Fernando Luis Bordignon

Aprendizado Extremo para Redes Neurais Fuzzy Baseadas em Uninormas

Campinas

2013

Universidade Estadual de Campinas
Faculdade de Engenharia Elétrica e de Computação

Fernando Luis Bordignon

Aprendizado Extremo para Redes Neurais Fuzzy Baseadas em Uninormas

Orientador: Prof. Dr. Fernando Antônio Campos Gomide

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Engenharia Elétrica da Faculdade de Engenharia Elétrica e de Computação da Universidade Estadual de Campinas para obtenção do título de Mestre em Engenharia Elétrica, na Área de Concentração: Engenharia de Computação.

Este exemplar corresponde à versão final da dissertação defendida pelo aluno Fernando Luis Bordignon, e orientada pelo Prof. Dr. Fernando Antônio Campos Gomide

---

Campinas
2013

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

B644a

Bordignon, Fernando Luis
Aprendizado extremo para redes neurais fuzzy baseadas em uninormas / Fernando Luis Bordignon. -- Campinas, SP: [s.n.], 2013.

Orientador: Fernando Antônio Campos Gomide.
Dissertação de Mestrado - Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Redes neurais híbridas. 2. Neurônios . 3. Aprendizagem. 4. Redes neurais (Computação). I. Gomide, Fernando Antônio Campos, 1951-. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

Título em Inglês: Extreme learning for uninorm-based fuzzy neural networks
Palavras-chave em Inglês: Hybrid neural networks, Neurons, Learning, Neural networks (computing)
Área de concentração: Engenharia da Computação
Titulação: Mestre em Engenharia Elétrica
Banca examinadora: Fernando Antônio Campos Gomide, Walmir Matos Caminhas, Akebo Yamakami
Data da defesa: 14-01-2013
Programa de Pós Graduação: Engenharia Elétrica

COMISSÃO JULGADORA - TESE DE MESTRADO

Candidato: Fernando Luis Bordignon

Data da Defesa: 14 de janeiro de 2013

Título da Tese: "Aprendizado Extremo para Redes Neurais Fuzzy Baseadas em Uninormas"

Prof. Dr. Fernando Antônio Campos Gomide (Presidente): _____

Prof. Dr. Walmir Matos Caminhas: _____

Prof. Dr. Akebo Yamakami: _____

Agradecimentos

Agradeço,

Ao Prof. Gomide pela orientação e à toda comunidade da FEEC pelo ótimo ambiente de pesquisa. À minha família pelo apoio e compreensão. À Vivi pelo carinho, amor e compreensão que apesar da distância prevaleceram fortes e constantes. À CAPES e ao CNPq, pelo apoio financeiro para apresentação de trabalhos e bolsa, respectivamente. Aos colegas e especialmente em memória ao companheiro de estudo e de vida Salomão, que as lembranças desta grande pessoa permaneçam na comunidade e com os que conviveram com ele.

Resumo

Sistemas evolutivos são sistemas com alto nível de adaptação capazes de modificar simultaneamente suas estruturas e parâmetros a partir de um fluxo de dados, recursivamente. Aprendizagem a partir de fluxos de dados é um problema contemporâneo e difícil devido à taxa de aumento da dimensão, tamanho e disponibilidade temporal de dados, criando dificuldades para métodos tradicionais de aprendizado. Esta dissertação, além de apresentar uma revisão da literatura de sistemas evolutivos e redes neurais *fuzzy*, aborda uma estrutura e introduz um método de aprendizagem evolutivo para treinar redes neurais híbridas baseadas em uninormas, usando conceitos de aprendizado extremo. Neurônios baseados em uninormas fundamentados nas normas e conormas triangulares, generalizam neurônios *fuzzy*. Uninormas trazem flexibilidade e generalidade a modelos neurais *fuzzy* pois elas podem se comportar como normas triangulares, conormas triangulares, ou de forma intermediária por meio do ajuste de elementos identidade. Este recurso adiciona uma forma de plasticidade em modelos de redes neurais. Um método de agrupamento recursivo para granularizar o espaço de entrada e um esquema baseado no aprendizado extremo compõem um algoritmo para treinar a rede neural. É provado que uma versão estática da rede neural *fuzzy* baseada em uninormas aproxima funções contínuas em domínios compactos, ou seja, é um aproximador universal. Postula-se, e experimentos computacionais endossam, que a rede neural *fuzzy* evolutiva compartilha capacidade de aproximação equivalente ou melhor, em ambientes dinâmicos, do que as suas equivalentes estáticas.

Palavras chave: redes neurais *fuzzy*; neurônios baseados em uninormas; sistemas evolutivos; aprendizado incremental; aprendizado extremo.

Abstract

Evolving systems are highly adaptive systems able to simultaneously modify their structures and parameters from a stream of data, online. Learning from data streams is a contemporary and challenging issue due to the increasing rate of the size and temporal availability of data, turning the application of traditional learning methods limited. This dissertation, in addition to reviewing the literature of evolving systems and neuro fuzzy networks, addresses a structure and introduces an evolving learning approach to train uninorm-based hybrid neural networks using extreme learning concepts. Uninorm-based neurons, rooted in triangular norms and conorms, generalize fuzzy neurons. Uninorms bring flexibility and generality to fuzzy neuron models as they can behave like triangular norms, triangular conorms, or in between by adjusting identity elements. This feature adds a form of plasticity in neural network modeling. An incremental clustering method is used to granulate the input space, and a scheme based on extreme learning is developed to train the neural network. It is proved that a static version of the uninorm-based neuro fuzzy network approximate continuous functions in compact domains, i.e. it is a universal approximator. It is postulated, and computational experiments endorse, that the evolving neuro fuzzy network share equivalent or better approximation capability in dynamic environments than their static counterparts.

Keywords: fuzzy neural networks; unineurons; uninorms; evolving systems; incremental learning; extreme learning.

Lista de Figuras

Lista de Tabelas

Lista de Acrônimos e Notação

RNA	Rede Neural Artificial
ECOS	*Evolving Connectionist Systems* (sistemas evolutivos conexionistas)
RMSE	*Root Mean Squared Error* (raiz quadrada do erro médio quadrático)
RLS	*Recursive Least Squares* (mínimos quadrados recursivo)
ELM	*Extreme Learning Machines* (métodos com aprendizado extremo)
MLP	*Multi Layer Perceptron* (perceptron de múltiplas camadas)
SVM	*Support Vector Machine* (máquina de vetor suporte)
eFS	*evolving Fuzzy Systems* (sistemas evolutivos *fuzzy*)
eTS	*evolving Takagi Sugeno*
ePL	*evolving Participatory Learning*
ANFIS	*Adaptive-Network-Based Fuzzy Inference System*
DENFIS	*Dynamic Evolving Neuro-Fuzzy Inference System*
FuNN	*Fuzzy Neural Network*
EFuNN	*Evolving Fuzzy Neural Network*
FlexFIS	*Flexible Fuzzy Inference System*
FCM	*Fuzzy C-Means*
ECM	*Evolving Clustering Method*
TSK	Modelo *fuzzy* do tipo Takagi-Sugeno-Kang

$\mathbf{A}$	matriz
$\mathbf{x}$	vetor
$\mathbf{A} > 0$	matriz $\mathbf{A}$ simétrica definida positiva
$\mathbf{A} \geq 0$	matriz $\mathbf{A}$ simétrica semi-definida positiva
$\mathbf{A}^T$	transposta da matriz $\mathbf{A}$
$\mathbb{R}$	conjunto dos números reais
$\mathbf{I}$	matriz identidade de dimensão apropriada
$\|\| \cdot \|\|$	norma Euclidiana

Trabalhos Publicados Pelo Autor

Bordignon, F. e Gomide, F. (2012), Extreme learning for evolving hybrid neural networks, Proceedings of the 12th Brazilian Symposium on Neural Networks. SBRN2012, Curitiba, PR, Brasil, pp. 196–201.

Bordignon, F. e Gomide, F. (2012), Capacidade de aproximação de redes neurais nebulosas baseadas em uninormas, Recentes Avanços em Sistemas Fuzzy, Anais do Segundo Congresso Brasileiro de Sistemas Fuzzy IICBSF, Natal, RN, Brasil, pp. 92–101.

Bordignon, F. e Gomide, F. (2012), Uninorm-Based Fuzzy Neural Networks and Approximation, Proceedings of the 2nd World Conference on Soft Computing, pp. 322-328. WConSC2012, Baku, Azerbaijan.

Bordignon, F. e Gomide, F. (2013), Uninorm Based Evolving Neural Networks and its Approximation Capabilities, *Neurocomputing*. **Submetido**

Sumário

Capítulo 1

Introdução

Sistemas *fuzzy* e redes neurais são aproximadores universais. Compartilham a habilidade de trabalhar em ambientes incertos, imprecisos e ruidosos. Ambos aproximam funções utilizando exemplos de entrada e saída. Basicamente sistemas *fuzzy* aproximam funções utilizando regras *fuzzy*, enquanto redes neurais utilizam uma estrutura conexionista e dados com exemplos de entradas e saídas para realizar seu treinamento.

Redes neurais artificiais do tipo MLP (perceptron de múltiplas camadas) são sistemas não lineares com aprendizado, cuja habilidade de generalização e tolerância a ruído emergem a partir de suas estruturas conexionistas e do processamento e representação distribuída (Lin e Lee, 1996, p. 479). Para utilizar uma rede neural é necessário especificar sua estrutura e utilizar um conjunto de dados de treinamento suficientemente representativo para obter os parâmetros da rede, via ciclos repetidos de treinamento. Como as redes têm seu conhecimento distribuído em unidades distintas de processamento, a interpretação direta do conhecimento codificado, em geral, é difícil.

Sistemas *fuzzy*, por outro lado, são modelos granulares que podem processar dados imprecisos. Estes sistemas representam o conhecimento em sua estrutura por meio de variáveis linguísticas e de relações via regras *fuzzy*. O processamento é uma forma de raciocínio aproximado. Como a representação do conhecimento é favorável ao entendimento humano, as relações entre as variáveis, muitas vezes, tornam-se transparentes.

Sistemas evolutivos, no contexto desta dissertação, são aqueles em que tanto sua estrutura quanto parâmetros são determinados simultaneamente e continuamente, sem provocar ruptura do modelo do sistema. Eles se adaptam de maneira recursiva e gradual, ou seja, o aprendizado acontece incrementalmente a medida em que exemplos de treinamento são disponibilizados. Este tipo de sistema permite processar fluxo de dados, proporcionando assim um mecanismo para

processamento *online*, para aplicações em tempo real e para o manuseio de grandes volumes de dados.

Uma classe de sistemas evolutivos foi concebida a partir da combinação de redes neurais com sistemas *fuzzy*. Esta classe é constituída de redes neurais *fuzzy*, isto é, redes neurais dotadas de mecanismos de aprendizagem simultânea da estrutura e parâmetros, com processamento que mimetiza o raciocínio aproximado.

1.1 Motivação e Relevância

Métodos tradicionais de aprendizado estão sendo revistos nos últimos anos, visto que a necessidade da capacidade de aprendizado *online* e o processamento de um volume grande de dados tornou-se importante (Angelov e Zhou, 2006; Leite et al., 2011). Os computadores estão conectados em um nível sem precedentes, gerando enorme quantidade de dados e grande interesse em entender e prever variáveis econômicas, técnicas e sociais.

A combinação de sistemas *fuzzy* com redes neurais visa aproveitar técnicas que são complementares, pois de modo geral redes neurais possuem habilidade de aprendizado e sistemas *fuzzy* possuem interpretabilidade. Espera-se que um sistema híbrido contenha características dos dois métodos. Além disso, essa combinação também facilita a concepção de sistemas evolutivos, pois a estrutura se traduz diretamente em forma de regras que podem ser adicionadas, removidas ou alteradas localmente. Estas regras representam uma combinação de sistemas locais simples que resulta em um modelo altamente não linear (Angelov; Filev e Kasabov, 2010).

Sistemas evolutivos são uma maneira inovadora e alternativa de adaptar, aprender e representar conhecimento sobre ambientes em constante mudança (Angelov e Zhou, 2006). A teoria de conjuntos *fuzzy* provê métodos sólidos e mecanismos eficientes para conceber sistemas evolutivos, como já foi reportado em trabalhos iniciais em modelagem baseada em regras *fuzzy* (Angelov, 1999) e modelos de inferência *fuzzy*-neurais (Kasabov e Song, 2002). Exemplos incluem uma abordagem mais geral chamada de sistemas evolutivos conexionistas, conhecido pela sigla em inglês ECOS (Kasabov, 2007); sistemas de inferência *fuzzy* flexíveis (FLEXFIS) (Lughofer, 2008 b); Takagi-Sugeno evolutivo (eTS) (Angelov e Filev, 2004) e suas versões aperfeiçoadas (Angelov e Filev, 2005; Angelov, 2010). Esforços recentes foram empregados em computação granular e modelagem evolutiva baseada na teoria de conjuntos *fuzzy* (FBeM) (Leite et al., 2011).

A partir da perspectiva de redes neurais, extensões do neurônio clássico utilizando normas e

conormas triangulares (t-normas e s-normas) foram estudadas extensivamente na literatura recente (Pedrycz, 2006; Hell; Costa e Gomide, 2008; Hell et al., 2009; Lemos; Caminhas e Gomide, 2010; Lemos et al., 2011). Em particular, a noção de uninormas foi explorada para generalizar os neurônios clássicos e sugerir neurônios lógicos. Uninormas trazem flexibilidade para o projeto de neurônios e redes neurais, além de naturalmente prover plasticidade neuronal por meio de um parâmetro extra chamado elemento de identidade. O valor do elemento identidade determina se uma uninorma se comporta como uma t-norma, s-norma ou um operador intermediário.

1.2 Objetivo

O objetivo do presente trabalho é propor e analisar métodos de aprendizado para redes neurais *fuzzy* baseadas em uninormas. Como em outros modelos que utilizam uninormas, obtém-se maior flexibilidade, plasticidade e acurácia comparado a propostas semelhantes (Lemos; Caminhas e Gomide, 2010). Para obter um modelo evolutivo *fuzzy*, além de manter a interpretabilidade do modelo, é necessário desenvolver um algoritmo recursivo de aprendizagem para atender os requisitos de processamento *online*, em tempo real e de grande volume de dados.

Para atingir o objetivo principal foram estipulados os seguintes objetivos intermediários: realizar uma revisão bibliográfica da literatura envolvendo sistemas evolutivos e redes neurais *fuzzy*; definir o tipo de estrutura da rede neural baseada em uninormas; desenvolver um algoritmo de treinamento que atenda os requisitos para um sistema evolutivo; verificar a capacidade de aproximação universal de redes baseadas em uninormas; analisar o comportamento da rede quanto a acurácia, tempo de execução e adaptabilidade do modelo.

Para validar o modelo proposto, foram feitos testes com conjunto de dados e problemas que são referência na literatura, além de testes estatísticos e comparações com métodos neurais *fuzzy* e evolutivos mais relevantes na literatura.

1.3 Organização do Texto

Esta dissertação está organizada da seguinte forma. O Capítulo 2 apresenta redes neurais *fuzzy* clássicas encontradas na literatura, define t-normas e s-normas e redes baseadas em neurônios *fuzzy*. O Capítulo 3 define o conceito de sistemas evolutivos, apresenta modelos evolutivos mais importantes e o método de aprendizado extremo.

O Capítulo 4 enfoca a principal contribuição desse trabalho: métodos de aprendizado para

modelos de redes neurais *fuzzy* baseadas em uninormas. Apresenta-se a definição de uninormas, a estrutura da rede, o construtor da uninorma utilizada, os métodos de agrupamento e o procedimento para determinação dos pesos e parâmetros. Mostra-se que a versão estática do modelo é um aproximador universal.

Resultados de experimentos computacionais realizados para analisar o desempenho do método proposto, fazem parte do Capítulo 5, que também descreve os procedimentos adotados para realizar os testes. Métodos evolutivos alternativos, sugeridos na literatura, são utilizados para resolver os mesmos problemas e comparados com os métodos sugeridos nesta dissertação. Conclusões, o resumo das contribuições da dissertação e sugestões de temas para trabalhos futuros constituem o Capítulo 6.

Capítulo 2

Redes Neurais *Fuzzy* Adaptativas

2.1 Introdução

Neste capítulo serão apresentadas redes neurais *fuzzy* predominantes na literatura que fazem uso de neurônios *fuzzy* do tipo *OR* e *AND* e de conceitos da teoria de conjuntos *fuzzy*. É definido o conceito de normas triangulares, neurônios baseados em t-normas e s-normas, e redes que fazem uso destes neurônios.

2.2 Redes Neurais *Fuzzy*

Redes neurais artificiais são modelos que emulam o poder computacional de processamento das redes neurais biológicas, com o objetivo de reproduzir as suas habilidades de adaptação e aprendizado. No entanto, um dos problemas de se usar redes neurais artificiais é que seus modelos, altamente conectados e em forma de caixa preta, dificultam a capacidade de explicar, de maneira compreensível por humanos, como a rede produz seus resultados (Mitra e Hayashi, 2000).

Combinar teoria de conjuntos *fuzzy* com redes neurais visa amenizar esse problema, dotando os sistemas conexionistas de atributos que permitam transparência e raciocínio aproximado. Para realizar essa integração, usualmente é utilizada uma estrutura que contém funções de pertinência como neurônios da camada de entrada e neurônios que desempenham funções de agregação ou neurônios tradicionais nas camadas seguintes. As variações nas conexões, tipos de neurônios utilizados e método de treinamento, definem os diferentes modelos presentes na literatura.

Redes neurais *fuzzy* podem ser do tipo *feedforward* ou recorrentes. O ajuste dos parâmetros

pode ser feito com mínimos quadrados recursivo e estático, ou via *backpropagation*. O treinamento via gradiente pode ser feito utilizando todo o conjunto de dados ou via aproximação local como em Hush e Horne (1993).

A rede neural *fuzzy* proposta em Lee e Lee (1975), utiliza um neurônio que generaliza o modelo clássico de McCulloch-Pitts, onde todas as entradas, saídas e pesos são conjuntos *fuzzy*. O treinamento é feito via generalização *fuzzy* do *backpropagation* (Lin e Lee, 1996, p. 643).

Outros modelos utilizam um tipo de neurônio artificial chamado de OWA (*ordered weighted averaging*), que é basicamente a soma dos produtos dos pesos pelas entradas ordenadas. Esse neurônio foi empregado para realizar as funções básicas da lógica *fuzzy*, como *AND*, *OR* e *NOT*. A partir de funções de pertinência na camada de entrada combinadas com os neurônios OWA, foram concebidas redes que ao final de seu treinamento poderiam ser parcialmente interpretadas.

Seguindo essa linha, Keller; Yager e Tahani (1992) utilizou redes neurais para implementar inferência *fuzzy*, onde certas propriedades como *modus ponens* foram provadas (Lin e Lee, 1996, p. 501). Redes mais flexíveis foram introduzidas para permitir maior capacidade de treinamento, pois as redes de inferência citadas acima utilizavam operadores lógicos fixos. As redes de agregação *fuzzy* (Keller; Krishnapuram e Rhee, 1992) possuem neurônios que podem ser qualquer operador de agregação *fuzzy* parametrizado. O treinamento é realizado com *backpropagation*.

Em Hayashi et al. (1992) foi implementado inferência como na abordagem Takagi-Sugeno, os parâmetros dos consequentes são otimizados utilizando um método de busca por padrões. Takagi et al. (1992) incorpora conhecimento na rede neural para facilitar análise de desempenho. Treinamento de redes neurais também foi empregado para resolver equações relacionais *fuzzy* (Wang, 1993). Treinamento via aprendizado evolucionário é introduzido por Machado e Rocha (1992), com aplicações em recuperação de informação em base de dados de linguagem natural, como arquivos médicos.

A *fuzzy adaptive learning control network* (FALCON) (Lin e Lee, 1991) consiste numa rede multicamada de propagação *feedforward*. Suas entradas e saídas são funções de pertinência, a camada intermediária consiste em neurônios do tipo AND que agregam as ativações da entrada e ativam as funções de pertinência da saída, formando regras. O aprendizado dá-se em duas etapas: a primeira é uma fase auto adaptativa baseada em mapas auto-organizáveis (Kohonen, 1982), que define a estrutura do modelo, a segunda é baseada em um aprendizado supervisionado via *backpropagation*, que ajusta os parâmetros das funções de pertinência e dos pesos entre as camadas (Lin e Lee, 1996, p. 535-536).

Kasabov (1996) propôs o modelo FuNN - *Fuzzy Neural Network* que possui a seguinte estru-

tura: camada de entrada representando as dimensões da entrada e o *bias*; camada dos termos antecedentes com funções de ativação *fuzzy*; camada das regras, onde as conexões representam os antecedentes; camada de ação, que são os consequentes das regras; e agregação geral.

O autor também define que a rede pode trabalhar de modo puramente *fuzzy*, eliminando as camadas das extremidades, 1 e 5, assim se trabalha com entradas e saídas como funções de pertinência. O aprendizado ocorre de forma análoga à redes neurais MLPs comuns, pois dependendo da escolha das funções, podemos obter um modelo bem próximo da MLP, mas com a vantagem de se poder explicar os resultados via regras *fuzzy*.

No modelo *Adaptive-Network-Based Fuzzy Inference System*, abreviado ANFIS (Shing e Jang, 1993), a primeira camada consiste em funções de pertinência. A segunda camada realiza a operação de uma t-norma. Na terceira camada faz-se uma normalização. A última produz a saída da rede como a média ponderada das saídas dos neurônios ativados, similarmente ao modelo Takagi-Sugeno-Kang (TSK) (Lin e Lee, 1996, p. 156).

O aprendizado dos parâmetros se dá em dois passos. O primeiro é um passo direto (*forward*), que fixa os parâmetros dos antecedentes e utiliza mínimos quadrados para estimar os parâmetros das funções dos consequentes. O segundo passo fixa os parâmetros dos consequentes e utiliza o método do gradiente para ajustar os parâmetros dos antecedentes de forma a minimizar o erro.

2.3 Redes Baseadas em Neurônios *Fuzzy*

Na teoria de conjuntos *fuzzy*, normas triangulares oferecem uma classe genérica de operadores de intersecção e união. As t-normas generalizam a operação de intersecção entre conjuntos *fuzzy*. S-norma é uma generalização de união entre conjuntos *fuzzy*.

Formalmente t-normas são operadores binários $t : [0,1] \times [0,1] \to [0,1]$ com as seguintes propriedades:

1. Comutatividade: $a\ t\ b = b\ t\ a$
2. Associatividade: $a\ t\ (b\ t\ c) = (a\ t\ b)\ t\ c$
3. Monotonicidade: se $b \leq c$, então $a\ t\ b \leq a\ t\ c$
4. Fronteira: $a\ t\ 1 = a$; a t 0=0

Escolhas comuns para t-normas são os operadores: $\min(a, b)$; produto ab; e Lukasiewicz $\max(a + b - 1, 0)$. As s-normas são operadores binários $s : [0,1] \times [0,1] \to [0,1]$ que são:

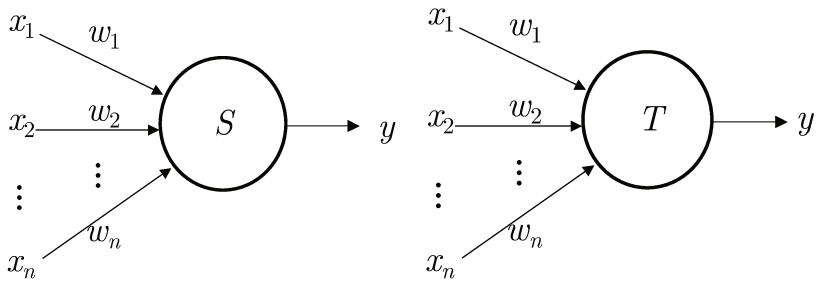
1. Comutativas: $a\ s\ b = b\ s\ a$
2. Associativas: $a\ s\ (b\ s\ c) = (a\ s\ b)\ s\ c$
3. Monotônicas: se $b \leq c,$ então $a\ s\ b \leq a\ s\ c$
4. Limitadas na fronteira: $a\ s\ 1 = 1$; a s 0=a

Exemplos de s-normas incluem os seguintes: $max(a, b)$; soma probabilística $a + b - ab$; e Lukasiewicz $\min(a + b, 1)$. Pedrycz e Rocha (1993) sugerem a utilização de neurônios baseados em normas triangulares para criar redes neurais *fuzzy*. Visando dotar redes neurais com maior interpretabilidade, principalmente, os neurônios OR e AND foram propostos. Os neurônios OR e AND são caracterizados por:

$$y = OR(\mathbf{w}, \mathbf{x}) = \overset{n}{\underset{i=1}{\mathrm{S}}}(w_i\, t\, x_i) \tag{2.1}$$

$$y = AND(\mathbf{w}, \mathbf{x}) = \overset{n}{\underset{i=1}{\mathrm{T}}}(w_i\, s\, x_i) \tag{2.2}$$

A Figura 2.1(a) apresenta o esquema do neurônio OR e a 2.1(b) o do neurônio AND.



(a) OR (b) AND

Figura 2.1: Neurônios *fuzzy* OR e AND.

Além das entradas convencionais, o modelo faz uso da negação da entrada, utilizando uma função de negação da lógica *fuzzy* para realizar esta operação. Esta função não é mostrada na estrutura da rede, pois o modelo simplesmente considera a negação como uma entrada a mais do sistema.

Para realizar o aprendizado de uma rede composta por esses neurônios via gradiente, é necessário derivar as t-normas e s-normas. Se escolhidas funções mais frequentemente utilizadas como min e max para realizar as t-normas e s-normas, gera-se uma descontinuidade em suas derivadas que podem penalizar o treinamento (Pedrycz e Rocha, 1993), como visto em (2.3) e (2.4).

$$\frac{\partial}{\partial x}\min(a,x) = \begin{cases} 1 & \text{se } x < a \\ 0 & \text{se } x > a \end{cases} \tag{2.3}$$

$$\frac{\partial}{\partial x}\max(a,x) = \begin{cases} 1 & \text{se } x > a \\ 0 & \text{se } x < a \end{cases} \tag{2.4}$$

Para atacar o problema, os autores apontam que a utilização de t-normas e s-normas paramétricas suaves, que tendem às padrões *min* e *max*, tem grande diferença conceitual mas apresentam um resultado similar no treinamento (Pedrycz e Rocha, 1993).

Também é discutido o aprendizado da estrutura via expansão e redução, além de ser possível inicializar com conhecimento de especialistas, onde conexões óbvias são inseridas na rede para facilitar a otimização da estrutura. Os autores apontam aplicações para este tipo de rede com bons resultados. São elas: tomada de decisão, problemas de diagnóstico e de controle. A Figura 2.2 exemplifica uma estrutura da rede baseada em neurônios *OR* e *AND*. Este tipo de rede já foi utilizado também para classificação em Caminhas et al. (1999), onde o aprendizado é feito via competição.

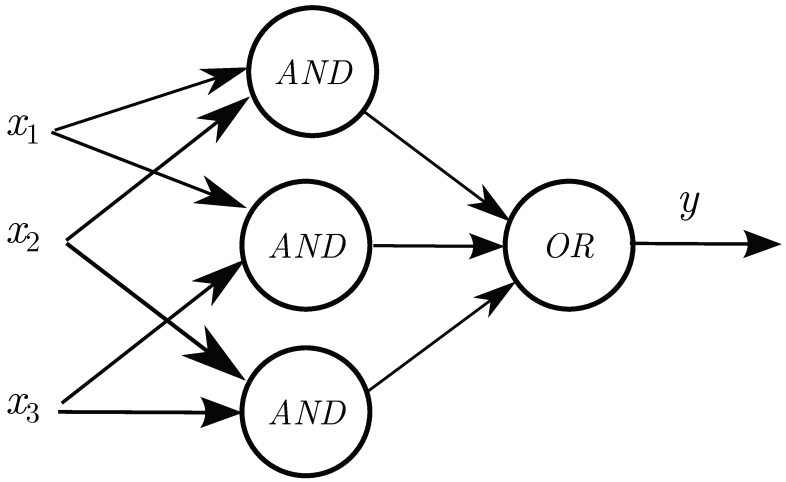


Figura 2.2: Rede neural *fuzzy* com neurônios *AND* e *OR*

Em Hell; Costa e Gomide (2008) são utilizados neurônios lógicos do tipo *OR* para conceber regras com as ativações da camada de entrada, suas saídas são agregadas diretamente por uma camada clássica de redes neurais. O método FCM é utilizado para definir a estrutura da rede e a solução, via métodos de otimização, de uma equação relacional *fuzzy* é utilizada para ajustar os pesos dos neurônios *fuzzy*.

2.4 Resumo

Este capítulo apresentou uma breve revisão e conceitos de redes neurais *fuzzy*, destacando modelos que utilizam t-normas e s-normas. O próximo capítulo trata de modelos evolutivos, e métodos de aprendizado incremental.

Capítulo 3

Sistemas Evolutivos e Aprendizado Extremo

3.1 Introdução

Presentemente a quantidade de dados disponível cresce de maneira sem precedentes. Dados são produzidos e disponibilizados não só em grande quantidade mas também são dinâmicos e apresentados em forma de fluxo (Angelov; Filev e Kasabov, 2010). Em certos contextos, como o da *web*, a quantidade gerada é tão grande que o antigo modelo de aquisição e armazenamento de todos os dados produzidos não faz mais sentido, ou seja, o custo de obter novos dados é menor do que o custo de armazenamento. Portanto, é preciso extrair e armazenar informação de forma compacta e rápida para poder descartar os dados irrelevantes.

Com este contexto em mente, novos métodos de aprendizado de máquina foram desenvolvidos, com o objetivo de adaptar à estas circunstâncias. Em particular, sistemas evolutivos propõem um aprendizado de forma incremental e adaptativa, onde a estrutura do modelo e seus parâmetros evoluem de forma gradual ao longo do tempo para acompanhar a dinâmica do fluxo de dados (Kasabov, 1998).

O sistema a ser modelado pode apresentar comportamentos que caracterizam *concept shifts* e *concept drifts*, isto é, mudanças repentinas ou graduais na dinâmica do sistema, respectivamente (Wang; Schlobach e Klein, 2010). Com isso, é gerado um compromisso entre adaptabilidade a curto prazo e sobrevivência a longo prazo do modelo. Portanto, é necessário que o modelo seja suficientemente flexível para se adaptar a mudanças abruptas do sistema, ao mesmo tempo em que se evita decisões de aprendizado catastróficas, evoluindo gradualmente. O modelo precisa manter-se sempre útil em qualquer instante.

Para tratar este novo problema de modelagem incremental de sistemas a partir de fluxos de dados, métodos com nível de adaptação maior, se comparados com os sistemas adaptativos convencionais, foram desenvolvidos. Métodos estes que também diferem da teoria de identificação de sistemas e do aprendizado de máquina e estatístico, onde usualmente assume-se que os processos possuem distribuição Gaussiana e natureza aleatória (Angelov; Filev e Kasabov, 2010). Na próxima seção serão apresentados modelos mais importantes que possuem as características mencionadas.

3.2 Modelos Evolutivos *Fuzzy*

Sistemas evolutivos *fuzzy* (eFS, *evolving Fuzzy Systems*) são sistemas adaptativos que constroem modelos a partir de fluxos de dados. Estes modelos podem ser vistos como uma combinação de modelos *fuzzy*, como um mecanismo evolutivo para representação e compactação de informação, e métodos recursivos de aprendizado de máquina (Lemos; Caminhas e Gomide, 2011).

Nas seções a seguir serão apresentados detalhadamente os modelos: eTS (Angelov, 2002), ePL (Lima; Gomide e Ballini, 2006), FBeM (Leite et al., 2011) e DENFIS (Kasabov e Song, 2002), pois estão diretamente relacionados com o presente trabalho. Além destes, outros modelos são proeminentes na literatura. A rede neural de inferência *fuzzy* autoconstrutiva proposta em Juang e Lin (1998), utiliza um método baseado em agrupamento alinhado para particionar o espaço de entrada recursivamente, definindo e eliminando conjuntos *fuzzy* de maneira incremental. As regras são do tipo TSK modificadas e o aprendizado dos consequentes é realizado via RLS.

O modelo FLEXFIS (*flexible fuzzy inference system*) (Lughofer, 2008 b), utiliza um algoritmo recursivo baseado em quantização vetorial chamado de eVQ (*evolving Vector Quantization*) (Lughofer, 2008 a). A estimativa dos parâmetros dos consequentes é realizada utilizando uma versão ponderada do RLS. Em Rubio (2009) é proposto um modelo de mapa auto-organizável evolutivo utilizando conjuntos *fuzzy* tipo 2. Este modelo é baseado em um algoritmo de agrupamento evolutivo que utiliza um limiar para a distância entre as amostras de dados e o vizinho mais próximo, identificando quando adicionar um novo grupo. O modelo também faz uso de um mecanismo de poda para identificar regras obsoletas.

Métodos alternativos para adaptar a estrutura de modelos evolutivos *fuzzy* são explorados na literatura. O sistema de inferência *fuzzy* sequencial adaptativo (SAFIS) (Rong et al., 2006)

utiliza um critério baseado em distância, em conjunto com uma medida de influência de novas regras para adaptar sua estrutura. A rede neural *fuzzy* auto-organizável (Leng; McGinnity e Prasad, 2005) emprega um critério de erro para avaliar o desempenho da generalização da rede em conjunto com métodos para adicionar e podar regras.

Modelos linguísticos também são destacados, como em Angelov e Zhou (2008) que propõe um classificador *fuzzy* utilizando o método de agrupamento do modelo eTS e em Filev e Angelov (2007) que utiliza modelagem linguística evolutiva para detecção de falhas. Estudos recentes concentram esforços em computação granular e suas extensões para modelagem, como modelagem evolutiva baseada em intervalos e em conjuntos *fuzzy* (Leite et al., 2011). Outras abordagens são baseadas em árvores *fuzzy*, como em Lemos; Caminhas e Gomide (2011), que utiliza uma árvore de regressão linear *fuzzy*, evoluindo a estrutura do modelo por meio de um método incremental baseado em testes estatísticos que avaliam a qualidade do modelo e de cada operação construtiva candidata.

3.2.1 eTS

O modelo eTS (*evolving* Takagi-Sugeno) (Angelov, 2002), é um modelo funcional do tipo Takagi-Sugeno, onde a base de regras e os parâmetros evoluem adicionando novas regras, modificando as regras existentes e ajustando os parâmetros para representar o conhecimento atual. As regras em modelos eTS são da seguinte forma:

$$\Re_i \ : \ \text{IF } \mathbf{x} \text{ é } \Gamma_i \text{ THEN } \ z_i = \gamma_{i0} + \sum_{j=1}^{n} \gamma_{ij} x_j$$
$$i = 1, \ldots, L^t$$

onde $\Re_i$ representa a i-ésima regra *fuzzy*, L^t o número de regras em $t; t = 1, \ldots ;$ $\mathbf{x} \in [0,1]^n$ é o vetor de entrada, z_i a saída da i-ésima regra; Γ_i o vetor de funções de pertinência e γ_{ij} os parâmetros dos consequentes. O modelo utiliza conjuntos dos antecedentes com funções de pertinência Gaussianas e atualiza os parâmetros dos consequentes com mínimos quadrados. A saída do modelo é determinada por uma média ponderada e normalizada pelas ativações a_i das funções de pertinência:

$$\hat{y}^t = \frac{\sum_{i=1}^{L^t} a_i z_i}{\sum_{i=1}^{L^t} a_i} \tag{3.1}$$

O procedimento recursivo para a evolução baseada em regras é resumido no Algoritmo 3.1.

Algoritmo 3.1 Aprendizado do modelo eTS

inicializar estrutura e parâmetros;
ler novo dado de entrada;
calcular o potencial do novo dado;
atualizar o potencial dos agrupamentos;
modificar ou atualizar a estrutura da base de regras;
calcular os parâmetros dos consequentes das regras;
calcular a saída do modelo;

O procedimento para agrupamento dos dados de entrada afim de gerar os antecedentes é uma variação do proposto por Yager e Filev (1994), onde a capacidade de um ponto se tornar um centro de um agrupamento é avaliada por seu potencial. Os potenciais dos dados são calculados por uma função de Cauchy. Basicamente se o potencial do novo dado é maior do que o potencial de grupos já existentes, é criado um novo grupo. Se um novo grupo está próximo de um grupo já existente, então o seu centro é o novo dado. Este procedimento é denominado eClustering+ e é abordado com mais detalhes na seção 4.4.2.

3.2.2 ePL

Outro paradigma de aprendizado evolutivo é o ePL (do inglês *evolving Participatory Learning*) (Lima; Gomide e Ballini, 2006). A estrutura do modelo é a mesma do eTS. No ePL os objetos de aprendizado são os centros dos agrupamentos $\mathbf{v}_i^t$, que são utilizados para as funções de pertinência dos antecedentes das regras. Aprendizado participativo assume que o aprendizado e o conhecimento sobre um sistema depende do que o mecanismo de aprendizado conhece sobre o sistema em si. É utilizado um esquema com uma medida de compatibilidade ρ e um índice de alerta γ. Se uma amostra de dados com alta compatibilidade é introduzida, o aprendizado procede. Se não, o algoritmo espera o índice de alerta ultrapassar um certo nível para reconhecer uma amostra com baixa compatibilidade como novo conhecimento para o modelo.

A compatibilidade da entrada $\mathbf{x}^t$ com cada grupo é calculada como segue:

$$\rho_i^t = 1 - \frac{||\mathbf{x}^t - \mathbf{v}_i||}{n} \tag{3.2}$$

onde $||\cdot||$ denota norma Euclidiana. O índice de alerta γ é atualizado utilizando:

$$\gamma_i^{t+1} = \gamma_i^t + \beta(1 - \rho_i^t - \gamma_i^t) \tag{3.3}$$

O valor de $\beta \in [0, 1]$ controla a taxa de mudança do índice de alerta: quanto mais perto β chega de um, mais rápido o sistema deve detectar variações de compatibilidade. Se o valor do índice de alerta exceder um limiar $\tau \in [0, 1]$, então um novo grupo é criado com centro em $\mathbf{x}^t$. Caso contrário, o grupo mais compatível, i.e. o grupo com o maior ρ, é atualizado utilizando:

$$\mathbf{v}_i^{t+1} = \mathbf{v}_i^t + \alpha(\rho_i^t)^{1-\gamma_i^{t+1}}(\mathbf{x}^t - \mathbf{v}_i^t) \tag{3.4}$$

Compatibilidade entre os centros dos grupos também são computadas para verificar se existe redundância:

$$\rho_{\mathbf{v}_i}^t = 1 - \sum_{j=1}^{L^t} ||\mathbf{v}_i - \mathbf{v}_j|| \tag{3.5}$$

Se $\rho_{\mathbf{v}_i}^t > \lambda$, $\lambda \in [0, 1]$, então o grupo é removido por ser considerado redundante. O método não necessita de uma entrada direta indicando o número de grupos, mas τ influencia diretamente no número final de grupos gerado. A estimativa dos parâmetros dos consequentes é feita via mínimos quadrados recursivo. Os passos principais do algoritmo de aprendizagem são resumidos no Algoritmo 3.2. A Figura 3.1 mostra um diagrama do aprendizado participativo.

Algoritmo 3.2 Aprendizado participativo

ler novo dado;
calcular o índice de compatibilidade ρ_i^t;
calcular o índice de alerta γ_i^t;
if $\gamma_i^t \geq \tau$ **then**
  $\mathbf{x}^t$ é um novo centro;
else
  atualizar centro mais próximo;
end if
calcular o índice de compatibilidade entre agrupamentos $\rho_{\mathbf{v}_i}^t$;
if $\rho_{\mathbf{v}_i}^t \geq \lambda$ **then**
  excluir $\mathbf{v}_i$;
end if
atualizar estrutura da base de regras;
calcular parâmetros dos consequentes;
calcular saída dos antecedentes;
calcular a saída do modelo;

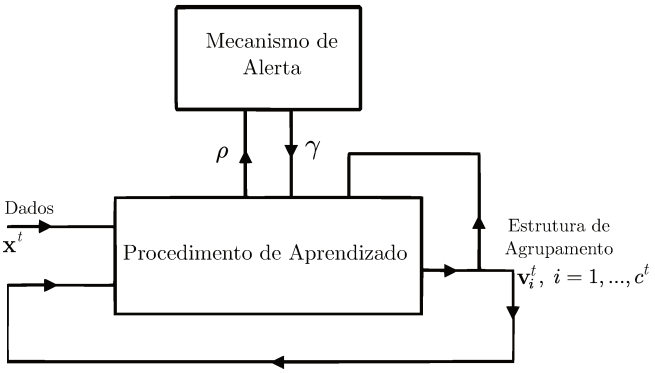


Figura 3.1: Aprendizado participativo

3.2.3 DENFIS

O modelo DENFIS (*dynamic evolving neural-fuzzy inference system*) proposto por Kasabov e Song (2002), tem estrutura similar a da EFuNN (Kasabov, 2001). Para particionar o espaço de entrada, o modelo DENFIS utiliza o método ECM (*evolving clustering method*), que resumidamente adiciona grupos se uma amostra de dado de treinamento apresentado está distante o suficiente dos grupos já formados. Para isso é utilizado um valor de limiar que influencia diretamente no número final de grupos encontrados. Os passos principais do ECM são demonstrados no Algoritmo 3.3, onde $\mathbf{c}_i$ representa as coordenadas dos centros do grupo i, $\mathbf{r}_i$ as dispersões das funções de ativação do grupo i, $\mathbf{d}_j$ a distância Euclidiana entre cada componente i da entrada para cada componente i do centro $\mathbf{c}_j$.

Algoritmo 3.3 ECM

```
iniciar o primeiro grupo c_1 = x^1;
iniciar a dispersão do grupo r_1 = 0;
while existirem dados de entrada do
  ler novo dado;
  calcular distâncias d_j entre x^t e os centros c_j;
  if ∃ d_il > min(r_ij), ∀ l, j then
    calcular os índices s_j = d_j + r_j;
    s* = min(s_ij) ∀ i, j;
    if s* < 2×limiar then
      criar novo grupo com centro c_{j+1} = x^t;
    else
      r_* correspondente ao s* é dividido por 2;
      c_* correspondente ao s* é atualizado;
    end if
  end if
end while
```

Este procedimento então é utilizado ao mesmo tempo em que os parâmetros dos consequentes são atualizados, a saída da rede é calculada da seguinte forma:

$$y^0 = \frac{\sum_{i=1}^{m} w_i f_i(x_1^0, x_2^0, \ldots, x_n^0)}{\sum_{i=1}^{m} w_i} \tag{3.6}$$

onde w_i é a distância entre o exemplo j e o seu respectivo centro. Alternativamente pode-se utilizar ativações das funções de pertinência como $\mathbf{w}^T = [w_1, \ldots, w_m]$.

No caso de utilizar funções do tipo $f_i = b_0 + b_1 x_1 + b_2 x_2 + \ldots + b_n x_n$ para construir um modelo Takagi-Sugeno de primeira ordem, o aprendizado é realizado via mínimos quadrados recursivo com fator de esquecimento λ, aqui reproduzido na sua forma ponderada por $\mathbf{w}$:

$$\mathbf{b}_{t+1} = \mathbf{b}_t + \mathbf{w}_{t+1}\mathbf{P}_{t+1}\mathbf{a}_{t+1}(\mathbf{y}_{t+1}^T - \mathbf{a}_{t+1}^T\mathbf{b}_t) \tag{3.7}$$

$$\mathbf{P}_{t+1} = \frac{1}{\lambda}\left[\mathbf{P}_t - \frac{\mathbf{w}_{t+1}\mathbf{P}_t\mathbf{a}_{t+1}\mathbf{a}_{t+1}^T\mathbf{P}_t}{\lambda + \mathbf{a}_{t+1}^T\mathbf{P}_t\mathbf{a}_{t+1}}\right] \tag{3.8}$$

onde $\mathbf{a}^T = [1, x_1, \ldots, x_n]$ e $\mathbf{b}^T = [b_0, b_1, \ldots, b_n]$.

3.2.4 FBeM

O modelo FBeM (*Fuzzy set Based granular evolving Model*) (Leite et al., 2011) utiliza funções de pertinência Gaussianas e respectivos graus de ativação $a^i,\ i = 1, \ldots, L^t$ para particionar o espaço de entrada e ponderar regras funcionais do tipo Takagi-Sugeno. Os consequentes são calculados por:

$$p^i = r_0^i + \sum_{j=1}^{n} r_j^i x_j \tag{3.9}$$

onde r_j^i são os parâmetros do consequente e p^i é a saída da i-ésima regra. A saída do modelo se dá por uma média ponderada:

$$p = \frac{\sum_{i=1}^{L^t} \min(a_1^i, \ldots, a_n^i)\, p^i}{\sum_{i=1}^{L^t} \min(a_1^i, \ldots, a_n^i)} \tag{3.10}$$

A criação de regras é feita quando não há ativação superior a um determinado limiar, ou seja, $\min(a_1^i, \ldots, a_n^i) \leq \rho\ \forall i$ onde $\rho \in [0, 1]$. A adaptação de regras consiste em contrair ou expandir os grânulos representados pelos conjuntos *fuzzy*, via alteração do raio das funções de pertinência baseada na frequência em que a função é ativada. Também é alterado seu valor

modal com a média das amostras que ativaram o grânulo. Os parâmetros dos consequentes são atualizados com mínimos quadrados recursivo.

O autor também discute um método para ajustar o limiar de criação dos grânulos baseado no erro que o usuário deseja atingir. Ao longo do tempo é possível aglutinar grupos para representar melhor grânulos maiores e remover redundância. Os centros das funções de pertinência são combinados proporcionalmente aos seus raios e o novo raio é determinado somando os dois antigos. É feita uma média dos parâmetros dos consequentes antigos para serem usados como parâmetros do novo consequente. Neste método também são removidos grânulos que não foram ativados por um determinado período de tempo. A visão geral do método é apresentada no Algoritmo 3.4.

Algoritmo 3.4 FBeM

```
iniciar o primeiro grupo;
while existirem dados de entrada do
  ler novo dado;
  if acomodar novas informações then
    criar novo grânulo e regra;
    adaptar regras e grânulos existentes;
  else
    descartar amostra;
  end if
  otimizar estrutura granular;
end while
```

3.3 Aprendizado Extremo

Aprendizado extremo foi proposto por Huang; Zhu e Siew (2004). A ideia consiste em treinar redes neurais com uma única camada intermediária sem utilizar métodos iterativos. O autor argumenta (e prova matematicamente) que os pesos da camada escondida não precisam ser ajustados, é necessário somente atribuir valores aleatórios a esses pesos e ajustar os pesos da camada de saída analiticamente. Aprendizado extremo apresenta vantagens sobre outros métodos para o aprendizado evolutivo e convencional em batelada, como: melhor generalização e procedimento não iterativo e livre de derivadas. De forma geral, uma rede de única camada escondida realiza a seguinte função:

$$f_j(\mathbf{x}) = \sum_{i=1}^{L} \beta_i G(\mathbf{x}, \mathbf{a}_i, b_i) \tag{3.11}$$

onde $G(\mathbf{x}, \mathbf{a}_i, b_i)$ denota a função de saída do neurônio i da camada intermediária com parâmetros $\mathbf{a}_i$ e b_i, L é o número de neurônios da camada intermediária e β_i é o peso entre o neurônio i e o neurônio da saída.

Desta forma é possível utilizar amostras de uma distribuição de probabilidades contínua, como a uniforme padrão $\mathcal{U}(0, 1)$, para inicializar os parâmetros $\mathbf{a}_i$. A seguir determina-se o valor das funções de saída $G(\cdot)$ dos neurônios da camada escondida para todo o conjunto de treinamento e utilizar o método dos mínimos quadrados para estimar os parâmetros β_i.

Huang e Chen (2007) prova que as *extreme learning machines* - ELMs, i.e. métodos que utilizam aprendizado extremo, são aproximadores universais contanto que o modelo tenha essa capacidade. Em outras palavras, treinar um modelo com aprendizado extremo obtém aproximadores universais se a estrutura do modelo permitir aproximação universal.

O modelo geral apresentado na Equação (3.11) foi obtido após a análise isolada de modelos como a rede de base radial (Huang e Siew, 2004) e modelos com regras *fuzzy* representando a camada escondida (Huang et al., 2005). É possível aplicar o método em modelos com aprendizado *online*, pois é possível utilizar a versão recursiva do método dos mínimos quadrados para ajustar os parâmetros da camada de saída. As equações dos métodos de mínimos quadrados recursivo e estático estão demonstradas nas Seções 4.6 e 4.7. Utilizando mínimos quadrados, já foi demonstrado que os pesos assumem os menores valores possíveis, o que favorece na capacidade de generalização da rede (Huang; Zhu e Siew, 2004).

A literatura destaca diversos modelos que utilizam *extreme learning*. Diferentes modelos utilizam variados tipos de neurônios na camada escondida, como: relações *fuzzy* (Huang et al., 2005) e neurônios complexos (Huang et al., 2008). ELM também foi empregado em máquinas de vetor suporte (SVM) (Liu; He e Shi, 2008) e aplicações como: avaliação de segurança em sistemas de potência (Xu et al., 2012), preservação de privacidade (Samet e Miri, 2012), detecção automática de ataque epiléptico em eletroencefalograma (Song; Crowcroft e Zhang, 2012) e reconhecimento de ações humanas (Minhas; Mohammed e Wu, 2012).

Outra extensão natural é o uso de aprendizado extremo para modelar sistemas evolutivos *fuzzy*. Pelo fato de ser livre de derivadas, o método pode ser utilizado em conjunto com neurônios que desempenham qualquer função não linear. É possível utilizá-lo para treinar redes que usem, por exemplo, uninormas em seus neurônios da camada intermediária. Devido a estes argumentos, o aprendizado extremo foi escolhido para realizar o treinamento da rede apresentada no próximo capítulo.

3.4 Resumo

Este capítulo discorreu sobre os modelos evolutivos predominantes na literatura, apresentando uma breve visão do estado da arte da área. Também foi introduzido o método para treinamento de redes neurais conhecido como aprendizado extremo (*extreme learning*). No próximo capítulo, serão utilizados esses conceitos para desenvolver o treinamento para uma rede neural baseada em uninormas.

Capítulo 4

Redes Baseadas em Uninormas com Aprendizado Extremo

4.1 Introdução

Este capítulo apresenta uma rede neural *fuzzy* evolutiva baseada em uninormas. O aprendizado envolve agrupamento para granularizar o espaço de entrada e aprendizado extremo para ajustar os pesos e demais parâmetros da rede. Com o uso de aprendizado extremo, a complexidade introduzida pelas uninormas é contornada, o que possibilita o uso desta em aplicações *online*. Introduz-se os conceitos de uninormas, *unineurons* e *uninetworks*. Detalha-se a estrutura da rede adotada, os métodos de agrupamento considerados e os métodos de ajuste dos parâmetros. O treinamento da versão estática da rede é resumido e a prova de aproximação universal apresentada.

4.2 *Unineurons* e *Uninetworks*

Normas triangulares possuem elemento identidade 1 para t-normas e 0 para t-conormas, também chamadas de s-normas. Uninormas as generalizam e unificam-as permitindo que o elemento neutro tenha seu valor no intervalo unitário, i.e. $e \in [0, 1]$. Portanto, as uninormas são t-normas para $e = 1$ e s-normas quando $e = 0$. Para os outros valores de e elas possuem características intermediárias. Formalmente uninorma é uma operação binária $u : [0, 1] \times [0, 1] \rightarrow [0, 1]$ que possui as seguintes propriedades:

1. Comutatividade: $a\ u\ b = b\ u\ a$
2. Associatividade: $a\ u\ (b\ u\ c) = (a\ u\ b)\ u\ c$
3. Monotonicidade: se $b \leq c$, então $a\ u\ b \leq a\ u\ c$
4. Identidade: $a\ u\ e = a,\ \forall\ a\ \in [0, 1]$

Uninormas também possuem a propriedade da dominância, mostrada na Figura 4.1, que define se a função nos intervalos $a < e, b > e$ e $a > e, b < e$ é mais próxima do operador *min* ou *max*.

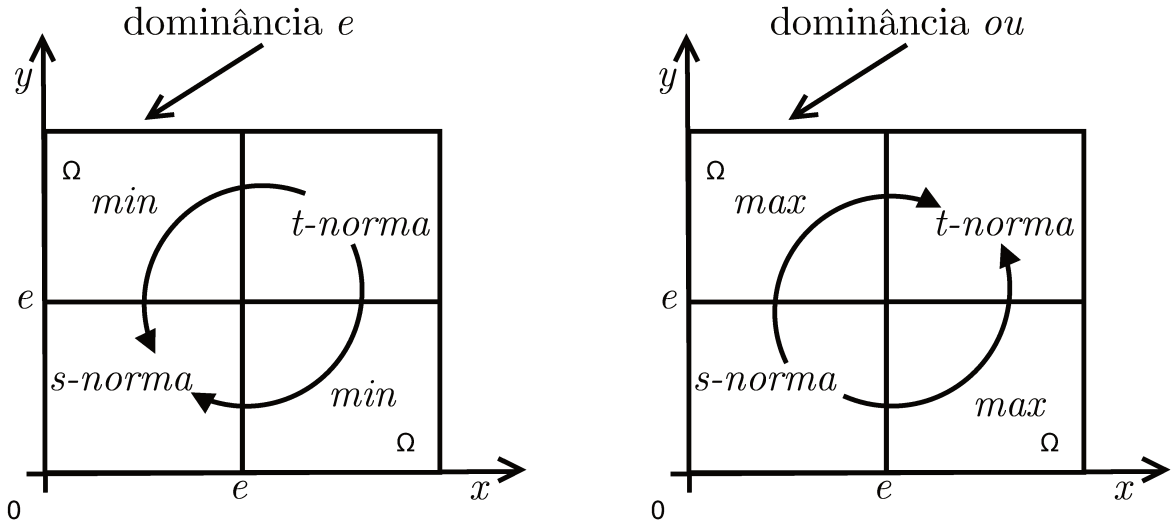


Figura 4.1: Uninorma com dominância *e* e *ou*

Uninormas foram utilizadas em trabalhos relacionados com redes *neuro-fuzzy* como em Pedrycz (2006), que explora o neurônio baseado em uninormas fazendo uma análise do seu comportamento, mas não propõe formas de treinamento de redes baseadas nesse neurônio. Hell et al. (2009) propõem um aprendizado por reforço para realizar o treinamento da camada intermediária. Em Lemos; Caminhas e Gomide (2010) é utilizado um algoritmo genético para ajustar os parâmetros da rede uma vez que a proposta sugere uma nova estrutura de rede. Esses autores sugerem que o uso de uninormas para o treinamento de redes *neuro-fuzzy* resulta em um modelo mais flexível e geral enquanto mantém certa transparência.

Neurônios construídos com uninormas possuem níveis de agregação local e global, pois um neurônio lógico possui uma formulação do tipo: $\underset{i=1}{\overset{n}{\mathrm{L}}}(w_i\ L_i\ x_i)$, onde L é um operador global e L_i local. Assim uninormas podem ser aplicadas a um destes níveis ou em ambos. O modelo de um neurônio baseado em uninormas adotado neste trabalho é mostrado na Figura 4.2. A saída y é computada da seguinte forma:

$$y = U_U(\mathbf{w}, \mathbf{a}) = \overset{n}{\underset{i=1}{\mathrm{U}}}(w_i \; u \; a_i) \tag{4.1}$$

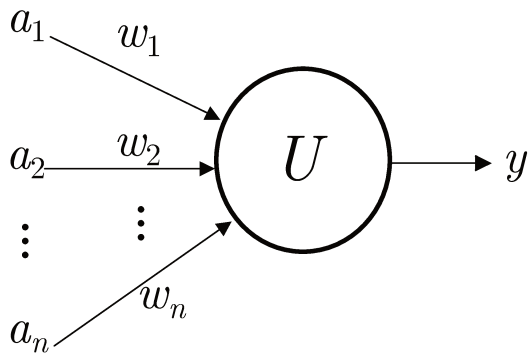


Figura 4.2: Neurônio baseado em uninorma

O aprendizado de redes utilizando este tipo de neurônio foi, em trabalhos iniciais, executado utilizando o método do gradiente. Uma proposta de uninorma modificada é exposta em Lemos; Caminhas e Gomide (2010). Nela a dominância da uninorma também é adicionada como parâmetro, provendo maior flexibilidade ao modelo. O aprendizado é feito via uma combinação do método de agrupamento FCM e um algoritmo genético com codificação real.

A utilização de uninormas traz maior flexibilidade ao custo de mais parâmetros para ajustar em tempo de treinamento. Este fato levou autores dos trabalhos citados a buscarem métodos que possam lidar com a complexidade extra, como algoritmos genéticos. Apesar dos resultados obtidos, esse tipo de treinamento é impraticável para uma aplicação em tempo real.

4.3 Topologia da Rede

O estrutura da rede neural *fuzzy* considerada neste trabalho foi estudada previamente em Hell; Costa e Gomide (2008), apresentando resultados para aproximação de funções. Este tipo de estrutura facilita adição e remoção de neurônios na camada intermediária, característica que possibilita a implementação de sistemas evolutivos. A rede é isomórfica à uma base de regras *fuzzy* e tem duas partes principais, uma correspondendo a um sistema de inferência *fuzzy* e outra correspondente a uma operação de agregação feita por uma rede neural. A Figura 4.3 exibe a estrutura da rede.

As primeiras duas camadas formam o sistema de inferência. A primeira camada é composta por funções de pertinência de conjuntos *fuzzy* que granularizam o espaço de entrada, caracteri-

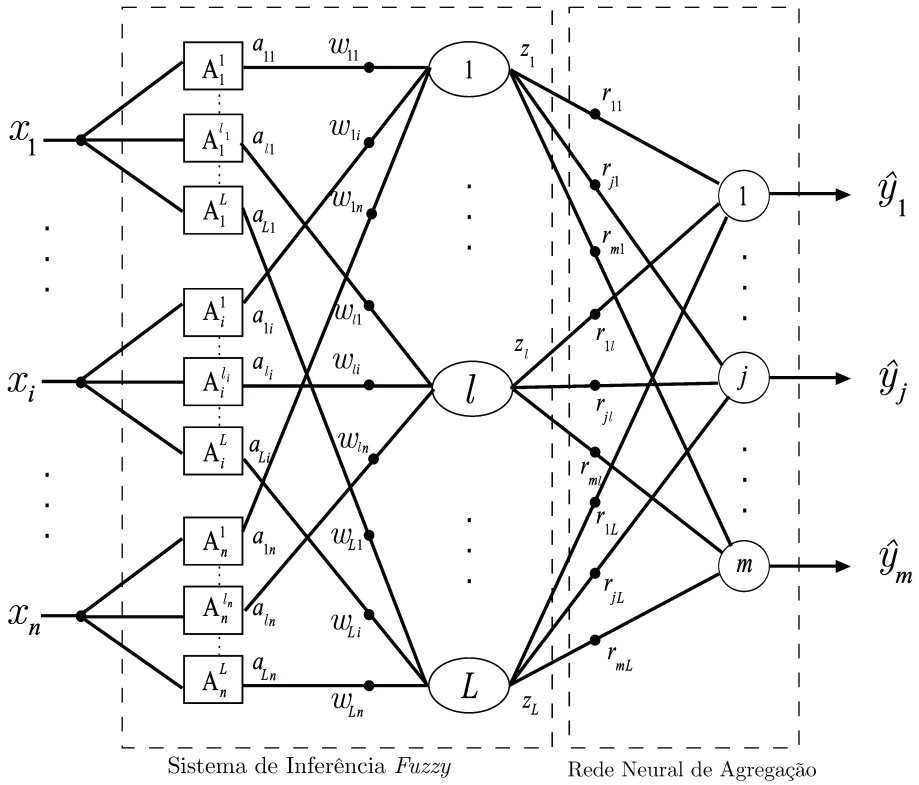


Figura 4.3: Estrutura da rede neural *fuzzy* baseada em uninormas

zando uma partição *fuzzy*. Foram utilizadas funções de pertinência Gaussianas centradas em $\mathbf{c}_l$ com dispersão σ. O grau de pertinência da entrada x_i é computado utilizando a Equação (4.2). Para cada componente x_i de um vetor de entrada $\mathbf{x}$ n-dimensional existem L conjuntos *fuzzy* $A_i^{l_i}$, $l_i = 1, \ldots, L$. L corresponde ao número de regras do sistema.

$$a_{li} = \exp\left(-\frac{(x_i - c_{li})^2}{2\sigma^2}\right) \tag{4.2}$$

onde $l = 1, \ldots, L$, $i = 1, \ldots, n$ e c_{li} é a i-ésima componente do l-ésimo centro de grupo. A dispersão σ é definida *a priori* e mantida constante durante o processo evolutivo.

A segunda camada contém neurônios do tipo U_U para agregar as saídas da camada de entrada ponderadas pelos pesos w_{li}:

$$z_l = \underset{i=1}{\overset{n}{\mathrm{U}}}(a_{li} \; u \; w_{li}) \tag{4.3}$$

aqui z_l , $l = 1, \ldots, L$ é a saída do l-ésimo neurônio da camada intermediária; U são uninormas definidas usando o construtor (4.4); w_{li} são os pesos sinápticos.

$$u(a,w) = \begin{cases} e + (1-e)\left(\frac{(a-e)}{(1-e)} \; s \; \frac{(w-e)}{(1-e)}\right) & \text{se } a, w \in (e,1] \\ e(\frac{a}{e} \; t \; \frac{w}{e}) & \text{caso contrário} \end{cases} \tag{4.4}$$

onde a t-norma t e a s-norma s são, respectivamente, o produto algébrico $a\,t\,w = aw$ e a soma probabilística $a\,s\,w = a+w-aw$. Esse tipo de construtor é sugerido em (Yager e Rybalov, 1996). Essa escolha resulta em uma superfície mais suave para a função de transferência do neurônio *fuzzy*, ao contrário da combinação *min-max* que gera mudanças abruptas nos extremos de (a, e) e (e, w) Yager e Rybalov (1996). A Figura 4.4 mostra a superfície gerada pela construção de uninorma utilizada neste trabalho.

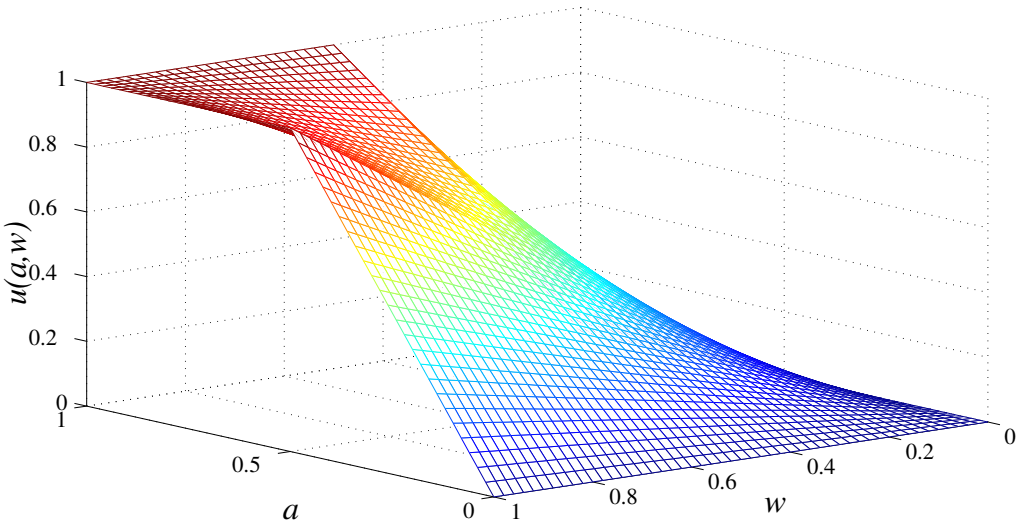


Figura 4.4: Superfície da uninorma utilizada

A terceira e última camada, que forma a segunda parte da rede, é uma camada clássica de neurônios com funções de ativação sigmoidal $f(x) = 1/(1+e^{-x})$, cujas saídas são:

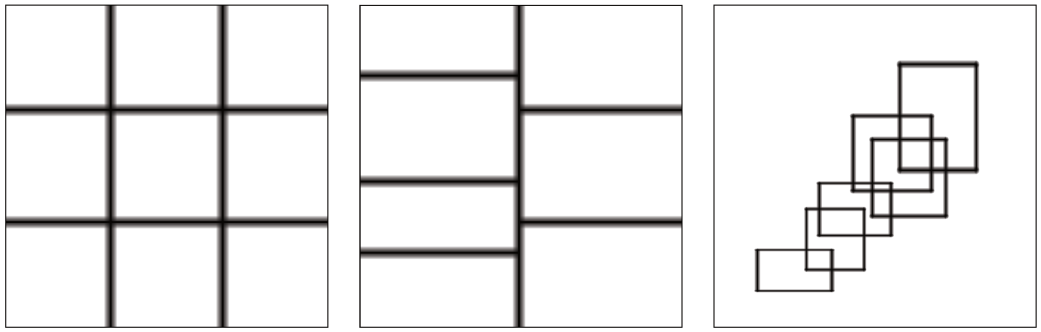
$$\hat{y}_j = f\left(\sum_{l=1}^{L} r_{jl} z_l\right) \tag{4.5}$$

onde $j = 1, \ldots, m$, m é a dimensão do espaço de saída e r_{jl} é o peso associado à conexão entre o l-ésimo neurônio da camada intermediária com a j-ésima saída.

4.4 Procedimentos de Agrupamento

A maioria das redes neurais *fuzzy* utilizam algum método de agrupamento para granularizar o espaço de entrada. O FCM (*fuzzy c-means*) é uma escolha frequente na literatura e foi utilizado para particionar o espaço de entrada no caso estático. Duas alternativas são apresentadas para o agrupamento recursivo: eClustering+, do método eTS+ (Angelov, 2010), e ePL, utilizando aprendizado participativo (Lima; Gomide e Ballini, 2006).

O espaço de entrada pode ser particionado basicamente em três formas (Jang e Sun, 1997, p. 86): grade, árvore e por grupos, conforme ilustra a Figura 4.5 para espaços de entrada de duas dimensões.



(a) Partição em grade (b) Partição em árvore (c) Partição em grupos

Figura 4.5: Tipos de partição do espaço de entrada.

Utilizar o modo grade pode ser interessante para aproximar funções onde se tem um hipercubo bem definido para a região de trabalho do modelo. Utilizar uma partição em forma de árvore é útil quando o sistema apresenta características hierárquicas. Particionar em grupos pode economizar recursos e criar modelos mais compactos e interpretáveis, de forma que podem existir regiões do espaço de entrada que não possuem amostras, ou não representam regiões de operação do sistema. Nas seções a seguir são apresentados os métodos utilizados em conjunto com os algoritmos de treinamento propostos.

4.4.1 FCM

Fuzzy C-Means (FCM) é um método de agrupamento estático onde uma amostra de dados pode pertencer a um ou mais grupos com um certo grau de pertinência (Bezdek; Ehrlich e Full, 1984). O objetivo do método é minimizar a seguinte função:

$$J_m = \sum_{i=1}^{N} \sum_{j=1}^{L} \mu_{ij}^m \, ||\mathbf{x}_i - \mathbf{c}_j||^2 \quad , \quad m \geq 1 \tag{4.6}$$

onde μ_{ij} é o grau de pertinência de $\mathbf{x}_i$ ao grupo $\mathbf{c}_j$.

A cada iteração μ_{ij} e os centros $\mathbf{c}_j$ são atualizados conforme:

$$\mu_{ij} = \left(\sum_{k=1}^{L} \left(\frac{\mathbf{x}_i - \mathbf{c}_j}{\mathbf{x}_i - \mathbf{c}_k} \right)^{\frac{2}{m-1}} \right)^{-1} \tag{4.7}$$

$$\mathbf{c}_j = \frac{\sum_{i=1}^{N} \mu_{ij} \mathbf{x}_i}{\sum_{i=1}^{N} \mu_{ij}} \tag{4.8}$$

O procedimento é executado até que não haja modificação significativa na estrutura do agrupamento, como por exemplo até que $\max \left\{ |\mu_{ij}^{t+1} - \mu_{ij}^{t}| \right\} < \epsilon$, onde ϵ é um valor escolhido previamente. A figura 4.6 mostra um exemplo de granularização feita de acordo com o algoritmo FCM.

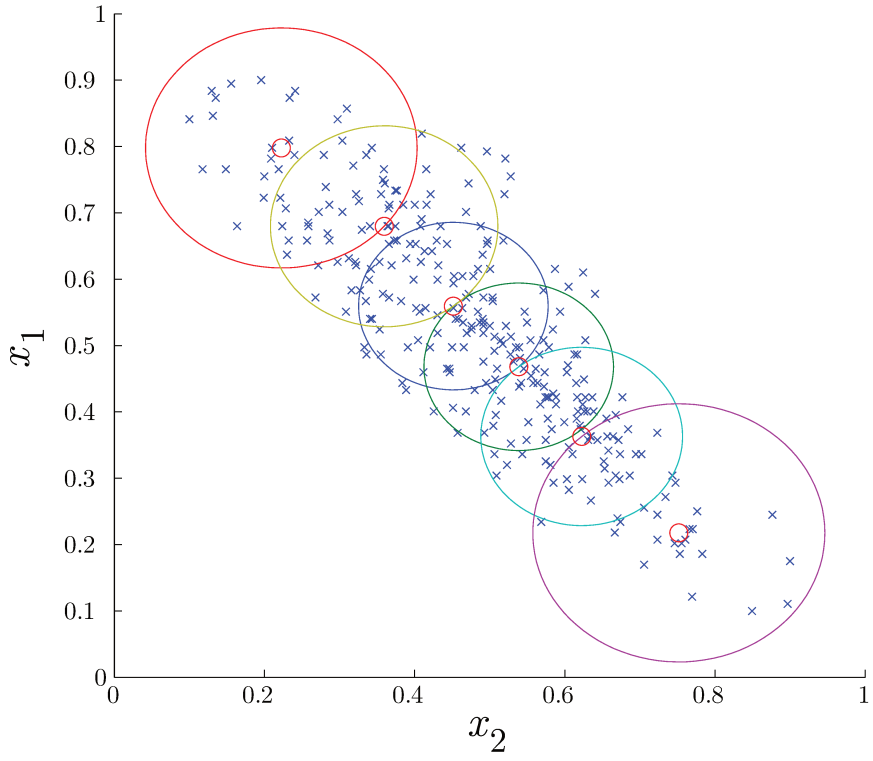


Figura 4.6: Exemplo de granularização feita pelo algoritmo FCM

Os passos principais do FCM estão listados no Algoritmo 4.1.

Algoritmo 4.1 FCM

ler a primeira amostra de dados
inicializar os graus de pertinência μ_{ij};
while condição de parada não satisfeita **do**
  atualizar centros com (4.8)
  atualizar pertinências com (4.7)
end while

GK-Clustering (Gustafson e Kessel, 1978) estende o FCM, pois possibilita a detecção de grupos não esféricos via uso de matriz de covariância, modificando a norma utilizada como distância. Desta forma o método consegue identificar grupos elipsoidais. Recentemente foi proposta uma versão recursiva do GK-Clustering por Dovzan e Skrjanc (2011) a partir do GK-Clustering estático.

4.4.2 eClustering+

Uma abordagem para agrupamento recursivo consiste em estimar a densidade em cada ponto utilizando a Equação (4.9):

$$D_t(o_t) = \left((t-1) \left(\sum_{j=1}^{n+m} o_{tj}^2 + 1 \right) + b_t - 2 \sum_{j=1}^{n+m} o_{tj} h_{tj} \right)^{-1} \tag{4.9}$$

com $o_t = [\mathbf{x}^T, \mathbf{y}^T]^T$; $D_1(o_1) = 1$; $b_t = b_{t-1} + \sum_{j=1}^{n+m} o_{(t-1)j}^2$; $b_1 = 0$; $h_{tj} = h_{(t-1)j} + o_{(t-1)j}$; $h_{1j} = 0$; $j = 1, \ldots, n+m$.

Recursivamente, a densidade para cada centro de grupo é calculada utilizando a Equação (4.10):

$$D_t(o^{i^*}) = \frac{t-1}{t-1+(t-2)\left(\frac{1}{D_{t-1}(o^{i^*})} - 1\right) + \sum_{j=1}^{n+m} (o_{tj} - o_{(t-1)j})} \tag{4.10}$$

inicializando com $D_1(o^{i^*}) = 1$.

Candidatos a centros de grupo são selecionados se a seguinte condição é verdadeira: condição A: $D_t(o_t) > \max D_t(o^{i^*})$ *ou* $D_t(o_t) < \min D_t(o^{i^*})$ onde i^* são os índices dos centros dos grupos. Para evitar sobreposição excessiva e redundância, o grupo é criado somente se não satisfaz: condição B: $\exists l \in [1, L^t] \mid a_{li} > e^{-1}, \forall i \in [1, n]$.

Outro ponto importante é o monitoramento da qualidade dos grupos. Segundo Angelov (2010) a qualidade das regras pode ser monitorada via: suporte, que representa o número de amostras de dados associado à regra; idade, indicando idade relativa e acumulada das regras;

utilidade, representando o acúmulo relativo da ativação das regras; e densidade local, que representa uma medida de poder de generalização.

Neste trabalho foi utilizada a medida de idade das regras para remover as que apresentarem idade maior do que um desvio padrão acima da médias das idades. A idade I é calculada utilizando o suporte S da seguinte forma:

$$S_l^t = S_l^{t-1} + 1 \text{ , se } l = \arg\max_{l=1}^{L^t} \sum_{i=1}^{n} a_{li} \tag{4.11}$$

onde a_l é a ativação como em (4.2) e a idade é computada com:

$$I_l^t = t - \frac{\sum_{i=1}^{S_l^t} E_i}{S_l^t} \tag{4.12}$$

onde E_i denota o instante de tempo em que cada amostra pertencente ao grupo foi apresentada ao sistema. As regras com idade maior que a média mais um desvio padrão são removidas, ou seja, se uma regra satisfaz: condição C: $I_l^t > \overline{I^t} + \widehat{I^t}$, onde $\overline{I^t}$ é a média das idades em t e $\widehat{I^t}$ o desvio padrão.

O Algoritmo 4.2 sumariza os passos do algoritmo eClustering+.

Algoritmo 4.2 eClustering+

```
ler a primeira amostra de dados
inicializar o primeiro centro o^{i*} ← o_1;
D_1(o_1) ← 1; b_1 ← 0; h_{1j} ← 0; D_1(o^{i*}) ← 1;
while existirem dados de entrada do
  ler o vetor entrada-saída o_t
  atualizar a densidade do ponto (4.9)
  atualizar as densidades dos grupos (4.10)
  if condição A é verdadeira then
    formar um novo centro de grupo
    L^t ← L^t + 1; o^{i*} ← o_t; D(o^{i*}) ← 1;
  end if
  if condição B é verdadeira then
    remover o grupo que ativou essa condição
  end if
  atualizar as idades dos grupos (4.10)
  if condição C é verdadeira then
    remover a regra que ativou essa condição
  end if
end while
```

4.4.3 Agrupamento ePL

A segunda alternativa para agrupamento recursivo é utilizar os procedimentos para agrupamento do modelo ePL, apresentado na Seção 3.2.2. Como o método já foi apresentado, nesta seção serão enumeradas as equações utilizadas para implementar o método. A compatibilidade da entrada $\mathbf{x}^t$ com cada grupo é calculada com:

$$\rho_i^t = 1 - \frac{||\mathbf{x}^t - \mathbf{v}_i||}{n} \tag{4.13}$$

onde $||\cdot||$ denota norma Euclidiana. O índice de alerta γ é atualizado:

$$\gamma_i^{t+1} = \gamma_i^t + \beta(1 - \rho_i^t - \gamma_i^t) \tag{4.14}$$

Se o valor do índice de alerta exceder um limiar $\tau \in [0, 1]$, então um novo grupo é criado com centro em $\mathbf{x}^t$. Caso contrário, o grupo mais compatível, i.e. o grupo com o maior ρ, é atualizado utilizando:

$$\mathbf{v}_i^{t+1} = \mathbf{v}_i^t + \alpha(\rho_i^t)^{1-\gamma_i^{t+1}}(\mathbf{x}^t - \mathbf{v}_i^t) \tag{4.15}$$

A compatibilidade entre os centros dos grupos são computadas para verificar se existe redundância:

$$\rho_{\mathbf{v}_i}^t = 1 - \sum_{j=1}^{L^t} ||\mathbf{v}_i - \mathbf{v}_j|| \tag{4.16}$$

Se $\rho_{\mathbf{v}_i}^t > \lambda$, $\lambda \in [0, 1]$, então o grupo é removido por ser considerado redundante. O Algoritmo 4.3 resume os passos do agrupamento do ePL.

4.5 Ajuste de Parâmetros

Como apresentado na Seção 3.3, o aprendizado extremo é uma alternativa interessante para treinar redes neurais com uma camada intermediária. Os autores do método demonstraram ser possível utilizá-lo em conjunto com vários tipos de funções de ativação não lineares nos neurônios da camada escondida (Huang; Zhu e Siew, 2004). Outro ponto que viabiliza a utilização de aprendizado extremo no modelo proposto é a possibilidade de realizar o ajuste dos parâmetros da última camada utilizando o método de mínimos quadrados recursivo.

O treinamento é realizado de forma semelhante aos outros modelos que utilizam aprendizado

Algoritmo 4.3 Agrupamento ePL

```
ler a primeira amostra de dados
inicializar o primeiro centro v_1^1 ← x^1; γ_1^1 ← 0;
while existem dados de entrada do
  ler o vetor entrada x^t
  for i = 1 até L^t do
    atualizar ρ_i^t com (4.13)
    atualizar γ_i^t com (4.14)
  end for
  if γ_i^t > τ, ∀i ∈ [1, L^t] then
    formar um novo grupo centrado em x^t
  else
    atualizar o centro com maior ρ utilizando (4.15)
  end if
  computar ρ_{v_i}^t utilizando (4.16)
  if ∃ ρ_{v_i}^t de forma que ρ_{v_i}^t > λ then
    v_i^t é redundante; remover o i-ésimo grupo
  end if
end while
```

extremo. Primeiramente, valores aleatórios em $[0, 1]$ são atribuídos aos pesos w_{li} e elementos identidade e dos neurônios U_U. Os parâmetros da última camada são atualizados via mínimos quadrados para o caso estático e mínimos quadrados recursivo para o caso evolutivo. Em Yen; Wang e Gillespie (1998) são sugeridas dois tipos de aprendizado para determinar os parâmetros de modelos *fuzzy*: local e global. No sentido do trabalho citado, o método empregado nesta dissertação realiza aprendizado global. Cada parâmetro dos consequentes contribui parcialmente para estimar a saída do modelo.

4.6 XUninet

A versão estática do treinamento para a rede baseada em uninormas é denominada XUninet (*e**X**treme learning **Uninet**work*). Para treinar a XUninet, a granularização do espaço de entrada é feita utilizando o algoritmo *fuzzy c-means* (FCM) (Bezdek; Ehrlich e Full, 1984). FCM é uma escolha comum para granularizar o espaço de entrada em trabalhos correlatos (Shing e Jang, 1993; Ballini e Gomide, 2002; Lemos; Caminhas e Gomide, 2010). A distância Euclidiana até o centro do grupo mais próximo foi utilizada como a dispersão σ dos grupos. Os centros dos grupos são projetados nos eixos correspondentes e são formados conjuntos *fuzzy* Gaussianos, centrados nas projeções dos respectivos valores modais (centros) e com dispersão σ;

O ajuste dos parâmetros restantes é realizado da seguinte forma: utiliza-se amostras da

distribuição uniforme padrão $\mathcal{U}(0,1)$ para iniciar os pesos e elementos identidade dos neurônios U_U, e a seguir o método dos mínimos quadrados é aplicado para ajustar os pesos da camada de saída $\mathbf{R}_{L\times m}$:

$$\mathbf{R} = (\mathbf{Z}^T\mathbf{Z})^{-1}\mathbf{Z}^T f^{-1}(\mathbf{Y}) \tag{4.17}$$

Onde $f^{-1}(\mathbf{Y}) = \log(\mathbf{Y}) - \log(1-\mathbf{Y})$ é a inversa da função de ativação da saída, e os valores de $\mathbf{Y}_{N\times m}$ são os N exemplos de dados esperados na saída para cada dimensão m. $\mathbf{Z}_{N\times L}$ é uma matriz cujos valores são as saídas dos L neurônios da segunda camada após propagar os N exemplos de treinamento através da rede. O Algoritmo 4.4 resume os principais passos do treinamento da XUninet.

Algoritmo 4.4 Procedimento de aprendizado da XUninet

ler e normalizar os dados
executar o FCM para particionar o espaço de entrada
atribuir aleatoriamente os valores dos pesos e elementos identidade
propagar os exemplos de entrada através da rede
utilizar mínimos quadrados

4.6.1 Aproximação Universal

Provas de aproximação universal são utilizadas para demonstrar a capacidade de aproximação funcional de redes neurais (Hornik; Stinchcombe e White, 1989; Park e Sandberg, 1991). Essas provas tem o objetivo de mostrar que uma rede neural pode aproximar uma função contínua com um erro arbitrário, dado um número suficiente de unidades de processamento. Esta seção apresenta uma prova para a versão *batch* da rede proposta, a XUninet. Espera-se que a versão evolutiva compartilhe das mesmas características. Está fora do escopo desta dissertação provar que a versão evolutiva é um aproximador universal.

Teorema 1. Seja uma rede XUninet, N pares de vetores entrada e saída correspondentes $(\mathbf{x}_t, \mathbf{y}_t)$ onde $\mathbf{x}_t \in \mathbb{R}^n$ e $\mathbf{y}_t \in \mathbb{R}^m$, com $N = L$ e funções de pertinência A^l centradas em $\mathbf{x}_t$ quando $l = t$ e raio tendendo a zero, i.e. $\sigma_l \to 0$. Para qualquer $\mathbf{w}$ e elementos de identidade e com valores em $(0,1)$ a matriz de saída da camada intermediária $\mathbf{Z}$ é inversível e $||\mathbf{Zr} - \mathbf{Y}|| = 0$ com probabilidade um.

Prova. Quando as funções de pertinência A^l estão centradas em $\mathbf{x}_t$ quando $l = t$ e têm raio tendendo a zero, as saídas dos neurônios a_l da camada de entrada serão iguais a 1 somente

quando $l = t$. Em outras palavras, as entradas do l-ésimo *unineuron* serão iguais a 1 somente quando $l = t$ enquanto todos os outros neurônios terão 0 em suas entradas. Denote por $z_l(1)$ a saída do l-ésimo *unineuron* quando todas as suas entradas são 1 e por $z_l(0)$ quando todas as entradas são 0. A matriz $\mathbf{Z_{N\times N}}$, cujas linhas correspondem às saídas dos N neurônios da camada intermediária para cada um dos N dados de entrada, terá a seguinte forma:

$$\mathbf{Z} = \begin{pmatrix} z_1(1) & z_2(0) & \cdots & z_N(0) \\ z_1(0) & z_2(1) & \cdots & z_N(0) \\ \vdots & \vdots & \ddots & \vdots \\ z_1(0) & z_2(0) & \cdots & z_N(1) \end{pmatrix}$$

A matriz $\mathbf{Z}$ pode ser reescrita como $\mathbf{Z} = \mathbf{B} + \mathbf{pq}^T$, onde $\mathbf{B}$ é

$$\mathbf{B} = \begin{pmatrix} z_1(1) - z_1(0) & & & 0 \\ & z_2(1) - z_2(0) & & \\ & & \ddots & \\ 0 & & & z_N(1) - z_N(0) \end{pmatrix}$$

e $\mathbf{p}$ é um vetor coluna $N\times 1$ com todas as N componentes iguais a 1 e $\mathbf{q} = [z_1(0), z_2(0), \ldots, z_N(0)]$. Dado que uninormas são monotônicas, e os pesos $w_{li} \in (0, 1)$, então $z_l(1) > z_l(0)$. Conclui-se que $\mathbf{B}$ é inversível pois $z_l(1) - z_l(0) > 0,\ \forall\ l$, isto é, seu determinante é positivo.

A fórmula de Sherman-Morrison (Sherman e Morrison, 1950) nos fornece um mecanismo para obter a inversa de $\mathbf{Z}$. Isto é, $\mathbf{Z^{-1}}$ pode ser reescrita por:

$$\mathbf{Z}^{-1} = \left(\mathbf{B} + \mathbf{pq^T}\right)^{-1} = \mathbf{B^{-1}} - \frac{\mathbf{B^{-1}pq^TB^{-1}}}{\mathbf{1} + \mathbf{q^TB^{-1}p}} \tag{4.18}$$

Já foi demonstrado que $\mathbf{B}^{-1}$ existe. Além disso, os elementos da diagonal de $\mathbf{B}$ são positivos, logo sua inversa $\mathbf{B^{-1}}$ também é diagonal com elementos positivos. Por construção $\mathbf{p}$ tem somente elementos positivos e os elementos de $\mathbf{q}$ são não negativos. Então $\mathbf{q^TB^{-1}p} \geq 0$ e o denominador $1 + \mathbf{q^TB^{-1}p} \geq 1$, portanto $\mathbf{Z}$ é inversível.

Note que assume-se os pesos $\mathbf{w} \in (0, 1)$ para evitar os casos degenerados onde $e = 1$ e $\mathbf{w} = 0$ ou $e = 0$ e $\mathbf{w} = 1$, o que pode resultar em $z_n(0) = z_n(1)$. Na prática essa questão não é muito importante pois se forem utilizadas amostras da distribuição uniforme padrão $\mathcal{U}(0, 1)$ para produzir os pesos e elementos de identidade, a probabilidade desses casos ocorrerem é

muito próxima a zero.

Teorema 2. Dado $\varepsilon > 0$ e o modelo XUninet, para N exemplos de treinamento distintos $(\mathbf{x}_t, \mathbf{y}_t)$ onde $\mathbf{x}_t \in \mathbb{R}^n$ e $\mathbf{y}_t \in \mathbb{R}^m$, com L neurônios na camada intermediária onde $L \leq N$, para qualquer $\mathbf{w}_{li}$ e elementos identidade e das uninormas escolhidos aleatoriamente em $(0, 1)$, com probabilidade um, $||\mathbf{Zr} - \mathbf{Y}|| < \varepsilon$.

Prova. A validade do teorema é imediata, ou então podemos utilizar $N = L$ e obter $||\mathbf{Zr} - \mathbf{Y}|| = 0$ como resultado do Teorema 1.

Os teoremas apresentados comprovam a capacidade da rede de armazenar conhecimento, no caso a aproximação de uma função. É esperado que quando temos menos neurônios do que amostras de dados, i.e. $L << N$, as regras sejam combinadas pela última camada de agregação. Com isso, amostras que estejam entre dois centros de grupos próximos, possuem um grau de compatibilidade com cada regra e apresentem um comportamento que possa ser aproximado pelas regras ativas. A última camada da rede agrega as contribuições das regras ativas.

4.7 eXUninet

A versão evolutiva da rede foi denominada eXUninet (***e**volving e**X**treme learning **Uni-net**work*). O treinamento da eXUninet é efetuado utilizando agrupamento recursivo e aprendizado extremo. Os passos principais são: atualizar a estrutura de agrupamento, ajustar os elementos identidade e das uninormas, os pesos dos neurônios *fuzzy* w_{li} e os pesos da camada de saída r_{jl}. Primeiramente são utilizadas amostras da distribuição uniforme padrão $\mathcal{U}(0, 1)$ para inicializar os pesos w_{li} e elementos identidade e dos neurônios U_U a seguir, utilizando o método de mínimos quadrados recursivo com fator de esquecimento λ, atualiza-se os pesos da camada de saída $\mathbf{r}^t = [r_{11}^t \dots r_{jl}^t \dots r_{mL^t}^t]$ utilizando as Equações (4.19)-(4.21):

$$\mathbf{J}^t = \mathbf{Q}^{t-1}\mathbf{z}^t\{\lambda + \mathbf{z}^t\mathbf{Q}^{t-1}(\mathbf{z}^t)^T\}^{-1} \tag{4.19}$$

$$\mathbf{Q}^t = (\mathbf{I}_{L^t} - \mathbf{J}^t\mathbf{z}^t)\lambda^{-1}\mathbf{Q}^{t-1} \tag{4.20}$$

$$\mathbf{r}^t = \mathbf{r}^{t-1} + (\mathbf{J}^t)^T(f^{-1}(\hat{\mathbf{y}}^t) - \mathbf{z}^t(\mathbf{r}^{t-1})^T) \tag{4.21}$$

onde $f^{-1}(\hat{\mathbf{y}}^t) = \log(\hat{\mathbf{y}}^t) - \log(1 - \hat{\mathbf{y}}^t)$. A inicialização de $\mathbf{Q}$ usualmente é $\mathbf{I}\omega$, $\omega = 1000$. Após esse ajuste, é utilizado um algoritmo de agrupamento recursivo, dentre os apresentados na Seção 4.4, para atualizar a estrutura do modelo. O Algoritmo 4.5 resume os principais passos

do treinamento da eXUninet.

Algoritmo 4.5 Aprendizado da eXUninet

ler a primeira amostra de dados $\mathbf{x}^1$
inicializar o primeiro centro de grupo em $\mathbf{x}^1$
inicializar a estrutura da rede e seus parâmetros
while existirem dados de entrada **do**
  ler o próximo vetor de entrada $\mathbf{x}^t$
  estimar a saída $\hat{\mathbf{y}}^t$
  ler o vetor de saída $\mathbf{y}^t$
  executar um passo do método de agrupamento
  atualizar a estrutura do modelo para corresponder ao agrupamento
  atualizar os parâmetros da rede utilizando (4.19) - (4.21)
end while

4.8 Resumo

Este capítulo apresentou a rede neural *fuzzy* proposta neste trabalho. Esta rede neural utiliza neurônios baseados em uninormas e agrupamento recursivo com aprendizado extremo Mostrou-se que, no caso particular em que a rede opera em modo estático, ela é um aproximador universal. No próximo capítulo serão apresentadas resultados computacionais com evidências empíricas para mostrar o teorema de aproximação funcional e comprovar a aplicabilidade dos modelos em problemas reais e *benchmarks* encontrados na literatura.

Capítulo 5

Resultados Computacionais

5.1 Introdução

Este capítulo apresenta os modelos e os respectivos resultados computacionais produzidos pela rede neural *fuzzy* detalhada no capítulo anterior. Simulações foram realizadas utilizando um computador pessoal com processador *dual-core* a $2.27GHz$ em ambiente MATLAB. A medida de erro utilizada é o RMSE - raiz quadrada do erro médio quadrático:

$$RMSE = \sqrt{\frac{1}{t}\sum_{j=1}^{t}(y^j - \hat{y}^j)^2} \tag{5.1}$$

ou o erro quadrático médio MSE (*mean squared error*):

$$MSE = \frac{1}{N}\sum_{j=1}^{N}(y^j - \hat{y}^j)^2 \tag{5.2}$$

Os dados foram normalizados no intervalo $[0.1, 0.9]$ para evitar problemas com a função de ativação dos neurônios da saída, pois a mesma não possui inversa nos pontos 0 e 1. Os erros foram computados para os dados normalizados.

Os métodos utilizados nesta seção fazem uso de parâmetros para ajustar o número final de neurônios na camada intermediária ou, equivalentemente, o número de regras fuzzy. Em alguns casos, é possível obter erros menores otimizando estes parâmetros. Contudo, o objetivo foi demonstrar que a rede neural *fuzzy* provê resultados competitivos com um menor tempo de treinamento. Como a eXUninet+ não usa nenhum parâmetro para realizar o agrupamento, quando

possível, os parâmetros dos outros métodos foram ajustados para utilizar aproximadamente o mesmo número de neurônios.

5.2 Resultados da XUninet

A proposta de aprendizado extremo tem a vantagem de melhor generalização e, por consequência, também evita o *overfitting* ou sobretreino. É possível então, utilizar todo o conjunto de treinamento para realizar o aprendizado. Não é preciso utilizar conjuntos de teste para parada precoce, pois não se trata de um procedimento iterativo (Huang; Zhu e Siew, 2004).

5.2.1 Forno à Gás de Box-Jenkins

O problema do forno à gás de Box e Jenkins (1970) é estudado extensivamente na literatura, sendo um padrão para testar e avaliar algoritmos e métodos de modelagem. A entrada do sistema é o fluxo de gás oxigênio-metano x e a é saída a concentração de dióxido de carbono emitido y. Estudos anteriores sugerem o uso de um preditor da seguinte forma (Leite et al., 2011):

$$\hat{y}^t = p(y^{t-1}, x^{t-4}) \tag{5.3}$$

Este problema é um problema de identificação de sistemas, pois consiste em criar um modelo que aproxime o forno à gás. Para reproduzir as mesmas condições dos métodos utilizados para comparação, os primeiros 200 exemplos de treinamento foram utilizados para treinar a rede, e os últimos 90 para testar e computar o RMSE. A Tabela 5.1 resume os resultados.

Tabela 5.1: Desempenho da XUninet para o Forno de Box-Jenkins

Modelo	Referência	Regras/Neurônios	RMSE
Transformada *Fuzzy*	Perfilieva (2006)	-	0.0906
Rede com UNI	Ballini e Gomide (2002)	6	0.0737
ANFIS	Shing e Jang (1993)	6	0.0526
Baseada em uninorma	Lemos; Caminhas e Gomide (2010)	6	0.0487
XUninet	-	6	0.0429

Neste caso, a XUninet obteve melhor resultado do que os outros métodos. A média do RMSE para as 100 execuções foi de 0.080848. O treinamento de uma única rede levou $100ms$,

o processo de executar o método 100 vezes levou menos de $20s$, assumidamente mais rápido do que Lemos; Caminhas e Gomide (2010) que utiliza um algoritmo genético para treinamento. A Figura 5.1 exibe os resultados da XUninet.

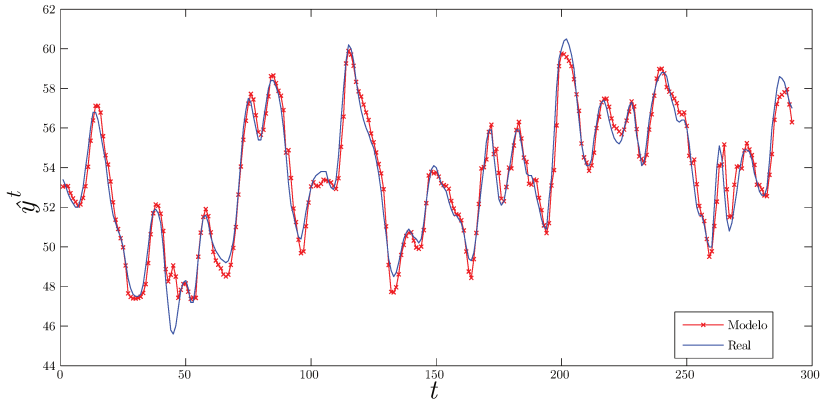


Figura 5.1: Identificação do forno de Box-Jenkins pela XUninet

A Figura 5.2 mostra o histograma do RMSE para as 100 execuções realizadas, o valor mais frequente no histograma foi de 0.06. O formato sugere uma concentração do erro mais próximo aos mínimos erros obtidos.

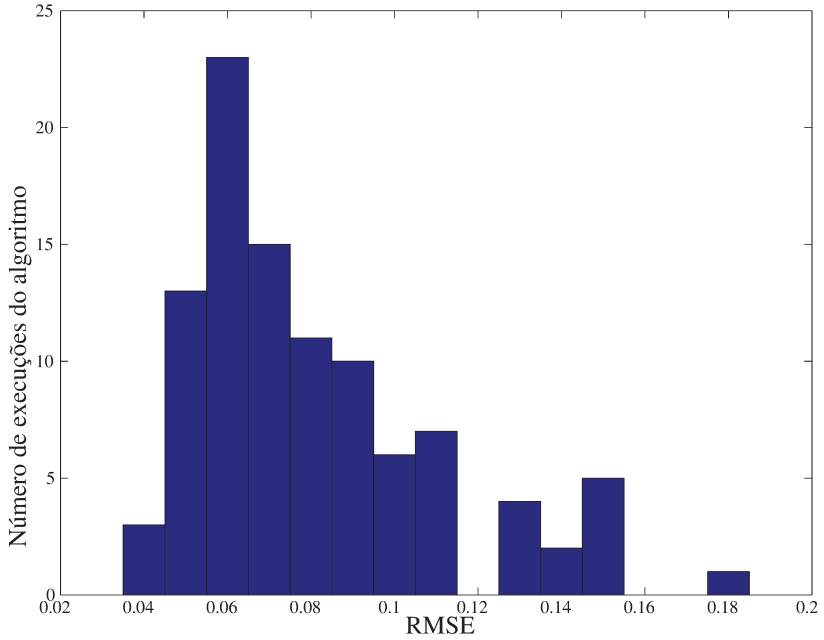


Figura 5.2: Histograma do RMSE da XUninet para Box-Jenkins

5.2.2 Aproximação de Função com Ruído

Nesta seção mostra-se a capacidade de aproximação de funções da XUninet quando os dados de aprendizagem estão contaminados por ruido. Para este propósito, utiliza-se a mesma função adotada por Delgado; Zuben e Gomide (2004):

$$F_1 : [0,1]^2 \rightarrow \Re,\ F_1(x_1, x_2) = f_1(x_1, x_2) + N[\mu, \sigma] \tag{5.4}$$

onde $f_1(x_1, x_2) = 1.9\,(1.35 + \exp(x_1)\sin(13(x_1 - 0.6)^2)\exp(-x_2)\sin(7x_2))$, $\mu = 0$ e $\sigma = 0.3$. O objetivo é treinar um modelo com dados gerados utilizando a função F_1 para obter um aproximador da função f_1.

O treinamento foi efetuado utilizando 15 pontos igualmente espaçados em $[0,1] \times [0,1]$, gerando uma grade de 15×15 pontos no universo das entradas. A Figura 5.3 mostra a função f_1 e os valores de F_1 utilizados para treinamento.

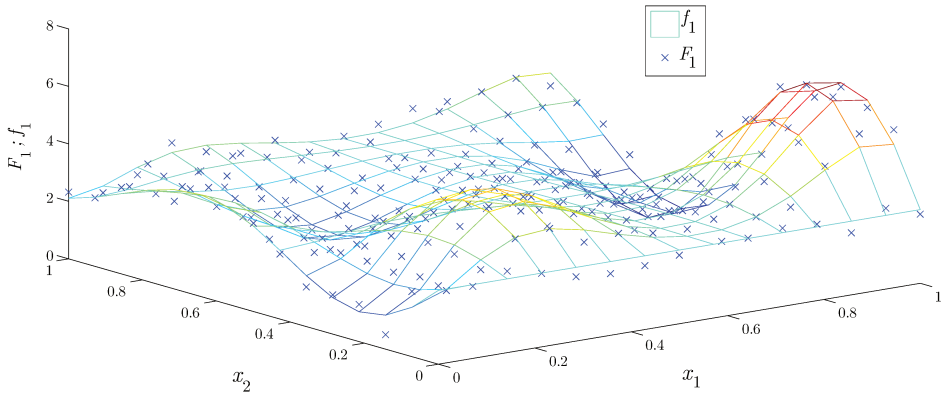


Figura 5.3: Função real f_1 e dados de treinamento gerados por F_1

Para avaliar a qualidade do modelo gerado utilizou-se neste caso, o MSE para comparar com resultados já existentes. O erro foi calculado utilizando as amostras de f_1 e a saída do modelo $\hat{y}$. O algoritmo de aprendizado foi executado 100 vezes e o melhor resultado escolhido. Cada execução durou em média $237ms$. Neste caso, a XUninet não supera os outros métodos caso seja utilizado o mesmo número de neurônios na sua camada intermediária que o número de neurônios/regras dos métodos alternativos. Com 20 neurônios na camada intermediária, a XUninet supera os métodos alternativos em termos do MSE. Contudo, em ambos casos a

XUninet supera o todos os métodos utilizados para comparação sob o ponto de vista de tempo de processamento. A Tabela 5.2 exibe os resultados e o número de regras utilizado.

Tabela 5.2: Desempenho da XUninet para a função F_1

Modelo	Referência	Neurônios/Regras	MSE
ANFIS	(Shing e Jang, 1993)	9	0.21
CoevolGFS	(Delgado; Zuben e Gomide, 2004)	8	0.13
XUninet	-	9	0.26
XUninet	-	20	0.12

A Figura 5.4 mostra a aproximação de f_1 feita pela a XUninet. Nesta figura, os valores de y correspondem aos valores reais de f_1 nos pontos x_1 e x_2 correspondentes.

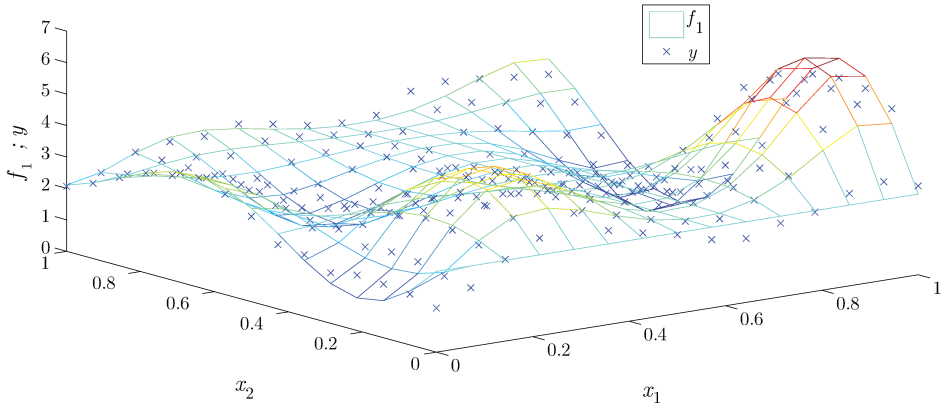


Figura 5.4: Aproximação da função f_1 pela XUninet

5.3 Resultados da eXUninet

Esta seção apresenta os resultados para a versão evolutiva da rede proposta. As duas abordagens para agrupamento serão identificadas com eXUninet+ quando eClustering+ for utilizado e eXUninet-ePL quando ePL for adotado. O fator de esquecimento λ foi fixado em 0.9. Esta escolha é justificada na Seção 5.3.6. Aqui o conjunto de dados foi percorrido sequencialmente e a saída da rede computada para a entrada correspondente. A seguir calcula-se o erro e atualiza-se o modelo, nesta ordem.

Os resultados reportados foram obtidos executando a eXUninet 100 vezes para avaliar o menor valor e a média dos erros. Foi utilizada a implementação do método DENFIS disponibi-

lizada por um dos autores do método, e a implementação do eTS de Dourado e Victor (2011) para comparação.

5.3.1 Função Seno

O objetivo deste exemplo é, através de um caso simples, testar e verificar as características da eXUninet. A função escolhida é a função seno:

$$y = sen(x) \tag{5.5}$$

Foram gerados dados entre 0 e 10π com intervalo de 0.02. O problema é aproximar a função seno. Ou seja, construir um modelo do tipo $\hat{y} = p(x)$. A Tabela 5.3 e a Figura 5.5 resumem os resultados. O algoritmo de aprendizagem extrema necessitou de 1.41 segundos em média.

Tabela 5.3: Desempenho da eXUninet na Aproximação da Função Seno

Modelo	Referência	Neurônios	RMSE mínimo	RMSE médio
DENFIS	Kasabov e Song (2002)	8	0.2530	0.2530
eXUninet+	-	6	0.0266	0.0405
eXUninet-ePL	-	6	0.0475	0.0530

Verifica-se que a eXUninet aproxima a função seno de forma mais eficiente que o DENFIS. Um teste-t para média (Kreyszig, 1970, p. 206) conclui que a média de erro da eXUninet é menor que o erro do DENFIS com 99% de confiança.

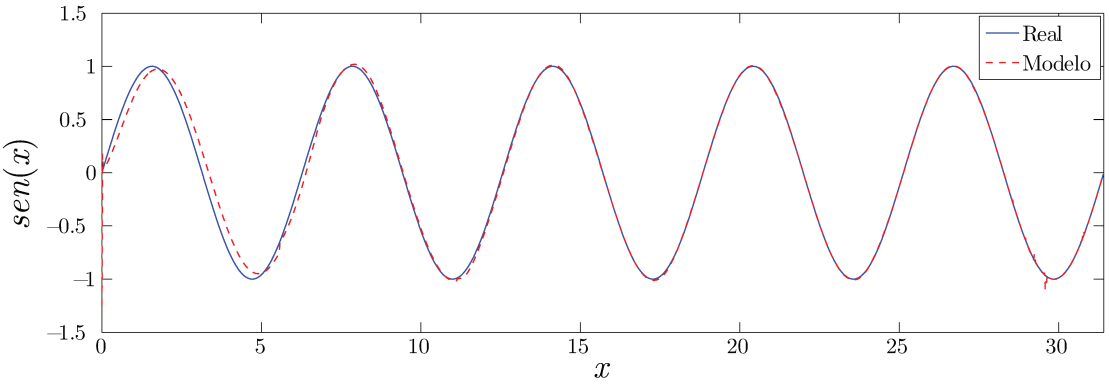


Figura 5.5: Aproximação da função seno pela eXUninet+

5.3.2 Seno Como uma Série Temporal

Séries temporais são sequências de amostras de dados que possuem um ordenamento temporal natural, pois podem representar eventos ou saídas de um sistema que ocorrem ou são amostrados num intervalo de tempo fixo. Este tipo de sequência necessita de uma análise diferente das aplicadas a problemas de aproximação de funções com entradas e saídas bem definidas. A ordem das amostras traz informações importantes dos eventos ou do sistema a ser modelado. Por isso usualmente são utilizadas amostras passadas da variável de saída para predição em instantes futuros dos valores da mesma variável.

É possível modelar o problema de estimar o valor da função seno como uma série temporal. Supondo que não é conhecida a relação entre o valor de $\sin(x)$ e x, é possível utilizar somente valores de $\sin(x)$ para a predição do valor da função em passos a frente. Para tanto pode ser modelado um preditor p do tipo:

$$\hat{y}^{t+4} = p(y^t, y^{t-4}, y^{t-8}) \tag{5.6}$$

sendo $y^t = sen(x^t)$.

Amostras com intervalo de 0.02 entre 0 e 10π foram utilizadas para a simulação. Os erros obtidos estão apresentados na Tabela 5.4 e a Figura 5.6 exibe o resultado da aproximação com a eXUninet+.

Tabela 5.4: Desempenho da eXUninet na Previsão da Função Seno como Série Temporal

Modelo	Referência	Neurônios	RMSE mínimo	RMSE médio
DENFIS	Kasabov e Song (2002)	10	0.1002	0.1002
eXUninet+	-	8	0.0227	0.0428
eXUninet-ePL	-	8	0.0154	0.0219

Verifica-se que a eXUninet obteve melhor desempenho comparado ao DENFIS. Nas próximas seções serão utilizados exemplos mais complexos e resultados de outros métodos serão apresentados para comparação.

5.3.3 Série Temporal de Mackey-Glass

A série temporal de Mackey e Glass (1977) é um exemplo utilizado amplamente na literatura, sendo uma referência para teste e avaliação de algoritmos e métodos de previsão de séries

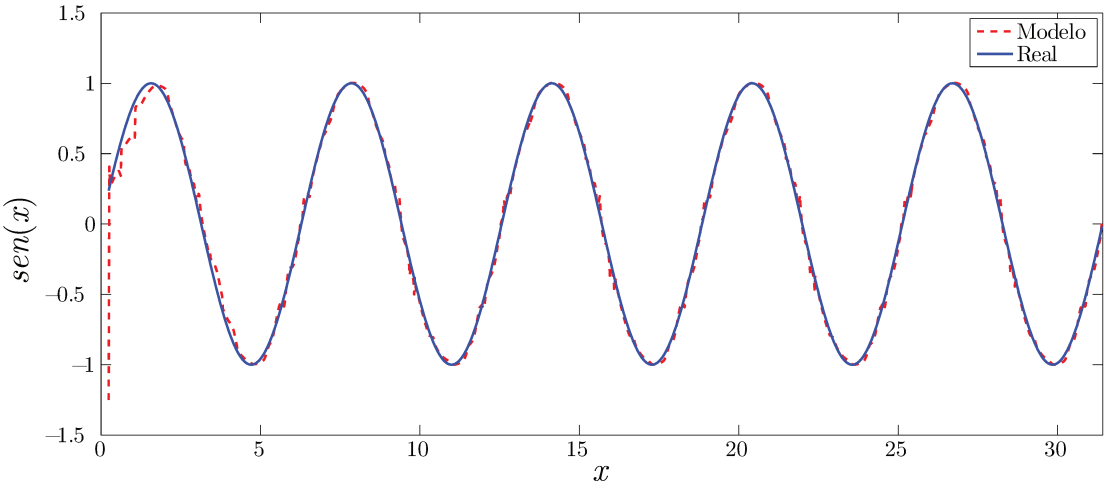


Figura 5.6: Resultado da eXUninet+ para a função seno como série temporal

temporais. Os valores da série são gerados utilizando:

$$\frac{dx}{dt} = \frac{Ax^{t-\tau}}{1 + (x^{t-\tau})^C} - Bx^t \ , \ A, B, C > 0 \tag{5.7}$$

Comportamentos semiperiódicos ou caóticos dependem dos valores dos parâmetros escolhidos. Estudos adotam: $A = 0.2$, $B = 0.1$, $C = 10$ e $\tau = 17$ (Angelov e Zhou, 2006; Kasabov e Song, 2002). Para estes valores, o sistema apresenta um comportamento semiperiódico, o que gera ciclos que nunca se repetem, ou seja, o comportamento é caótico. As principais séries temporais de relevância prática são de natureza caótica, elevando a importância do estudo desta classe de problemas. Além disso, comportamento complexo obtido não é facilmente modelado por métodos clássicos de identificação de sistemas.

Os dados utilizados nesta seção correspondem a 3200 amostras geradas via (5.7) com passo de integração igual a 0.1. As amostras no intervalo $201 \leq t \leq 3200$ foram utilizadas para treinamento e teste simultaneamente. O objetivo é prever o valor de x^{t+85} passos a frente, ou seja:

$$\hat{x}^{t+85} = p(x^t, x^{t-6}, x^{t-12}, x^{t-18}) \tag{5.8}$$

Resultados comparando o melhor e a média dos erros com abordagens alternativas são resu-

midos na Tabela 5.5.

Tabela 5.5: Desempenho da eXUninet para a Previsão da Série de Mackey-Glass

Modelo	Referência	Neurônios/Regras	RMSE min.	RMSE médio
eTS	Angelov e Zhou (2006)	24	0.0779	0.0779
DENFIS	Kasabov e Song (2002)	25	0.0730	0.0730
FBeM	Leite et al. (2011)	26	0.0968	0.0968
eXUninet+	-	27	0.0442	0.0579
eXUninet-ePL	-	27	0.0505	0.0597

Para este caso, o erro médio da eXUninet é menor que os outros métodos com 99% de confiança utilizando o teste-t para a média (Kreyszig, 1970, p. 206). O desvio padrão do erro para a eXUninet+ foi de 0.0074 e de 0.0046 para a eXUninet-ePL. O treinamento de uma instância da eXUninet+ requer em média $6.87s$, o que representa uma média de $2.1ms$ por amostra. Um tempo menor de processamento é importante de ser observado pois, em alguns casos, sistemas evolutivos necessitam operar em tempo real. A Figura 5.7 exibe os resultados da eXUninet+ para as primeiras 500 amostras de dados.

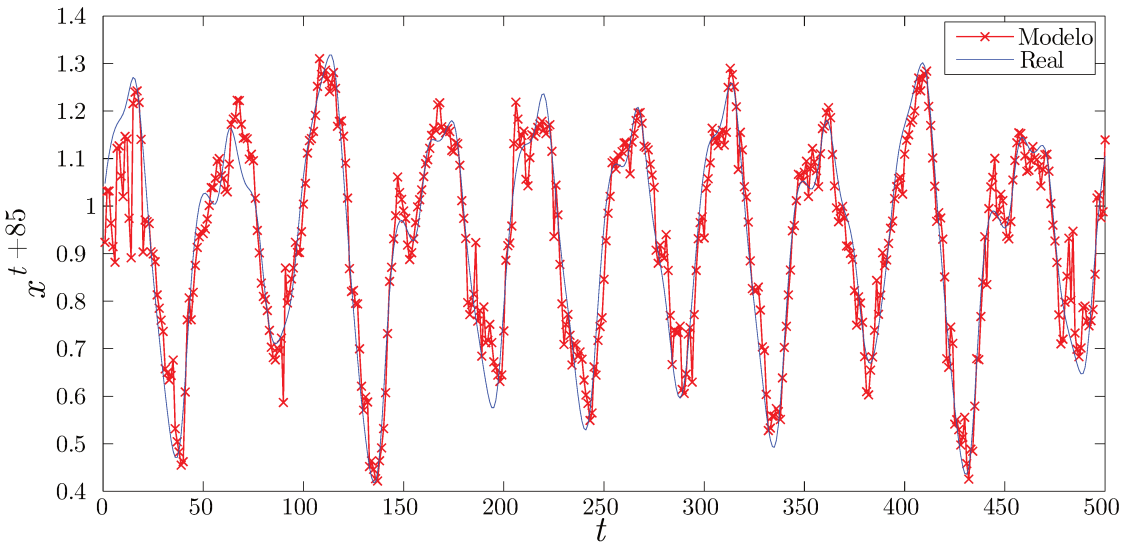


Figura 5.7: Previsão da série de Mackey-Glass pela eXUninet+

5.3.4 Forno à Gás de Box-Jenkins

O problema de identificação do forno à gás de Box-Jenkins, apresentado na Seção 5.2.1, também foi utilizado para avaliar o desempenho da eXUninet supondo um modelo da mesma

forma:

$$\hat{y}^t = p(y^{t-1}, x^{t-4}) \tag{5.9}$$

A Tabela 5.6 resume os resultados.

Tabela 5.6: Desempenho da eXUninet na Identificação do Forno de Box-Jenkins

Modelo	Referência	Neurônios/Regras	RMSE min.	RMSE médio
eTS	Angelov e Zhou (2006)	12	0.0796	0.0796
DENFIS	Kasabov e Song (2002)	12	0.0190	0.0190
FBeM	Leite et al. (2011)	3	0.0421	0.0421
eXUninet+	-	11	0.0588	0.0694
eXUninet-ePL	-	12	0.0523	0.0612

Neste caso, a eXUninet obteve melhor resultado do que o eTS com 99% de confiança para o teste-t da média, mas produz um erro maior do que os outros métodos. O treinamento de uma instância levou em média $318ms$, correspondendo a $1.1ms$ por amostra. A Figura 5.8 exibe os resultados da eXUninet-ePL.

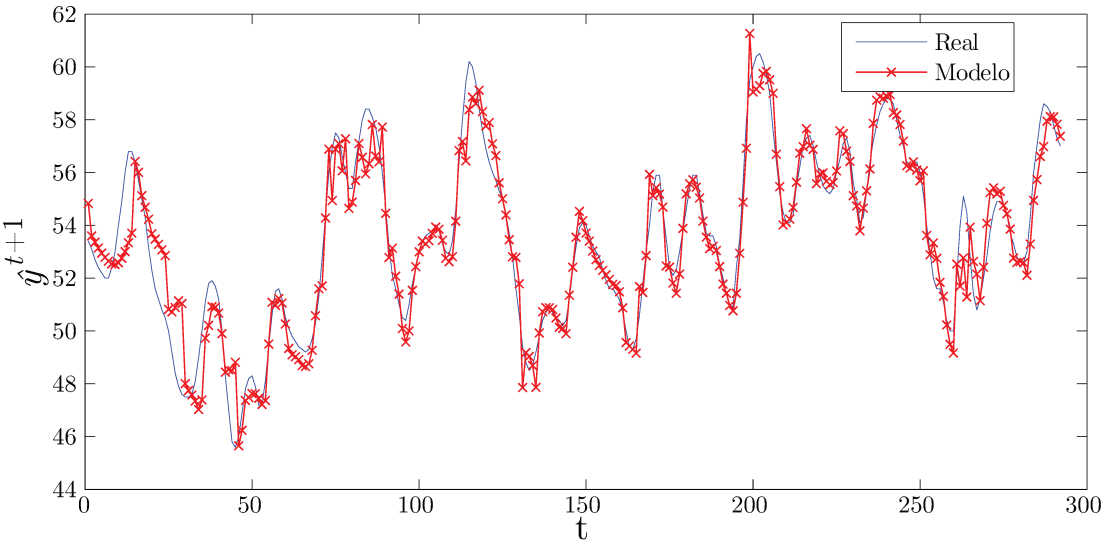


Figura 5.8: Identificação do forno à gás de Box-Jenkins pela eXUninet+

5.3.5 Série Temporal com *Concept Drift* e *Shift*

Narendra e Parthasarathy (1990) propõe *benchmarks* para modelagem não linear de proces-

sos visando a verificação da capacidade de aproximação de redes neurais. Um dos problemas propostos sugere gerar dados de aprendizagem seguindo:

$$y^{t+1} = \frac{ay^t}{1 + b(y^t)^2} + c(u^t)^3 \tag{5.10}$$

onde $u^t = sen(2\pi t/25) + sen(2\pi t/10)$, $a = b = c = 1$ e $y^0 = 0$.

Dados foram gerados para $t = 0, \ldots, 400$. Para simular uma mudança abrupta no comportamento do sistema, caracterizando um *concept shift*, o valor dos parâmetros a, b e c são alterados no instante $t = 200$ para: $a = 3.5$, $b = 0.8$ e $c = 1.5$. A tarefa da rede consiste em modelar um preditor p para realizar a previsão um passo a frente dado 4 passos anteriores, como segue:

$$\hat{y}^t = p(y^{t-1}, y^{t-2}, y^{t-3}, y^{t-4}) \tag{5.11}$$

A Tabela 5.7 exibe os erros obtidos pela eXUninet e modelos utilizados para comparação. A Figura 5.9 mostra o comportamento da série temporal e o efeito introduzido, juntamente com o resultado da aproximação.

Tabela 5.7: Desempenho da eXUninet para a Série Temporal com *Concept Shift*

Modelo	Referência	Neurônios/Regras	RMSE min.	RMSE médio
eTS	Angelov e Zhou (2006)	8	0.0993	0.0993
DENFIS	Kasabov e Song (2002)	12	0.0562	0.0562
eXUninet+	-	21	0.0725	0.0913
eXUninet-ePL	-	21	0.0781	0.0935

Nota-se que a eXUninet não conseguiu resultados melhores que o DENFIS apesar de ter utilizado um número maior de neurônios. Contudo quando comparada com o eTS, foi obtém-se menor erro médio com 99% de confiança do teste-t para a média. O treinamento durou $513ms$.

O segundo teste efetuado consiste em variar os parâmetros a, b e c da mesma forma descrita no teste anterior, mas agora é somada uma forma de onda senoidal representando uma variação gradual e sistemática no comportamento sistema, caracterizando *concept drift*. A variação dos parâmetros é feita de acordo com as expressões (5.12-5.15).

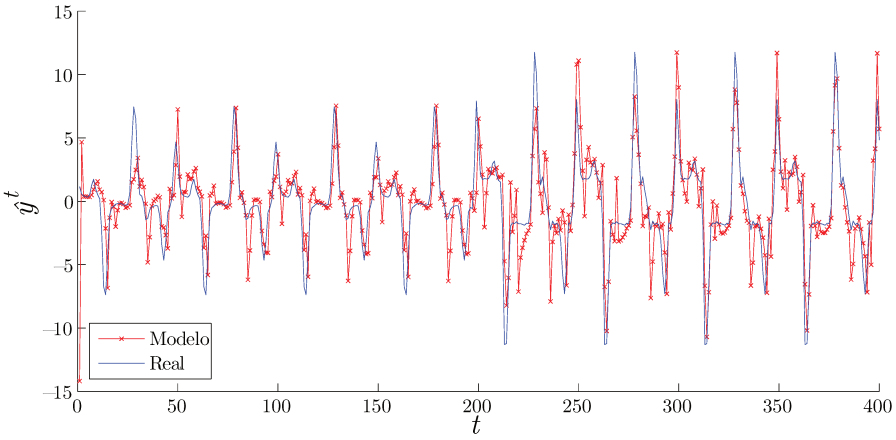


Figura 5.9: Série temporal com *concept shift* e previsão da eXUninet+

$$s = 0.5sen(0.2t) \tag{5.12}$$

$$a = s + \begin{cases} 1 & \text{se } t < 200 \\ 3.5 & \text{caso contrário.} \end{cases} \tag{5.13}$$

$$b = s + \begin{cases} 1 & \text{se } t < 200 \\ 0.8 & \text{caso contrário.} \end{cases} \tag{5.14}$$

$$c = s + \begin{cases} 1 & \text{se } t < 200 \\ 1.5 & \text{caso contrário.} \end{cases} \tag{5.15}$$

Os valores e variações de cada parâmetro são demonstrados na Figura 5.10. Para este caso

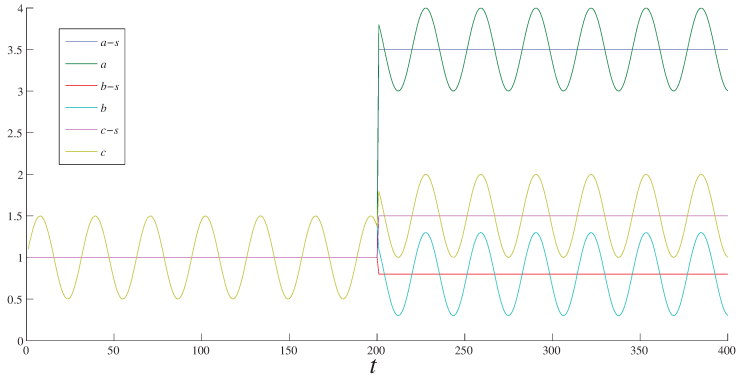


Figura 5.10: Variação dos parâmetros para teste com *concept drift*

foram obtidos os resultados da Tabela 5.8. Aqui a eXUninet obteve erro próximo ao eTS e

Tabela 5.8: Desempenho da eXUninet para a Série Temporal com *Concept Drift*

Modelo	Referência	Regras	RMSE min.	RMSE médio
eTS	Angelov e Zhou (2006)	11	0.0856	0.0856
DENFIS	Kasabov e Song (2002)	16	0.0482	0.0482
eXUninet+	-	16	0.0752	0.0864
eXUninet-ePL	-	15	0.0762	0.0868

não houve diferença significativa do ponto de vista de um teste-t para a média com 99% de confiança. Já o DENFIS obteve melhor erro com o mesmo número de regras. A Figura 5.11 mostra as previsões feitas pela eXUninet+.

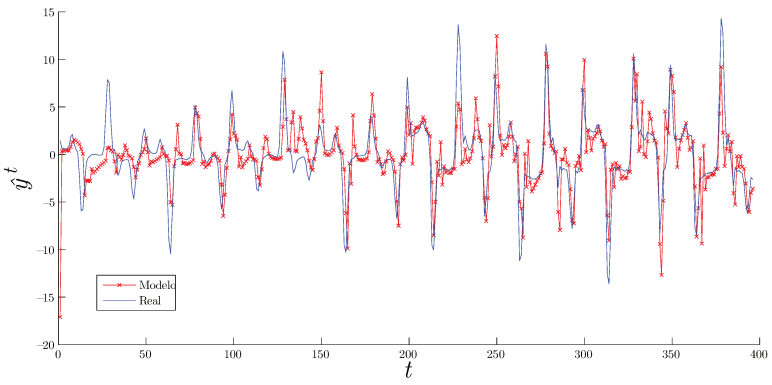


Figura 5.11: Série temporal com *concept drift* e previsão da eXUninet+

5.3.6 Análise de Adaptabilidade e Sobrevivência

Uma questão que surge na modelagem de sistemas evolutivos é o compromisso entre a adaptabilidade a curto prazo e a sobrevivência a longo prazo. Idealmente o modelo deve se adaptar rapidamente a mudanças no ambiente, sem provocar a ruptura do modelo corrente. Ou seja, detectar *concept drifts* e *shifts* e tomar ações para corrigir os seus parâmetros e estrutura, alterando-os de forma gradual.

O algoritmo RLS com fator de esquecimento λ estipula por quanto tempo uma amostra de dado contribui para a estimativa atual dos pesos da rede. Combinando a ideia do fator de esquecimento com a expansão e contração do número de regras, é possível controlar a adaptabilidade do modelo. Sistemas evolutivos necessitam de uma capacidade de adaptação para lidar com ambientes em constante mudança e reduzindo o valor do fator de esquecimento, o modelo fica

mais suscetível a detectar *concept drifts* e *shifts*. Portanto, é possível avaliar o quão adaptável é um modelo e quanto conhecimento sobre o sistema sobrevive quando a adaptabilidade é variada.

Para demonstrar o controle sobre a adaptabilidade, foi realizado um teste com a eXUninet-ePL utilizando $\lambda = 0.9$, e com os parâmetros do agrupamento ePL ajustados para gerar 12 regras. Para comparação foi realizada outra simulação com $\lambda = 0.995$ e ajustes para gerar 27 regras. Ambos utilizaram o conjunto de dados Mackey-Glass como na Seção 5.3.3, mas utilizando somente as 500 amostras iniciais. O ajuste de parâmetros dos modelos foi interrompido em $t = 250$ e nos 250 instantes restantes foi utilizado o modelo somente para estimar a saída. Foi calculado o RMSE-treino para os 250 primeiros e o RMSE-teste para os últimos 250. Os erros são comparados na Tabela 5.9.

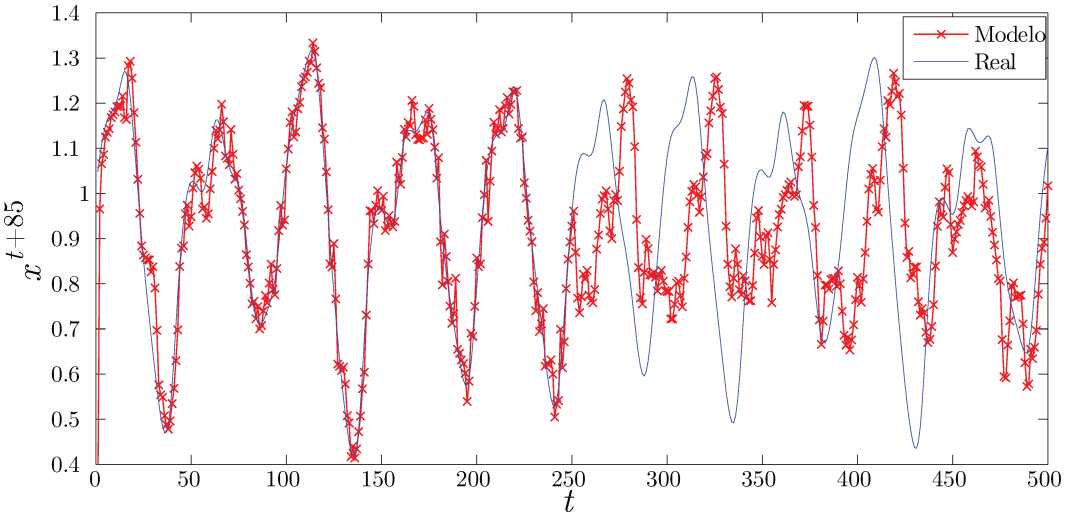
Tabela 5.9: Teste de Adaptabilidade vs. Sobrevivência

λ	Regras	RMSE-treino	RMSE-teste
0.995	27	0.115	0.123
0.900	12	0.074	0.190

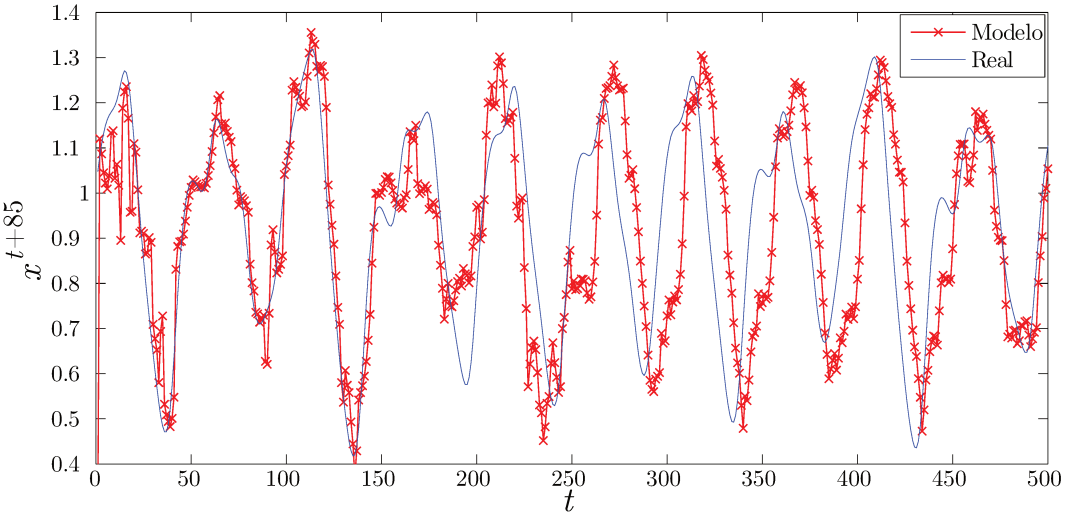
A Figura 5.12 ilustra o *trade-off* entre adaptabilidade e sobrevivência no caso dos testes realizados nesta seção. Na Figura 5.12(a) é observado o melhor desempenho da rede com 12 regras até o instante 250. Após esse instante, o modelo de 27 regras da Figura 5.12(b) possui melhor desempenho pois possui mais regras e informações do sistema sobrevivem nos seus parâmetros.

5.4 Resumo

Os resultados sugerem que o algoritmo de treinamento da rede neural *fuzzy* com aprendizagem extrema proposto é simples, rápido e competitivo. O modelo representado pela rede neural fuzzzy é capaz de adaptar-se a mudanças de conceito sem ruptura, ou seja, o modelo fornece saídas próximas às do sistema real durante toda a sua execução. O controle do nível de adaptabilidade também se mostrou importante, pois é mais uma ferramenta de modelagem para sistemas dinâmicos com *concept drift*.



(a) $\lambda = 0.900$



(b) $\lambda = 0.995$

Figura 5.12: Aprendizado terminado em $t = 250$ para valores de λ distintos.

Capítulo 6

Conclusão

A principal contribuição do presente trabalho é a proposta de algoritmos de aprendizado baseados em agrupamento e aprendizagem extrema para redes neurais *fuzzy* baseadas em uninorma. Além do aprendizado ser rápido e apresentar resultados competitivos, a interpretabilidade do modelo é mantida. Outras contribuições foram: a prova de aproximação universal da versão estática com treinamento em batelada da rede e discussão da adaptabilidade versus sobrevivência dos modelos evolutivos.

Os resultados de simulações comprovaram que em certos casos a eXUninet tem um melhor desempenho devido a flexibilidade que as uninormas oferecem. O aprendizado extremo mostrou-se eficiente para o treinamento da eXUninet. Aliado aos algoritmos de agrupamento dos modelos ePL e eTS, foi possível realizar o treinamento de redes baseadas em uninormas, de forma recursiva e com um controle da adaptabilidade. Esta combinação de métodos provê a alta capacidade de adaptação da estrutura do modelo e o ajuste dos parâmetros utilizando fluxo de dados, caracterizando um modelo evolutivo.

Trabalhos futuros deverão explorar a prova de aproximação universal para a rede na sua versão evolutiva. Outra contribuição interessante seria a utilização de nullnormas para construir os neurônios da rede, pois nullnormas também fornecem alta flexibilidade e generalizam t-normas e s-normas. Testes em conjuntos de dados que apresentam *concept drifts* precisam ser mais explorados para avaliar melhor a capacidade de aprendizado do modelo, bem como a realização de testes estatísticos mais avançados considerando variações nos parâmetros dos modelos.

Outra possibilidade é a extensão do modelo para tratar recorrências nas conexões dos neurônios. Utilizar métodos de agrupamento recursivo como o GK recursivo (Dovzan e Skrjanc, 2011) pode ser interessante em comparações futuras de algoritmos evolutivos. Analisar a interpretabilidade dos *unineurons* também é oportuno, pois o comportamento intermediário desses

neurônios, quando os elementos neutros de suas uninormas estão entre 0 e 1 ainda não foi detalhado na literatura.

Bibliografia

Angelov, P. (1999), Evolving fuzzy rule-based models, Proc. 8th IFSA World Congress, Taipei, Taiwan, Vol. 1, pp. 19 –23.

Angelov, P. (2002), Evolving Rule-Based Models: A Tool for Design of Flexible Adaptive Systems, Physica-Verlag, Heidelberg, Alemanha.

Angelov, P. (2010), Evolving takagi-sugeno fuzzy systems from streaming data, P. Angelov; D. Filev e N. Kasabov, (Eds.), Evolving Intelligent Systems, John Wiley & Sons, Inc., Hoboken, NJ, EUA, pp. 21–50.

Angelov, P. e Filev, D. (2004), An approach to online identification of takagi-sugeno fuzzy models, IEEE Trans. on Systems, Man, and Cybernetics, Part B: Cybernetics **34**(1), 484 – 498.

Angelov, P. e Filev, D. (2005), SimpLets: a simplified method for learning evolving takagi-sugeno fuzzy models, Proc. of the 14th IEEE International Conference on Fuzzy Systems, 2005., Reno, NV, EUA, pp. 1068 –1073.

Angelov, P. e Zhou, X. (2006), Evolving fuzzy systems from data streams in real-time, 2006 International Symposium on Evolving Fuzzy Systems, Lake District, Reino Unido, pp. 29 –35.

Angelov, P. e Zhou, X. (2008), Evolving fuzzy-rule-based classifiers from data streams., IEEE Trans. on Fuzzy Systems **16**(6), 1462–1475.

Angelov, P.; Filev, D. e Kasabov, N. (2010), Evolving Intelligent Systems: Methodology and Applications, John Wiley & Sons, Hoboken, NJ, EUA.

Ballini, R. e Gomide, F. (2002), Learning in recurrent, hybrid neurofuzzy networks, Proc. of the 2002 IEEE International Conference on Fuzzy Systems, Vol. 1, Havaí, EUA, pp. 785 –790.

Bezdek, J.; Ehrlich, R. e Full, W. (1984), FCM: The fuzzy c-means clustering algorithm, Computers and Geosciences **10**(2-3), 191–203.

Box, G. e Jenkins, G. (1970), Time Series Analysis: Forecasting and Control, Holden-Day, San Francisco, CA, EUA.

Caminhas, W.; Tavares, H.; Gomide, F. e Pedrycz, W. (1999), Fuzzy set based neural networks: Structure, learning and application., JACIII **3**(3), 151–157.

Delgado, M.; Zuben, F. e Gomide, F. (2004), Coevolutionary genetic fuzzy systems: a hierarchical collaborative approach, Fuzzy Sets and Systems **141**(1), 89 – 106.

Dourado, A. e Victor, J. (2011), Evolving fuzzy systems from data: the computational eFS-Lab(oratory), World Conference on Soft Computing, San Francisco, CA, EUA, pp. 183–189.

Dovzan, D. e Skrjanc, I. (2011), Recursive clustering based on a gustafson-kessel algorithm, Evolving Systems **2**, 15–24.

Filev, D. e Angelov, P. (2007), Algorithms for Real-time Clustering and Generation of Rules from Data, John Wiley & Sons, Hoboken, NJ, EUA, pp. 354–369.

Gustafson, D. e Kessel, W. (1978), Fuzzy clustering with a fuzzy covariance matrix, IEEE Conference on Decision and Control, Vol. 17, San Diego, CA, EUA, pp. 761 –766.

Hayashi, I.; Nomura, H.; Yamasaki, H. e Wakami, N. (1992), Construction of fuzzy inference rules by NDF and NDFL, International Journal of Approximate Reasoning **6**(2), 241 – 266.

Hell, M.; Costa, P. e Gomide, F. (2008), Hybrid neurofuzzy computing with nullneurons, IEEE International Joint Conference on Neural Networks., Hong Kong, China, pp. 3653 –3659.

Hell, M.; Costa, P. e Gomide, F. (2008), Participatory learning in power transformers thermal modeling, IEEE Trans. on Power Delivery **23**(4), 2058 –2067.

Hell, M.; Gomide, F.; Ballini, R. e Costa, P. (2009), Uninetworks in time series forecasting, Proc. of the 2009 Annual Meeting of the North American Fuzzy Information Processing Society, Cincinnati, OH, EUA, pp. 1 –6.

Hornik, K.; Stinchcombe, M. e White, H. (1989), Multilayer feedforward networks are universal approximators, Neural Networks **2**(5), 359 – 366.

Huang, G. e Chen, L. (2007), Convex incremental extreme learning machine, Neurocomputing **70**(16-18), 3056 – 3062.

Huang, G. e Siew, C. (2004), Extreme learning machine: RBF network case, Proc. 8th Int. Conf. Control, Autom., Robot., Vision, ICARCV 2004, Kunning, China, pp. 1029–1036.

Huang, G.; Li, M.; Chen, L. e Siew, C. (2008), Incremental extreme learning machine with fully complex hidden nodes, Neurocomputing **71**(4-6), 576–583.

Huang, G.; Liang, N.; Rong, H.; Saratchandran, P. e Sundararajan, N. (2005), On-line sequential extreme learning machine, M. H. Hamza, (Ed.), Computational Intelligence 05, IASTED/ACTA Press, Calgary, Alberta, Canadá, pp. 232–237.

Huang, G.; Zhu, Q. e Siew, C. (2004), Extreme learning machine: a new learning scheme of feedforward neural networks, Proc. of the IEEE International Joint Conference on Neural Networks, Vol. 2, Budapeste, Hungria, pp. 985 – 990.

Hush, D. e Horne, B. (1993), Progress in supervised neural networks, IEEE Signal Processing Magazine **10**(1), 8 –39.

Jang, J. e Sun, C. (1997), Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence, Prentice-Hall, Inc., Upper Saddle River, NJ, EUA.

Juang, C. e Lin, C. (1998), An online self-constructing neural fuzzy inference network and its applications, IEEE Trans. on Fuzzy Systems **6**(1), 12–32.

Kasabov, N. (1996), Adaptable neuro production systems, Neurocomputing **13**(2-4), 95 – 117.

Kasabov, N. (1998), ECOS: Evolving connectionist systems and the eco learning paradigm, ICONIP'98, Kitakyushu, Japão, pp. 123 –128.

Kasabov, N. (2001), Evolving fuzzy neural networks for supervised/unsupervised online knowledge-based learning, IEEE Trans. on Systems, Man, and Cybernetics, Part B: Cybernetics **31**(6), 902 –918.

Kasabov, N. (2007), Evolving Connectionist Systems: The Knowledge Engineering Approach, Evolving Connectionist Systems, Springer, Londres, Reino Unido.

Kasabov, N. e Song, Q. (2002), DENFIS: dynamic evolving neural-fuzzy inference system and its application for time-series prediction, IEEE Trans. on Fuzzy Systems **10**(2), 144 –154.

Keller, J.; Krishnapuram, R. e Rhee, F. (1992), Evidence aggregation networks for fuzzy logic inference, IEEE Trans. on Neural Networks **3**(5), 761–769.

Keller, J.; Yager, R. e Tahani, H. (1992), Neural network implementation of fuzzy logic, Fuzzy Sets and Systems **45**(1), 1 – 12.

Kohonen, T. (1982), Self-organized formation of topologically correct feature maps, Biological Cybernetics **43**, 59–69.

Kreyszig, E. (1970), Introductory Mathematical Statistics: Principles and Methods, John Wiley & Sons, Hoboken, NJ, EUA.

Lee, S. e Lee, E. (1975), Fuzzy neural networks, Mathematical Biosciences **23**(1-2), 151–177.

Leite, D.; Gomide, F.; Ballini, R. e Costa, P. (2011), Fuzzy granular evolving modeling for time series prediction, Proc. of IEEE International Conference on Fuzzy Systems, Taipei, Taiwan, pp. 2794–2801.

Lemos, A.; Caminhas, W. e Gomide, F. (2010), New uninorm-based neuron model and fuzzy neural networks, Proc. of the 2009 Annual Meeting of the North American Fuzzy Information Processing Society, Toronto, Canadá, pp. 1 –6.

Lemos, A.; Caminhas, W. e Gomide, F. (2011), Fuzzy evolving linear regression trees, Evolving Systems **2**, 1–14.

Lemos, A.; Caminhas, W. e Gomide, F. (2011), Multivariable gaussian evolving fuzzy modeling system, IEEE Trans. on Fuzzy Systems **19**(1), 91–104.

Lemos, A.; Kreinovich, V.; Caminhas, W. e Gomide, F. (2011), Universal approximation with uninorm-based fuzzy neural networks, Proc. of the 2009 Annual Meeting of the North American Fuzzy Information Processing Society, El Paso, EUA, pp. 1 –6.

Leng, G.; McGinnity, T. e Prasad, G. (2005), An approach for on-line extraction of fuzzy rules using a self-organising fuzzy neural network, Fuzzy Sets and Systems **150**(2), 211 – 243.

Lima, E.; Gomide, F. e Ballini, R. (2006), Participatory evolving fuzzy modeling, International Symposium on Evolving Fuzzy Systems, Lake District, Reino Unido, pp. 36 –41.

Lin, C. e Lee, C. (1991), Neural-network-based fuzzy logic control and decision system, IEEE Trans. on Computers **40**(12), 1320 –1336.

Lin, C. e Lee, C. (1996), Neural Fuzzy Systems: A Neuro-Fuzzy Synergism to Intelligent Systems, Prentice-Hall, Upper Saddle River, NJ, USA.

Liu, Q.; He, Q. e Shi, Z. (2008), Extreme support vector machine classifier, Proc. of the 12th Pacific-Asia conference on Advances in knowledge discovery and data mining, PAKDD'08, Springer-Verlag, Heidelberg, Alemanha, pp. 222–233.

Lughofer, E. (2008 a), Extensions of vector quantization for incremental clustering, Pattern Recognition **41**(3), 995–1011.

Lughofer, E. (2008 b), Flexfis: A robust incremental learning approach for evolving takagi-sugeno fuzzy models, IEEE Trans. on Fuzzy Systems **16**(6), 1393–1410.

Machado, R. e Rocha, A. (1992), Evolutive fuzzy neural networks, IEEE International Conference on Fuzzy Systems, San Diego, CA, EUA, pp. 493 –500.

Mackey, M. e Glass, L. (1977), Oscillation and Chaos in Physiological Control Systems, Science **197**(4300), 287–289.

Minhas, R.; Mohammed, A. e Wu, Q. (2012), Incremental learning in human action recognition based on snippets, IEEE Trans. on Circuits and Systems for Video Technology **22**(11), 1529 –1541.

Mitra, S. e Hayashi, Y. (2000), Neuro-fuzzy rule generation: survey in soft computing framework, IEEE Trans. on Neural Networks **11**(3), 748 –768.

Narendra, K. e Parthasarathy, K. (1990), Identification and control of dynamical systems using neural networks, IEEE Trans. on Neural Networks **1**(1), 4–27.

Park, J. e Sandberg, I. (1991), Universal approximation using radial-basis-function networks, Neural Computation **3**(2), 246–257.

Pedrycz, W. (2006), Logic-based fuzzy neurocomputing with unineurons, IEEE Trans. on Fuzzy Systems **14**(6), 860 –873.

Pedrycz, W. e Rocha, A. (1993), Fuzzy-set based models of neurons and knowledge-based networks, IEEE Trans. on Fuzzy Systems **1**(4), 254 –266.

Perfilieva, I. (2006), Fuzzy transforms: Theory and applications, Fuzzy Sets and Systems **157**(8), 993 – 1023.

Rong, H.; Sundararajan, N.; Huang, G. e Saratchandran, P. (2006), Sequential adaptive fuzzy inference system (safis) for nonlinear system identification and prediction, Fuzzy Sets and Systems **157**(9), 1260–1275.

Rubio, J. (2009), SOFMLS: online self-organizing fuzzy modified least-squares network, IEEE Trans. on Fuzzy Systems **17**(6), 1296–1309.

Samet, S. e Miri, A. (2012), Privacy-preserving back-propagation and extreme learning machine algorithms, Data & Knowledge Engineering **79-80**, 40–61.

Sherman, J. e Morrison, W. (1950), Adjustment of an inverse matrix corresponding to a change in one element of a given matrix, The Annals of Mathematical Statistics **21**(1), 124–127.

Shing, J. e Jang, R. (1993), ANFIS: Adaptive-network-based fuzzy inference system, IEEE Trans. on Systems, Man, and Cybernetics **23**, 665–685.

Song, Y.; Crowcroft, J. e Zhang, J. (2012), Automatic epileptic seizure detection in eegs based on optimized sample entropy and extreme learning machine, Journal of Neuroscience Methods **210**(2), 132 – 146.

Takagi, H.; Suzuki, N.; Koda, T. e Kojima, Y. (1992), Neural networks designed on approximate reasoning architecture and their applications, IEEE Trans. on Neural Networks **3**(5), 752 – 760.

Wang, L.-X. (1993), Solving fuzzy relational equations through network training, Second IEEE International Conference on Fuzzy Systems, San Francisco, CA, EUA, pp. 956 –960 vol.2.

Wang, S.; Schlobach, S. e Klein, M. (2010), What is concept drift and how to measure it?, P. Cimiano e H. Pinto, (Eds.), Knowledge Engineering and Management by the Masses, Vol. 6317, Springer, Berlin Heidelberg, pp. 241–256.

Xu, Y.; Dong, Z.; Zhao, J.; Zhang, P. e Wong, K. (2012), A reliable intelligent system for real-time dynamic security assessment of power systems, IEEE Trans. on Power Systems **27**(3), 1253 –1263.

Yager, R. e Filev, D. (1994), Approximate clustering via the mountain method, IEEE Trans. on Systems, Man and Cybernetics **24**(8), 1279 –1284.

Yager, R. e Rybalov, A. (1996), Uninorm aggregation operators, Fuzzy Sets and Systems **80**(1), 111–120.

Yen, J.; Wang, L. e Gillespie, C. (1998), Improving the interpretability of TSK fuzzy models by combining global learning and local learning, IEEE Transactions on Fuzzy Systems **6**(4), 530 –537.