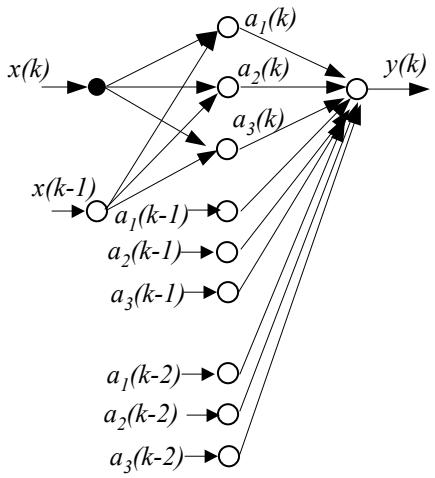

Figura 2.11 Rede TDNN para processamento temporal (Soto, 1999).

Nesta rede, existem sinapses novas, mas a rede final formada não conduz à forma da rede estática padrão, totalmente interconectada, pois os neurônios novos na camada oculta não estão conectados aos elementos de entrada. Este fato faz a rede TDNN perder certa "simetria" no processo da exploração "*feedforward*" e de retropropagação, tornando-a mais complexa (Soto, 1999).

A rede neural com atraso no tempo utilizada neste trabalho é a denominada rede neural FIR (*Finite Response Impulse*) onde cada sinapse é formada por um filtro FIR linear que representa a natureza temporal do problema. Pode-se dizer que este tipo de rede engloba todos os outros métodos de sua classe. Na seção seguinte será detalhada.

2.4.8.2 Redes neurais FIR

Nas redes neurais FIR as sinapses são modeladas por um filtro linear invariante no tempo. Nesta modelagem que considera memória em todos os neurônios a "simetria" da rede é mantida, pois utiliza um modelo fundamentado na técnica de processamento de sinais.

Soto (1999) mostra a formulação de uma rede neural FIR detalhadamente, considerando para o modelo filtros sinápticos de tempo contínuo. A figura 2.12 ilustra o modelo FIR do neurônio com N entradas.

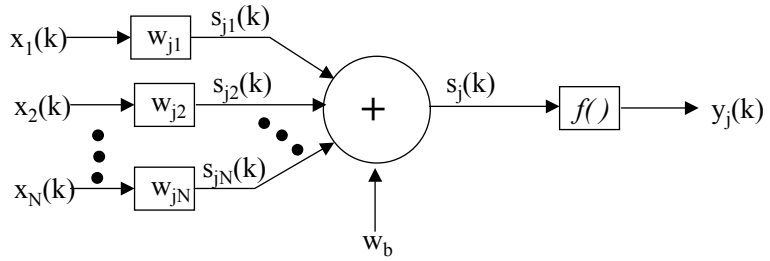

Figura 2.12 Modelo FIR do neurônio (Soto, 1999).

Onde:

$x_1(k)$, $x_2(k)$ e $x_N(k)$ são as entradas do neurônio,

w_{ji} é o vetor de pesos da sinapse do neurônio j,

$s_{ji}(k)$ é o potencial sináptico do neurônio j,

$s_j(k)$ é a soma dos potenciais sinápticos do neurônio j,

w_b é o bias,

$y_j(k)$ é a saída do neurônio após passar pela função de ativação do neurônio j.

Neste modelo o potencial sináptico $s_{ji}(k)$ é a resposta de um filtro FIR de ordem M, onde M representa o número total de atrasos unitários considerados no desenho do filtro FIR. Este potencial sináptico $s_{ji}(k)$ é definida pela equação 2.43, onde k representa o tempo discreto.

$$s_{ji}(k) = \sum_{n=0}^{M} w_{ji}(n) x_i(k-n) + w_b \tag{2.43}$$

Então a soma dos potenciais sinápticos, é definido por:

$$s_j(k) = \sum_{i=1}^{N} \sum_{n=0}^{M} w_{ji}(n) x_i(k-n) + w_b \tag{2.44}$$

Na expressão acima, o comportamento temporal é representado pelo somatório interno e o comportamento espacial pelo somatório externo.

A figura 2.13 mostra à esquerda uma rede FIR com 4 camadas: a camada de entrada (camada 0, l=0), duas camadas ocultas (camadas l=1 e l=2) e a camada de saída (camada 3, l =3). A direita tem-se a ampliação das sinapses entre os neurônios da camada l=1 e o neurônio do meio da camada l=2.

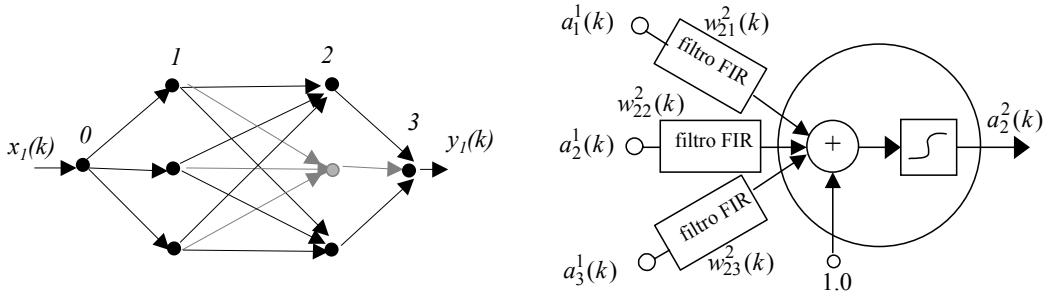

Figura 2.13 Neurônio FIR numa rede multicamada (Soto, 1999).

Para treinamento da rede FIR, utiliza-se o algoritmo de retro-propagação temporal (Wan, 1994).

O algoritmo de retro-propagação temporal pode ser definido pelas equações (2.45), (2.46) e (2.47).

$$w_{ji}(k+1) = w_{ji}(k) + \Delta w_{ji}(k) \tag{2.45}$$

$$\Delta w_{ji}^{L-n}(k) = \eta \delta_j^{L-n}(k - nM) \cdot a_i^{L-1-n}(k - nM) \tag{2.46}$$

$$\delta_j^{L-n}(k - nM) = \begin{cases} 2e_j(k)\varphi'(s_j^L(k)) & n = 0 \\ \varphi'(s_j^{L-n}(k - nM) \cdot \sum_m \delta_m^{L+1-n}(k - nM) \cdot w_{jm}^{L+1-n} & 1 \le l \le L-1 \end{cases} \tag{2.47}$$

2.4.8.3
Redes neurais recorrentes

Nestas redes o comportamento dinâmico é simulado por conexões de realimentação, como ilustra a figura 2.14. A realimentação pode ser de dois tipos: a) local que é situada ao nível de neurônio; b) global que envolve alguma(s) camada(s) completa(s). As redes recorrentes podem ser utilizadas de duas maneiras: a) memórias associativas; b) mapeamento entrada-saída; sendo a segunda maneira utilizada para o processamento de séries temporais.

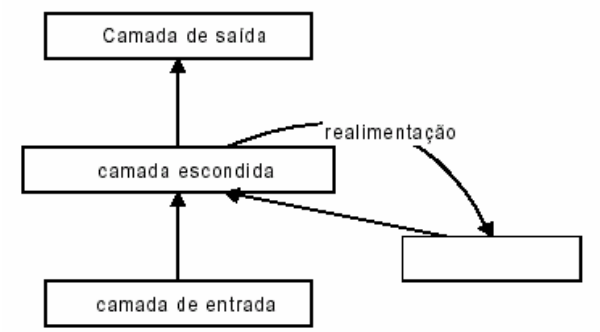

Figura 2.14 Esquema de processamento temporal utilizando redes neurais recorrentes (Soto 1999).

Um tipo de rede neural temporal com realimentação global é rede de Elman. Nesta rede cada um dos neurônios da camada oculta tem realimentação para as unidades de contexto, como ilustra a figura 2.15. Esta é a rede neural recorrente utilizada neste trabalho, na próxima seção é apresentada a sua formulação.

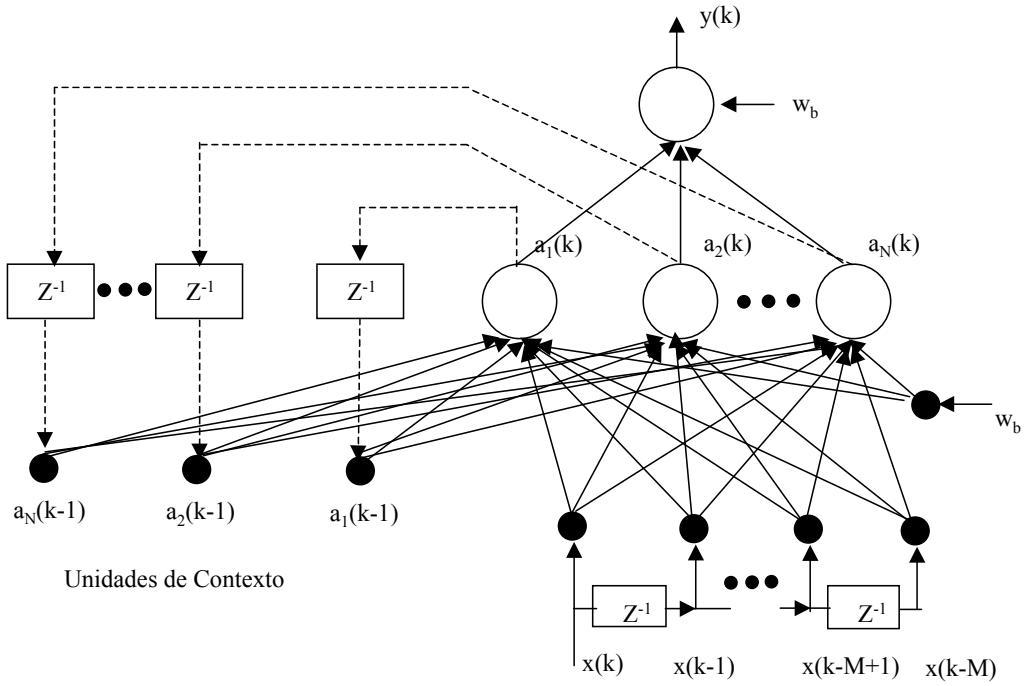

Figura 2.15 Rede de Elman (Soto, 1999).

Na rede de Williams-Zipser a(s) camada(s) oculta(s) possuem unidades de contexto e um laço de realimentação da saída da rede para as unidades escondidas (Figura 2.16).

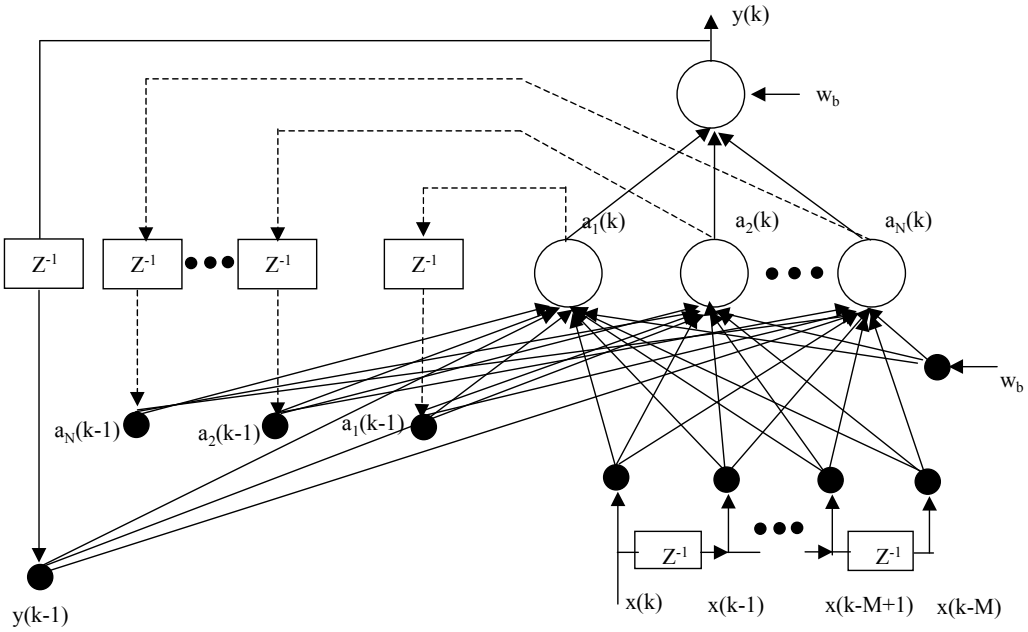

Figura 2.16 Rede de Williams-Zipser (Soto, 1999).

Na rede tipo Jordan considera-se somente realimentação dos valores de ativação de saída para as unidades de contexto (figura 2.17).

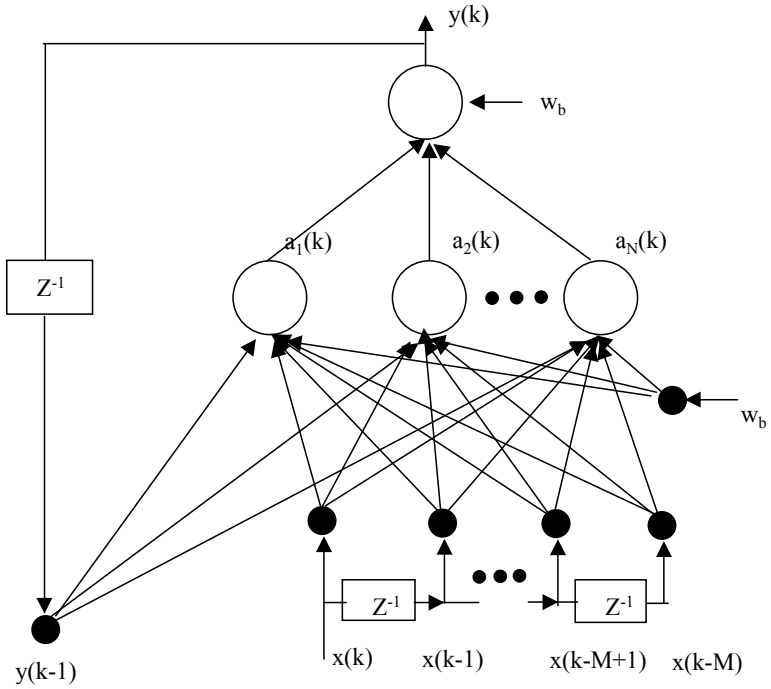

Figura 2.17 Rede de Jordan (Soto, 1999).

2.4.8.4
Redes neurais Elman

Na rede neural Elman ou rede completamente recorrente (Figura 2.18), a saída de cada neurônio da camada oculta realimenta todos os neurônios da camada oculta. Neste tipo de arquitetura que utiliza o conceito de realimentação global, cada neurônio da camada oculta está completamente conectado ao vetor de estados a(k-1).

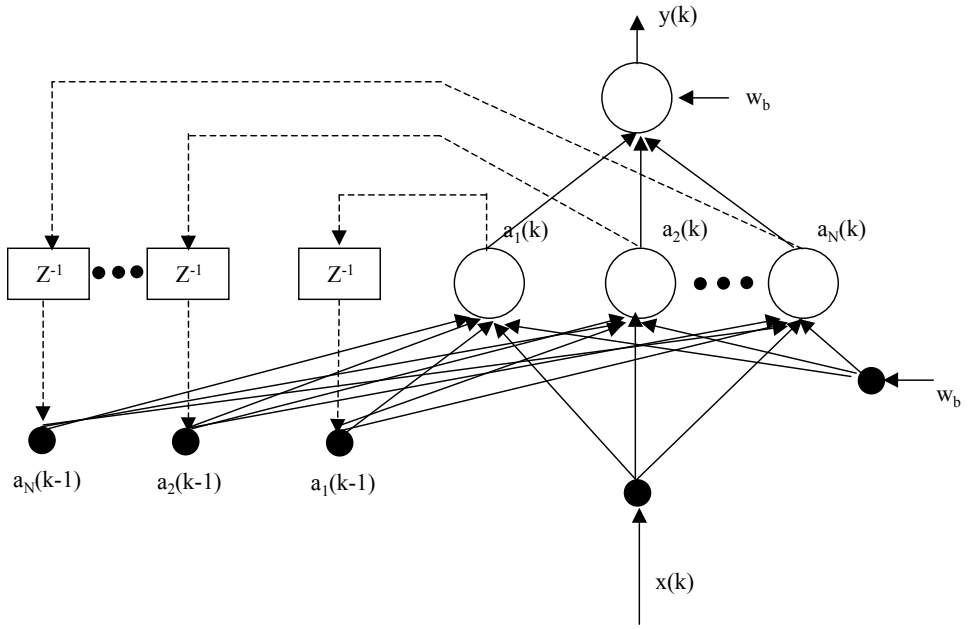

Figura 2.18 Rede completamente recorrente – Rede de Elman. (Soto, 1999).

onde

x(k) é a entrada externa no tempo discreto k,

a(k) é o vetor contendo as saídas dos N neurônios da camada oculta no tempo discreto k,

$z^h(k)$ é o conjunto de entradas dos neurônios da camada oculta, denominado vetor de estados,

y(k) é à saída da rede no tempo discreto k.

A seguir, são mostradas as equações para uma rede neural Elman com uma entrada e uma saída (Figura 2.18). As entradas da camada oculta são definidas por:

$$z_i^h(k) = \begin{cases} x_i(k) & , \quad i \in l \\ a_i(k-1) & , \quad i \in H \end{cases} \tag{2.48}$$

onde

l representa uma camada externa,

H representa o conjunto de saídas dos neurônios da camada oculta.

A saída do j-ésimo neurônio da camada oculta é:

$$a_j(k) = \varphi(\sum_{i \in l} w_{ji} x_i(k) + \sum_{i \in H} w_{ji} a_i(k-1) + w_{jb}) \quad , \quad j \in H \tag{2.49}$$

onde

φ() é a função de ativação não-linear,

w_{jb} é o peso do bias.

Para o neurônio da camada de saída, a saída y(k) é representada pela equação 2.50, onde "o" representa camada de saída.

$$y(k) = f(\sum_{i \in H} w_{oi} a_i(k) + w_{bo}) \tag{2.50}$$

O algoritmo de treinamento da rede neural Elman é o algoritmo de regularização bayseana. É importante notar que a quantidade de pesos sinápticos aumenta, já que as entradas da camada oculta são as entradas externas e os valores de saídas da camada oculta no tempo anterior.

Deve-se tomar o cuidado de considerar valores iniciais das saídas dos neurônios ocultos e de saída no tempo (k=0). Este cuidado permite que as saídas no tempo k=0 não influenciem a saída da rede na primeira interação. Adotam-se como valores iniciais:

$$\begin{matrix} a_j(0) = 0 \\ y(0) = 0 \end{matrix} \quad , \quad j \in H \tag{2.51}$$

2.5
Métodos Geoestatisticos

Métodos geoestatísticos foram desenvolvidos na década de 60 pelo engenheiro francês Matheron (1962, 1970) para o estudo de variáveis regionalizadas. As variáveis regionalizadas apresentam uma característica importante que é a sua continuidade , cuja avaliação está fundamentada nos

princípios da regressão linear que, na estatística, investiga a dependência entre variáveis de um fenômeno.

Na geoestatística, baseada na teoria das funções aleatórias, procura-se avaliar a dependência da variável com ela própria, avaliada em diferentes posições separadas por um vetor $\vec{h}$. A continuidade da variável pode ser descrita através de 3 parâmetros inter-relacionados: a função de covariância $C(h)$, o correlograma $\rho(h)$ e o variograma $\gamma(h)$.

2.5.1
Função Covariância

Neste trabalho adotaremos o símbolo ~ (til) para referenciar os parâmetros do modelo e o símbolo ^ (acento circunflexo) para os valores estimados. Então $\tilde{m}$ corresponde à média dos valores idealizados como realizações de uma função aleatória, m significa a média aritmética entre valores observados.

A função covariância, como o nome indica, representa a variação entre variáveis separadas pela distância h.

$$\tilde{C}_v(h) = Cov\{V(x).V(x+h)\} \tag{2.52}$$

$$\tilde{C}_v(h) = E\{V(x).V(x+h)\} - E\{V(x)\}.E\{V(x+h)\} \tag{2.53}$$

Onde o valor esperado da variável aleatória $E\{V(x)\}$ denota a sua média $\tilde{m}$. Para funções aleatórias estacionárias, $E\{V(x)\} = E\{V(x+h)\}$, resultando:

$$\tilde{C}_v(h) = E\{V(x).V(x+h)\} - E\{V(x)\}^2 \tag{2.54}$$

2.5.2
Função correlação ou correlograma

É expressa pela normalização da função covariância em relação aos desvios padrões.

$$\widetilde{\rho}_V(h) = \frac{Cov\{V(x)\cdot V(x+h)\}}{\sqrt{Var\{V(x)\}\cdot Var\{V(x+h)\}}} \tag{2.55}$$

Como a covariância entre variáveis na mesma posição é a variância da função aleatória,

$$\widetilde{C}_V(0) = Cov\{V(x)\cdot V(x)\} = Var\{V(x)\} \tag{2.56}$$

$$\widetilde{C}_V(0) = E\{V^2(x)\} - E\{V(x)\}^2 \tag{2.57}$$

Então

$$\widetilde{\rho}_V(h) = \frac{\widetilde{C}_V(h)}{\widetilde{C}_V(0)} \tag{2.58}$$

2.5.3 Variograma

O variograma é expresso matematicamente como a metade do quadrado das diferenças esperado entre variáveis aleatórias distanciadas de h,

$$\widetilde{\gamma}_v(h) = \frac{1}{2}E\{[V(x) - V(x+h)]^2\} \tag{2.59}$$

$$\widetilde{\gamma}_v(h) = \frac{1}{2}E\{V^2(x)\} - E\{V(x)\cdot V(x+h)\} + \frac{1}{2}E\{V^2(x+h)\} \tag{2.60}$$

Considerando que para funções aleatórias estacionárias $E\{V^2(x)\} = E\{V^2(x+h)\}$, a expressão acima pode ser escrita como

$$\widetilde{\gamma}_v(h) = E\{V^2(x)\} - E\{V(x)\cdot V(x+h)\} \tag{2.61}$$

Adicionando e subtraindo $E\{V(x)\}^2$ ao segundo termo da equação acima,

$$\widetilde{\gamma}_v(h) = E\{V^2(x)\} - E\{V(x)\}^2 - \{E[V(x)\cdot V(x+h)] - E\{V(x)\}^2\} \tag{2.62}$$

Com auxilio das equações (2.54), (2.56) e (2.61) obtemos,

$$\widetilde{\gamma}_v(h) = \widetilde{C}_V(0) - \widetilde{C}_V(h) \tag{2.63}$$

A equação (2.54) pode ser escrita como

$$\widetilde{C}_v(h) = \widetilde{C}_V(0) - \widetilde{\gamma}_v(h) \tag{2.64}$$

$$\widetilde{C}_v(h) = \widetilde{\gamma}_v(\infty) - \widetilde{\gamma}_v(h) \tag{2.65}$$

As funções covariância, correlograma e variograma fornecem as mesmas informações de maneira ligeiramente diferente, como mostra a figura 2.19. O correlograma inicia com o valor 1, tendendo a zero com o crescimento de h, enquanto que a função covariância decresce de forma similar a partir do valor inicial $\widetilde{C}(0) = \widetilde{\sigma}^2$. O variograma inicia em 0 aumentando até o valor máximo $\widetilde{\gamma}^2(\infty) = \widetilde{\sigma}^2$, referenciando como patamar ("*Sill*").

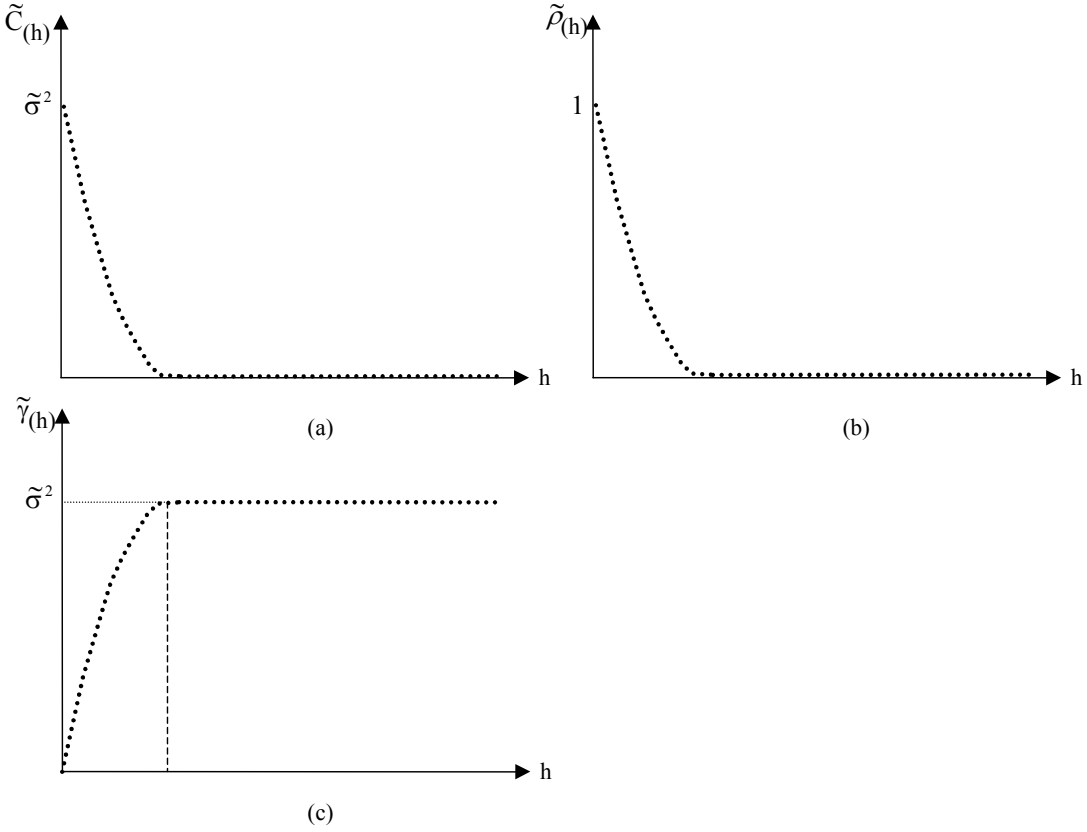

Figura 2.19 Esquema de funções: (a) covariância, (b) correlograma e (c) variograma (Xavier, 1999).

Na figura 2.20 são descritas as principais características da função variograma $\tilde{\gamma}(h)$:

a) Alcance a ("*range*"): distância a partir da qual os valores do variograma permanecem essencialmente constantes.

b) Patamar C + C_0 ("*sill*"): valor máximo do variograma para distâncias muito grandes $\tilde{\gamma}(\infty)$. Expressa a variância $\tilde{\sigma}^2$ da variável aleatória.

c) Efeito pepita C_0: representa descontinuidades na origem causada por diversos fatores, como por exemplo erros de amostragem.

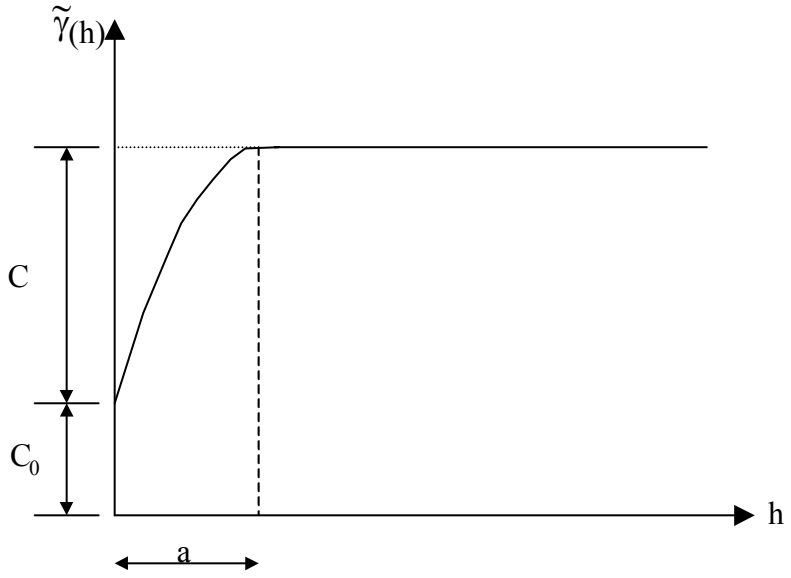

Figura 2.20 Esquema básico de uma função variograma (Xavier, 1999).

2.5.4
Modelos Variográficos

Os modelos variográficos são utilizados para se obter valores de $\gamma(h)$ em distâncias e direções para as quais não se dispõem de observações experimentais. Já que no método de krigagem ordinária, estes valores $\gamma(h)$ serão necessários.

Como os resultados das estimativas devem existir e serem únicos, o sistema de equações lineares gerados pela krigagem ordinária necessita possuir uma matriz positivo-definida que, por sua vez, impõe a condição de que modelos variográficos sejam construídos com auxilio de funções positivo-definidas. Dentre estes principais modelos podem ser citados (Xavier, 1999):

a) Modelo esférico

Modelo variográfico mais freqüentemente usado, expresso por:

$$\gamma(h) = \begin{cases} C \cdot \left[1.5 \cdot \left(\frac{h}{a} \right) - 0.5 \cdot \left(\frac{h}{a} \right)^3 \right] & para \cdot \cdot h \le a \\ C & para \cdot \cdot h > a \end{cases} \tag{2.66}$$

b) Modelo exponencial:

$$\gamma(h) = C\left(1 - e^{-\frac{3h}{a}} \right) \tag{2.67}$$

c) Modelo Gaussiano:

Modelo variográfico freqüentemente usado para fenômenos naturais com elevada continuidade.

$$\gamma(h) = C\left(1 - e^{-\frac{3h^2}{a^2}} \right) \tag{2.68}$$

O efeito pepita pode ser considerado somado-se ao modelo variográfico como uma constante C_0.

2.5.5 Krigagem ordinária

O método de krigagem ordinária é associado na língua inglesa, a sigla B.L.U.E significando "*the best linear unbiased estimator*". Linear porque suas estimativas são feitas por combinações lineares, "*unbiased*"(sem viés) porque o erro de estimação esperado no modelo é nulo ($m_R = 0$) e "*best*" porque seu objetivo é minimizar a variância dos erros $\widetilde{\sigma}_R^2$.

O erro de estimação é expresso por

$$R(x_0) = \hat{V}(x_0) - V(x_0) \tag{2.69}$$

e sua variância pode ser definida como:

$$Var\{R(x_0)\} = Var\left\{\hat{V}(x_0) - V(x_0)\right\} \tag{2.70}$$

$$Var\{R(x_0)\} = Cov\left\{\hat{V}(x_0)\cdot\hat{V}(x_0)\right\} - 2\cdot Cov\left\{\hat{V}(x_0)\cdot V(x_0)\right\} + Cov\{V(x_0)\cdot V(x_0)\} \tag{2.71}$$

O termo $Cov\left\{\hat{V}(x_0)\cdot\hat{V}(x_0)\right\}$ representa a própria variância de $\hat{V}(x_0)$, ou seja

$$Cov\left\{\hat{V}(x_0)\cdot\hat{V}(x_0)\right\} = Var\left\{\hat{V}(x_0)\right\} \tag{2.72}$$

Sabendo que a estimativa $\hat{V}(x_0)$ é expressa pela combinação linear de $V(x_i)$ para *i = 1,2,...,n*

$$\hat{V}(x_0) = \sum_{i=1}^{n} w_i \cdot V(x_i) \tag{2.73}$$

Resulta

$$Cov\left\{\hat{V}(x_0)\cdot\hat{V}(x_0)\right\} = Var\left\{\sum_{i=1}^{n} w_i \cdot V(x_i)\right\} \tag{2.74}$$

$$Cov\left\{\hat{V}(x_0)\cdot\hat{V}(x_0)\right\} = \sum_{i=1}^{n}\sum_{j=1}^{n} w_i \cdot w_j \cdot Cov\left\{V(x_i)\cdot V(x_j)\right\} \tag{2.75}$$

$$Cov\left\{\hat{V}(x_0)\cdot\hat{V}(x_0)\right\} = \sum_{i=1}^{n}\sum_{j=1}^{n} w_i \cdot w_j \cdot \widetilde{C}_{ij} \tag{2.76}$$

O termo $2\cdot Cov\left\{\hat{V}(x_0)\cdot V(x_0)\right\}$pode ser escrito como

$$2\cdot Cov\left\{\hat{V}(x_0)\cdot V(x_0)\right\} = 2\cdot Cov\left\{\left(\sum_{i=1}^{n} w_i \cdot V(x_i)\right)\cdot V(x_0)\right\} \tag{2.77}$$

$$2\cdot Cov\left\{\hat{V}(x_0)\cdot V(x_0)\right\} = 2\cdot E\left\{\sum_{i=1}^{n} w_i \cdot V(x_i)\cdot V(x_0)\right\} - 2\cdot E\left\{\sum_{i=1}^{n} w_i \cdot V(x_i)\right\}\cdot E\{V(x_0)\} \tag{2.78}$$

$$2\cdot Cov\left\{\hat{V}(x_0)\cdot V(x_0)\right\} = 2\cdot \sum_{i=1}^{n} w_i \cdot\left\{E\{V(x_i)\cdot V(x_0)\} - \cdot E\{V(x_i)\}\cdot E\{V(x_0)\}\right\} \tag{2.79}$$

$$2 \cdot Cov\left\{\hat{V}(x_0) \cdot V(x_0)\right\} = 2 \cdot \sum_{i=1}^{n} w_i \cdot \widetilde{C}_{i0} \tag{2.80}$$

O termo $Cov\{V(x_0) \cdot V(x_0)\}$ pode ser expresso como

$$Cov\{V(x_0) \cdot V(x_0)\} = Var\{V(x_0)\} \tag{2.81}$$

$$Cov\{V(x_0) \cdot V(x_0)\} = \widetilde{\sigma}^2 \tag{2.82}$$

Assim, a equação (2.71) pode ser então finalmente escrita como

$$\widetilde{\sigma}_R^2 = \widetilde{\sigma}^2 + \sum_{i=1}^{n}\sum_{j=1}^{n} w_i \cdot w_j \cdot \widetilde{C}_{ij} - 2 \cdot \sum_{i=1}^{n} w_i \cdot \widetilde{C}_{i0} \tag{2.83}$$

Uma vez selecionado o modelo do variograma (ou da função covariância) é possível determinar-se $\widetilde{\sigma}^2$ e todas as covariâncias $\widetilde{C}_{ij}$.

A minimização da função a n variáveis produz então um sistema de n equações a n incógnitas que pode ser resolvido por qualquer método da álgebra linear para solução de um sistema de equações lineares.

Para garantir que a condição de não viés seja automaticamente satisfeita é acrescentado à equação (2.83) a condição de restrição $\left(\sum_{i=1}^{n} w_i = 1 \right)$, resultando em um problema de minimização com restrição. Então este problema de minimização com restrição deve ser resolvido pelo método do multiplicador de Lagrange (2.84).

$$\widetilde{\sigma}_R^2 = \widetilde{\sigma}^2 + \sum_{i=1}^{n}\sum_{j=1}^{n} w_i \cdot w_j \cdot \widetilde{C}_{ij} - 2 \cdot \sum_{i=1}^{n} w_i \cdot \widetilde{C}_{i0} + 2 \cdot \mu \cdot \left(\sum_{i=1}^{n} w_i - 1 \right) \tag{2.84}$$

O multiplicador de Lagrange μ introduz uma nova incógnita no problema, agora expresso sem restrição já que a condição de não viés é automaticamente satisfeita por (2.84).

$$\frac{\partial \widetilde{\sigma}_R^2}{\partial w_i} = 0 \qquad \text{para } i = 1,...,n \tag{2.85}$$

$$\frac{\partial \tilde{\sigma}_R^2}{\partial \mu} = 0 \tag{2.86}$$

produz um sistema de (n+1) equações que pode ser expresso sob forma matricial da seguinte maneira:

$$\begin{bmatrix} \tilde{C}_{11} & ... & \tilde{C}_{1n} & 1 \\ \tilde{C}_{21} & ... & \tilde{C}_{2n} & 1 \\ . & . & . & . \\ . & . & . & . \\ . & . & . & . \\ \tilde{C}_{n1} & ... & \tilde{C}_{nn} & 1 \\ 1 & ... & 1 & 0 \end{bmatrix} \cdot \begin{Bmatrix} w_1 \\ w_2 \\ . \\ . \\ . \\ w_n \\ \mu \end{Bmatrix} = \begin{Bmatrix} \tilde{C}_{10} \\ . \\ . \\ . \\ . \\ \tilde{C}_{n0} \\ 1 \end{Bmatrix} \tag{2.87}$$

ou,

$$[C] \cdot \{W\} = \{D\} \tag{2.88}$$

possibilitando que as incógnitas $w_i, ..., w_n$ sejam obtidas sem maiores dificuldades através do método de eliminação de Gauss, por exemplo:

O valor da variância do erro minimizada pode enfim ser calculada como:

$$\tilde{\sigma}_R^2 = \tilde{\sigma}^2 - \{W\}^T \cdot \{D\} \tag{2.89}$$