2
Fundamentação Teórica

2.1.
Introdução

Neste capítulo são apresentados os conceitos fundamentais sobre Projeto de Experimentos e sobre as técnicas de inteligência computacional empregadas neste trabalho para construção do aproximador de função: redes neurais artificiais *Multilayer Perceptron* (*MLP*), com aprendizado com retropropagação do erro (*backpropagation*), e o modelo neuro-fuzzy *Adaptive-Network-Based Fuzzy Inference Systems (ANFIS).*

2.2.
Projeto de Experimentos

2.2.1.
Introdução

Projeto de experimentos (*Design of Experiments* – DOE) (Montgomery, 2000) é uma técnica utilizada para analisar o efeito da variação simultânea de variáveis (fatores) com o objetivo de obter o máximo de informação com menor número de experimentos.

O projeto de experimentos se inicia determinando os objetivos dos mesmos e selecionando os fatores que serão objeto de estudo. Um projeto de experimentos bem definido objetiva maximizar a quantidade de informação que se pode obter com uma quantidade determinada de esforço ou minimizar o esforço para uma quantidade de informação desejada.

Um experimento é composto por uma ou mais saídas, um conjunto de entrada com fatores controláveis e outro conjunto com fatores que não se pode, ou não se deseja, controlar (Figura 1).

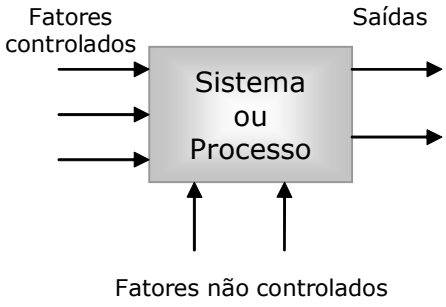

Figura 1 – Estrutura de um experimento.

O projeto de experimentos pode ser utilizado para atingir diversos objetivos, entre eles (NIST/SEMATECH, 2008):

- *Screening Experiments* – Identificar, entre fatores candidatos, os que mais influenciam a saída;
- *Comparative experiments* – Concluir a respeito de um importante fator;
- *Modelagem de Superfície de Resposta* para:
  - Maximizar ou minimizar a saída;
  - Reduzir variação de saída, identificando regiões onde um processo é mais fácil de ser gerenciado;
  - Tornar um processo robusto e;
- *Regression Modeling* – Estimar um modelo preciso, quantificando a dependência da saída em função das entradas, particularmente útil para sistemas com poucos fatores.

Para entender o projeto de experimentos, algumas definições básicas devem ser descritas, como apresentadas a seguir.

Fator ⇨ Um fator é uma variável, controlada ou não, que exerce influência sobre a saída do sistema.

Nível ⇨ Os níveis de um fator são os valores deste fator que estão sendo examinados. Como exemplo, seja um experimento onde se deseja observar a qualidade de um produto submetido a quatro diferentes condições de temperatura durante sua produção. Neste caso, as quatro condições são os níveis deste fator.

Efeito principal ⇨ O efeito principal de um fator é aquele associado à sensibilidade do sistema a um fator isoladamente. Na Figura 2 tem-se dois fatores (*A* e *B*) com dois níveis cada, simbolizados pelos sinais "-" e "+". Assim,

"*A*-" significa o fator *A* experimentado no nível "-", significando, por exemplo, uma temperatura baixa. Pode-se observar na figura o efeito da variação de nível de cada fator sobre a saída. Percebe-se que os fatores são independentes. Em outras palavras, a mudança do nível de um fator influencia na saída de forma independente dos demais. Ainda pela Figura 2, pode-se observar que quando o fator *A* muda do nível "-" para "+", a variação na saída é sempre de 0,5, independente do fator *B* estar no nível "-" (0,1 – 0,6) ou "+" (0,3 – 0,8). A mesma análise pode ser realizada para o fator *B*.

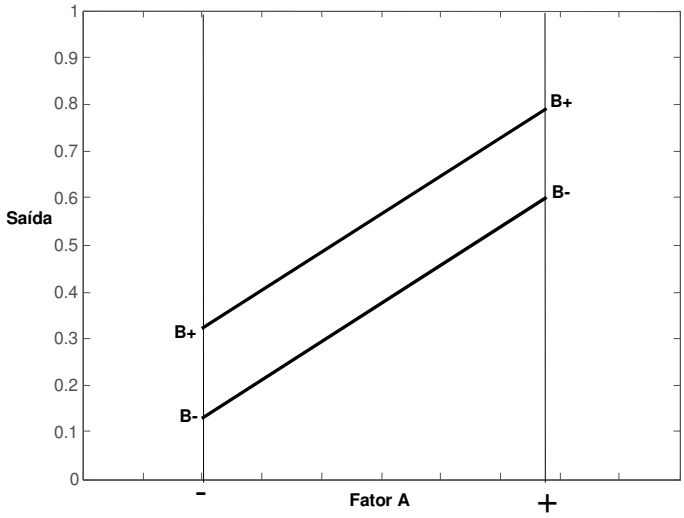

Figura 2 – Visualização gráfica do efeito principal dos fatores A e B.

Efeito de interação ⇨ O efeito de interação é observado quando um fator em conjunto com outro influencia na saída do sistema (Figura 3). Neste caso, fazendo uma análise semelhante à realizada anteriormente, pode-se observar que quando o fator *A* muda do nível "-" para "+", a variação na saída é de 0,5 quando o fator *B* está no nível "-" (0,1 – 0,6); já quando o fator *B* está no nível "+", a variação é de apenas 0,1 (0,3 – 0,4), indicando que há efeito resultante da interação entre os fatores *A* e *B*.

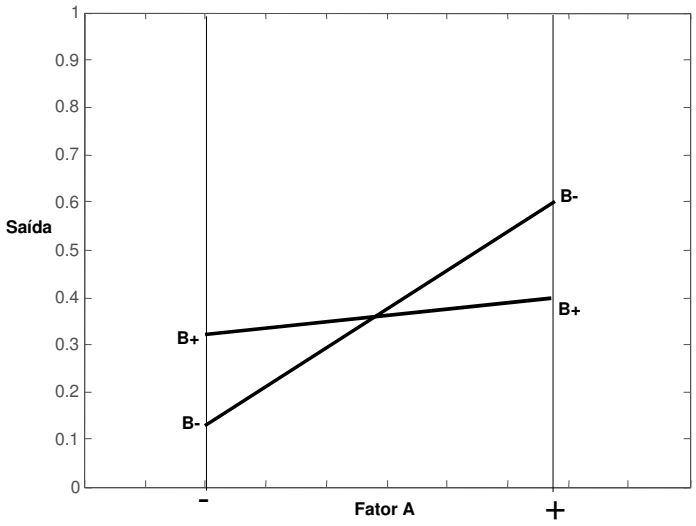

Figura 3 – Visualização gráfica do efeito de interação dos fatores *A* e *B*.

2.2.2. Projeto de Experimentos Fatoriais Completo

Em um projeto de experimentos fatoriais completo, todas as combinações dos fatores são analisadas. A quantidade de ensaios necessária para que se possa realizar todas as combinações de níveis dos fatores se dá pela equação 1.

$$q = \prod_{i=1}^{k} n_i \qquad (1)$$

onde:

q é a quantidade de ensaios ou execuções,

k é a quantidade de fatores e

n é a quantidade de níveis de cada fator.

Desta forma, um projeto de experimentos fatoriais completo é considerado aceitável quando somente alguns fatores precisam ser investigados, tornando inviável quando existem muitos fatores. Isso acontece porque todas as combinações possíveis dos fatores de controle são levadas em consideração; assim, executando um projeto de experimentos fatoriais completo é possível estimar tanto os efeitos principais quanto os provenientes das interações entre fatores. Na Tabela 1 tem-se um projeto de experimentos fatoriais completo com

dois fatores (*A* e *B*) com dois níveis cada, simbolizados pelos sinais de "+" e "-", e sua interação. Os sinais da coluna de interação são obtidos multiplicando os sinais das colunas dos respectivos efeitos principais. Os sinais referentes aos níveis dos fatores poderiam simbolizar alto e baixo ou "operador João" e "operador José".

Ensaio	Fator *A*	Fator *B*	Interação *AB*
1	-	-	+
2	-	+	-
3	+	-	-
4	+	+	+

Tabela 1 – Projeto de experimentos fatoriais completo com dois fatores com dois níveis cada e sua interação.

2.2.3. Projeto de Experimentos Fatoriais Fracionado

Diferentemente do caso anterior, em um projeto de experimentos fatoriais fracionado apenas uma fração adequadamente escolhida das combinações requeridas de um projeto e experimentos fatoriais completo é selecionada para ser executada.

A necessidade do projeto de experimentos fatoriais fracionado se dá, principalmente, pelo incremento da quantidade de ensaios (execuções) necessárias ao projeto de experimentos fatoriais completo quando são acrescentados novos fatores. Para exemplificar, admitindo-se apenas seis fatores com dois níveis cada, são necessários 64 experimentos. Se o número de fatores aumentar para oito, serão necessários 256 experimentos (Eq. 1).

Diante do, normalmente alto, custo de experimentação e de observações onde se têm muitos fatores controlados, uma estratégia que reduza a quantidade de experimentos se faz necessária. A solução se dá pelo uso de uma fração dos experimentos determinados pelo projeto de experimentos fatoriais completo. O interesse é determinar quais ensaios devem ser realizados e quais não devem.

Existem diversas estratégias para garantir uma escolha apropriada dos ensaios a realizar. Entendem-se como apropriada, neste caso, a garantia das propriedades de balanceamento (caso em que a combinação dos níveis tem a mesma quantidade de observações) e ortogonalidade entre fatores (Montgomery, 2000). A técnica utilizada para se obter essa fração é baseada no confundimento (*confounding* ou *alises*) e se constitui em uma forma de arranjar

as combinações de um ensaio fatorial completo em blocos com pequenos números de combinações, aproveitando-se das interações não significativas para tal fim. Neste caso, substituem-se as interações entre fatores consideradas como não influentes por outros fatores. O preço a pagar pelo uso das interações de alta ordem é que se torna impossível distinguir o efeito resultante destas interações e o efeito resultante do fator que está sendo substituído (confundindo). De fato, o efeito será resultante da soma dos efeitos da interação e do fator confundido. Isto não se torna um problema, desde que se possa assumir a hipótese de que as interações de alta ordem são insignificantes e podem ser negligenciadas. Tomando como referência a Tabela 1, na Tabela 2 pode-se observar a substituição da interação entre os fatores *A* e *B* pelo fator *C*.

Ensaio	Fator *A*	Fator *B*	Fator *C*
1	-	-	+
2	-	+	-
3	+	-	-
4	+	+	+

Tabela 2 – Projeto de experimentos fatoriais fracionado com 3 fatores e 2 níveis cada. O fator *C* se confunde com o efeito de interação dos fatores *A* e *B*.

Um projeto de experimentos fatoriais fracionado do tipo 2^{k-p} contém 2^{k-p} experimentos (ou combinações) e são chamados como $1/2^{p}$ frações de 2^{k}, ou simplesmente 2^{k-p} delineamento fatorial fracionário

O principal uso de experimentos fatoriais fracionados é na identificação dos fatores influentes (*screening expriments*) (Montgomery, 2000).

O grau em que os efeitos se associam é denominado resolução. Um projeto de experimentos é de uma resolução *R* se nenhum *"p"* efeito de fator é confundido com outro efeito menor que *"R-p"* fatores. Os mais utilizados são os de resolução III, IV e V:

Resolução III: Neste tipo de projeto nenhum efeito principal se confunde com outro efeito principal, porém podem se confundir com efeitos de interações de dois fatores (primeira ordem);

Resolução IV: Neste tipo de projeto nenhum efeito principal se confunde com outro efeito principal ou com interações de primeira ordem, porém interações de primeira ordem podem se confundir entre si;

Resolução V: Não há confundimento entre efeitos principais ou o efeito de interação de dois fatores é confundido com efeitos principais ou de interações de dois fatores. Apenas efeitos de interação de dois fatores se confundem com efeitos de interação de três fatores.

Desta forma, sempre que se desejar diferenciar os efeitos resultantes dos fatores principais e os efeitos resultantes da interação entre fatores, se desejará, por conseqüência, a maior resolução que seja possível para o grau de fracionamento necessário. Porém, no limite, a técnica apresentada permite que k fatores possam ter seus efeitos analisados com apenas $k+1$ observações (Lochner, 1990), lidando com a perda de possibilidade da distinção entre os efeitos determinados pelos fatores principais e os oriundos de interação de fatores.

Para exemplificar um projeto de experimentos fatoriais fracionado, pode-se considerar um projeto de experimentos fatoriais completo de dois níveis (por exemplo, alto e baixo) para 3 fatores (*A*, *B* e *C*), onde se deseja reduzir à metade a quantidade de observações.

A quantidade necessária de observações em um projeto de experimentos fatoriais completo seria de 2^3 execuções (equação 1), indicados na Figura 4 pelos vértices do cubo e na Tabela 3.

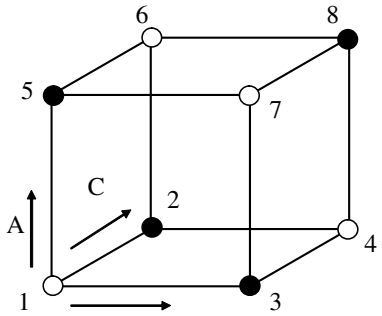

Figura 4 – Visão geométrica de projeto de experimentos fatoriais com três fatores com dois níveis cada.

Ensaio	Fator A	Fator B	Fator C	Interação AB	Interação AC	Interação BC	Interação ABC
1	-	-	-	+	+	+	-
2	-	-	+	+	-	-	+
3	-	+	-	-	+	-	+
4	-	+	+	-	-	+	-
5	+	-	-	-	-	+	+
6	+	-	+	-	+	-	-
7	+	+	-	+	-	-	-
8	+	+	+	+	+	+	+

Tabela 3 – Projeto de experimentos fatoriais completo com 3 fatores e 2 níveis cada. Referência à Figura 4.

Entretanto é possível reduzir a quantidade de experimentos necessários de acordo com a técnica de confundimento. Neste exemplo realiza-se um fracionamento à metade (2^{3-1}) indicados na Figura 4 pelos pontos escuros nos vértices do cubo e na Tabela 4. A aplicação da técnica de confundimento resultou em um projeto experimental com 4 ensaios, como na Tabela 1, onde fator *C* foi confundido com a interação do fator *A* com o fator *B*. Assim, neste projeto de experimentos fatoriais fracionado, pode-se analisar a influência dos fatores *A*, *B* e *C*. Deve-se observar que não se pode distinguir a influência na saída obtida pela análise do fator *C* da influência na saída originada pelo efeito de interação dos fatores *A* e *B*.

Ensaio	Fator A	Fator B	Fator C
2	-	-	+
3	-	+	-
5	+	-	-
8	+	+	+

Tabela 4 – Projeto de experimento fatoriais fracionado à metade com 3 fatores e 2 níveis cada referente à Figura 4.

2.3.
Redes Neurais Artificiais

Redes Neurais Artificiais (Haykin, 2001) são modelos computacionais não lineares, inspirados na estrutura de neurônios interconectados existente no cérebro humano, capazes de realizar operações de aprendizado, associação, generalização e abstração. As redes neurais são compostas por diversos elementos processadores (neurônios artificiais), altamente interconectados, que efetuam operações simples, transmitindo seus resultados aos processadores vizinhos. A habilidade das redes neurais em realizar mapeamentos não-lineares entre suas entradas e saídas as tem tornado prósperas no reconhecimento de padrões (Bishop, 1995) e na modelagem de sistemas complexos.

Na literatura, pode-se encontrar muitos tipos de redes neurais, com diferentes arquiteturas e algoritmos de aprendizado. Embora exista uma grande quantidade de arquiteturas de redes neurais, a estrutura multicamada (MLP – *Multilayer Perceptron*) é a mais conhecida e utilizada (Hush, 1993), devido à capacidade de aproximação universal e de generalização para uma ampla classe de problemas (Iyoda, 2000).

Deste modo, neste trabalho, optou-se por avaliar o desempenho da arquitetura *multilayer perceptron feedforward* com a seleção de padrões de treinamento através da técnica de análise de experimentos. O algoritmo de aprendizado utilizado foi o *Backpropagation* (Haykin, 2001), no qual utiliza-se um conjunto de exemplos de entrada-saída para se efetuar o aprendizado supervisionado. A partir das entradas apresentadas, a rede realiza seu processamento e a saída obtida é comparada com a saída esperada. A partir do erro obtido, um processo de ajuste de pesos é aplicado, buscando-se minimizar o erro, conforme detalhado nas seções subseqüentes.

2.3.1.
Arquitetura Multilayer Perceptron

A arquitetura *Multilayer Perceptron* (MLP) é a mais utilizada nas aplicações de engenharia. Conforme indicado pelo próprio nome, a arquitetura MLP é organizada em diversas camadas: uma camada de entrada, formada pelos neurônios que estão conectados às entradas globais da rede, isto é, recebem os atributos de entrada; uma camada de saída, contendo os neurônios que apresentam as saídas da rede neural ao ambiente externo; e uma ou mais camadas intermediárias (ou escondidas), compostas de neurônios cujas

entradas e saídas estão conectadas somente a outros neurônios, não havendo interação com o ambiente externo à rede.

Portanto, de modo geral, esta arquitetura possui um vetor de entradas $\mathbf{x}$ de dimensão d $\mathbf{x} = [x_1, x_2, \mathrm{K}, x_d]$; um vetor de c saídas $\mathbf{y} = [y_1, y_2, \mathrm{K}, y_c]$; e M neurônios na camada escondida (pode possuir mais de uma camada escondida).

Todos os parâmetros adaptáveis (pesos e *bias*) desta arquitetura são agrupados convenientemente em um único vetor w-dimensional $\mathbf{w} = [w_1, w_2, \mathrm{K}, w_W]$ para facilitar tratamentos analíticos.

Além disso, para o cálculo dos parâmetros adaptáveis (treinamento) $\mathbf{w} = [w_1, w_2, \mathrm{K}, w_W]$, utiliza-se um conjunto de observações (dados de treinamento) $D = \left\{ \left(x^{(1)}, t^{(1)}\right), \left(x^{(2)}, t^{(2)}\right), \mathrm{K}, \left(x^{(N)}, t^{(N)}\right) \right\}$ de algum processo a ser modelado, onde $\mathbf{x} = \left(x^{(1)}, x^{(2)}, \mathrm{K}, x^{(N)}\right)$ é o conjunto de dados de entrada e $\mathbf{t} = \left(t^{(1)}, t^{(2)}, \mathrm{K}, t^{(N)}\right)$ é o conjunto de dados contendo a saída desejada (Figura 5).

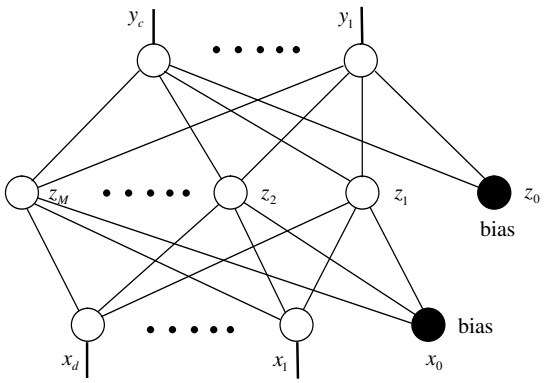

Figura 5 - Representação geral da arquitetura *multilayer perceptron* (*feedforward*).

2.3.2. Algoritmos de Aprendizado *Backpropagation*

Redes neurais com treinamento por *Backpropagation (BP)* (Haykin, 2001; Wassermann, 1993) são redes *feedforward,* de uma ou mais camadas escondidas, cujo algoritmo de aprendizado define uma maneira sistemática de atualização dos pesos das diversas camadas baseada na idéia que os erros dos neurônios das camadas escondidas são determinados pela retropropagação reversa dos erros dos neurônios da camada de saída.

O treinamento, supervisionado, é baseado no método do gradiente decrescente (*gradient descent*), buscando minimizar o erro global da camada de saída. Deste modo, a atualização do peso (Δw_{ji}) é proporcional ao negativo da derivada parcial do erro com relação ao próprio peso:

$$\Delta\omega_{ji} = -\eta \frac{\partial E_{SSE}}{\partial \omega_{ji}} \qquad (2)$$

onde η é a taxa de aprendizado e E_{SSE} (*Sum of Squared Errors*) é a função erro definida como:

$$E_{SSE} = \frac{1}{2}\sum_{p=1}^{N_p}\sum_{j=1}^{N_0}\left(t_j^p - s_j^p\right)^2 \qquad (3)$$

onde N_O é o número de processadores da camada de saída, N_p é o número de padrões de treinamento e t_j^p é o valor esperado na saída do processador j *ao se apresentar o padrão* p.

Derivando a equação 2, chega-se à seguinte fórmula:

$$\Delta\omega_{ji} = \eta \cdot s_i \cdot e_j$$
$$e_j = \begin{cases} \left(t_j - s_j\right) f'\left(net_j\right) & \text{se } j \in \text{camada de saída} \\ f'\left(net_j\right)\sum_{k=1}^{N}\omega_{kj}e_k & \text{se } j \in \text{camada de escondida} \end{cases} \qquad (4)$$

onde η é a taxa de aprendizado; s_i é a entrada associada ao peso w_{ji}; e_j é o erro do *j*-ésimo processador; t_j é o valor desejado de saída do processador *j*; s_j é o seu estado de ativação; net_j é o seu potencial interno; *f'* é a derivada da função de ativação; e N é o número de processadores na camada seguinte à camada do processador *j*.

A taxa de aprendizado η é um parâmetro importante a ser definido no aprendizado. Esta não deve ser nem muito pequena, causando um treinamento muito lento, nem muito grande, gerando oscilações. Quando a taxa de aprendizado é pequena, e dependendo da inicialização dos pesos (feita de forma aleatória), a Rede Neural pode ficar presa em um mínimo local. Quando a taxa

de aprendizado é grande, a Rede Neural pode nunca conseguir chegar ao mínimo global, pois os valores dos pesos são grandes. A solução para este problema é utilizar uma taxa de aprendizado adaptativa. Além deste parâmetro, pode-se também utilizar um termo α de momento (Haykin, 2001), proporcional à variação no valor do peso sináptico no passo anterior. Deste modo, a equação de atualização do peso sináptico w_{ji} é modificada da seguinte forma:

$$\Delta\omega_{ji}(t+1) = \eta \cdot s_i \cdot e_j + \alpha\Delta\omega_{ji}(t) \qquad (5)$$

A utilização do termo de momento tem a função de acelerar a convergência da rede, sem causar oscilações.

Como se pode verificar, o algoritmo de aprendizado do *Backpropagation* tem duas fases, para cada padrão apresentado: *Feedforward* e *Feedbackward*. Na primeira etapa, as entradas se propagam pela rede, da camada de entrada até a camada de saída, gerando a saída da rede em resposta ao padrão apresentado. Na segunda etapa, os erros se propagam na direção contrária ao fluxo de dados, indo da camada de saída até a primeira camada escondida, atualizando os pesos sinápticos. Este procedimento de aprendizado é repetido diversas vezes, até que, para todos os processadores da camada de saída e para todos os padrões de treinamento, o erro seja menor do que o especificado.

Foi demonstrado que o algoritmo *Backpropagation* é um aproximador universal (Hornik, 1989), sendo capaz de aprender qualquer mapeamento de entrada-saída. Entretanto, apesar do grande sucesso do *Backpropagation* nas mais diferentes aplicações, existem alguns problemas básicos: a definição do tamanho da rede, o longo processo de treinamento, e fenômenos como paralisia da rede (contornado diminuindo o valor de η) e mínimo local (que pode ser solucionado utilizando-se métodos estatísticos).

A definição do tamanho da rede, isto é, o número de camadas escondidas e número de processadores em cada uma dessas camadas, é um compromisso entre convergência e generalização. Convergência é a capacidade da Rede Neural de aprender todos os padrões do conjunto de treinamento. Generalização é a capacidade de responder corretamente aos padrões nunca vistos (conjunto de teste). O objetivo é utilizar a menor rede possível, de forma a se obter uma boa generalização, que seja capaz de aprender todos os padrões (Feitosa, 1999).

Para garantir a capacidade de generalização da rede, uma ferramenta padrão da estatística conhecida como validação cruzada fornece um princípio orientador atraente (Stone, 1974, 1978). A técnica consiste em dividir os dados disponíveis primeiramente em duas partes: conjunto de treinamento e conjunto de testes. O conjunto de treinamento é dividido adicionalmente em dois subconjuntos disjuntos: subconjunto de estimação e subconjunto de validação (Haykin, 2001). Com isto, é possível verificar se o modelo, com um conjunto de dados diferente daquele usado para estimar os parâmetros da rede, responde corretamente aos padrões apresentados.

O método de treinamento com parada antecipada (*early stopping*) faz uso dos subconjuntos de estimação e validação para determinar o momento de parada do treinamento, evitando que a rede se ajuste excessivamente ao subconjunto de estimação. O método consiste em apresentar o subconjunto de validação ao modelo ajustado após um determinado tempo de treinamento (Haykin, 2001). Quando o erro obtido a partir do subconjunto de validação – erro de validação – cresce de forma consistente, o treinamento da rede é interrompido (Figura 6).

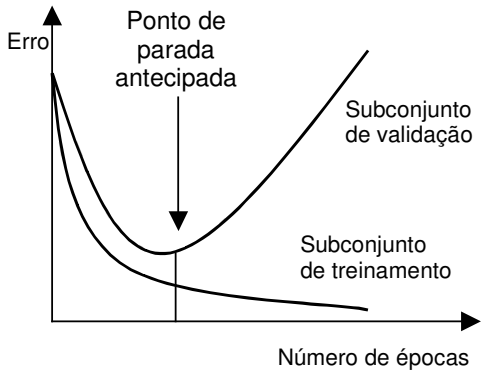

Figura 6 – Identificação do ponto de interrupção do treinamento com o método da parada antecipada.

2.4.
Modelos Neuro-*Fuzzy*

2.4.1.
Sistemas de Inferência *Fuzzy*

Sistemas de Inferência *Fuzzy* (*FIS - Fuzzy Inference System*) são modelos computacionais baseados em conceitos da teoria de conjuntos *fuzzy*, regras *fuzzy* e raciocínio *fuzzy*.

Um sistema de inferência *fuzzy* emprega regras *fuzzy* (*fuzzy if-then rules*) para modelar aspectos qualitativos do conhecimento e processamento humano, sem utilizar uma análise quantitativa precisa (Jang, 1993) (Jang, 1997) (Klir, 1995). Basicamente um *FIS* é composto por quatro blocos funcionais (Mendel, 1995):

- Módulo de "Fuzzificação" ⇨ transforma entradas precisas (*crisp*) – resultantes de medições ou observações –, em graus de pertinência associados aos conjuntos fuzzy (de entrada) relevantes. Neste módulo ocorre também a ativação das regras relevantes para uma dada situação.
- Base de conhecimentos ⇨ composta de uma base de regras *fuzzy*. As regras**,** geralmente fornecidas por especialistas, em forma de sentenças lingüísticas, constituem um dos aspectos fundamentais no desempenho de um *FIS*.
- Máquina de Inferência ⇨ É o módulo onde ocorre o raciocínio (inferência), a partir das entradas "fuzzificadas", gerando a saída (*fuzzy*) resultante a partir das regras ativadas na etapa de "fuzzificação";
- Módulo de "Defuzzificação" ⇨ transforma os resultados da inferência *fuzzy* em saída precisa (*crisp*). Uma vez obtido o conjunto *fuzzy* de saída, através do processo de inferência, este módulo efetua a interpretação dessa informação, gerando uma saída precisa.

As regras *fuzzy* (Zadeh, 1965) são expressões na forma:

SE <ANTECEDENTE> ENTÃO <CONSEQUENTE>

Antecedentes e conseqüentes são proposições do tipo "x é A" onde x é uma variável lingüística e A é um conjunto *fuzzy,* associado a uma função de pertinência.

Regras *fuzzy* são apropriadas para representar o conhecimento lingüístico/empírico, copiando o modo de raciocínio humano que nos torna capazes de tomar decisões em ambientes imprecisos e complexos (Jang, 1993).

Este *FIS*, inicialmente concebido por Zadeh e outros pesquisadores, entre os quais E. H. Mamdani, é muitas vezes referenciado como sendo do tipo *Mamdani*. Outros modelos de inferência foram propostos posteriormente, podendo ser classificados basicamente em três tipos (Wang, 1994):

- Modelo Mamdani: Apresenta conjuntos *fuzzy* nos antecedentes e conseqüentes das regras;
- Modelo Takagi e Sugeno (Takagi, 1985): Difere do Mamdani na parte do conseqüente da regra. Neste caso, o conseqüente é uma função (geralmente linear) das variáveis dos antecedentes: se x_1 é A_1 e x_2 é A_2, então $z = f(x_1, x_2)$. A saída é calculada através da média ponderada das saídas de cada regra;
- Modelo Tsukamoto: Os conseqüentes das regras são funções monotônicas de pertinência. O valor preciso é induzido pelo nível de ativação de cada regra, e a saída final é obtida pela média ponderada da saída de cada regra.

Os *FIS* tiveram suas aplicações iniciais concentradas na área de controle, onde, conforme mencionado, se procurava modelar por meio de regras lingüísticas o modo aproximado de raciocínio, tentando imitar a habilidade humana de tomar decisões racionais em um ambiente de incerteza e imprecisão. Posteriormente, a gama de aplicações cresceu consideravelmente, principalmente quando os *FIS* se tornaram também capazes de lidar com o conhecimento objetivo – expresso em dados numéricos, por exemplo. Um passo importante neste sentido foi o aparecimento dos sistemas híbridos *neuro-fuzzy*, onde um *FIS* é estruturado segundo uma rede neural, cujas camadas correspondem às diversas fases do processo de inferência. Esta hibridização faz uso da capacidade de aprendizado das redes neurais para a sintonia dos parâmetros de sistemas *fuzzy*, facilitando bastante o seu projeto e permitindo aplicações de sucesso em áreas como classificação, previsão de séries temporais e aproximação de funções.

2.4.2.
Sistemas Neuro-*Fuzzy*

Os sistemas neuro-*fuzzy* têm atraído o interesse da comunidade científica nestes últimos anos pois têm demonstrado um melhor desempenho na solução de problemas quando comparados com sistemas puros (Buckley, 1994; Buckley, 1995). Diversas estruturas foram propostas na literatura (Lin, 2001; Lee, 2000; Zhang, 1999).

Sistemas neuro-*fuzzy* estão entre os sistemas híbridos mais pesquisados atualmente, sendo este a fusão de duas ferramentas já conhecidas; Redes Neurais Artificiais e a Lógica *Fuzzy*, no qual agregam-se as características de transparência de raciocínio da lógica *fuzzy* juntamente com a capacidade de aprendizado e generalização das redes neurais artificiais.

Estes sistemas agregam as propriedades de aproximação universal e aprendizagem das redes neurais com a facilidade de manipulação de informação lingüística e dedutiva dos *FIS* em um único sistema, resultando em um modelo eficiente e robusto, com capacidade para tratar as incertezas existentes na informação (Vellasco, 2008a) (Vellasco, 2008b).

2.4.3.
ANFIS

O Sistema de Inferência Neuro-*Fuzzy* Adaptativo (ANFIS – *Adaptive-Neuro-Fuzzy Inference System*) foi proposto por Jang (1993) e se tornou muito popular por associar vantagens de duas importantes técnicas de modelagem – Redes Neurais Artificiais e Sistemas de Inferência *Fuzzy*. A idéia básica é de implementar um *FIS* através de uma arquitetura paralela distribuída, neste caso, a de uma Rede Neural Artificial, de tal forma que os algoritmos de aprendizado possam ser usados para ajustar os parâmetros do Sistema de Inferência *Fuzzy*.

O ajuste destes parâmetros é efetuado utilizando o algoritmo de retropropagação ou uma combinação deste com um algoritmo do tipo mínimos quadrados (*Least Squares*) (Chung, 2000). O modelo ANFIS é do tipo Takagi-Sugeno (Takagi, 1985), com funções lineares ou constantes nos conseqüentes das regras que formam o sistema, tendo estas regras pesos unitários. Na Figura 7 tem-se a arquitetura do ANFIS.

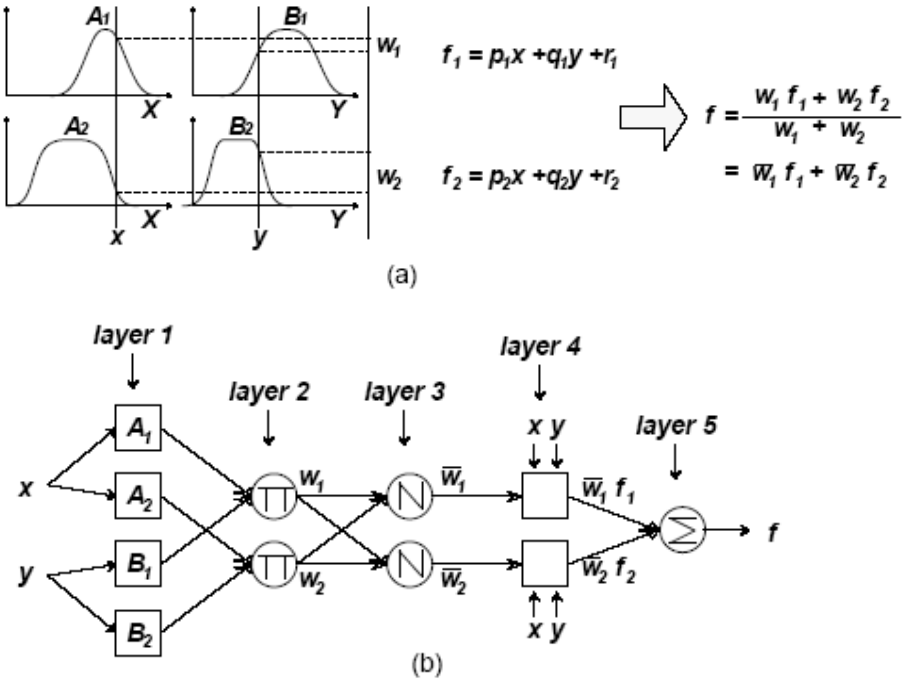

Figura 7 – Arquitetura do ANFIS (Jang,1993).

Na primeira camada (*layer 1*) calcula-se o grau de pertinência com que as entradas precisas satisfazem os termos lingüísticos associados a estes nós (equivalente ao Módulo de "Fuzzificação"), através da função $O_i^1 = \mu_{A_i}(x)$, que determina o grau de pertinência da entrada x no conjunto *fuzzy* A_i. Usualmente $\mu_{A_i}(x)$ é uma função de pertinência (FP) em forma de sino com máximo igual a 1 e mínimo igual a 0 (Eq. 6).

$$\mu_{A_i}(x) = \frac{1}{1 + \left[\left(\frac{x - c_i}{a_i}\right)^2\right]^{b_i}} \qquad (6)$$

onde:

c_i Determina o centro da FP

a_i É a metade da largura da FP

b_i Controla a inclinação no ponto onde FP = 0,5

Na segunda camada (*layer 2*), cada nó corresponde a uma regra. Cada nó calcula o grau de ativação de cada regra, ou seja, com que grau o conseqüente

da regra está sendo atendido. Os neurônios desta camada executam a operação de interseção (geralmente produto) (Eq. 7).

$$\omega_i = \mu_{A_i}(x) \times \mu_{B_i}(y), i = 1,2 \quad (7)$$

Na terceira camada (*layer 3*) é realizada a normalização, onde cada nó calcula a razão entre o nível de disparo da regra i pela soma dos níveis de disparo de todas as regras (Eq. 8). Esta normalização é importante para completar a etapa de "defuzzificação", efetuada na camada 5.

$$\varpi_i = \frac{\omega_i}{\omega_1 + \omega_2}, i = 1,2 \quad (8)$$

Já na quarta camada (*layer 4*) as saídas do neurônio são calculadas pelo produto entre os níveis de disparo normalizados e o valor do conseqüente da regra em si (Eq. 9).

$$O_i^4 = \varpi_i f_i = \varpi_i (p_i x + q_i y + r_i) \quad (9)$$

Na camada cinco (*layer 5*) calcula-se a saída precisa, somando todos os sinais de entrada do nó, já normalizados (Eq. 10).

$$O_i^5 = \sum_i \varpi_i f_i = \frac{\sum_i \omega_i f_i}{\sum_i \omega_i} \quad (10)$$

Dentre as principais aplicações do modelo ANFIS tem-se a sua utilização em problemas de aproximação de função e identificação de sistemas. Deste modo, este trabalho teve como objetivo comparar o desempenho desse modelo com as redes *Multilayer Perceptrons*, em conjunto com a análise de experimentos.