3
Sistema Neuro-Fuzzy Hierárquico BSP (NFHB)

3.1.
Introdução

Os sistemas neuro-fuzzy descritos na seção anterior possuem limitações por terem uma capacidade reduzida de criação de sua própria estrutura e de receberem um número maior de variáveis de entrada. Os sistemas neuro-fuzzy hierárquicos eliminam essas limitações por serem capazes de criar sua estrutura de acordo com o problema e, conseqüentemente, são capazes de receber um maior número de variáveis de entrada. A criação automática da estrutura através de um algoritmo de aprendizado é uma importante característica quando se pretende desenvolver um sistema completamente automático..

Conforme descrito no capítulo anterior, os modelos neuro-fuzzy realizam particionamentos no espaço de entrada. Esse particionamento depende do formato das funções de pertinência utilizadas e também da relação entre as regras fuzzy que compõem o sistema. Na Figura 7, são mostrados os principais particionamentos realizados pelos sistemas neuro-fuzzy.

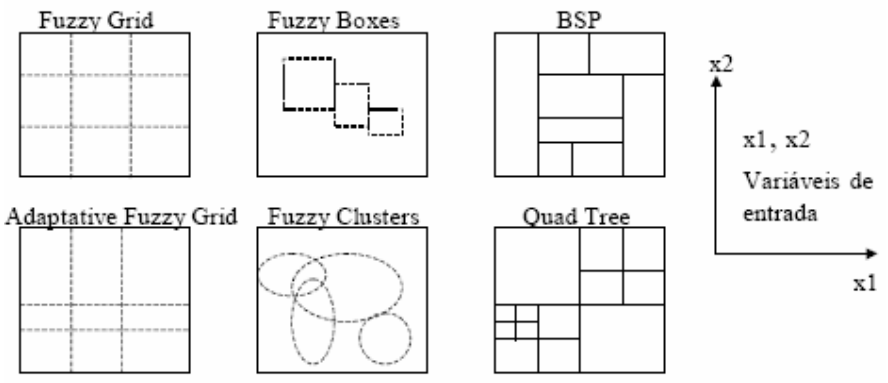

Figura 7: Tipos Comuns de Particionamento dos Sistemas Neuro-Fuzzy

O particionamento *Fuzzy Grid* ocorre quando os antecedentes das regras não são atualizados, ou seja, o particionamento é fixo. No particionamento *Adaptive Fuzzy Grid*, utilizado pelos modelos NEFCLASS e ANFIS, as funções

de pertinência podem ter seus parâmetros atualizados no treinamento e, por isso, ele é considerado adaptativo. O particionamento *Fuzzy-Box* é o utilizado pelo FSOM, como descrito anteriormente, onde os centros das "caixas" são definidos pelos mapas auto-organizáveis. O *Fuzzy-Cluster* é utilizado nas Redes Neurais do tipo Radial Basis Functions (RBFs), ou Funções de Base Radial, onde as partições são definidas por normais multivariadas.

Os particionamentos Binary Space Particioning (BSP) e Quad-Tree são característicos dos sistemas neuro-fuzzy hierárquicos, pois vão particionando o espaço de entrada em partes cada vez menores localmente. Sendo assim, são utilizados algoritmos recursivos para gerar particionamentos destes tipos, produzindo estruturas hierárquicas.

Devido à flexibilidade do particionamento BSP e das vantagens dos sistemas neuro-fuzzy hierárquicos, principalmente pela facilidade de criar sua estrutura automaticamente, o Sistema Neuro-Fuzzy Hierárquico BSP (NFHB) [SOUZ99] foi escolhido como base para a geração de um sistema de mineração de dados completamente automático. Para isso, neste capítulo serão descritas características importantes desse modelo, destacando seus principais parâmetros. Em seguida, o particionamento BSP será descrito em mais detalhes. Ainda neste capítulo, a célula básica NFHB será descrita, assim como a arquitetura do sistema, seus métodos para atualização dos pesos fuzzy (antecedentes e conseqüentes) e as estratégias para seleção das variáveis de entrada. Ao final, será realizado um resumo dos parâmetros do sistema.

3.2. O Particionamento BSP

O particionamento BSP [CHIN89] [CHRY92] realiza partições sucessivas no espaço de entrada das variáveis, sempre dividindo o mesmo em duas novas partições. O particionamento é realizado de forma recursiva até o espaço ser suficientemente dividido para que o sistema correspondente possa armazenar as informações do problema em questão.

Em um problema comum, o espaço de entrada tem como dimensão o número de variáveis de entrada. O particionamento, em cada etapa, é realizado por um hiperplano de dimensão d-1 (onde d é o número de variáveis), sendo esse ortogonal ao eixo da variável que tem seu domínio particionado em dois. O hiperplano corta uma das partições existentes em duas novas partições, de

tamanhos não necessariamente iguais. Várias etapas ocorrem até que o espaço de entrada esteja suficientemente particionado para o problema em questão.

Na Figura 8 é ilustrado um exemplo de particionamento BSP. No exemplo, o espaço de entrada, representado pelas variáveis x_1 e x_2, é bidimensional para facilitar a visualização. Inicialmente, a variável x_2 sofre uma partição em seu domínio criando duas partições no espaço inicial, um superior e outro inferior. Em seguida, a partição de cima é dividida em duas outras, chamadas *A* e *B*, particionando o domínio da variável x_1. A partir deste momento, a partição inferior recebe sucessivos particionamentos verticais e horizontais, criando as partições *C*, *D*, *E* e *F*.

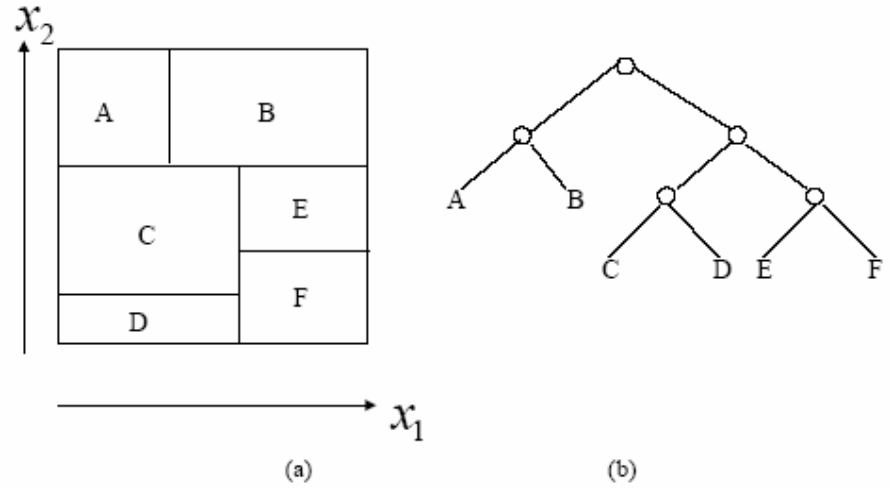

Figura 8: (a) Exemplo de Particionamento BSP; (b) Árvore do Particionamento

A Figura 8(a) ilustra o espaço inicial particionado, enquanto a Figura 8(b) demonstra que o particionamento pode ser representado na forma de uma árvore. Os ramos da árvore representam todos os particionamentos ao longo do processo, e as folhas representam a configuração final de partições, demonstrado nos *labels* da figura.

Como pode ser visto, este particionamento é flexível, podendo assumir as mais diversas configurações o que dá potencialidade aos sistemas neuro-fuzzy que a utilizam.

3.3. A Célula Básica NFHB

A arquitetura do sistema NFHB é composta de células básicas, denominadas “células Neuro-Fuzzy BSP”. No processo de criação da estrutura

hierárquica, o particionamento do espaço de entrada é realizado em conseqüência da criação de novas células.

A Figura 9 mostra um esquema simplificado da célula básica NFHB. Como pode ser visto, a célula possui dois conseqüentes, representados por d_1 e d_2 ($d_i s$), uma saída *y* e um valor de entrada, denominado *x*.

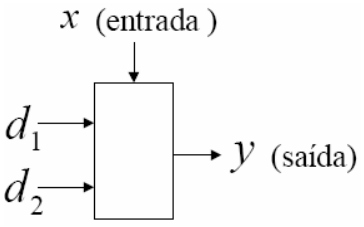

Figura 9: Esquema Simplificado de uma Célula Básica NFHB

Na Figura 10, um esquema detalhado representa o que ocorre no interior da célula. Cada célula possui uma variável lingüística que contém dois conjuntos fuzzy: baixo (ρ) e alto (μ). Os valores $\rho(x)$ e $\mu(x)$ são as avaliações para o valor de entrada *x* das funções de pertinência referentes aos conjuntos fuzzy citados.

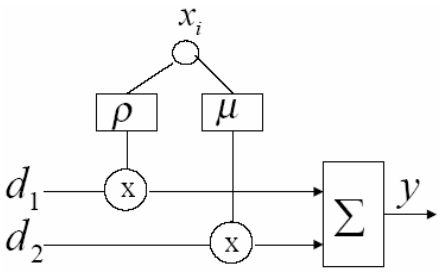

Figura 10: Interior de uma Célula Básica NFHB

Dessa forma, a partir de uma célula NFHB, pode-se extrair duas regras fuzzy, conforme apresentado seguir:

$$\text{Regra } 1: Se\ x \in \rho\ então\ y = d_1$$
$$\text{Regra } 2: Se\ x \in \mu\ então\ y = d_2$$

O formato das funções de pertinência, representado na Figura 11, é sigmoidal (complementares em 1), e suas equações são:

$$\mu(x) = sig(x,a,b) = \frac{1}{1+e^{-a.(x-b)}} \qquad \text{Eq. 8}$$

$$\rho(x) = 1 - \mu(x) \qquad \text{Eq. 9}$$

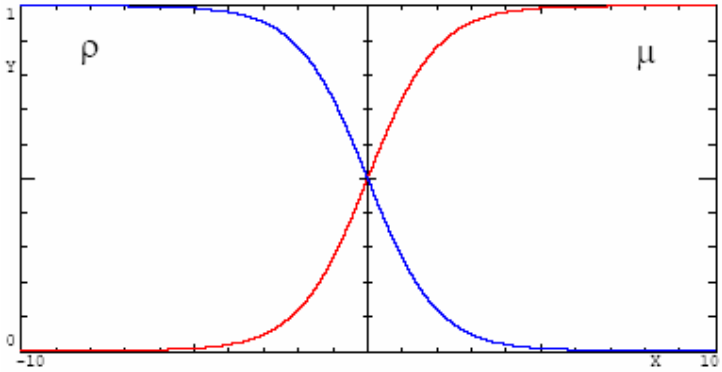

Figura 11: Formato das Funções de Pertinência da Célula NFHB

Na Eq. 8, as constantes *a* e *b* são parâmetros da função sigmóide, onde *a* é a inclinação da reta tangente da função no ponto de inflexão e *b* o valor de *x* para o qual a interseção das duas curvas ocorre (ponto de inflexão).

Sendo assim, conclui-se que, para todo valor de *x*:

$$\rho(x) + \mu(x) = 1 \qquad \text{Eq. 10}$$

Os valores de $\rho(x)$ e $\mu(x)$ representam os antecedentes das regras que irão indicar o quanto os conseqüentes afetarão a saída *y*.

Os conseqüentes $d_i s$ podem ser de três tipos:

- *Singleton (Sugeno de ordem zero)*: um valor real;
- *Combinação linear das entradas (Sugeno de 1ª ordem)*: neste caso, o conseqüente deve ser calculado a partir de uma combinação linear dos valores assumidos pelo padrão de entrada sendo avaliado, ou seja:

$$d_i = \sum_{k=0}^{n} w_k . x_k \qquad \text{Eq. 11}$$

onde x_k representa a *k-ésima* entrada do sistema e w_k representa o coeficiente que as multiplicam. *N* é o número total de entradas do sistema.

- *Outra célula NFHB*: é através deste tipo de conseqüente que é possível montar uma estrutura hierárquica, onde o espaço de entrada pode ser sucessivamente particionado.

A recursividade aparece quando o conseqüente de uma célula é outra célula NFHB, pois é a partir desta célula filha que novas regras podem ser encadeadas, formando uma hierarquia.

A partir das duas regras armazenadas na célula, pode-se calcular a saída *crisp*, ou defuzzificada da mesma. O método de defuzzificação utilizado é a média ponderada, ou seja:

$$y = \frac{\rho(x).d_1 + \mu(x).d_2}{\rho(x) + \mu(x)} \qquad \text{Eq. 12}$$

Essa expressão pode ser simplificada a partir da Eq. 10, aproveitando que as funções de pertinência são complementares em 1, logo:

$$y = \rho(x).d_1 + \mu(x).d_2 \qquad \text{Eq. 13}$$

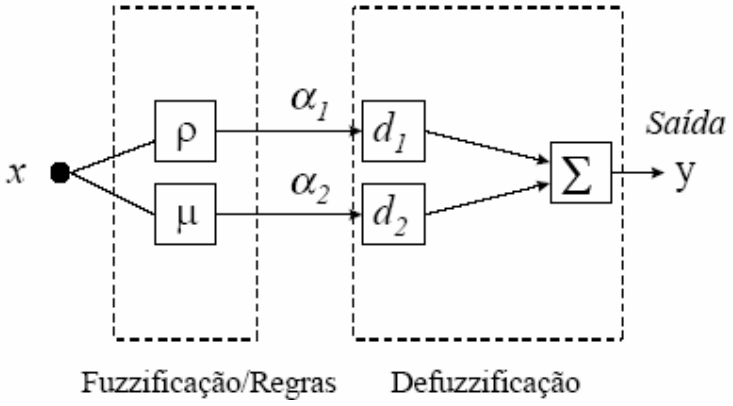

Figura 12: Célula NFHB na forma de uma Rede Neuro-Fuzzy

Desse modo, pode-se reescrever a Eq. 12 utilizando esses valores, da seguinte forma:

$$y = \alpha_1.d_1 + \alpha_2.d_2 \qquad \text{Eq. 14}$$

Como foi visto no item anterior, a estrutura hierárquica binária pode ser visualizada como uma árvore. No NFHB, os nós intermediários representam as células e as folhas são conseqüentes Sugeno (*singletons* ou combinações lineares das entradas) das regras. Como a estrutura é hierárquica, as saídas das células de hierarquia inferior correspondem a conseqüentes de células em níveis mais altos. Este assunto será detalhado na seção a seguir.

3.4. A Arquitetura NFHB

Um exemplo de uma estrutura NFHB pode ser visualizado na Figura 13. Nesse exemplo percebe-se a estrutura hierárquica, onde saídas de células de níveis inferiores são conseqüentes de células de níveis superiores. A saída da célula localizada na raiz representa a saída do sistema como um todo.

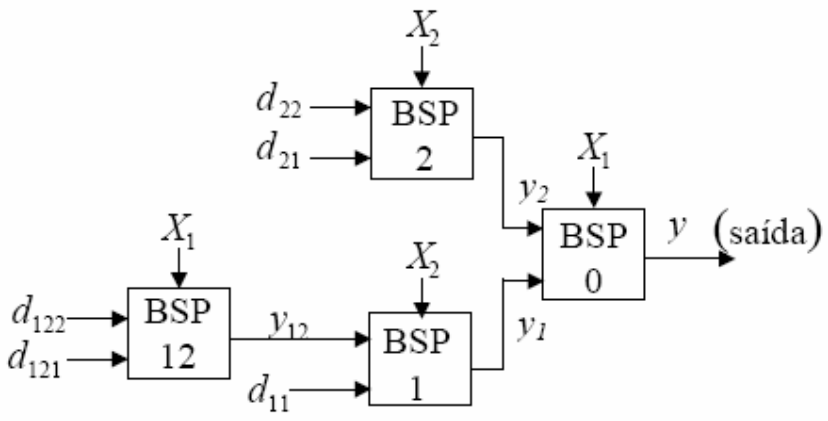

Figura 13: Exemplo de arquitetura NFHB

Na Figura 14, o particionamento do espaço de entrada correspondente ao exemplo da Figura 13 é ilustrado. Cada partição que não é dividida ao final do treinamento é chamada de bi-partição. Nota-se que cada célula da estrutura, ao ser inserida, divide uma partição existente em duas novas, sendo que no final do processo existem tantas bi-partições quanto $d_i s$ no sistema.

No exemplo, a célula "BSP 0" é a raiz e representa todo o domínio da variável x_1, sua entrada. Essa célula, através de sua função de pertinência, mais especificamente no ponto de valor *b* (parâmetro da sigmóide), divide o domínio da variável x_1 em duas novas partições, identificadas por **1** e **2**.

No segundo nível da hierarquia aparecem duas células, filhas da raiz, que dividem seus espaços em mais duas partições cada uma. Ambas recebem como

entrada a variável x_2, porém uma atua na partição **1** e outra, na partição **2**. As novas partições são denominadas **11**, **12** (partição **1**), **21** e **22** (partição **2**). No exemplo ainda é realizado um outro particionamento pela célula "BSP 12", relativo a variável x_1.

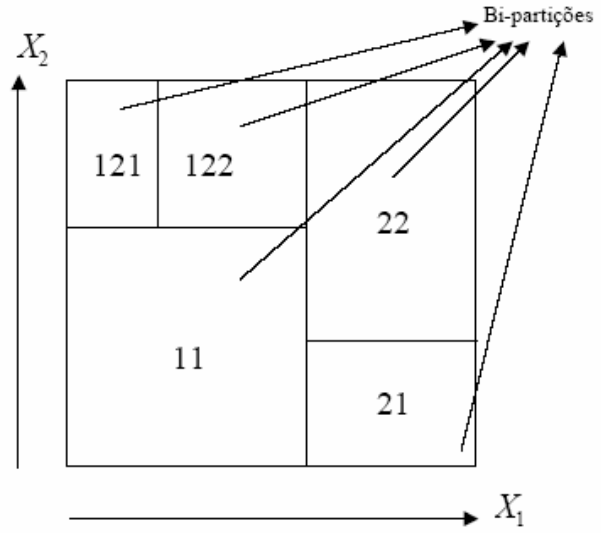

Figura 14: Particionamento Correspondente a Estrutura do Exemplo

Sendo assim, é possível construir uma árvore de particionamentos, ilustrada na Figura 15. Os nós da árvore representados por círculos são as células, portanto nós intermediários. Já as folhas, representadas por quadrados, são as bi-partições, ou seja, os $d_i s$.

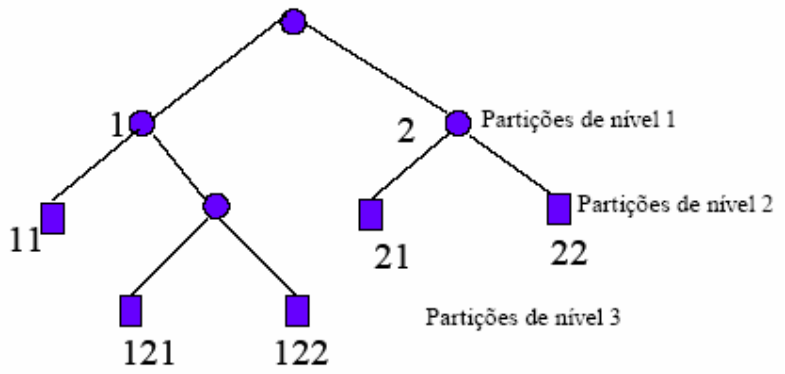

Figura 15: Árvore de Particionamentos

A partir da forma como as regras fuzzy são armazenadas nas células e da estrutura demonstrada no exemplo, pode-se extrair o conjunto de regras gerado. As regras extraídas são apresentadas a seguir.

$$
\begin{aligned}
&Se \quad x_1 \in \rho_0 \quad então \\
&\qquad Se \quad x_2 \in \rho_1 \quad então \quad y = d_{11} \\
&\qquad Se \quad x_2 \in \mu_1 \quad então \\
&\qquad\qquad Se \quad x_1 \in \rho_{12} \quad então \quad y = d_{121} \\
&\qquad\qquad Se \quad x_1 \in \mu_{12} \quad então \quad y = d_{122} \\
&Se \quad x_1 \in \mu_0 \quad então \\
&\qquad Se \quad x_2 \in \rho_2 \quad então \quad y = d_{21} \\
&\qquad Se \quad x_2 \in \mu_2 \quad então \quad y = d_{22}
\end{aligned}
$$

Onde:

- ρ_0 e μ_0 correspondem às funções de pertinência da célula "BSP 0" (partição de nível 0);
- ρ_1 e μ_1 correspondem às funções de pertinência da subdivisão da partição **1**, na célula "BSP 1";
- ρ_2 e μ_2 correspondem às funções de pertinência da subdivisão da partição **2**, na célula "BSP 2";
- ρ_{12} e μ_{12} correspondem às funções de pertinência da subdivisão da partição **12**, na célula "BSP 12";

Nota-se pela própria estrutura de regras a arquitetura hierárquica refletida nas mesmas. Ao se avaliar um novo padrão, percorre-se as regras extraindo os valores de *y* para cada uma delas, recolocando-os como conseqüentes das células mais próximas a raiz na hierarquia, até que se obtenha a saída da célula raiz. Obviamente o *y* correspondente à bi-partição onde se encontra o padrão será o que mais contribuirá para a saída final da estrutura.

Percebe-se que, neste modelo, a mesma variável é utilizada como entrada para todas as células de um mesmo nível. A ordem com que as variáveis devem ser apresentadas deve ser definida *a priori*, normalmente por algum algoritmo de seleção de variáveis. Os algoritmos utilizados pelo sistema serão descritos na seção 3.7.

3.5. O Algoritmo de Aprendizado

Para se criar uma arquitetura como a do exemplo do item anterior, é necessário realizar um treinamento supervisionado com dados do problema

estudado. É a partir deste treinamento que novas células são criadas, os pesos fuzzy são ajustados e, conseqüentemente, a estrutura é montada.

Os pesos fuzzy no caso do sistema NFHB são os parâmetros *a* e *b* que definem o formato das funções de pertinência (sigmóides) e o valores dos $d_i s$ que correspondem aos valores dos conseqüentes.

O algoritmo de treinamento do sistema NFHB é mostrado na Figura 16.

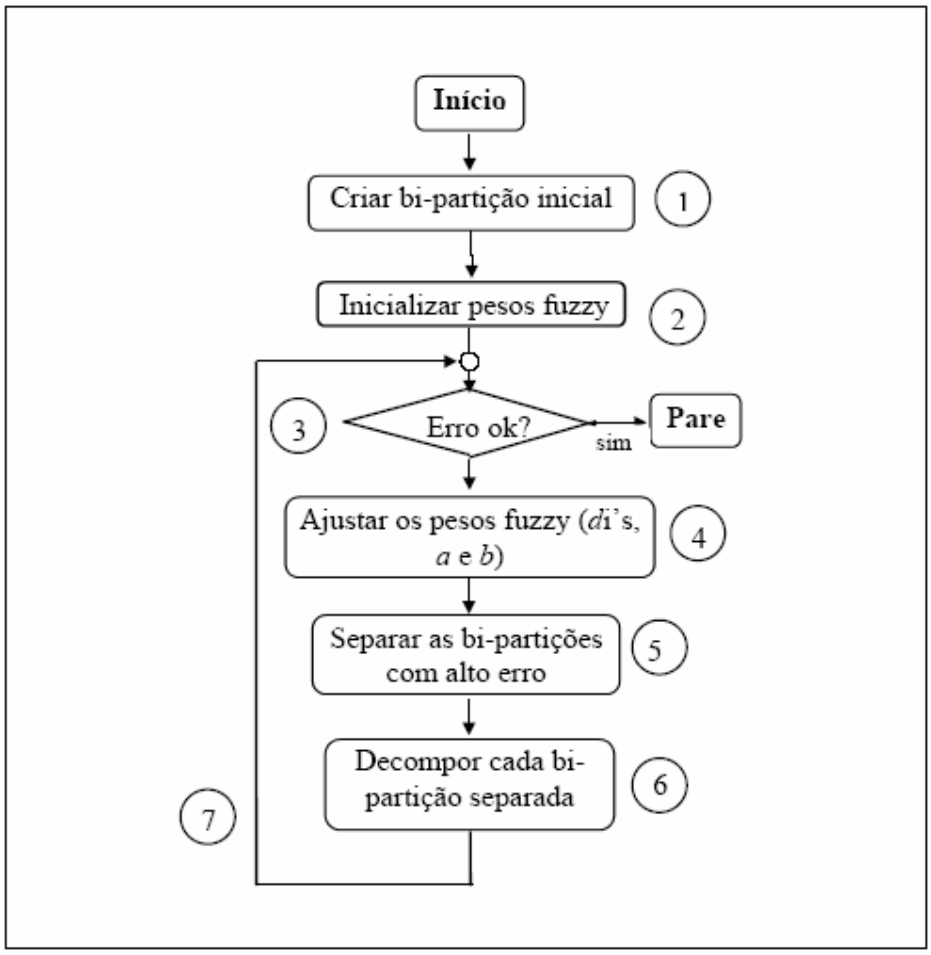

Figura 16: Algoritmo de Aprendizado do Sistema NFHB

O algoritmo é composto de 7 etapas conforme a figura demonstra. Inicialmente, é criada a partição inicial que corresponde à raiz da estrutura.

Em seguida os pesos fuzzy são inicializados. Os parâmetros *a* e *b* da função de pertinência são inicializados da seguinte forma.

$$a = \frac{a^*}{I} \qquad \text{Eq. 15}$$

$$b = (LimS + LimI)/2 \qquad \text{Eq. 16}$$

onde:

- *LimS* é o limite superior do domínio da variável *X;*
- *LimI* é o limite inferior do domínio da variável *X;*
- a^* é um parâmetro do sistema que define este valor;
- $I = LimS - LimI$

A partir deste ponto, inicia-se um *loop*. O critério de parada desse *loop* é composto por duas possibilidades de parada: número máximo de ciclos ou por validação cruzada.

O número máximo de ciclos de treinamento é estipulado inicialmente e quando esse valor é atingido o algoritmo pára, de modo a evitar que a estrutura fique possivelmente treinando indefinidamente.

O segundo critério segue o princípio da validação cruzada [MITC97]. Quando o sistema vai ser treinado, os dados disponíveis são separados em dois conjuntos: um de treinamento e outro de validação. O conjunto de treinamento é utilizado para atualizar os pesos fuzzy e para qualquer outro cálculo no processo de criação da estrutura. O conjunto de validação é utilizado para calcular o erro médio quadrático (ERMS) da estrutura, dado pela equação 17, após um ciclo de treinamento. Esse erro calculado é chamado de erro de validação justamente porque esses dados não foram utilizados para atualizar a estrutura nem seus parâmetros, porém estão sendo utilizados para validá-la.

$$E_{rms} = \sqrt{\frac{1}{L}\sum_{n=1}^{L}(y_n - y_n^d)^2} \qquad \text{Eq. 17}$$

Onde:

- *L* é o número de padrões levados em consideração no cálculo;
- *n* é o índice de um padrão no cálculo;
- y_n é a saída calculada pelo sistema NFHB para o padrão de índice *n*;
- y_n^d é a saída desejada para o padrão de índice *n*.

Conforme mostrado na Figura 17, o erro de treinamento, calculado a partir do conjunto de dados correspondente, é praticamente sempre decrescente. Já o

erro de validação sofre uma oscilação inicial e começa a decrescer, sendo que em um dado momento retoma um crescimento. Esse ponto onde a curva do erro de validação atinge o seu valor mínimo é o ponto ótimo de parada.

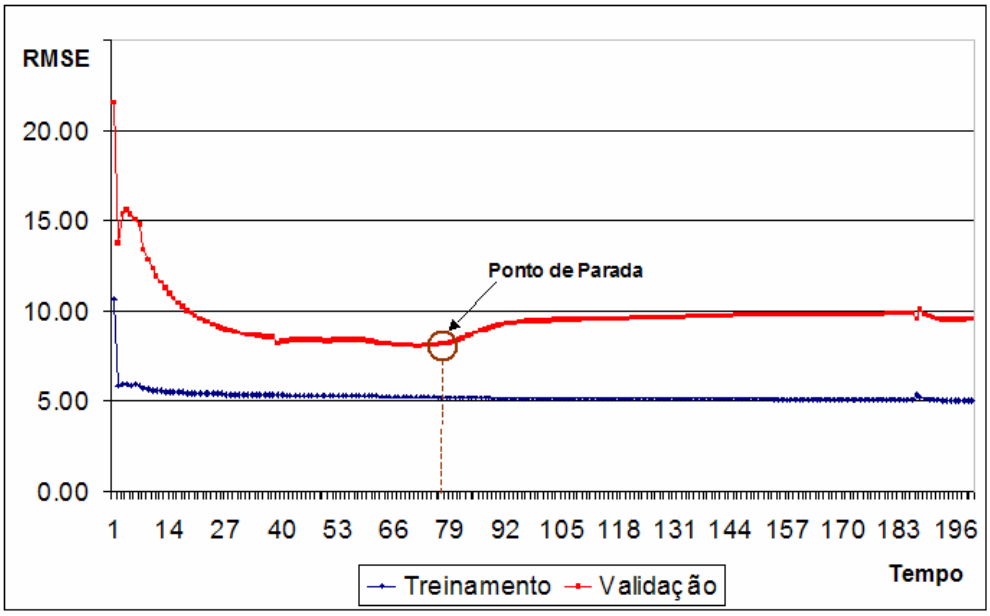

Figura 17: Gráfico de Exemplo de Erros de Treinamento e Validação

O erro de validação retoma um crescimento porque a estrutura cresceu demasiadamente, perdendo em generalização e ficando muito específica para os dados de treinamento, gerando *overfitting*. A Figura 17 apresenta um exemplo de gráfico onde os valores dos erros de treinamento e validação são plotados com relação ao número de ciclos de treinamento. Quando o erro de validação do sistema cresce continuamente ao longo de um número estipulado de ciclos de treinamento, o treinamento também é interrompido..

Cada iteração do *loop* de treinamento inicia ajustando os pesos fuzzy. Os pesos dos antecedentes são representados pelos parâmetros *a* e *b* das funções sigmóides, atualizadas através do método RProp, descrito à frente. Os pesos fuzzy dos conseqüentes são os $d_i s$, atualizados por um método iterativo para o cálculo dos mínimos quadrados ordinários, chamado Método de *Gauss-Seidel*, também explicado mais adiante.

O passo 6 indicado depende de um parâmetro do sistema. Esse parâmetro define se são considerados os erros das bi-partições no momento em que estas são separadas para particionamento. Se o parâmetro definir que não devem ser usados os erros, todas as bi-partições da estrutura são selecionadas. Caso o parâmetro defina por utilizar os erros parciais das bi-partições, esse é

comparado com o valor do ERMS de treinamento (equação 17), e se estiver acima desse valor, a bi-partição é selecionada. O valor do erro parcial de uma bi-partição ***i*** é dado pela equação 18.

$$E^i_{rms} = \sqrt{\frac{\sum_{n=1}^{L} \Pi^i_n (y_n - y^d_n)^2}{\sum_{n=1}^{L} \Pi^i_n}} \qquad \text{Eq. 18}$$

Onde:

- *L* é o número de padrões levados em consideração no cálculo;
- *n* é o índice de um padrão no cálculo;
- y_n é a saída calculada pelo sistema NFHB para o padrão de índice *n*;
- y^d_n é a saída desejada para o padrão de índice *n*;
- *i* é a bi-partição para a qual o erro está sendo calculado;
- Π^i_n é o fator pi-alpha para o padrão *n*, na bi-partição *i*. O fator pi-alpha é a multiplicação do nível de disparo de todas as regras da hierarquia, a partir da raiz, que são acionadas até chegar na bi-partição sendo analisada.

Após a seleção das candidatas ao particionamento, um teste é realizado. Existe um parâmetro do sistema chamado taxa de decomposição, identificado por δ. O algoritmo define que caso o número de padrões incidentes na bi-partição em questão dividido pelo número total de padrões de treinamento seja menor que a taxa de decomposição, a bi-partição não deve ser particionada. Assim, esse parâmetro evita que a estrutura NFHB cresça demasiadamente em pontos que possuem poucos padrões incidentes.

Para as bi-partições que passam pela taxa de decomposição, a estratégia de particionamento é estudada. Um parâmetro do sistema define se os conseqüentes do tipo Sugeno de 1ª ordem (combinação linear das entradas), devem ser utilizados ou não. Caso o parâmetro defina que sim, um outro parâmetro deve ser analisado: o nível em que a combinação linear deve ser utilizada. Este parâmetro define o nível, a partir do qual este tipo de conseqüente não deve ser mais usado. Sendo assim, caso esteja sendo utilizada a combinação linear e a bi-partição a ser particionada esteja em um nível permitido, o particionamento segue a ordem definida na Figura 18.

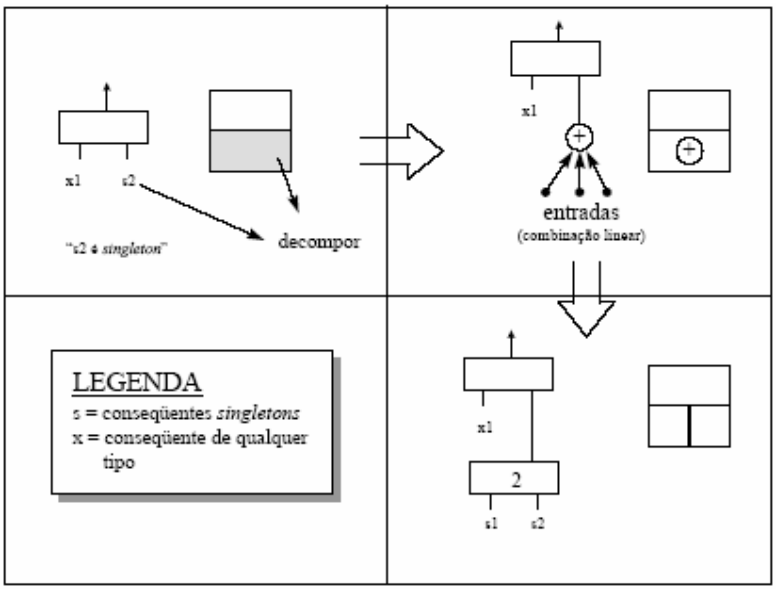

Figura 18: Mecanismo de particionamento com uso de combinação linear das entradas

Como pode ser visto na figura, se o conseqüente for um s*ingleton*, ele transforma-se em uma combinação linear das entradas e, se for combinação linear, torna-se a saída de uma nova célula. Caso não esteja sendo utilizada a combinação linear ou o nível da bi-partição seja superior ao nível especificado pelo parâmetro, conseqüentes do tipo *Singleton* geram uma nova célula.

No momento em que novas células são criadas, essas devem ser inicializadas, com sua saída ocupando o lugar do conseqüente d_i correspondente. A inicialização dos pesos fuzzy de uma nova célula segue o mesmo processo descrito pelas equações 11 e 12, porém os valores de *LimS* e *LimI* são relativos ao subespaço onde se encontram.

Na próxima seção serão descritos os algoritmos utilizados na atualização dos pesos fuzzy, referente ao passo 4 do algoritmo de aprendizado descrito, discriminando, também, seus parâmetros específicos.

3.6. Técnicas para Atualização dos Pesos Fuzzy

À medida que a estrutura NFHB vai evoluindo ao longo do treinamento, novas células vão surgindo e com elas novos pesos fuzzy. Esses pesos devem ser atualizados a cada iteração para que a estrutura se ajuste ao problema em questão.

Os pesos fuzzy se dividem em dois tipos: os que contribuem linearmente para a saída do sistema e os que contribuem não-linearmente para a saída.

O primeiro tipo é representado pelos $d_i s$, ou seja, pelos valores dos conseqüentes. Em problemas lineares normalmente são utilizados métodos de aproximação por MQO e, neste caso, está sendo utilizado o método de Gauss-Seidel, descrito a seguir.

Os parâmetros *a* e *b* das funções de pertinência representam a parte não linear dos pesos fuzzy e, conseqüentemente, não podem ser atualizados por MQO. Dessa forma, foi utilizado um método baseado em *gradient descent*, o RProp [RUSS99].

A seguir, os métodos de Gauss-Seidel e RProp serão descritos, enfatizando seus parâmetros de configuração.

3.6.1. Método de Gauss-Seidel

Como descrito nos itens anteriores, os conseqüentes do sistema NFHB, quando não são representados pela saída de uma célula, são do tipo *Singleton* ou Sugeno de 1ª Ordem. No caso do *Singleton*, apenas um valor é utilizado como conseqüente para cada bi-partição. Já no caso de combinação linear, existem vários valores que são multiplicados pelas entradas gerando um valor final para o conseqüente. Em ambos os casos existe uma relação linear da entrada com a saída, já que a defuzzificação também apresenta uma relação linear. Dessa forma, para esses parâmetros pode-se montar um sistema linear, como o que se segue.

$$A.w = b \qquad \text{Eq. 19}$$

onde *A* é uma matriz gerada a partir dos valores dos atributos preditivos dos padrões de treinamento, *b* é um vetor com as saídas e w é o vetor de incógnitas, neste caso os pesos fuzzy, que se deseja descobrir.

A solução desta equação linear por mínimos quadrados diz que se quer encontrar w de forma a minimizar o quadrado $\|b - A.w\|^2$. Em casos em que a matriz *A* não é quadrada, utiliza-se a pseudo-inversa da matriz *A*, para encontrar a melhor solução de w, denominada w^*.

$$w^* = (A^T A)^{-1} A^T b \quad \text{Eq. 20}$$

Onde A^T é a transposta da matriz *A* e $(A^T A)^{-1} A^T$ é a pseudo-inversa da matriz *A*, caso $A^T A$ seja não-singular.

Essa é uma solução analítica para o problema em questão, porém não se pode confiar na não-singularidade da matriz $A^T A$. Outra alternativa é a solução iterativa ou através de um método de gradiente descendente do seguinte sistema de equações lineares.

$$A^T A w^* = A^T b \quad \text{Eq. 21}$$

O método de Jacobi é um método iterativo que resolve o problema. Supondo que a equação 21 pode ser escrita na forma como está escrita a equação 22, pode-se definir que a *i-ésima* equação do sistema de *n* equações é a seguinte.

$$\sum_{j=1}^{n} a_{i,j}.w_j = b_i \quad \text{Eq. 22}$$

A partir de manipulação da equação 22 isola-se o peso w_i, que pode ser aproximado por:

$$w_i = (b_i - \sum_{i \neq j} a_{ij} w_j) \Big/ a_{i,i} \quad \text{Eq. 23}$$

De maneira natural, a partir da equação 23, pode-se gerar o método iterativo, como se segue.

$$w_i^k = (b_i - \sum_{i \neq j} a_{ij} w_i^{k-1}) \Big/ a_{i,i} \quad \text{Eq. 24}$$

Como pode ser visto na equação 24 do método de Jacobi, os valores de w utilizados são sempre da iteração anterior (*k*-1). O método de Gauss-Seidel difere do método de Jacobi nesse aspecto. Esse método utiliza os valores de w_i já atualizados na iteração corrente, modificando o cálculo como a seguir:

$$w_i^k = (b_i - \sum_{j<i} a_{ij} w_i^k - \sum_{j>i} a_{ij} w_i^{k-1}) \Big/ a_{i,i} \quad \text{Eq. 25}$$

Dessa forma, o método converge mais rapidamente. O método de Gauss-Seidel contribui com mais um parâmetro para o sistema, o número de ciclos que o mesmo irá rodar. Como método iterativo, esse é executado em *loop* até que um critério de parada seja alcançado. Nesse caso, o critério de parada é o número máximo de iterações que pára o método quando alcançado.

Mais detalhes sobre esse método podem ser encontradas em [SOUZ99].

3.6.2. Método Resilient Back Propagation (RProp)

Os pesos fuzzy referentes aos parâmetros que definem o formato (inclinação e transição fuzzy) das funções de pertinência contribuem de maneira não-linear para a saída. Sendo assim, não se pode utilizar o método dos mínimos quadrados como foi feito para os conseqüentes, sendo que métodos baseados em *gradient descent* são mais indicados para atualização.

Este método tem como princípio básico a minimização do erro quadrático de um peso, dado pela equação 26.

$$\varepsilon = (y - y^d)^2 \quad \text{Eq. 26}$$

onde:

- y é a saída do sistema;
- y^d é a saída desejada;
- ε é o erro quadrático a ser minimizado.

O método atualiza o peso utilizando o gradiente do erro (ε) em relação ao peso em questão. A equação a seguir descreve um exemplo desse processo, para um parâmetro "a".

$$a^{k+1} = a^k - \eta \frac{\partial \varepsilon}{\partial a} \quad \text{Eq. 27}$$

onde:

- a^{k+1} é o novo valor do peso sendo atualizado;
- a^{k} é o valor do peso no passo anterior;
- η é a taxa de aprendizado;
- $\frac{\partial \varepsilon}{\partial a}$ é o gradiente do erro em relação ao peso a.

Apesar de relativamente simples, esse método não foi utilizado. O método de RProp foi escolhido por ser muito mais eficiente quando se trata de rapidez de convergência.

O método apresentado utiliza apenas uma taxa de aprendizado (η) para todos os parâmetros, o que o torna lento. O RProp surgiu justamente para ser um método que, apesar de consumir mais recursos computacionais, converge muito mais rapidamente.

O RProp define uma taxa de aprendizado própria para cada parâmetro sendo atualizado, além do mesmo ter seu valor modificado a cada iteração. As equações que definem a forma como os pesos são atualizados são mostradas a seguir.

$$W^{k+1} = W^{k} - \Delta W, \quad se \quad \frac{\partial \varepsilon}{\partial W} > 0 \qquad \text{Eq. 28}$$

$$W^{k+1} = W^{k} + \Delta W, \quad se \quad \frac{\partial \varepsilon}{\partial W} < 0 \qquad \text{Eq. 29}$$

$$W^{k+1} = W^{k}, \quad se \quad \frac{\partial \varepsilon}{\partial W} = 0 \qquad \text{Eq. 30}$$

Podemos observar dessas equações que:

- Se o gradiente do erro estiver crescendo em relação ao peso, deve-se subtrair o valor de ΔW;
- Se o gradiente do erro estiver decrescendo em relação ao peso, deve-se somar o valor de ΔW;
- Se o gradiente do erro for zero, deve-se manter o valor de W inalterado (ponto de erro mínimo);

Com essa política de atualização de parâmetros, é possível obter uma convergência muito mais rápida. Os valores de ΔW também devem ser

alterados ao longo das iterações. Essa alteração segue a política definida pelas equações a seguir.

$$\Delta W^{k+1} = \lambda^{+} * \Delta W^{k}, \quad se \quad \frac{\partial \varepsilon}{\partial W}^{k} * \frac{\partial \varepsilon}{\partial W}^{k+1} > 0 \qquad \text{Eq. 31}$$

$$\Delta W^{k+1} = \lambda^{-} * \Delta W^{k}, \quad se \quad \frac{\partial \varepsilon}{\partial W}^{k} * \frac{\partial \varepsilon}{\partial W}^{k+1} < 0 \qquad \text{Eq. 32}$$

$$\Delta W^{k+1} = \Delta W^{k}, \quad se \quad \frac{\partial \varepsilon}{\partial W}^{k} * \frac{\partial \varepsilon}{\partial W}^{k+1} = 0 \qquad \text{Eq. 33}$$

Onde:

- λ^{+} é o fator de dilatação do ΔW ;
- λ^{-} é o fator de contração do ΔW .

A partir dessas equações observa-se que, se o gradiente do erro em relação ao peso mantém o sinal ao longo de duas iterações, a variação no peso tem o seu valor aumentado através da multiplicação por um fator de dilatação com valor maior que 1. Entretanto, se o valor do gradiente muda de sinal, significa que o peso está oscilando em torno do erro mínimo e, sendo assim, o valor de ΔW é contraído por um fator menor que 1.

Os valores dos fatores de dilatação e contração λ^{+} e λ^{-} são próximos de 1,2 e 0,5, respectivamente e são parâmetros de entrada para o sistema.

Da mesma forma que o Método de Gauss-Seidel descrito anteriormente, o RProp é um método iterativo e, dessa forma, contribui com mais um parâmetro para o sistema, o número de ciclos que o algoritmo deve rodar.

O algoritmo RProp ainda contribui com mais um parâmetro para o sistema como um todo, o *step*. O *step* é um valor, definido *a priori*, utilizado para inicializar e determinar os valores máximos de ΔW, sendo assim, pode influenciar bastante no algoritmo.

No caso do sistema NFHB, os pesos sendo atualizados são os valores de **a** e **b** das funções de pertinência. Os valores máximos de Δa e Δb são determinados da seguinte forma.

$$\Delta a^{\max} = a/(4 * step) \qquad \text{Eq. 34}$$

$$\Delta b^{\max} = (LimS - LimI)/(4 * step) \qquad \text{Eq. 35}$$

onde:

- $\Delta a^{\max}$ é o valor máximo que o Δa pode assumir;
- a é o valor corrente do peso na célula em questão;
- $\Delta b^{\max}$ é o valor máximo que o Δb pode assumir;
- $LimS$ é o limite superior do domínio da variável de entrada, na bi-partição em questão;
- $LimI$ é o limite inferior do domínio da variável de entrada, na bi-partição em questão.

O valor ¼ multiplicando as expressões foi obtido heuristicamente. A partir das equações 34 e 35, pode-se determinar as equações que descrevem o valor inicial de Δa e Δb, como a seguir.

$$\Delta a^{ini} = \Delta a^{\max} / (4 * step) \quad \text{Eq. 36}$$

$$\Delta b^{ini} = \Delta b^{\max} / (4 * step) \quad \text{Eq. 37}$$

Onde:

- Δa^{ini} é o valor inicial de Δa;
- Δb^{ini} é o valor inicial de Δb.

Mais uma vez o fator ¼ foi obtido heuristicamente e, como pode ser visto, o parâmetro *step* é decisivo na determinação dos passos iniciais do algoritmo na atualização dos pesos.

3.7. Estratégias para Seleção de Variáveis de Entrada

Um importante parâmetro do sistema NFHB determina qual das estratégias de seleção de variáveis será utilizada para um determinado conjunto de dados sendo analisado. Esse parâmetro é do tipo inteiro, que identifica um dos métodos sendo utilizados.

O método de seleção de variáveis irá determinar a ordem em que as variáveis serão apresentadas à estrutura NFHB sendo construída. Como foi visto na seção 3.4, um determinado nível da estrutura, incluindo todas as células que se encontram nesse nível, recebe a mesma variável de entrada. Dessa forma, a

ordem das variáveis perante os níveis é fundamental para os resultados do sistema, os quais podem ser completamente diferentes a partir de uma pequena alteração.

Desse modo, é de suma importância para o sistema a utilização de bons métodos para seleção da ordem das variáveis. Esta seção descreve brevemente os três métodos utilizados neste trabalho: o Método da Efetividade de Uma Entrada Singular, ou *Single Input Effectiveness* (SIE) [CAO 97]; o Método do Estimador do Mínimo Quadrado ou *Least Square Estimator* (LSE) [CHUN00] e o método do modelo adaptado Anfis [JANG93] [JANG94].

Os dois primeiros são chamados métodos livres do modelo ou *model free* [CONT02], pois não necessitam de um modelo de mineração para determinar a ordem das variáveis. Os métodos do tipo *model free* são de especial interesse, pois geralmente são mais eficientes computacionalmente, além do fato de, em sendo independentes do modelo, permitirem uma maior flexibilidade desse. Já o modelo adaptado Anfis utiliza um mini modelo Anfis para determinar a ordem das variáveis. Um maior detalhamento desses métodos será realizado nos itens seguintes.

3.7.1. Método SIE

Este método baseia-se no cálculo do grau de efetividade das variáveis de entrada na saída, ou SIE.

o problema em questão pode ser descrito como um sistema linear, na forma $y = Gu$, de *m* saídas e *r* entradas. O princípio deste método consiste em descobrir a importância das variáveis a partir da representação do vetor de entradas como a soma de duas projeções ortogonais $u_n + u_c$, onde u_n é a projeção sobre o espaço nulo da matriz de transferência G do sistema e u_c é a projeção sobre o complemento ortogonal do espaço nulo de G.

Dessa forma, a efetividade de cada variável pode ser calculada a partir da norma da projeção u_c sobre o canal *i* relativo a variável em questão, já que a componente *i* da projeção u_n não importa para a saída por pertencer ao espaço nulo como pode ser visto na equação 38.

$$y = Gu = Gu_n + Gu_c = Gu_c \qquad \text{Eq. 38}$$

A partir dos conceitos apresentados, é possível descrever a efetividade do conjunto de entradas η como:

$$\eta = \frac{\|u_c\|}{\|u\|} \quad \text{Eq. 39}$$

Da mesma forma, a não-efetividade do conjunto de entradas ζ pode ser definida como:

$$\zeta = \frac{\|u_n\|}{\|u\|} \quad \text{Eq. 40}$$

Sendo assim, tem-se que $\eta^2 + \zeta^2 = 1$, desde que $\|u\|^2 = \|u_n\|^2 + \|u_c\|^2$. Como pode ser visto em [CONT02], pode-se estender essa relação para as variáveis *i*, obtendo a expressão $\eta_i^2 + \zeta_i^2 = 1$, onde os valores η_i e ζ_i são as efetividade e não-efetividade singulares, respectivamente, para a variável de índice *i*. As variáveis são, assim, ordenadas conforme os seus graus de efetividade, onde a que apresenta maior valor deve ser a primeira a ser utilizada no sistema.

O método SIE ainda é capaz de descartar algumas variáveis do sistema através do seu grau efetividade singular. Se a efetividade singular de uma variável não estiver acima de um limiar (ex.: $(\eta_i \leq 0{,}2)$), denominado limiar SIE, essa deve ser descartada.

Caso esse teste seja feito e variáveis sejam descartadas, o cálculo das efetividades e testes de relevância devem ser refeitos, pois os valores e a ordem das variáveis podem mudar já que uma ou mais colunas da matriz de entrada foram retiradas. Esse processo é realizado até que não seja retirada mais nenhuma variável.

Neste trabalho, o limiar SIE utilizado foi de 0,01. Maiores informações sobre este método podem ser encontradas em [CONT02] e [CAO 97].

3.7.2. Método LSE

O algoritmo do Método LSE tem como objetivo principal descobrir a importância da *i*-ésima variável x_i estimando o *i*-ésimo fator b_i da função F (equação 41) que descreve a variação da saída Δy em relação às variações das variáveis de entrada Δx_i.

$$F = \Delta y = b_1[\Delta x_1] + b_2[\Delta x_2] + ... + b_n[\Delta x_n] \quad \text{Eq. 41}$$

Os valores dos componentes do vetor Δy são obtidos subtraindo-se os valores da variável de saída para todos os padrões da base de dados combinados dois a dois. Da mesma forma, os valores dos vetores Δx_i são obtidos subtraindo-se os valores correspondentes que as variáveis x_i assumem.

Após a realização da estimativa dos valores dos $b_i s$, o fator com valor mais alto corresponderá à variável que mais contribui para a saída, ou seja, a variável de maior importância. Essa deve ser a primeira variável selecionada pelo modelo.

O algoritmo para estimar os valores dos parâmetros b_i é baseado em um estimador por mínimos quadrados, onde o objetivo é minimizar o quadrado $\left\|\Delta x b - \Delta y\right\|^2$. Um melhor detalhamento do algoritmo pode ser encontrado em [CHUN00] e em [CONT02].

3.7.3. Método do Modelo Adaptado Anfis

Diferentemente dos métodos apresentados anteriormente, o método do modelo adaptado Anfis necessita da definição de um outro modelo para determinar a ordem de entrada das variáveis.

O algoritmo cria um sistema Anfis com duas entradas. Utilizando particionamento fixo, o sistema Anfis divide o espaço das duas variáveis em "n" divisões (número arbitrado inicialmente, neste caso 4). Depois de criada a estrutura, testa-se todas as variáveis sendo selecionadas, duas a duas. No teste, a atualização dos conseqüentes é feita por MQO e os erros médios quadráticos de cada par testado são guardados para posterior ordenação crescente.

Após o cálculo de todos os erros, esses são listados de forma crescente. Ao final tem-se a lista de variáveis na ordem em que devem ser apresentadas à estrutura NFHB.

Apesar de ser dependente do modelo e de parecer ineficiente computacionalmente, esse método demonstrou ser relativamente rápido em sua execução.

3.8. Resumo dos Parâmetros do Sistema NFHB

O objetivo principal deste capítulo foi descrever o sistema NFHB e seus principais parâmetros de configuração. A Tabela 1 resume os parâmetros descritos.

No capítulo seguinte será descrito um modelo de otimização criado para descobrir automaticamente os valores dos parâmetros aqui explicitados, de forma a se obter resultados próximos e até mesmo melhores que os obtidos com a participação de especialistas.

Tabela 1: Resumo dos Parâmetros de Entrada do Sistema NFHB

Nome	Tipo	Função	Contexto
a^*	Real	Inicializa o valor do parâmetro *a* referente às funções de pertinência dos antecedentes das células (sigmóide).	Antecedentes
UsarTolerânciaSeparação	Booleano	Indica se o erro relativo das bi-partições serão levados em consideração ou não na separação para particionamento.	Particionamento
δ	Real	Determina o número mínimo de padrões incidentes em qualquer uma das bi-partições na estrutura NFHB gerada.	Particionamento
UsarCombinaçãoLinear	Booleano	Determina se serão utilizados consequentes de Sugeno de ordem 1, ou seja, combinações lineares das entradas.	Conseqüentes
NívelCombinaçãoLinear	Inteiro	Nível a partir do qual não serão mais utilizadas combinações lineares como consequentes (utilizado somente caso o parâmetro anterior seja verdadeiro).	Conseqüentes
numCiclosMQO	Inteiro	Número máximo de ciclos do método de Gauss-Seidel.	Conseqüentes
λ^+	Real	Dilata a variação do peso no algoritmo Rprop.	Antecedentes
λ^-	Real	Contrai a variação do peso no algoritmo Rprop.	Antecedentes
numCiclosRProp	Inteiro	Número máximo de ciclos do algoritmo Rprop.	Antecedentes
step	Inteiro	Influencia na determinação dos valores máximo e inicial da variação dos pesos no algoritmo Rprop.	Antecedentes
MétodoSeleçãoVariáveis	Inteiro	Determina qual dos métodos de seleção de variáveis será utilizado.	Inicialização