UNIVERSIDADE ESTADUAL DE CAMPINAS
Faculdade de Engenharia Elétrica e de Computação

Alisson da Silva Porto

Modelagem Nebulosa Evolutiva Min–Max

CAMPINAS

2018

Alisson da Silva Porto

Modelagem Nebulosa Evolutiva Min–Max

Dissertação apresentada à Faculdade de Engenharia Elétrica e de Computação da Universidade Estadual de Campinas como parte dos requisitos exigidos para a obtenção do título de Mestre em Engenharia Elétrica, na Área de Automação.

Orientador: Prof. Dr. Fernando Antônio Campos Gomide

Este exemplar corresponde à versão final da dissertação defendida pelo aluno Alisson da Silva Porto, e orientada pela Prof. Dr. Fernando Antônio Campos Gomide

CAMPINAS

2018

Agência(s) de fomento e nº(s) de processo(s): CNPq, 159678/2015-3

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca da Área de Engenharia e Arquitetura
Luciana Pietrosanto Milla - CRB 8/8129

P838m

Porto, Alisson da Silva, 1991-
Modelagem nebulosa evolutiva min-max / Alisson da Silva Porto. – Campinas, SP : [s.n.], 2018.

Orientador: Fernando Antônio Campos Gomide.
Dissertação (mestrado) – Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Sistemas nebulosos. 2. Algoritmos evolutivos. 3. Sistemas de computação adaptativos. 4. Modelos de regressão (Estatística). 5. Previsão de séries temporais. I. Gomide, Fernando Antônio Campos, 1951-. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

Informações para Biblioteca Digital

Título em outro idioma: Evolving fuzzy min-max modeling
Palavras-chave em inglês:
Fuzzy Systems
Evolving algorithms
Adaptive computing systems
Regression models (Statistic)
Time series forecasting
Área de concentração: Automação
Titulação: Mestre em Engenharia Elétrica
Banca examinadora:
Fernando Antônio Campos Gomide [Orientador]
Walmir Matos Caminhas
Levy Boccato
Data de defesa: 28-09-2018
Programa de Pós-Graduação: Engenharia Elétrica

COMISSÃO JULGADORA - DISSERTAÇÃO DE MESTRADO

Candidato: Alisson da Silva Porto RA: 180570
Data da Defesa: 28 de setembro de 2018
Título da Tese: Modelagem Nebulosa Evolutiva Min–Max

Prof. Dr. Fernando Antônio Campos Gomide (FEEC/UNICAMP) - Presidente
Prof. Dr. Walmir Matos Caminhas (UFMG) - Membro Titular
Prof. Dr. Levy Boccato (FEEC/UNICAMP) - Membro Titular

Ata de defesa, com as respectivas assinaturas dos membros da Comissão Julgadora, encontra-se no processo de vida acadêmica do aluno.

Dedico este trabalho aos meus pais,
Delmar Porto Malheiros e Maria Zélia da Silva Porto.

Agradecimentos

À minha família, em especial meus pais Maria Zélia da Silva Porto e Delmar Porto Malheiros, pelo carinho e pelo apoio incondicional.

À Flávia, por me incentivar e apoiar inúmeras vezes nesse período.

Ao professor Fernando Gomide, por me aceitar no programa de mestrado, além da paciência e dedicação durante a elaboração deste trabalho.

Aos colegas da república estudantil onde morei e do Laboratório de Computação e Automação Industrial, pela amizade e companheirismo nesse período do mestrado.

Aos professores e funcionários da Faculdade de Engenharia Elétrica e Computação, em especial ao professor Fernando Von Zuben pelo excelente curso de Redes Neurais Artificiais.

Ao CNPq pelo apoio financeiro.

Resumo

Métodos nebulosos evolutivos são uma alternativa para modelar sistemas não-estacionários usando fluxos de dados. Este trabalho propõe um algoritmo evolutivo Min-Max para modelar sistemas com bases de regras nebulosas funcionais. O algoritmo processa um fluxo de dados para adaptar continuamente um modelo funcional que descreve o processo gerador dos dados. O algoritmo é incremental, aprende com apenas uma apresentação do conjunto de dados, processa uma amostra por vez, não armazena dados antigos e realiza todos os procedimentos dinamicamente, de acordo com a aquisição de dados de entrada. O modelo tem duas partes principais: estrutural e paramétrica. A parte estrutural é composta pelas regras nebulosas, regras estas definidas por um algoritmo de agrupamento que divide os dados no espaço de entrada em grânulos no formato de hiper-retângulos, atribuindo uma regra nebulosa a cada hiper-retângulo. A base de regras é vazia inicialmente, e regras são adicionadas gradualmente de acordo com a aquisição de dados de entrada. O algoritmo de agrupamento é uma versão aprimorada do aprendizado Min-Max, que ajusta o tamanho dos hiper-retângulos automaticamente de acordo com a dispersão dos dados de entrada. A parte paramétrica do modelo é constituída por funções afins locais atribuídas a cada regra com coeficientes atualizados pelo algoritmo de quadrados mínimos recursivo. O modelo gera a saída a partir da combinação das saídas individuais das regras, em que cada saída tem um peso diferente, determinado pelo nível de ativação normalizado da regra correspondente. O algoritmo nebuloso evolutivo Min-Max também possui mecanismos de gerenciamento da base de regras e estimação automática de parâmetros de aprendizado. O gerenciamento da base consiste na identificação de regras redundantes (que são mescladas) ou obsoletas (que são excluídas), permitindo que a base de regras esteja sempre atualizada e que o método seja auto-adaptável a ambientes não estacionários. A estimação automática de parâmetros, em contrapartida, torna o algoritmo mais autônomo, atenuando perdas de desempenho provenientes de escolhas inadequadas de parâmetros de aprendizado. O algoritmo é aplicado em problemas de previsão de séries temporais e identificação de sistemas e comparado a abordagens clássicas e evolutivas representativas do estado da arte na área.

Palavras-chaves: Sistemas nebulosos evolutivos, modelagem de sistemas, aprendizado incremental, regressão

Abstract

Evolving fuzzy systems are an appealing approach to deal with nonstationary system modeling using data streams. This work introduces an evolving fuzzy Min-Max algorithm for fuzzy rule-based systems modeling. The algorithm processes an incoming data stream to construct and continuously update a fuzzy functional model of the data generator process. The algorithm is incremental, learns with only one pass in the dataset, process one data sample at a time, and does not perform any retraining or store past data. The model has two parts: structural and parametric. The structural part is a set of functional fuzzy rules formed by a clustering algorithm that granulates the input data space into data granules in the form of hyperboxes. To each hyperbox corresponds a fuzzy rule. The rule base is initially empty, and rules are gradually added as new incoming data are input. The clustering algorithm is an enhanced Min-Max learning approach that automatically adjusts hyperboxes sizes based on input data dispersion. The parametric part of the model is formed by local affine functions associated with each fuzzy rule. The parameters of the local functions are updated using the recursive least squares algorithm. The model output is produced combining the local affine functions weighted by the normalized firing degrees of the active rules. The evolving fuzzy Min-Max algorithm also has a rule base management method to allow concise and updated rule bases by identifying redundant rules (which are merged) or obsolete rules (which are deleted). The rule base management mechanism makes the model parsimoniously adaptive in nonstationary environments. Another property of the evolving fuzzy algorithm is the dynamic estimation of some learning parameters, which increases the algorithm autonomy, and mitigates the deterioration of the prediction performance due to unsuitable initial choices of the learning parameters. The proposed algorithm is applied in system identification and time series forecasting tasks, and its performance is compared with that of evolving and classic regression algorithms representative of the current state of the art in the area.

Keywords: Evolving fuzzy systems, system modeling, incremental learning, regression

Lista de ilustrações

Lista de tabelas

Sumário

1 Introdução

1.1 Motivação

A modelagem de sistemas dinâmicos é de fundamental importância nas aplicações industriais e nos processos de tomada de decisão. Inicialmente, a maioria dos processos industriais eram conduzidos manualmente, demandando diversos funcionários para o seu controle e supervisão. Ao longo do século passado, no entanto, diversos modelos foram desenvolvidos a fim de se automatizar estes processos. Um modelo matemático pode ser descrito como a representação dos aspectos essenciais de um sistema, sintetizando o conhecimento desse sistema de uma forma útil e aplicável (EYKHOFF, 1974). A modelagem de um processo produz, portanto, uma representação simplificada dos fenômenos que interferem no funcionamento deste processo, com diferentes níveis de precisão, de acordo com a aplicação desejada. Após a modelagem de um sistema, é possível projetar controladores e atuadores para operá-lo automaticamente, diminuindo ou até mesmo excluindo a necessidade de operadores humanos. Este tipo de abordagem resulta em ganho de produtividade e redução de custos, aumentando os lucros das companhias (LUGHOFER, 2011).

A primeira geração de modelos foram os analíticos, também conhecidos como modelos *white box*. Estes modelos consistem na completa descrição matemática do processo, baseando-se nas fórmulas derivadas da física, matemática e biologia, cobrindo todos os estados possíveis do sistema, inclusive em condições extremas. Este tipo de abordagem, entretanto, demanda um conhecimento profundo sobre todos os fatores relevantes ao sistema a ser modelado, o que se torna impraticável à medida que a complexidade aumenta. Além disso, uma vez que o modelo está construído, qualquer modificação das características do sistema requer uma nova análise e reconstrução do modelo, o que é algo demorado e dispendioso.

Os problemas práticos da modelagem analítica motivaram o surgimento de uma nova abordagem para modelagem de sistemas. Em vez da análise profunda das características físicas do processo, utiliza-se o conhecimento de especialistas para a construção dos modelos. Esses especialistas eram, geralmente, operadores humanos com vasta experiência prática, capazes de manipular o sistema em diversos regimes e identificar até mesmo falhas menos frequentes. Em vez de usar fórmulas matemáticas diretas, o conhecimento desses especialistas era transmitido através da fala, representado em variáveis linguísticas (e.g. alto, baixo, pouco, muito, etc) (PEDRYCZ; GOMIDE, 2007), as quais eram então inseridas em modelos matemáticos específicos. Alguns exemplos desses modelos são os sistemas nebulosos, desenvolvidos com base na teoria de conjuntos nebulosos, proposta por Zadeh (ZADEH, 1965). Esta abordagem constitui uma alternativa interessante para a representação matemática das variáveis linguísticas, pois é capaz de lidar com informações imprecisas e difusas de uma maneira semelhante ao raciocínio

humano.

A modelagem de sistemas baseada na experiência de especialistas possibilitou a representação de conhecimento de uma forma mais sucinta, sem a necessidade de um estudo profundo de todos os estados e modos de operação do sistema. Existiam, no entanto, algumas desvantagens importantes. Uma delas era o fato do conhecimento do especialista ser baseado na sua interpretação particular do sistema. Essa interpretação está geralmente ligada a fatores subjetivos, que poderiam variar de especialista para especialista. Além disso, nem sempre era possível contar com um operador com vasta experiência prática, principalmente devido ao tempo necessário para a aquisição desse conhecimento. Estes fatores revelaram a necessidade de se desenvolver abordagens independentes do conhecimento prévio sobre o sistema, e imunes a fatores subjetivos da interpretação humana. Esta demanda motivou o surgimento de técnicas capazes de aprender o comportamento de um sistema de forma automática e objetiva. Esse conjunto de técnicas recebeu a denominação de modelos *data driven*. Em vez de utilizar o conhecimento de especialistas, estas técnicas se baseiam nos dados gerados por um sistema para modelá-lo, os quais são geralmente obtidos por sistemas de leitura (e.g sensores), são pré-processados e inseridos em um algoritmo computacional que extrai conhecimento desse conjunto de dados. O processo de extração de conhecimento dos dados é chamado de treinamento do modelo. Uma das vantagens das abordagens *data driven* é a natureza genérica dos algoritmos, significando que não é necessário saber o tipo de sistema que gerou os dados (físico, químico, biológico...), tampouco as leis que regem os fenômenos naturais envolvidos neste sistema. Devido a sua natureza genérica, esse conjunto de técnicas é classificado como modelos *black box*. Alguns exemplos destas técnicas são as redes neurais artificiais (HAYKIN, 1998), (DUDA *et al.*, 2000), máquinas de vetores suporte (BURGES, 1998), (VAPNIK, 1998) e métodos de aprendizado estatístico (HASTIE *et al.*, 2009), (JAMES *et al.*, 2017).

As abordagens *data driven* simplificaram e agilizaram a modelagem de sistemas, sendo amplamente utilizadas em diversas aplicações. Apesar desse relevante avanço, existiam questões que ainda inviabilizavam a utilização destes métodos em algumas situações. Uma delas é que, geralmente, esses modelos precisam de um número considerável de dados, com baixo nível de ruído, para que o algoritmo possa construir um modelo confiável do sistema. Esse conjunto de dados precisa ser gravado em uma unidade de armazenamento (e.g. disco rígido), onde o algoritmo de aprendizado faz diversas leituras no conjunto completo para treinar o modelo, o que é conhecido como treinamento *offline* ou em batelada. Em aplicações com grandes conjuntos de dados, este processo exige considerável quantidade de recursos computacionais, tanto de memória quanto de processamento, além de um intervalo de tempo relevante para o treinamento do modelo. Outra questão importante é que muitos desses algoritmos geram modelos estáticos do sistema, o que significa que o algoritmo não faz nenhuma revisão deste modelo após a fase de treinamento. Quando o processo é estacionário, as características do sistema se mantém inalteradas ao longo do tempo, e portanto, um modelo construído com dados atuais se manterá igualmente válido no futuro, representando com fidelidade a natureza do sistema. Na

maioria dos sistemas dinâmicos reais, no entanto, a estacionariedade não é observada. Nesses sistemas, as distribuições estatísticas dos dados mudam ao longo do tempo, modificando as relações entre dados de entrada e saída. O sistema em que essas mudanças ocorrem é chamado de não estacionário, e as mudanças nas distribuições de dados são conhecidas como *concept drift* (quando as mudanças são suaves e gradativas) ou como *concept shift* (quando as mudanças são bruscas e repentinas) (GAMA *et al.*, 2014), (TSYMBAL, 2004). Em distribuições não estacionárias, um modelo não adaptativo treinado sobre a falsa suposição de estacionariedade está fadado a se tornar obsoleto, e terá um desempenho aquém do ideal na melhor das hipóteses, ou falhará gravemente na pior delas (DITZLER *et al.*, 2015). Em processos não estacionários, portanto, o modelo precisa ser constantemente revisto para se manter atualizado e representar adequadamente o sistema. Nos algoritmos em batelada, no entanto, essa revisão implicaria uma atualização completa do modelo, descartando-se todo o conhecimento antigo e começando-se um novo treinamento do início com os dados mais recentes. Em algumas aplicações, esta abordagem simplesmente não é possível, pois o custo computacional e tempo necessários para se realizar treinamentos recorrentes são proibitivos para as demandas da aplicação.

As aplicações de que se tratam o parágrafo anterior são particularmente importantes em problemas de previsão em tempo real, que envolvem o processamento não de um conjunto finito de dados, mas de um fluxo contínuo. Esses tipos de aplicações emergiram devido às recentes evoluções nas redes de comunicação e sensoriamento, gerando grandes quantidades de informação provenientes de fontes distribuídas e conectadas. Essas tecnologias deram origem à transmissão automática de informações, onde os dados não são inseridos apenas por usuários, mas também por outros computadores (MUTHUKRISHNAN, 2005). Nesses tipos de sistemas, os dados são gerados e coletados em alta velocidade, o que significa que o ritmo de geração dos dados é alto quando comparado à capacidade computacional (GAMA, 2012). Além da alta velocidade, esses fluxos de dados são geralmente não estacionários, o que impossibilita completamente a modelagem desses sistemas utilizando-se algoritmos em batelada. Essas aplicações deram origem à demanda por métodos capazes de processar fluxos de dados sem a necessidade de retreinamento, auto adaptáveis a processos não estacionários e ainda com baixo custo computacional. Essas demandas não poderiam ser satisfeitas com métodos de aprendizado em batelada, tampouco por modelos estáticos que assumem processos constantes ao longo do tempo.

Neste cenário, os sistemas nebulosos evolutivos surgiram como uma alternativa interessante. Esses métodos são capazes de processar os dados de forma incremental, têm capacidade de aprender distribuições de dados com algoritmos rápidos e possuem meios de auto revisão da sua estrutura para se adaptar a mudanças no ambiente gerador de amostras. Desde o surgimento do algoritmo Takagi Sugeno evolutivo (ANGELOV; FILEV, 2004), diversas contribuições enriqueceram a literatura de métodos evolutivos para aplicações de modelagem e previsão em tempo real.

Esta dissertação propõe mais uma contribuição à literatura de métodos nebulosos evolutivos, chamada de algoritmo nebuloso evolutivo Min-Max. Baseando-se nas principais motivações descritas acima, o método prioriza a rapidez e simplicidade do algoritmo de aprendizado. Além disso, o modelo é capaz de se auto adaptar de maneira *online*, tanto na sua estrutura (gerenciamento constante da base de regras) quanto nos seus parâmetros (quadrados mínimos com *forgetting factor* e ajuste dos parâmetros de aprendizado). Desta forma, acredita-se que o método proposto é capaz de suprir as demandas das tarefas de previsão em tempo real e modelagem, podendo processar fluxos contínuos de dados de natureza genérica.

1.2 Objetivos

O objetivo principal deste trabalho é desenvolver um método evolutivo de regressão baseado no algoritmo de aprendizado das redes neuro nebulosas Min-Max. A ideia é utilizar as mesmas regras nebulosas na etapa de clusterização e empregar operações simples no algoritmo de aprendizado, como máximo, mínimo e comparações. No entanto, deseja-se implementar algumas melhorias ao algoritmo Min-Max original, possibilitando que os hiper-retângulos sejam ajustados de acordo com a dispersão dos dados de entrada, o que, possivelmente, resultaria em melhores desempenhos em ambientes não estacionários. Desta forma, é possível construir um algoritmo rápido, eficiente e relativamente simples. Definidos os grupos na fase de clusterização, um modelo linear é ajustado para cada grupo utilizando o método de quadrados mínimos recursivo. O algoritmo é incremental, extraindo conhecimento dos dados em tempo real, processando uma amostra por vez e não armazenando dados antigos. O método também gerencia a base de regras, identificando grupos obsoletos ou redundantes, a fim de se eliminar complexidade desnecessária e se ajustar às mudanças nas distribuições de dados (i.e. *concept drift* e *concept shift*).

Outro objetivo considerado neste trabalho é a estimação de alguns parâmetros de aprendizado. Um desses parâmetros é o que determina a largura máxima dos hiper-retângulos. Nas técnicas Min-Max existentes na literatura, esse parâmetro é geralmente fornecido pelo usuário, se mantém fixo ao longo do treinamento do modelo e tem forte influência no desempenho do algoritmo. Por esse motivo, este trabalho sugere uma forma de ajustar o valor inicial fornecido pelo usuário. Além disso, este trabalho propõe um método de fusão de regras independente do usuário, baseando-se na interseção, volume e centro das regras para determinar se um par de regras é redundante ou não.

Apesar da técnica proposta utilizar operações simples em seu método de aprendizado, deseja-se que ela tenha desempenho comparável a técnicas similares ou mais complexas. Para constatar se esse objetivo foi atingido, o algoritmo de regressão é testado em cinco conjuntos de dados distintos, e seus resultados são comparados com outras técnicas da literatura.

1.3 Organização do trabalho

Este capítulo apresentou o tema de modelagem e previsão em tempo real. O capítulo descreveu demandas específicas destas aplicações, que ainda não são atendidas por métodos clássicos de modelagem de sistemas. Essas demandas motivaram a elaboração do algoritmo de regressão proposto nesse trabalho. Os temas abordados nos próximos capítulos são detalhados abaixo.

O Capítulo 2 aborda o aprendizado Min-Max. Esta técnica é particularmente importante por dar origem ao algoritmo de aprendizado empregado no método proposto nesta dissertação. O capítulo detalha o algoritmo Min-Max pioneiro, proposto por Simpson (SIMPSON, 1992), e depois apresenta métodos relacionados para aplicações em classificação, clusterização e regressão *offline*.

O Capítulo 3 detalha os métodos nebulosos evolutivos. Uma revisão da literatura dessas técnicas é apresentada, dando maior ênfase a três métodos em particular, e abordando resumidamente outros algoritmos.

O Capítulo 4 se baseia nos conceitos dos Capítulos 2 e 3 para detalhar o algoritmo de regressão proposto neste trabalho. O algoritmo nebuloso evolutivo Min-Max (eFMM) é abordado quanto às suas características estruturais e paramétricas. O capítulo também detalha o algoritmo de aprendizado do modelo.

O Capítulo 5 aplica o eFMM em problemas de previsão de séries temporais e identificação de sistemas. O eFMM é comparado com técnicas relevantes da literatura em duas métricas de desempenho e no tempo de execução do algoritmo. O capítulo também avalia a estacionariedade dos conjuntos de dados utilizados nos experimentos, a fim de atestar a natureza genérica do modelo.

Finalmente, o Capítulo 6 conclui os conceitos trazidos nesta dissertação. O capítulo detalha as contribuições trazidas pelo eFMM e faz uma comparação final do algoritmo com técnicas evolutivas similares, baseando-se nos resultados apresentados no Capítulo 5. Algumas sugestões para investigações futuras também são apresentadas.

2 Aprendizado Min–Max

2.1 Introdução

O aprendizado Min-Max é um conjunto de técnicas que combinam a eficiência computacional das redes neurais com a capacidade de lidar com informações imprecisas dos sistemas nebulosos. Os trabalhos pioneiros desta abordagem foram uma rede de classificação (SIMPSON, 1992) e uma de clusterização (SIMPSON, 1993). A ideia principal dos algoritmos Min-Max é a utilização de estruturas paramétricas retangulares como regras nebulosas, e a agregação destas estruturas para formar classes na saída. A principal vantagem das estruturas citadas é o fato de elas serem definidas por apenas dois pontos, independentemente da dimensão de entrada. Além disso, o ajuste destas estruturas é realizado a partir de operações simples, como máximo, mínimo e comparações. Como resultado, obtém-se algoritmos computacionalmente rápidos e eficientes, capazes de aprender fronteiras não lineares entre classes com apenas uma época de treinamento, sem a necessidade de armazenar dados antigos. Todas essas características despertaram o interesse por esta abordagem, o que culminou na publicação de diversos trabalhos propondo atualizações dos algoritmos originais de Simpson (GABRYS; BARGIELA, 2000), (NANDEDKAR; BISWAS, 2007), (SEERA *et al.*, 2015), (MOHAMMED; LIM, 2015).

Este capítulo detalha o algoritmo de classificação original de Simpson na Seção 2.2. A Seção 2.3 cita trabalhos que propuseram modificações no algoritmo de treinamento original, enquanto as Seções 2.4 e 2.5 apresentam abordagens que utilizam as regras nebulosas de Simpson em algoritmos de treinamento diferentes. A Seção 2.6, em contrapartida, comenta brevemente sobre alguns algoritmos Min-Max de regressão encontrados na literatura. Por fim, a Seção 2.7 resume os temas abordados neste capítulo.

2.2 Rede Min–Max de classificação

Na rede Min-Max de classificação, cada neurônio é representado por uma função de pertinência n-dimensional, onde n é a dimensão do padrão de entrada. O núcleo (i.e. região com nível de pertinência igual a 1) desta função é um paralelepípedo de n dimensões. Este trabalho utilizará o termo hiper-retângulo para referir-se ao núcleo desta função de pertinência. A Figura 2.1 ilustra um hiper-retângulo de 3 dimensões.

Cada hiper-retângulo da rede é descrito pela seguinte quádrupla

$$B_i = \{I^n, V_i, W_i, b_i(x_h, V_i, W_i)\} \tag{2.1}$$

onde I^n é o hipercubo unitário n-dimensional, $x_h \in I^n$ é um padrão de entrada, V_i e W_i são os pontos de mínimo e máximo do i-ésimo hiper-retângulo e b_i é a função de pertinência associada

a B_i. Cada hiper-retângulo está associado a uma classe, e gera, para cada amostra recebida, uma ativação na saída da rede igual ao nível de pertinência da amostra. Cada hiper-retângulo está associado a somente uma classe, mas uma classe pode conter vários hiper-retângulos associados a ela. Desta forma, é necessário combinar as ativações de diversos hiper-retângulos para uma mesma classe

$$c_k = \bigcup_{i \in K} B_i \tag{2.2}$$

onde K é o conjunto de hiper-retângulos associados à classe c_k. Na rede de Simpson, a operação de união é realizada como $\max_{i \in K} b_i$.

2.2.1 Estrutura da rede

A estrutura da rede Min-Max é ilustrada na Figura 2.2. Os padrões de entrada são inseridos na primeira camada da rede, cuja j-ésima componente é x_j. A segunda camada é formada pelos hiper-retângulos, que são os neurônios da rede neuro nebulosa. As funções de ativação destes neurônios são as respectivas funções de pertinência. A conexão entre primeira e segunda camadas é realizada pelos pontos de máximo e mínimo dos hiper-retângulos, W e V, que são ajustados por um algoritmo de treinamento (ver Seção 2.2.3). A terceira camada é a saída da rede, onde cada nó representa uma classe. A conexão entre segunda e terceira camadas é composta por valores binários, armazenados em uma matriz $U = [u_{ik}]$. Os elementos dessa matriz são:

$$u_{ik} = \begin{cases} 1, & \text{se } b_i \text{ pertence à classe } c_k \\ 0, & \text{caso contrário} \end{cases} \tag{2.3}$$

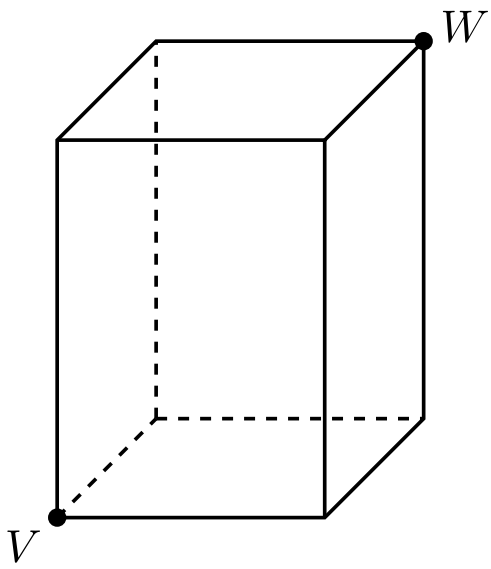


Figura 2.1 – Hiper-retângulo no I^3

Desta forma, cada hiper-retângulo estará conectado apenas à classe correspondente. A saída é obtida por:

$$c_k = \max_{i=1,\ldots,m} b_i u_{ik} \tag{2.4}$$

e a classe com maior nível de ativação é aquela que classifica uma amostra. Esta técnica recebe o nome *winner-takes-all* na literatura (KOHONEN, 1989).

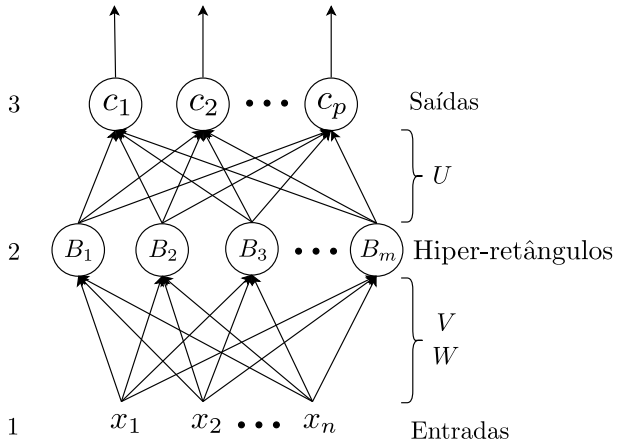


Figura 2.2 – Estrutura da rede Min-Max

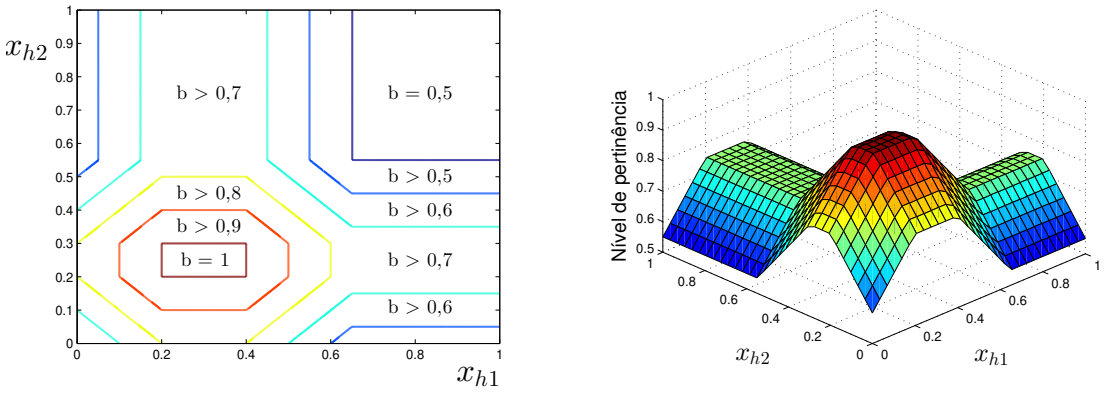
2.2.2 Função de pertinência

A função de pertinência b, ilustrada na Figura 2.3, é:

$$\begin{aligned} b_i(x_h, V_i, W_i) \quad = \quad & \frac{1}{2n} \sum_{j=1}^{n} \big[max(0, 1 - max(0, \gamma\, min(1, x_{jh} - w_{ji}))) \\ & + max(0, 1 - max(0, \gamma\, min(1, v_{ji} - x_{jh}))) \big] \end{aligned} \tag{2.5}$$

onde n é a dimensão do espaço de entrada, i é o índice do i-ésimo hiper-retângulo, γ é um parâmetro que regula o decaimento da função de pertinência, e x_{jh}, v_{ji} *e* w_{ji} são as componentes individuais do h-ésimo padrão de entrada, ponto de mínimo e máximo do hiper-retângulo, respectivamente. É importante lembrar que a expressão (2.5) gera valores de pertinência adequados somente quando o hiper-retângulo e os padrões de entrada estão contidos no hipercubo unitário I^n.

O nível de pertinência é igual a 1 (i.e. total compatibilidade) para todo ponto no interior ou na fronteira de um hiper-retângulo. Para os pontos exteriores, o valor da função decai proporcionalmente à diferença elemento a elemento entre a entrada e os pontos de máximo e mínimo. Esta função tem o valor mínimo de 0,5 para qualquer ponto do espaço I^n. Esta característica pode ser indesejável em algumas aplicações, e portanto, outras funções foram propostas em trabalhos posteriores, como descrito na próxima seção.



(a) Curvas de nível (b) Tridimensional

Figura 2.3 – Função de pertinência: V= (0,2 0,2), W= (0,4 0,3) e $\gamma = 4$

Os núcleos das funções de pertinência da rede Min-Max são estruturas paramétricas, cujas fronteiras são hiperplanos paralelos aos eixos de coordenadas. No entanto, é possível haver fronteiras não lineares entre as classes devido às funções de pertinência. Este conceito é ilustrado na Figura 2.4.

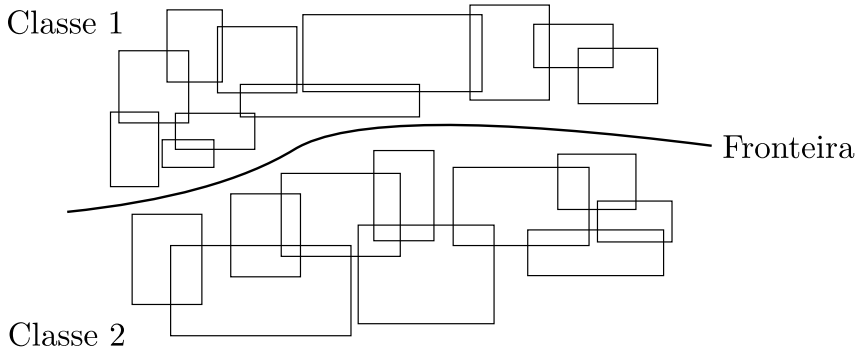


Figura 2.4 – Fronteira não linear entre classes 1 e 2

2.2.3 Algoritmo de treinamento

O treinamento da rede neuro nebulosa Min-Max original proposta por Simpson consiste em três etapas: Expansão, teste de sobreposição e contração. As três etapas são detalhadas a seguir. Um padrão de entrada da rede é composto pelo par ordenado $\{x_h, d_h\}$, sendo x_h o padrão de entrada, $h = 1,2,3,...,N$, N o número total de padrões, e $d_h = 1,2,...,C$ é a classe associada ao padrão x_h, sendo C o número total de classes.

Inicialmente, a rede não tem nenhum hiper-retângulo. Quando o primeiro padrão de entrada é recebido ele se torna o primeiro hiper-retângulo fazendo-se $V_i = W_i = x_h$, e $L_i = d_h$, onde L_i denota a classe correspondente ao hiper-retângulo B_i. Através do algoritmo de treinamento, a rede vai incorporando conhecimento com a adição ou o ajuste de hiper-retângulos. O treinamento necessita de apenas uma época, ou seja, uma única apresentação do conjunto completo de amostras. O treinamento é *online*, significando que cada amostra é lida, processada e

descartada antes do processamento da próxima amostra.

Expansão

Na etapa de expansão, a rede recebe o padrão de entrada e procura um hiper-retângulo que possa se expandir para incorporar a amostra. A expansão de um hiper-retângulo requer as condições:

1. $L_i = d_h$
2. $n\theta \geq \sum_{j=1}^{n} max(w_{ji}, x_{jh}) - min(v_{ji}, x_{jh})$

O parâmetro θ é definido pelo usuário e controla o tamanho máximo do hiper-retângulo. Este parâmetro tem forte influência no desempenho da rede (SIMPSON, 1992). As condições 1 e 2 são testadas no hiper-retângulo com maior nível de pertinência com relação ao padrão de entrada. Caso ele não satisfaça uma destas condições, escolhe-se o hiper-retângulo com segundo maior nível de pertinência, e assim sucessivamente. Quando um hiper-retângulo satisfaz as condições, o processo de expansão é realizado da seguinte forma:

$$v_{ji} = min(v_{ji}, x_{jh}) \qquad\qquad w_{ji} = max(w_{ji}, x_{jh}) \tag{2.6}$$

Se a nova amostra for externa ao hiper-retângulo, ela ficará na fronteira deste após o processo de expansão. Caso ela já esteja no interior, o processo de expansão não modifica a configuração original da rede. Se nenhum hiper-retângulo satisfizer as condições de expansão 1 e 2, a nova amostra passa a ser um novo hiper-retângulo com $V_i = W_i = x_h$, e $L_i = d_h$.

Teste de sobreposição

Ao fim de um processo de expansão, é possível que o hiper-retângulo expandido se sobreponha a outro em um subconjunto de dimensões. Esta situação é ilustrada na Figura 2.5. Quando isso acontece, os padrões contidos no espaço comum compartilhado por mais de um hiper-retângulo têm nível de pertinência igual a 1 para todos eles. Isto gera uma indeterminação e, portanto, não é um efeito desejável. Caso as classes dos hiper-retângulos envolvidos sejam iguais, não há problema, pois todos os hiper-retângulos que englobam a amostra na região de conflito ativarão a mesma classe na saída. Se houver classes divergentes, no entanto, algo precisa ser feito para que um único padrão não pertença plenamente a mais de uma classe. Para contornar este problema, é realizado um teste de sobreposição entre pares de hiper-retângulos toda vez que um hiper-retângulo se expande, a fim de encontrar possíveis regiões de conflito. Este teste é realizado entre o hiper-retângulo recém expandido, i, e todos os hiper-retângulos com classe diferente de i, ou seja, $L_l \neq L_i$. O teste de sobreposição entre os hiper-retângulos i e l é descrito abaixo:

Para $j = 1,2,...,n$ **Fazer:**

$$
\begin{aligned}
&caso\ 1: v_{ji} < v_{jl} < w_{ji} < w_{jl},\\
&\qquad \tau^{novo} = min(w_{ji} - v_{jl}, \tau^{antigo})\\
&caso\ 2: v_{jl} < v_{ji} < w_{jl} < w_{ji}\\
&\qquad \tau^{novo} = min(w_{jl} - v_{ji}, \tau^{antigo})\\
&caso\ 3: v_{ji} < v_{jl} < w_{jl} < w_{ji}\\
&\qquad \tau^{novo} = min(min(w_{jl} - v_{ji}, w_{ji} - v_{jl}), \tau^{antigo})\\
&caso\ 4: v_{jl} < v_{ji} < w_{ji} < w_{jl}\\
&\qquad \tau^{novo} = min(min(w_{ji} - v_{jl}, w_{jl} - v_{ji}), \tau^{antigo})\\
&\qquad \textbf{Se } \tau^{antigo} > \tau^{novo}, \textbf{ Então:}\\
&\qquad \Delta = j, n_{caso} = 1,2,3 \textit{ ou } 4 \text{ e } \tau^{antigo} = \tau^{novo}\\
&\qquad \textbf{Fim Se}\\
&\textbf{Fim Para}
\end{aligned}
\tag{2.7}
$$

O teste acima funciona da seguinte maneira: Para cada uma das n dimensões, o teste identifica se existe uma sobreposição entre os hiper-retângulos i e l na dimensão j. Em caso positivo, classifica-se a sobreposição em um dos quatro casos possíveis. A Figura 2.6 ilustra estes quatro casos, considerando-se o eixo horizontal. Inicialmente, $\tau^{antigo} = 1$. Toda vez que uma sobreposição é identificada em uma dada dimensão, o seu tamanho é calculado, armazenado em τ^{novo} e comparado com τ^{antigo}. Caso $\tau^{antigo} > \tau^{novo}$, então a sobreposição identificada na dimensão j é menor do que todas as identificadas anteriormente, e as variáveis Δ e n_{caso} têm os seus valores atualizados. O algoritmo apresentado acima identifica a dimensão onde ocorre a menor sobreposição entre um par de hiper-retângulos. Na etapa de contração, apresentada a seguir, os hiper-retângulos serão modificados apenas nesta dimensão onde ocorre a sobreposição mínima, de modo a garantir a menor modificação possível na estrutura existente.

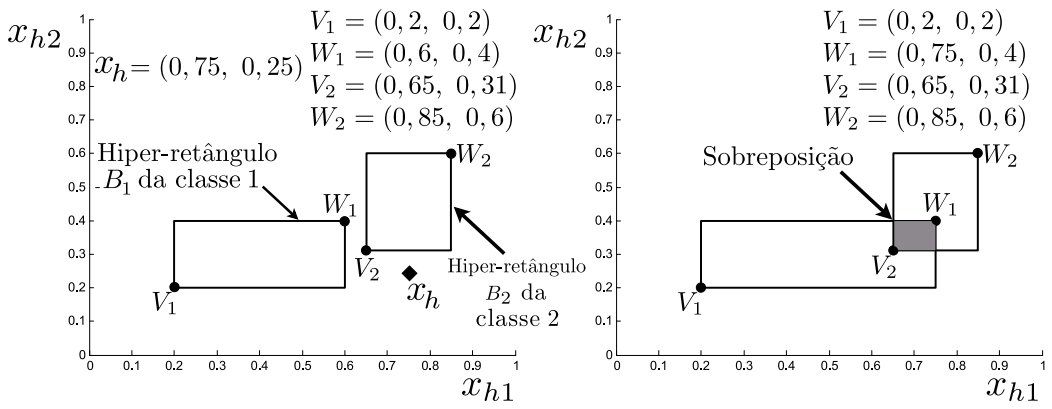
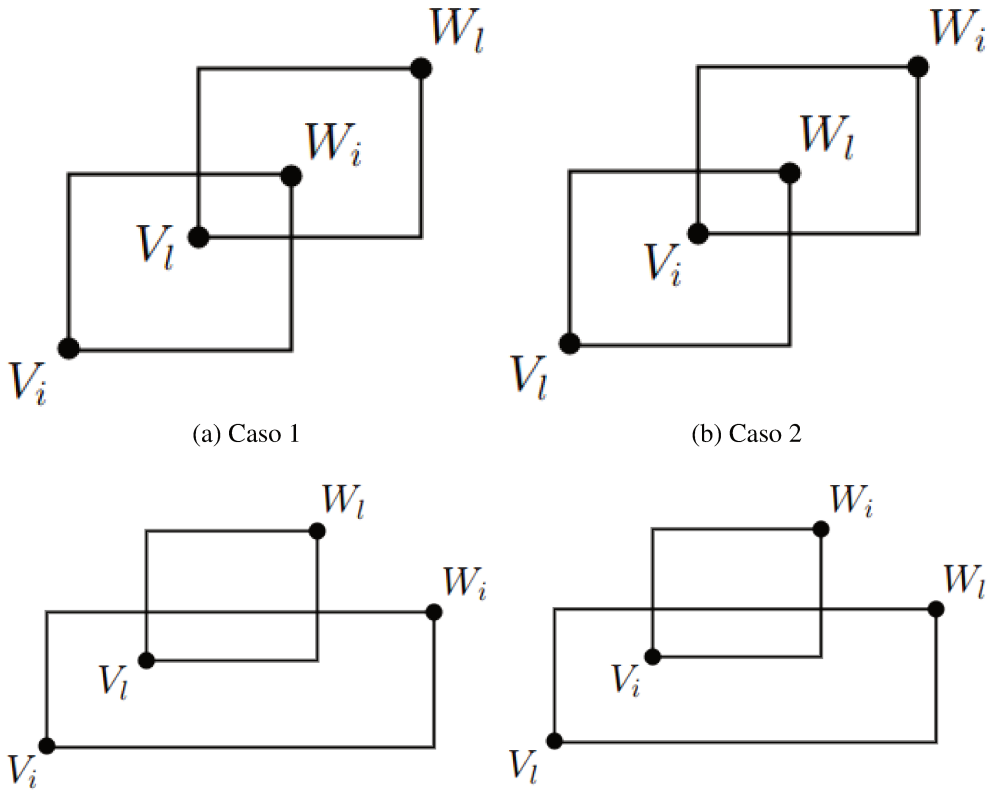


Figura 2.5 – Sobreposição entre hiper-retângulos 1 e 2 após expansão



(a) Caso 1 (b) Caso 2

(c) Caso 3 (d) Caso 4

Figura 2.6 – Casos de sobreposição no eixo horizontal

Contração

Após a detecção de uma sobreposição, é necessário eliminar o problema utilizando a contração. Para tal, os pontos de máximo e mínimo dos hiper-retângulos envolvidos são ajustados na dimensão onde ocorre a menor sobreposição, Δ, encontrada no teste de sobreposição. A Figura 2.7 ilustra a contração para eliminar a sobreposição na dimensão horizontal entre os hiper-retângulos i e l. Para realizar a contração, Simpson definiu uma regra para cada um dos quatro casos de sobreposições possíveis, sendo que os casos 3 e 4 foram subdivididos em dois:

$$
\begin{aligned}
&caso\ 1 : v_{\Delta i} < v_{\Delta l} < w_{\Delta i} < w_{\Delta l}, \\
&\qquad w_{\Delta i}^{novo} = v_{\Delta l}^{novo} = \frac{w_{\Delta i}^{antigo} + v_{\Delta l}^{antigo}}{2} \\
&caso\ 2 : v_{\Delta l} < v_{\Delta i} < w_{\Delta l} < w_{\Delta i} \\
&\qquad w_{\Delta l}^{novo} = v_{\Delta i}^{novo} = \frac{w_{\Delta l}^{antigo} + v_{\Delta i}^{antigo}}{2} \\
&caso\ 3a : v_{\Delta i} < v_{\Delta l} < w_{\Delta l} < w_{\Delta i}\ e\ (w_{\Delta l} - v_{\Delta i}) < (w_{\Delta i} - v_{\Delta l}) \\
&\qquad v_{\Delta i}^{novo} = w_{\Delta l}^{antigo} \\
&caso\ 3b : v_{\Delta i} < v_{\Delta l} < w_{\Delta l} < w_{\Delta i}\ e\ (w_{\Delta l} - v_{\Delta i}) > (w_{\Delta i} - v_{\Delta l}) \\
&\qquad w_{\Delta i}^{novo} = v_{\Delta l}^{antigo} \\
&caso\ 4a : v_{\Delta l} < v_{\Delta i} < w_{\Delta i} < w_{\Delta l}\ e\ (w_{\Delta l} - v_{\Delta i}) < (w_{\Delta i} - v_{\Delta l}) \\
&\qquad w_{\Delta l}^{novo} = v_{\Delta i}^{antigo} \\
&caso\ 4b : v_{\Delta l} < v_{\Delta i} < w_{\Delta i} < w_{\Delta l}\ e\ (w_{\Delta l} - v_{\Delta i}) < (w_{\Delta i} - v_{\Delta l}) \\
&\qquad v_{\Delta l}^{novo} = w_{\Delta i}^{antigo}
\end{aligned}
\tag{2.8}
$$

Figura 2.7 – Contração

2.3 Atualizações do algoritmo original

O próprio Simpson publicou uma segunda versão de seu algoritmo para lidar com tarefas de clusterização (SIMPSON, 1993). Este novo trabalho propôs uma nova função de pertinência, que pode atribuir valor zero a amostras distantes do hiper-retângulo. Além disso, um novo processo de contração elimina sobreposições em todas as dimensões, a fim de produzir grupos mais compactos. Posteriormente, Gabrys (GABRYS; BARGIELA, 2000) propôs uma única rede Min-Max capaz de lidar com ambos os problemas, classificação e clusterização. A esta rede foi dado o nome *General Fuzzy Min-Max Neural Network* (GFMN). Outra característica apresentada foi a possibilidade de processar dados intervalares. Desta forma, um padrão de entrada pode ser caracterizado não por um, mas por dois pontos, x_h^l e x_h^u, sendo estes, respectivamente, os pontos de mínimo e máximo do padrão de entrada h.

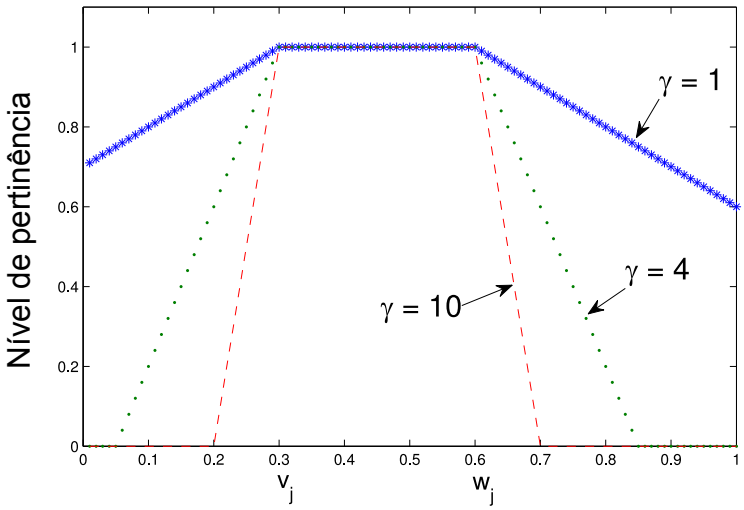


Figura 2.8 – Função de pertinência (2.9)

Uma terceira modificação do algoritmo foi a função de pertinência. Considerou-se que as funções utilizadas por Simpson atribuíam valores muito altos para amostras em regiões onde deveriam ter menor nível. Além disso, foi mostrado que elas atribuem níveis de pertinência iguais para amostras com distâncias diferentes para um certo hiper-retângulo. Por estes motivos, uma nova função foi introduzida:

$$b_j(x_h, V_i, W_i) = min(min([1 - f(x^u_{jh} - w_{ji}, \gamma_j)], [1 - f(v_{ji} - x^l_{jh}, \gamma_j)]) \tag{2.9}$$

$$f(r\gamma) = \begin{cases} 1, & \text{se} \quad r\gamma \geq 1 \\ r\gamma, & \text{se} \ 0 \leq r\gamma \leq 1 \\ 0, & \text{se} \quad r\gamma \leq 0 \end{cases}$$

onde $x^u_{jh}, x^l_{jh}, w_{ji}, v_{ji}$ são os pontos de máximo e mínimo do padrão de entrada h e do hiper-retângulo i, respectivamente, e γ_j o fator de decaimento da dimensão j, conforme ilustra a Figura 2.8. Outra modificação apresentada por Gabrys é uma nova regra para o tamanho máximo permitido a um hiper-retângulo. Para se expandir, um hiper-retângulo B_i deve satisfazer a seguinte condição

$$w_{ji} - v_{ji} \leq \theta \qquad\qquad j = 1, 2, ..., n \tag{2.10}$$

Também baseando-se no trabalho de Simpson, Mohammed propôs algumas modificações no algoritmo original (MOHAMMED; LIM, 2015). Neste artigo, Mohammed lista falhas no processo de identificação de sobreposições, uma vez que alguns tipos de sobreposições eram ignoradas e poderiam causar erros de classificação. Para resolver o problema, Mohammed propôs nove regras na etapa de teste de sobreposição no lugar das quatro propostas por Simpson.

Uma atualização desta primeira versão é apresentada pelo mesmo autor em (MOHAMMED; LIM, 2017). Este trabalho propõe duas técnicas para refinar o algoritmo de aprendizado. A primeira delas incorpora o método k-vizinhos mais próximos na rede Min-Max. Quando uma nova amostra é recebida, em vez de se testar a condição de expansão apenas do hiper-retângulo mais influente, testa-se os k hiper-retângulos mais influentes pertencentes à mesma classe. Desta forma, evita-se a criação excessiva de hiper-retângulos, o que simplifica a estrutura da rede. A segunda modificação sugerida neste trabalho é a exclusão de hiper-retângulos com baixo desempenho (i.e. *pruning*). O desempenho dos hiper-retângulos é quantificado como a razão das amostras classificadas corretamente sobre o total de amostras classificadas por um hiper-retângulo B_i. Esta abordagem torna o modelo mais robusto na presença de ruído, o que melhora o desempenho de classificação (MOHAMMED; LIM, 2017).

Em outro trabalho, Seera (SEERA *et al.*, 2015) apresentou uma rede Min-Max para clusterização. Esta rede utiliza a mesma função de pertinência usada por Simpson em (SIMPSON, 1993). No entanto, a rede implementa uma restrição extra para que um hiper-retângulo possa incorporar uma amostra. Para tal, calcula-se o centro de todos os hiper-retângulos usando-se a média recursiva das amostras incorporadas pelo hiper-retângulo. Após a fase de contração, é verificado se os centros estão contidos no interior dos respectivos hiper-retângulos. Se essa condição for violada, o hiper-retângulo mantém a configuração anterior, e a nova amostra forma um novo hiper-retângulo. Esta ideia é ilustrada na Figura 2.9. Neste exemplo, as estrelas azul e verde representam os centros dos respectivos hiper-retângulos. Com a chegada da amostra amarela, um dos hiper-retângulos se expande para incorporá-la, gerando uma sobreposição. O processo de contração é realizado conforme mostra a Figura 2.9(c), fazendo com que o centro azul disponha-se no exterior do hiper-retângulo. Como isto caracteriza a violação da regra, os dois hiper-retângulos voltam a ter na Figura 2.9(d) a mesma configuração da Figura 2.9(a), e a nova amostra passa a ser um novo hiper-retângulo.

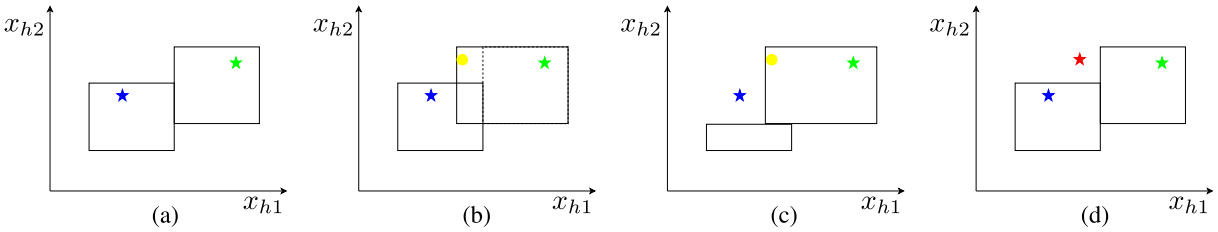


Figura 2.9 – Regra de contração de Seera (SEERA *et al.*, 2015)

2.4 Técnicas que não realizam contração

No algoritmo original de Simpson, o problema de conflito entre hiper-retângulos de classes diferentes é solucionado utilizando a abordagem de detecção de sobreposição e contração, como citado anteriormente. No entanto, alguns autores criticaram esta técnica, pelo fato de que ela pode degradar o conhecimento existente na estrutura da rede, além de tornar o desem-

penho do algoritmo muito dependente do tamanho máximo do hiper-retângulo, θ. (NANDEDKAR; BISWAS, 2007), (BARGIELA *et al.*, 2004).

Para evitar as críticas citadas acima, uma alternativa ao processo de contração foi utilizada (NANDEDKAR; BISWAS, 2007), (ZHANG *et al.*, 2011). Nesta nova técnica, um neurônio especial é utilizado nas regiões de interseção entre hiper-retângulos de classes diferentes, eliminando-se a necessidade de contraí-los, assim como ilustra a Figura 2.10. O neurônio especial tem as mesmas dimensões da área de conflito e uma função de pertinência própria para definir a qual classe pertence uma amostra contida em seu interior. No trabalho (ZHANG *et al.*, 2011), os neurônios tradicionais recebem o nome CLN (*Classifying Neuron*), e os neurônios especiais, OLN (*Overlap Neuron*).

Apesar da abordagem de Zhang resolver o problema da sobreposição entre hiper-retângulos sem realizar contração, ela precisa de neurônios adicionais para este propósito, aumentando a complexidade da rede. Para contornar este problema, uma abordagem semelhante àquela de Zhang, mas que não utiliza neurônios adicionais, é sugerida em (MA *et al.*, 2012).

2.5 Abordagens Min-Max *offline*

Existem algumas redes Min-Max que, apesar de utilizar o hiper-retângulo como neurônio, utilizam algoritmos de aprendizagem bem diferentes daquele proposto por Simpson. Alguns exemplos são os trabalhos baseados na técnica *Adaptive Resolution Classifier* - ARC (RIZZI *et al.*, 2002).

O algoritmo ARC cria inicialmente um grande hiper-retângulo H_0 que contém todas as amostras de um conjunto de treinamento com todas as classes. Este hiper-retângulo inicial é então sucessivamente dividido, a fim de separar amostras de classes diferentes em hiper-retângulos distintos. O objetivo final é obter hiper-retângulos que contenham amostras de somente uma classe em seu interior. Um algoritmo muito parecido com o ARC é sugerido em (TAGLIAFERRI *et al.*, 2001), o qual foi denominado TDFMM (*Top down fuzzy Min-Max*). Nesta rede, cada hiper-retângulo tem um centroide e uma função de pertinência Gaussiana.

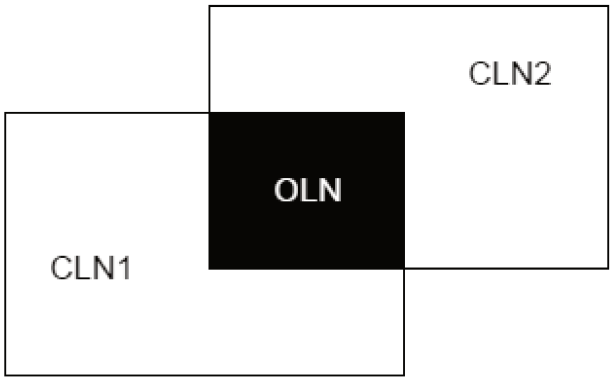


Figura 2.10 – Neurônio especial OLN

O nível de pertinência decai de acordo com a distância para o centroide, assumindo valores inferiores a 1 no interior do hiper-retângulo, o que difere da abordagem original de Simpson.

O trabalho (SHINDE *et al.*, 2015) descreve uma técnica para extrair regras nebulosas do tipo *if-then* de uma rede GFMN previamente treinada (ver seção 2.3). Estas regras permitem a descrição dos critérios de classificação adotados pela rede em formas de palavras em vez de números, tornando-as mais interpretáveis.

2.6 Redes Min-Max de regressão

Comparando-se com as redes Min-Max aplicadas à classificação, existem poucas publicações dedicadas à regressão. Uma técnica *offline* é apresentada em (TAGLIAFERRI *et al.*, 2001). O algoritmo de aprendizado desta rede é dividido em duas etapas, o ajuste estrutural e o ajuste paramétrico. O ajuste estrutural tem o objetivo de encontrar as regras nebulosas que compõem o sistema. Por outro lado, o ajuste paramétrico tem a função de encontrar os parâmetros das funções de pertinência e os pesos associados a cada regra. Para a primeira etapa, utiliza-se a rede TDFMM, proposta no mesmo artigo e citada na Seção 2.5. A segunda etapa é executada pelo algoritmo de gradiente, com o erro quadrático como função objetivo.

Outra rede que utiliza a técnica Min-Max para regressão é descrita em (MASCIOLI; MARTINELLI, 1998). Nesta técnica, no entanto, a rede Min-Max é utilizada apenas para a identificação do centro dos *clusters* de dados. Estes centros são posteriormente utilizados para construir uma rede ANFIS (*Adaptive Neuro-Fuzzy Inference System*) (JANG, 1993).

2.7 Resumo

Este capítulo apresentou o aprendizado Min-Max, a motivação por trás desta técnica e o algoritmo original de Simpson. Trabalhos que atualizaram o algoritmo original ou utilizaram o neurônio proposto por Simpson em algoritmos diferentes também foram discutidos. O capítulo resumiu diversas técnicas de classificação e clusterização, mas poucas de regressão, devido à escassez de publicações Min-Max dedicadas a este fim. Esse fato é uma das motivações para o desenvolvimento dessa dissertação. O próximo capítulo apresenta os métodos nebulosos evolutivos, que é a segunda categoria em que se enquadra o modelo proposto neste trabalho.

3 Métodos Nebulosos Evolutivos

Este capítulo apresenta os métodos nebulosos evolutivos. A Seção 3.1 introduz o tema, enquanto a Seção 3.2 descreve com maior detalhe uma categoria específica de modelos. A Seção 3.3 traz uma revisão da literatura de métodos evolutivos, dando ênfase a três modelos em particular. Por fim, a Seção 3.4 resume os conceitos abordados neste capítulo.

3.1 Introdução

O conceito de sistemas nebulosos evolutivos surgiu por volta do início deste século para suprir a demanda por sistemas flexíveis, robustos e interpretáveis, para aplicações na indústria, sistemas autônomos, sensores inteligentes, etc, (ANGELOV, 2008). Estas aplicações requerem o processamento de grandes fluxos de dados em tempo real, o que exige um algoritmo rápido e recursivo (aprender com apenas uma apresentação do conjunto de dados), incremental (processar apenas um dado de cada vez), eficiente em termos de uso de memória (não armazenar todos os dados antigos) e adaptativo (ajustar a estrutura do modelo na presença de *data shifts* e *data drifts*) (BARUAH; ANGELOV, 2011). Estas propriedades não estão presentes nos métodos offline, que exigem o conjunto completo de dados antes do início do treinamento, não sendo capazes de se adaptar a novas informações. Neste cenário, os sistemas evolutivos ganharam destaque, pois apresentam todas as características para suprir a demanda por processamento em tempo real, além de ter desempenhos competitivos com métodos clássicos offline.

Os sistemas nebulosos evolutivos são modelos matemáticos específicos que identificam relações funcionais entre dois conjuntos de dados, sendo eles os dados de entrada e os de saída. O princípio de funcionamento de um sistema nebuloso é dividir o espaço de entrada em diversas regiões, e atribuir uma regra nebulosa a cada região. Uma regra determina uma relação local entre o espaço de entrada e o de saída, que é válida na região de influência da regra. De uma maneira geral, uma regra nebulosa tem o seguinte formato:

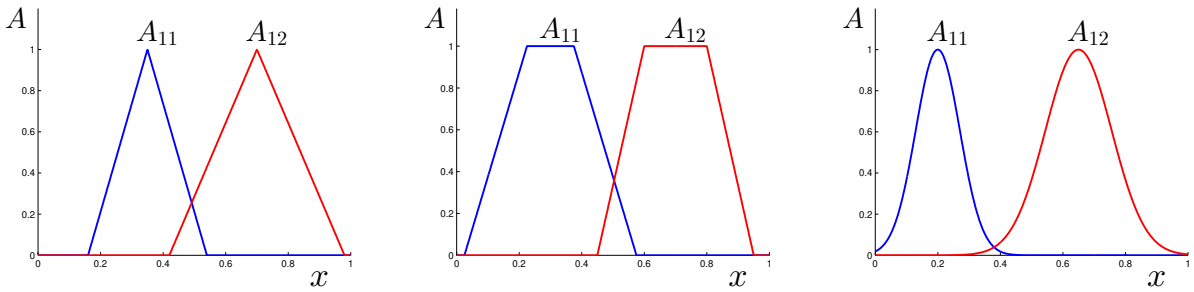
$$R_i\text{: SE } x_1 \text{ é } A_{1i} \text{ E } x_2 \text{ é } A_{2i} \ldots \text{ E } x_n \text{ é } A_{ni} \text{ ENTÃO } \bar{y} = B_i \tag{3.1}$$

onde R_i é a i-ésima regra nebulosa, x_j é a j-ésima componente do vetor de entrada, n é a dimensão do espaço de entrada e $\bar{y}$ é a saída da regra R_i. Cada um dos termos A_{ji} representam uma função de pertinência da regra R_i na dimensão j, sendo que cada uma dessas funções atribui um nível de pertinência para a componente x_j correspondente. Posteriormente, estes n níveis de pertinência unidimensionais passam por uma operação de agregação para se obter o nível de ativação da regra R_i. Este processo será detalhado na próxima seção. O termo B_i é a saída determinada localmente pela regra R_i. Se B_i for um conjunto nebuloso, diz-se que esse

sistema é um modelo nebuloso linguístico, enquanto se B_i for uma função das variáveis de entrada, o sistema recebe o nome de modelo nebuloso funcional. A saída do sistema é obtida pela agregação das contribuições locais das R regras existentes. Este conjunto de R regras recebe o nome de base de regras.

Outros dois conceitos importantes que se referem às regras nebulosas são os antecedentes e os consequentes. Os antecedentes designam a parte da regra associada ao espaço de entrada, sendo portanto a estrutura que delimita a zona de influência desta regra. Os consequentes, no entanto, dizem respeito à saída da regra, sendo formados por constantes, conjuntos nebulosos ou relações funcionais, assim como explicado anteriormente. Em (3.1), os termos A_{ji} formam o antecedente da regra R_i, enquanto o termo B_i representa o consequente desta mesma regra.

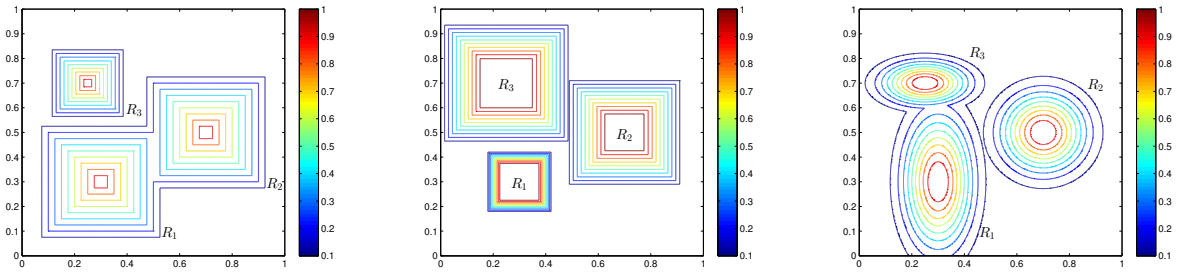
A região de influência de uma regra é delimitada pela função de pertinência associada a ela. Esta função tem imagem no intervalo $[0,1]$, sendo que o valor 1 é atribuído para regiões do espaço de entrada em total conformidade com a condição descrita pela regra em questão, enquanto o valor 0 é atribuído a regiões em que a regra não exerce nenhuma influência. Funções de pertinência de regras diferentes podem se sobrepor umas às outras, desde que não haja sobreposição entre regiões de pertinência completa (valor igual a 1). As funções de pertinência podem ter diversos formatos, assim como ilustra a Figura 3.1. A Figura 3.2, em contrapartida, mostra três regras em um espaço de entrada bidimensional, associadas aos três tipos de função ilustrados na figura anterior. Os níveis de pertinência estão associados às cores indicadas na barra lateral em cada imagem. Nesta última imagem, os centros das três regras estão fixos ($[0.3\ 0.3; 0.7\ 0.5; 0.25\ 0.7]$), mas as regiões de influência são distintas em cada um dos três casos, formando sobreposições diferentes entre as regras.



(a) Triangular (b) Trapezoidal (c) Gaussiana

Figura 3.1 – Exemplos de função de pertinência

É importante fazer um comparativo entre os sistemas nebulosos e os algoritmos do tipo *Piecewise Linear* (JOHANSSON, 2002). Este último consiste na divisão do espaço de entrada em subespaços de fronteiras rígidas, sendo que a saída dentro destes subespaços é formada exclusivamente por uma relação linear local. Nos sistemas nebulosos, entretanto, o espaço é dividido em regiões de fronteiras difusas, que se sobrepõem umas às outras, e a saída



(a) Triangular (b) Trapezoidal (c) Gaussiana

Figura 3.2 – Regras no espaço de entrada

é formada pela combinação de saídas individuais das regras, permitindo transições suaves entre estes modelos locais. Além disso, esta abordagem permite construir relações entre a entrada e a saída com menor número de regras (TAKAGI; SUGENO, 1985).

Quando um modelo nebuloso recebe a qualificação de evolutivo, isto significa que ele pode modificar a estrutura da base de regras dinamicamente, usando um fluxo de dados de entrada. Para processar esse fluxo de dados, o modelo é dotado de um algoritmo de aprendizado que tem duas funções principais: atualização dos antecedentes e dos consequentes. A atualização dos antecedentes envolve a criação, modificação, exclusão e, em alguns casos, fusão de regras. Para isso, são utilizados algoritmos de agrupamento, que decidem se o dado mais recente pode ser representado adequadamente pelas regras existentes, ou se é necessário criar uma nova regra. O ajuste dos consequentes, por outro lado, consiste na atualização das saídas das regras. O algoritmo de aprendizado também pode ser dotado de métricas de avaliação da qualidade da base de regras, a fim de identificar regras obsoletas, que podem ser excluídas, e regras redundantes, que podem ser unidas em uma única regra.

Deve-se notar que o termo evolutivo utilizado neste contexto difere do termo evolutivo (ou evolucionário) comumente usado em algoritmos inspirados na teoria da evolução das espécies (e.g. algoritmos genéticos), uma vez que o primeiro refere ao processo contínuo de auto-desenvolvimento de uma entidade individual, enquanto o segundo está associado à geração de populações de indivíduos por reprodução, mutação e seleção natural (BARUAH; ANGELOV, 2011).

A próxima seção explica os modelos nebulosos funcionais, que é a categoria em que se encontra o algoritmo proposto nesta dissertação.

3.2 Modelos nebulosos funcionais

Em um modelo nebuloso funcional, os antecedentes são termos linguísticos, enquanto os consequentes são funções das variáveis de entrada. De uma forma geral, uma regra

de um modelo nebuloso funcional tem o seguinte formato:

$$R_i\text{: SE } x_1 \text{ é } A_{1i} \text{ E } x_2 \text{ é } A_{2i} \ldots \text{ E } x_n \text{ é } A_{ni} \text{ ENTÃO } \bar{y} = f_i(x) \tag{3.2}$$

onde R_i é a i-ésima regra nebulosa, x_j é a j-ésima componente do vetor de entrada $x = [x_1, x_2, \ldots, x_n]$, n é a dimensão do espaço de entrada e $\bar{y}$ é a saída da regra R_i. A_{ji} é a função de pertinência correspondente à j-ésima componente do antecedente da i-ésima regra.

A saída $\bar{y}$ é gerada por uma função f_i arbitrária das variáveis de entrada. Os tipos mais comuns de funções, no entanto, são polinômios. Neste caso, o grau do polinômio indica a ordem do modelo funcional. Quando o polinômio tem grau zero, o modelo é chamado de funcional de ordem zero. Os modelos funcionais recebem o nome de Takagi Sugeno (TS) (TAKAGI; SUGENO, 1985). Os modelos TS são amplamente utilizados nos algoritmos evolutivos, pois além de se mostraram bastante eficazes na modelagem de sistemas não lineares, podem ser ajustados recursivamente pela técnica de quadrados mínimos recursivos.

Para uma entrada x, o nível de ativação da regra R_i é obtido usando uma operação entre os níveis de pertinência de cada componente de x conforme:

$$\tau_i = A_{1i}(x_1)\ t\ A_{2i}(x_2)\ t\ \ldots\ t\ A_{ni}(x_n) = \underset{j=1}{\overset{n}{T}} A_{ji}(x_j) \tag{3.3}$$

A operação descrita pela expressão (3.3) é realizada por uma *t-norma*. Uma *t-norma* é um operador binário $t : [0,1] \times [0,1] \to [0,1]$ que satisfaz as seguintes condições (PEDRYCZ; GOMIDE, 2007):

1. Comutatividade: $atb = bta$
2. Associatividade: $at(btc) = (atb)tc$
3. Monotonicidade: Se $b \leq c$, então $atb \leq atc$
4. Condições de fronteira: $at1 = a$, $at0 = 0$

onde $a, b, c \in [0,1]$. As *t-normas* mais utilizadas na literatura de modelos nebulosos são o mínimo e o produto, sendo que nestes casos a operação descrita por (3.3) é:

$$\tau_i = min\left(A_{1i}(x_1), A_{2i}(x_2), \ldots, A_{ni}(x_n)\right) \tag{3.4}$$

$$\tau_i = A_{1i}(x_1)A_{2i}(x_2)\ldots A_{ni}(x_n) \tag{3.5}$$

O intervalo em que $\tau_i > 0$ define a região de influência da regra i, ou seja, é a região do espaço de entrada em que a relação funcional $f_i(x)$ é válida.

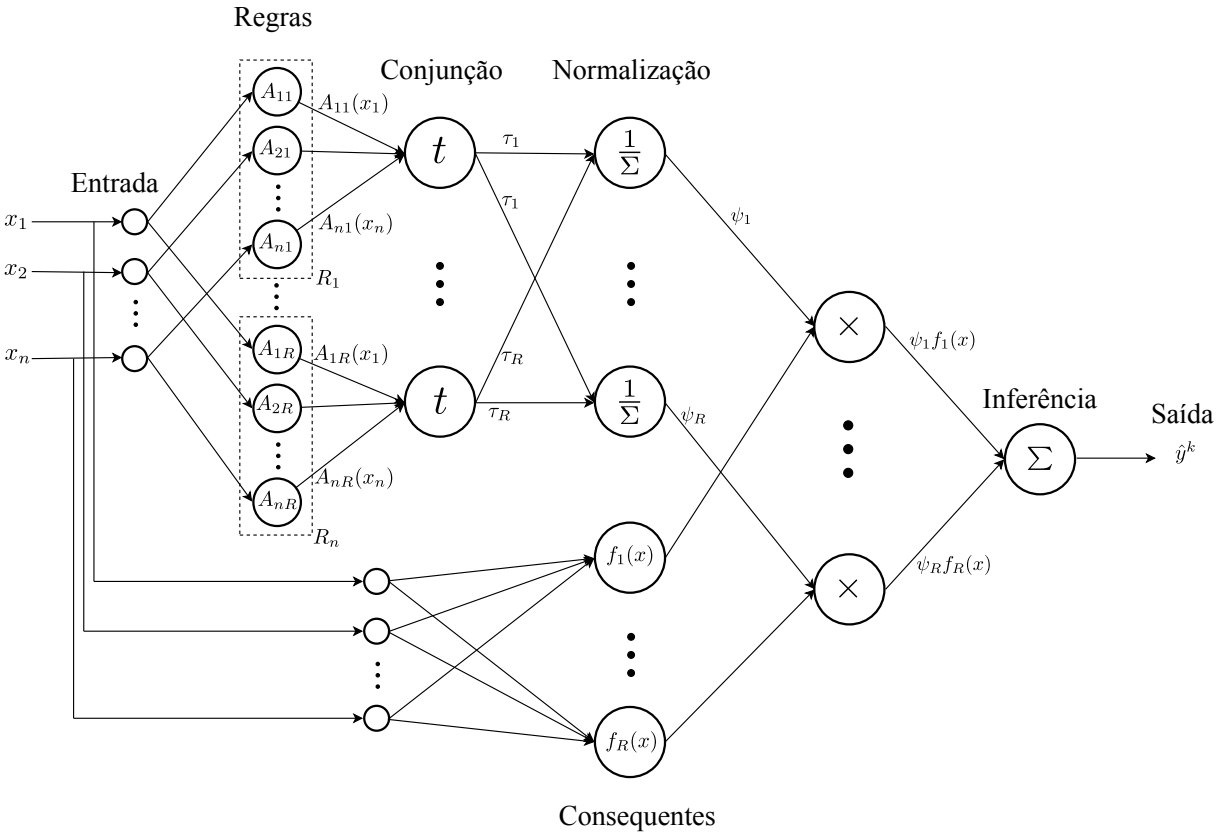


Figura 3.3 – Modelo nebuloso funcional

Após obtidos os níveis de ativação τ_i, é necessário agregar estes R níveis para inferir a saída do modelo. Para isto, combina-se ponderadamente as saídas das R regras, sendo que o peso desta combinação é dado pela ativação normalizada de cada uma delas:

$$\psi_i = \frac{\tau_i}{\sum_{l=1}^{R} \tau_l} \tag{3.6}$$

$$\hat{y} = \sum_{i=1}^{R} \psi_i f_i(x) \tag{3.7}$$

onde $\hat{y}$ é a saída do modelo e ψ_i é o nível de ativação normalizado da regra i.

A Figura 3.3 ilustra a arquitetura de um modelo nebuloso funcional.

3.3 Revisão da literatura

A seguir, esta seção detalha três algoritmos relevantes na literatura de sistemas nebulosos, sendo eles o eTS (Seção 3.3.1), o ePL (Seção 3.3.2), e o FLEXFIS (Seção 3.3.3), juntamente com as suas respectivas atualizações de acordo com o estado da arte na área. Posteriormente, a Seção 3.3.4 apresenta brevemente outras estratégias evolutivas.

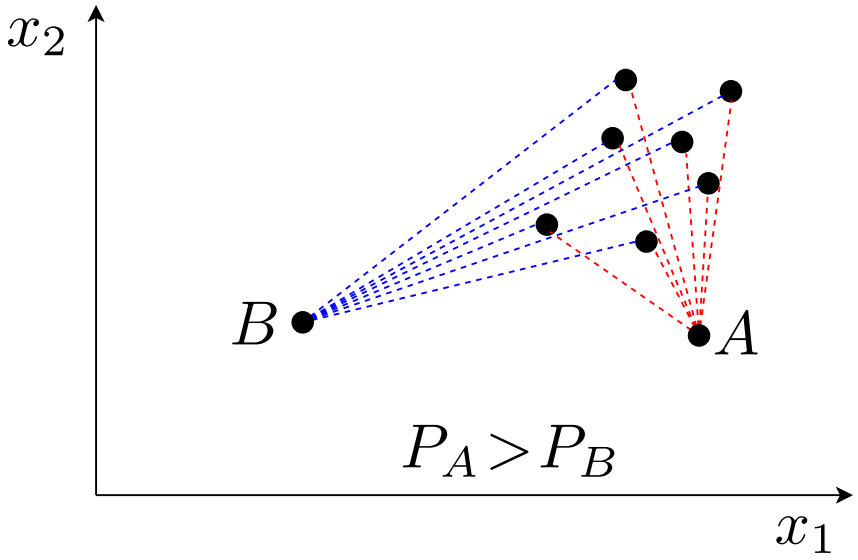


Figura 3.4 – Potencial

3.3.1 eTS

O trabalho (ANGELOV; FILEV, 2004) apresentou um método para criar regras do tipo Takagi-Sugeno recursivamente, a partir de um fluxo de dados de entrada. Denominado *Evolving Takagi Sugeno* (eTS), o algoritmo modifica os antecedentes das regras a partir de um agrupamento não supervisionado, adicionando novas regras ou modificando alguma existente. Em cada instante, deve-se decidir como a base de regras será atualizada, ajustando os parâmetros de acordo com o dado mais recente (ANGELOV, 2002). O eTS baseia-se na ideia de que a representatividade de um dado pode ser medida por uma função que define o potencial relativo à cada centro de grupo (YAGER; FILEV, 1994). O potencial representa uma medida de proximidade espacial entre um ponto z^k e os demais pontos. O potencial é calculado por:

$$P(z^k) = \frac{1}{1 + \frac{1}{k-1}\sum_{l=1}^{k-1}\sum_{j=1}^{n+1}\left(z_j^k - z_j^l\right)^2} \tag{3.8}$$

onde $z^k = [(x^k)^T, (y^k)]^T$, sendo x^k a entrada e y^k a saída no passo k. A expressão (3.8) é utilizada no eTS pela possibilidade de ser transformada em uma expressão recursiva, mas existem outras propostas de funções para o cálculo do potencial (ANGELOV; FILEV, 2004), (ANGELOV; ZHOU, 2006).

A expressão (3.8) atribui valores mais altos para dados situados em regiões de densidade de dados maior. A Figura 3.4 ilustra este conceito. Nesta figura, é possível ver que o ponto A situa-se mais próximo dos demais pontos do que o ponto B, e por isso $P_A > P_B$. Quanto mais concentrados estiverem os dados em uma certa região, maior será o potencial neste local. Intuitivamente, essa concentração de dados é caracterizada por pontos próximos entre si e, portanto, o potencial é uma função monotônica e inversamente proporcional às distâncias entre as observações (ANGELOV; FILEV, 2004).

A principal característica do algoritmo de agrupamento do eTS baseia-se na premissa de que pontos com maior potencial são melhores candidatos a se tornar centros de grupos. Esta abordagem baseia-se no algoritmo *Subtractive Clustering* (CHIU, 1994), que é uma técnica *offline* que realiza clusterização usando a função potencial. O algoritmo de agrupamento do eTS é uma extensão do *Subtractive Clustering* para o caso *online*. Para isso, a expressão (3.8) é modificada para se tornar recursiva:

$$P(z^k) = \frac{k-1}{(k-1)(\nu^k+1)+\sigma^k-2\upsilon^k} \tag{3.9}$$

$$\nu^k = \sum_{j=1}^{n+1}\left(z_j^k\right), \quad \sigma^k = \sigma^{k-1} + \sum_{j=1}^{n+1}\left(z_j^{k-1}\right)^2, \quad \upsilon^k = \sum_{j=1}^{n+1} z_j^k \beta_j^k, \quad \beta_j^k = \beta_j^{k-1} + z_j^{k-1} \tag{3.10}$$

A cada passo, o eTS calcula o potencial do dado atual z^k e realiza uma ação, de acordo com uma das seguintes condições (ANGELOV; ZHOU, 2006), (LIMA, 2008):

1. se uma observação possuir o potencial maior que o potencial de todos os centros atuais, então esse ponto será um centro de grupo;
2. se, além da condição anterior, o potencial do ponto for próximo o suficiente ao potencial de algum centro, então esse centro será substituído por essa observação;
3. caso contrário, a base de regras permanece como está, sem modificação.

Esta abordagem apresenta algumas características importantes. Uma delas é que as atualizações solicitadas pelas Condições 1 e 2 buscam a criação de grupos cada vez mais representativos, com maior poder de generalização. Isto ocorre porque, para um ponto se tornar um novo centro de grupo, é necessário que ele apresente um potencial maior do que todos os centros de grupos existentes. Além disso, se o ponto z^k também estiver próximo de um centro de grupo, este centro é substituído, o que torna a base de regras mais concisa. A proximidade entre um centro de grupo e o ponto z^k, de que se trata a Condição 2, é constatada por:

$$\frac{P(z^k)}{\max\limits_{1\leq l\leq R} P^k(c_l)} - \frac{\delta_{min}}{r} \geq 1 \tag{3.11}$$

onde R é o número de regras na base, $r = [0.3, 0.5]$ é uma constante que define o raio da regra e δ_{min} é a distância Euclidiana entre o ponto z^k e o centro de grupo mais próximo.

Outra característica interessante do algoritmo de agrupamento do eTS é a robustez na presença de *outliers*. Como os *outliers* apresentam, geralmente, grande distância com relação aos demais dados, o seu potencial é baixo, e a chance dele se tornar um centro de grupo é mínima.

Após realizar uma das três ações descritas acima, o modelo atualiza o potencial dos centros de todos os grupos existentes recursivamente:

$$P^k(c_i) = \frac{(k-1)P^{k-1}(c_i)}{k-2+P^{k-1}(c_i)+P^{k-1}(c_i)\sum_{j=1}^{n+1}\left(c_{ji}^k - c_{ji}^{k-1}\right)^2} \tag{3.12}$$

onde c_{ji}^k é a j-ésima componente do i-ésimo centro no instante k.

A razão para a atualização do potencial dos centros dos grupos a cada instante é que o potencial depende da proximidade de um ponto com relação a todos os demais pontos, e portanto, o potencial é influenciado pelos novos pontos adquiridos pelo modelo.

Após a atualização dos antecedentes, o modelo atualiza os consequentes das regras. Para isso, utiliza-se uma modalidade de quadrados mínimos recursivos ponderados pelo nível de pertinência (*fuzzily weighted recursive least squares* - fwRLS) (ANGELOV; FILEV, 2004), (LUGHOFER, 2011). Nesta abordagem, as equações para o ajuste recursivo dos parâmetros são:

$$\theta_i^k = \theta_i^{k-1} + Q_i^k \bar{x}^k \psi_i(x^k)\left(y^k - (\bar{x}^k)^T\theta_i\right) \tag{3.13}$$

$$Q_i^k = Q_i^{k-1} - \frac{Q_i^{k-1}\bar{x}^k\left(\bar{x}^k\right)^T Q_i^{k-1}}{\frac{1}{\psi_i(x^k)} + (\bar{x}^k)^T Q_i^{k-1}\bar{x}^k} \tag{3.14}$$

onde $Q^0 = \omega E$, $\omega = [10^3, 10^5]$, E é a matriz identidade, $\bar{x} = [1, x_1, x_2, ..., x_n]$ é o vetor de entrada estendido e $\psi_i(x^k)$ é o nível de pertinência normalizado do ponto x^k para a regra i (que é equivalente ao nível de ativação normalizado desta mesma regra).

A saída do modelo é calculada pela média ponderada das saídas individuais de cada regra, onde o peso desta ponderação é, novamente, o nível de pertinência normalizado:

$$\mu_{ji} = exp\left(\frac{-4}{r^2}(x_j - c_{ji})\right)^2, \quad \tau_i = \mu_{1i}\ t\ \mu_{2i}\ t\ ...\ t\ \mu_{ni} \tag{3.15}$$

$$\hat{y}^{k+1} = \sum_{i=1}^{R}\psi_i\theta_i^T\bar{x}, \quad \psi_i = \frac{\tau_i}{\sum_{l=1}^{R}\tau_l}, \quad \bar{x} = [1, x_1, x_2, ..., x_n]^T \tag{3.16}$$

onde μ_{ji} é o nível de pertinência na j-ésima dimensão, t representa uma t-norma (e.g. mínimo, produto) e θ_i é o vetor contendo os parâmetros do consequente. O algoritmo 3.1 resume o eTS.

O trabalho (ANGELOV; FILEV, 2005) apresentou uma atualização do algoritmo eTS. Denominado Simpl_eTS, a principal contribuição deste trabalho é uma abordagem que

3.1 Algoritmo eTS

1: Definir o raio r
2: Inicializar o centro $c_1 = z^1$
3: **Para** $k = 1,2,3,...$ **fazer**
4: Ler o ponto de entrada z^k
5: Calcular o potencial do ponto z^k recursivamente:
6:

$$P(z^k) = \frac{k-1}{(k-1)(v^k+1)+\sigma^k-2\upsilon^k}$$

$$v^k = \sum_{j=1}^{n+1}\left(z_j^k\right), \quad \sigma^k = \sigma^{k-1} + \sum_{j=1}^{n+1}\left(z_j^{k-1}\right)^2, \quad \upsilon^k = \sum_{j=1}^{n+1} z_j^k \beta_j^k, \quad \beta_j^k = \beta_j^{k-1} + z_j^{k-1}$$

7: Atualizar o potencial dos centros dos grupos existentes:
8:

$$P^k(c_i) = \frac{(k-1)P^{k-1}(c_i)}{k-2+P^{k-1}(c_i)+P^{k-1}(c_i)\sum_{j=1}^{n+1}\left(c_{ji}^k - c_{ji}^{k-1}\right)^2}$$

9: Comparar o potencial do ponto z^k com o potencial atualizado dos centros dos grupos
10: **Se** $P(z^k) > P(c_i)$ para $i = 1,2,...,R$ **E** $\frac{P(z^k)}{\max\limits_{1\le l\le R} P^k(c_l)} - \frac{\delta_{min}}{r} \ge 1$ **Então**
11: O ponto z^k substitui o centro c_i, $c_i = z^k$, $P^k(c_i) = P(z^k)$
12: **Senão Se** $P(z^k) > P(c_i)$ **Então**
13: Ponto z^k é ponto focal de um novo grupo: $R = R+1$, $c_R = z^k$, $P^k(c_i) = P(z^k)$
14: **Senão** Os grupos ficam inalterados
15: **Fim Se**
16: Atualizar os consequentes das regras usando fwRLS (expressões (3.13) e (3.14))
17: Gerar a saída: $\hat{y}^{k+1} = \sum_{i=1}^{R} \psi_i \theta_i^T \bar{x}$, $\psi_i = \frac{\tau_i}{\sum_{l=1}^{R}\tau_l}$, $\bar{x} = [1, x_1, x_2, ..., x_n]^T$
18: **Fim Para**

simplifica os cálculos realizados no eTS. Para isso, o conceito de potencial dos dados é substituído pela ideia de dispersão:

$$S(z^k) = \frac{1}{N(n+m)} \sum_{l=1}^{N}\sum_{j=1}^{n+m}\left(z_j^k - z_j^l\right)^2 \tag{3.17}$$

onde $S(z^k)$ é a dispersão dos dados ao redor do ponto z^k, N é o total de pontos, e n e m são as dimensões de entrada e saída, respectivamente.

A expressão (3.17) tem valor máximo de 1, quando todos os pontos se coincidem no mesmo local do espaço, e valor mínimo de 0, quando o ponto z^k se encontra no vértice do hipercubo unitário em que todas as coordenadas são iguais a 1 e todos os demais pontos se situam no vértice oposto, onde as componentes são iguais a 0. Esta expressão é mais simples computacionalmente do que a função de potencial, pois apresenta uma divisão sobre números inteiros,

enquanto a função potencial apresenta divisão sobre números reais (ANGELOV; FILEV, 2005).

O algoritmo de agrupamento do Simpl_eTS cria novas regras toda vez que o dado mais recente possui dispersão $S(z^k)$ maior do que as dispersões de todos os centros de grupos ou quando $S(z^k)$ for menor do que todas as demais dispersões:

$$S(z^k) > \max_{1\leq l\leq R} S(c_l) \textbf{ OU } S(z^k) < \min_{1\leq l\leq R} S(c_l) \tag{3.18}$$

Se z^k possuir grande distância para os centros de grupos c_i, o valor $S(z^k)$ tende a ser alto, podendo satisfazer a primeira condição indicada em (3.18). Por outro lado, se $S(z^k)$ for menor do que todas as dispersões $S(c_i)$ (o que satisfaz a segunda condição em (3.18)), provavelmente a concentração de dados é mais alta ao redor de z^k do que nas imediações dos centros de grupos. Ambas as situações habilitam z^k como um candidato a formar um novo grupo, e isto justifica a abordagem descrita pela Condição (3.18). Um aspecto negativo do uso desta condição com relação ao uso do potencial é que o algoritmo perde robustez frente a outliers. Como estes pontos têm grandes distâncias para os grupos existentes, eles podem apresentar alto valor $S(z^k)$, e, portanto, satisfazer a Condição (3.18).

O algoritmo Simpl_eTS também introduziu uma técnica de exclusão de regras para simplificar a estrutura do modelo. O critério de exclusão consiste em contar o número de pontos associados a uma regra (N_i), e caso este número represente uma fração muito pequena do total de pontos recebidos pelo modelo (Por exemplo, $(N_i/N) < 0,01$), então a regra R_i é excluída da base de regras. Esta funcionalidade atenua a desvantagem descrita no parágrafo anterior.

Uma nova atualização do eTS foi apresentada em (ANGELOV; ZHOU, 2006). Esta técnica, que recebeu o nome de xTS (*Extended Takagi Sugeno*), também utiliza o potencial dos dados para realizar a clusterização no espaço de entrada/saída, e estendeu o eTS para o caso de múltiplas saídas (*MIMO - Multiple Input, Multiple Output*).

Outras contribuições trazidas neste trabalho foram a adaptação automática da zona de influência das regras e a apresentação de índices de qualidade de clusters. A adaptação das zonas de influência é realizada recursivamente:

$$\begin{aligned} r_{ji}^k &= \rho r_{ji}^{k-1} + (1-\rho)\sigma_{ji}^k \\ \sigma_{ji}^k &= \sqrt{\frac{1}{N_i^k}\sum_{l=1}^{N_i^k}\left(c_{ji}-x_j\right)^2} \end{aligned} \tag{3.19}$$

onde N_i^k é o número de pontos associados à regra i no passo k, ρ é uma constante (valor sugerido de 0,5) que representa a compatibilidade entre as informações novas e as antigas e σ_{ji}^k é a dispersão local dos dados no espaço de entrada, similar à variância nas distribuições de proba-

bilidade. Desta forma, a zona de influência das regras é constantemente ajustada utilizando-se a dispersão local dos dados, em vez de ser fixada por um valor r pré-definido pelo usuário.

Os índices de qualidade de cluster são utilizados para identificar as regras que, devido à natureza dinâmica e não estacionária da distribuição de dados, não são mais compatíveis com o estado atual do processo gerador de amostras. Desta maneira, estas regras perdem a sua representatividade e capacidade de generalização, e precisam ser excluídas ou ignoradas, para que o desempenho de previsão do algoritmo não se deteriore. Os dois índices de qualidade apresentados em (ANGELOV; ZHOU, 2006) são:

- Idade: Este valor está dentro do intervalo $(0,k]$, sendo que uma regra associada a dados recentes tem idade mais próxima de 0, enquanto regras associadas somente a dados antigos é considerada "mais velha", tendo idade mais próxima de k. A idade da regra i no passo k é calculada usando:

$$idade_i^k = k - \frac{2A_i^k}{(k+1)};\ \ i = [1,R] \tag{3.20}$$

$$A_i^k = \sum_{p=1}^{k} L_i^p,\ \ L_i^p = \begin{cases} p, & \text{se } \operatorname{argmin}_{1\leq l\leq R} \|z^p - c_l\|^2 = i \\ 0, & \text{se } \operatorname{argmin}_{1\leq l\leq R} \|z^p - c_l\|^2 \neq i \end{cases} \tag{3.21}$$

- Suporte: É o número de pontos associados a uma regra. Quanto maior o seu suporte, mais representativa é uma regra. O suporte da regra i é atualizado fazendo-se:

$$N_i^{k+1} = \begin{cases} N_i^k + 1, \text{ se } \operatorname{argmin}_{1\leq l\leq R} \left\|z^k - c_l\right\|^2 = i \\ N_i^k, \text{ caso contrário} \end{cases} \tag{3.22}$$

Outra proposta que atualizou o algoritmo eTS foi apresentada em (ANGELOV, 2010). Esta nova abordagem recebeu o nome de eTS+, e também utiliza o potencial para clusterizar os dados no espaço de entrada e saída, mas adicionou uma condição extra para a criação de novos grupos. Para formular esta nova condição, calcula-se recursivamente o número de pontos que tem baixo nível de pertinência para todas as regras existentes na base:

$$O^k = \begin{cases} O^{k-1} + 1, \text{ se } \left(z_j^k - c_{ji}\right) > 2\sigma_{ji}, \text{ para } \forall j = [1,2,...,n+m], \text{ e para } \forall i = [1,R] \\ O^{k-1}, \text{ caso contrário} \end{cases} \tag{3.23}$$

onde O^k é o número de pontos isolados (i.e. com nível de pertinência inferior a e^{-2} para todas as regras) no instante k, e σ_{ji} é a dispersão da regra i na dimensão j.

O fundamento por trás desta condição é que, se existe um grande número de pontos com baixo nível de pertinência para todas as regras, então a base de regras não está representando adequadamente a distribuição de dados atual. Desta forma, se o valor de O^k for alto, a

criação de novos grupos é estimulada, promovendo a renovação da base de regras. O controle é feito pelo coeficiente η:

$$\eta P(z^k) > \max_{1 \leq l \leq R} P(c_i) \tag{3.24}$$

$$\eta = \begin{cases} 1, \text{ se } \mu_{ji}(x^k) > e^{-2}, \forall j, \forall i \text{ ou } k < 3 \\ \frac{O^k - 3}{\log k}, \text{ caso contrário} \end{cases} \quad k = 2,3,... \tag{3.25}$$

onde $\eta > 1$ é o número normalizado de pontos isolados.

Se a Condição (3.24) for satisfeita, o ponto z^k se torna um novo centro de grupo. Se o número de pontos isolados cresce, o valor de η também cresce, o que aumenta as chances da condição descrita acima ser satisfeita.

Outra contribuição trazida neste trabalho foi outra métrica de qualidade de clusters, chamada de índice de utilidade. A utilidade da regra i no passo k é calculada usando:

$$U_i^k = \frac{\sum_{p=1}^{k} \psi^p}{k - I^{i*}}, i = 1,2,...,R \tag{3.26}$$

onde ψ^p é o nível de pertinência normalizado do ponto de índice p para a regra i, e I^{i*} é o índice do passo em que a regra i foi criada.

O valor da utilidade indica o quanto uma regra foi utilizada desde o momento da sua criação. As regras pouco utilizadas, naturalmente, tem valor de utilidade baixo, e são candidatas a exclusão. Em (ANGELOV, 2010), o critério para excluir uma regra é:

$$\textbf{SE } U < \varepsilon, \textbf{ ENTÃO } R = R - 1 \tag{3.27}$$

sendo $\varepsilon \in [0.03, 0.1]$ um limiar fornecido pelo usuário.

3.3.2 ePL

Os trabalhos (LIMA *et al.*, 2010) e (LIMA, 2008) apresentaram um algoritmo com uma proposta um pouco diferente do eTS. Chamada de *Evolving Participatory Learning* – ePL, esta técnica também processa os dados em tempo real, atualizando os antecedentes e consequentes a cada passo. O ePL, no entanto, não utiliza os conceitos de potencial ou dispersão para realizar o agrupamento. Em vez disso, utiliza-se o aprendizado participativo (YAGER, 1990), onde o ritmo de atualização da base de regras depende da compatibilidade entre o ponto mais recente x^k e a estrutura atual do modelo. Esta compatibilidade é medida por meio de dois coeficientes, sendo eles ρ e a. O primeiro é o índice de compatibilidade, que indica o quanto o ponto x^k é compatível com a estrutura atual da base de regras. O segundo coeficiente, chamado de índice de alerta, atua como um crítico à estrutura atual do modelo, indicando a necessidade

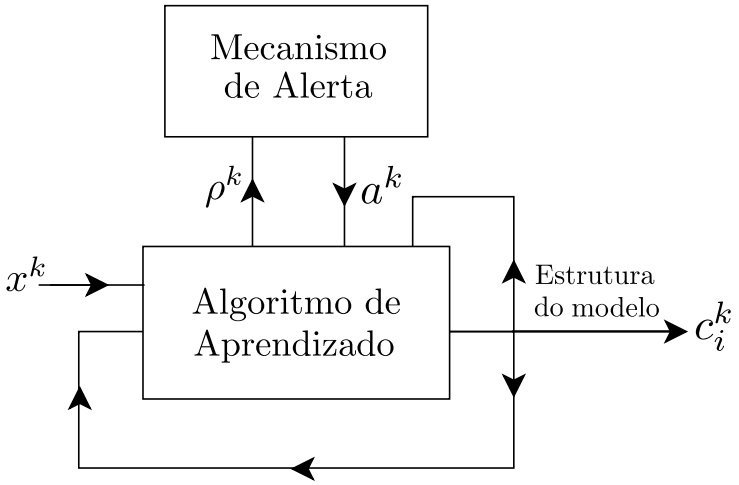


Figura 3.5 – Aprendizado participativo

de revisá-lo frente à nova informação trazida pelos dados mais recentes. O índice de alerta pode ser interpretado como um complemento à confiança na base de regras atual para representar a distribuição de dados (LIMA *et al.*, 2010). A Figura 3.5 ilustra o mecanismo utilizado pelo aprendizado participativo.

A cada passo, o algoritmo participativo pode criar uma nova regra ou modificar alguma existente. Para isso, o algoritmo calcula os índices de compatibilidade e de alerta utilizando x^k:

$$\rho_i^k = 1 - \frac{\left\|x^k - c_i^k\right\|}{n} \tag{3.28}$$

$$a_i^k = a_i^{k-1} + \beta\left(1 - \rho_i^k - a_i^{k-1}\right) \tag{3.29}$$

onde ρ_i^k é o índice de compatibilidade da regra i para o ponto x^k, a_i^k é o índice de alerta da regra i no passo k, n é a dimensão do espaço de entrada e $\beta \in [0,1]$ controla o ritmo de atualização do índice de alerta. Quanto mais próximo o valor de β for de 1, mais rápido o modelo capta variações de compatibilidade (LIMA *et al.*, 2010).

Caso o índice de alerta de todas as R regras seja maior do que um limiar $\tau \in [0,1]$, definido pelo usuário, então o modelo cria uma nova regra. Se essa condição não for satisfeita, o centro da regra mais influente é modificado da seguinte maneira:

$$i = \underset{l}{\operatorname{argmax}}\{\rho_l^k\}$$

$$c_i^k = (1-G)c_i^{k-1} + Gx^k \tag{3.30}$$

É possível verificar que o centro atualizado, c_i^k, é resultado de uma combinação convexa entre a sua configuração antiga (c_i^{k-1}) e o ponto mais recente (x^k). O peso dessa combinação convexa, G, é calculado da seguinte maneira:

$$G = \alpha \left(\rho_i^k\right)^{1-a_i^k} \tag{3.31}$$

onde $\alpha \in [0,1]$ é o parâmetro de aprendizado (i.e. *learning rate*). O valor de α é definido pelo usuário, e influencia na velocidade de atualização do modelo.

A expressão (3.31) demonstra que o parâmetro G é modulado pelo índice de alerta, $a_i^k \in [0,1]$. Desta forma, se $a_i^k = 0$, isto indica que a configuração atual da base de regras é totalmente compatível com a informação trazida por x^k, e $G = \alpha\left(\rho_i^k\right)$, acelerando o aprendizado. Por outro lado, se o índice de alerta possui valor elevado, a magnitude de G diminui, evitando que o centro c_i absorva a informação de x^k quando a compatibilidade for baixa. Esta abordagem protege a estrutura do modelo contra informações espúrias trazidas por pontos muito distantes dos centros de grupos, permitindo que o usuário escolha valores mais elevados para o parâmetro α (LIMA *et al.*, 2010).

Devido à natureza dinâmica de criação e atualização de regras, é possível que dois centros de grupos fiquem muito próximos um do outro, o que criaria uma redundância na base de regras. Por este motivo, o ePL tem um mecanismo de identificação de regras redundantes. Para isso, calcula-se a compatibilidade entre os pares de regras existentes no modelo, utilizando-se uma expressão similar a (3.28). Caso a compatibilidade entre um par de regras exceda um valor pré-estabelecido, esse par de regras é considerado redundante:

$$\rho_{il}^k = 1 - \sum_{j=1}^{n} \left\| c_i^k - c_l^k \right\| \tag{3.32}$$

$$\rho_{il}^k \geq \lambda \tag{3.33}$$

onde ρ_{il}^k é a compatibilidade entre as regras i e l e λ é um parâmetro definido pelo usuário. Caso a Condição (3.33) seja satisfeita, as regras i e l podem ser substituídas por uma regra resultante da combinação convexa entre ambas, ou uma das duas pode ser excluída.

Posteriormente, o trabalho (MACIEL *et al.*, 2012) apresentou atualizações para o ePL. Denominado *Enhanced Evolving Participatory Learning* – ePL+, este algoritmo propôs um método de avaliação constante da qualidade da base de regras e adaptação da zona de influência dos clusters.

A qualidade da base de regras é medida pelo mesmo método adotado pelo eTS+, calculando-se a utilidade das regras com (3.26) . Após este cálculo, as regras que possuírem utilidade menor do que um limiar pré-definido são excluídas da base de regras. Já a adaptação da zona de influência é realizada com o mesmo método adotado no algoritmo xTS, utilizando-se a expressão (3.19). Outra atualização trazida neste trabalho foi a extensão do ePL para o caso de múltiplas entradas e múltiplas saídas (*Multiple Input Multiple Output* - MIMO).

Outro algoritmo baseado no aprendizado participativo foi apresentado em (LEMOS *et al.*, 2011). Esta técnica recebeu o nome de *Multivariable Gaussian Evolving Fuzzy* – eMG, e apresentou regras nebulosas com formato de elipses n-dimensionais com eixos em qualquer direção do espaço. Isto é realizado pela seguinte função de pertinência:

$$H_i = exp\left[-\frac{1}{2}(x-c_i)^T\Sigma_i^{-1}(x-c_i)\right] \tag{3.34}$$

onde H_i é a função de pertinência associada à regra i e $\Sigma_i \in \mathbb{R}^{n\times n}$ é uma matriz simétrica e definida positiva. Os elementos de Σ_i definem a dispersão e orientação da elipse, enquanto o centro c_i define o ponto focal da função de pertinência (i.e. o ponto com nível máximo de pertinência). A Figura 3.6 ilustra a estrutura de duas regras no espaço bidimensional, com as respectivas curvas de nível e os pontos no espaço de entrada. Nas curvas, os tons vermelhos indicam níveis de pertinência mais elevados.

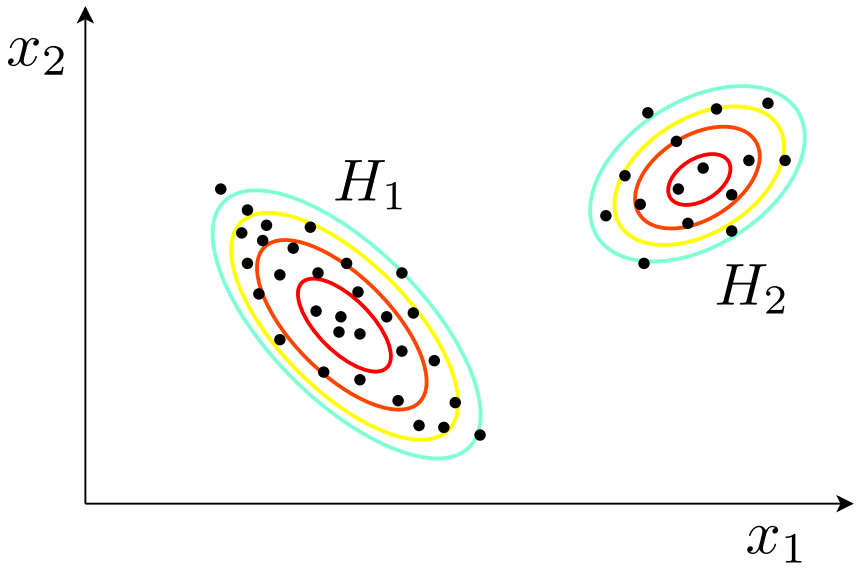


Figura 3.6 – Estrutura das regras no espaço de entrada

3.3.3 FLEXFIS

Outra abordagem de aprendizado nebuloso incremental é apresentada em (LUGHOFER, 2008). Este algoritmo recebeu o nome de FLEXFIS (*Flexible Fuzzy Inference System*), e utiliza uma versão de uma técnica chamada de quantização vetorial (GRAY, 1984) para construir regras nebulosas no espaço conjunto de entrada e saída. A essas regras nebulosas, o algoritmo associa funções de pertinência Gaussianas com matrizes de dispersão diagonais. Como resultado, a região do espaço com igual nível de pertinência para uma regra nebulosa tem um formato elipsoidal, com eixos paralelos às coordenadas de entrada/saída, assim como ilustra a Figura 3.7. Os eixos destas elipses tem comprimento de $2\sigma_{ji}$ entre o centro e a fronteira, sendo σ_{ji} a dispersão dos dados na direção j ao redor da regra i.

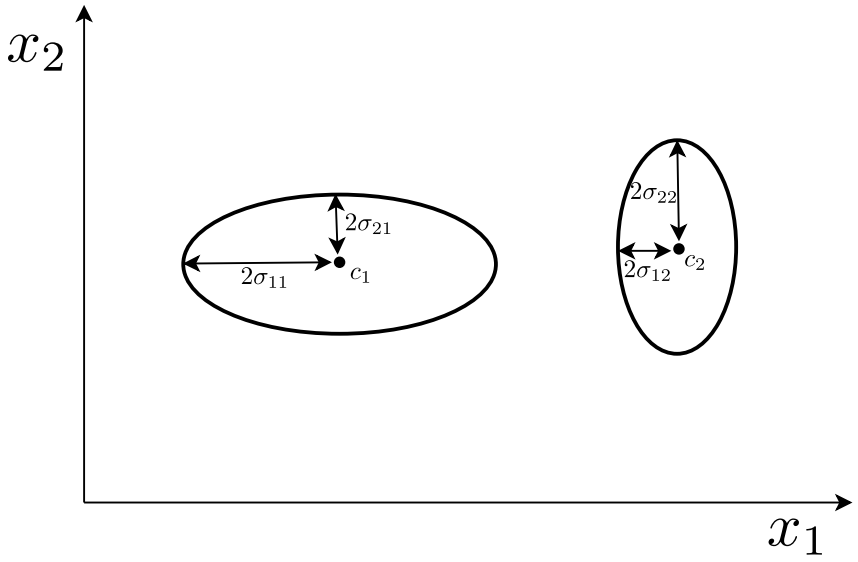


Figura 3.7 – Clusters elipsoidais e dispersões

A principal motivação do FLEXFIS é a criação de um modelo que associa adequadamente a adaptação dos parâmetros não lineares dos antecedentes com o ajuste dos parâmetros lineares dos consequentes (LUGHOFER, 2008). Esta abordagem tem o propósito de dimensionar os parâmetros do consequente considerando a natureza dinâmica da criação e modificação de regras, já que os quadrados mínimos convencionais assumem que os demais parâmetros são estáticos. Este procedimento é realizado aplicando correções ao vetor de parâmetros do consequente e à matriz Q (ver (3.13)), da seguinte forma:

$$\theta_i = \theta_i + \zeta_\theta \tag{3.35}$$

$$Q_i = Q_i + \zeta_Q \tag{3.36}$$

onde ζ_θ e ζ_P são o vetor e a matriz de correção, respectivamente. Após corrigir o vetor θ_i e a matriz Q_i, utiliza-se os quadrados mínimos ponderados convencionais para atualizar os consequentes, assim como mostra as expressões (3.13) e (3.14). Esta abordagem tem a vantagem de considerar as mudanças dos antecedentes no cálculo dos consequentes, mas tem a desvantagem de demandar o armazenamento de dados passados para calcular o vetor e matriz de correção, o que não é uma característica desejável para algoritmos incrementais. Após esta descrição sobre as atualizações dos consequentes no FLEXFIS, os próximos parágrafos explicam o algoritmo de agrupamento do modelo.

A cada novo dado recebido, o algoritmo FLEXFIS pode criar uma nova regra ou atualizar alguma previamente existente. A primeira etapa para decidir entre estas duas ações é encontrar a regra mais influente para o ponto mais recente $z^k = [\left(x^k\right)^T, \left(y^k\right)^T]^T$. O algoritmo FLEXFIS aplica um método diferente para escolher o cluster mais influente quando um novo dado é recebido. No lugar de se calcular a distância do dado até o centro do cluster, assim como ocorre no método de quantização vetorial, calcula-se a distância Euclidiana entre z^k e as

superfícies das regras existentes no modelo, tal como ilustra a Figura 3.8. Isto é realizado da seguinte maneira: Traça-se inicialmente uma reta ligando z^k até o centro c_i. Depois, encontra-se o ponto desta reta que intercepta a superfície da regra, e calcula-se a distância deste ponto até z^k. A seguinte equação realiza estes procedimentos:

$$dist_i = (1-t)\sqrt{\sum_{j=1}^{n+1}\left(x_j - c_{ji}\right)^2} \tag{3.37}$$

$$t = \frac{1}{\sqrt{\frac{\sum_{j=1}^{n+1}\left(x_j - c_{ji}\right)^2}{\sigma_{ji}^2}}} \tag{3.38}$$

onde $dist_i$ é a distância entre o dado mais recente z^k e a superfície da regra i. A justificativa para esta abordagem é que ela evita a geração desnecessária de clusters, principalmente nas redondezas de regras com grande zona de influência. No caso em que a distância entre z^k e o centro c_i da regra mais influente é muito grande, o algoritmo pode decidir criar uma nova regra, mesmo que z^k esteja dentro da zona de influência da regra i (LUGHOFER, 2008).

Após calcular todas as distâncias $dist_i$, $i = 1,2,...,R$, encontra-se a menor dentre elas $(l = \arg\min_i\{dist_i\})$, e considera-se a regra l como a mais influente. A distância para a regra mais influente é então comparada com um parâmetro ρ pré definido. Se $dist_l < \rho$, então z^k é utilizado para atualizar a regra l. Esta atualização envolve o ajuste do centro e da zona de influência. O ajuste recursivo do centro é realizado pela seguinte equação:

$$c_l = c_l + \eta_l(x^k - c_l) \tag{3.39}$$

$$\eta_l = \frac{init_{gain}}{k_l} \tag{3.40}$$

onde k_l é o contador de pontos incorporados pela regra l e $init_{gain}$ é um parâmetro definido pelo usuário, com valor sugerido de 0,5. É interessante observar que o fator η_l diminui quando o número de pontos incorporados pela regra aumenta, o que diminui a taxa de atualização do centro da regra. O intuito desta abordagem é fazer com que as regras sejam mais estáveis após incorporarem uma grande quantidade de dados, pois acredita-se que nesta etapa elas já estejam bem estabelecidas, e permitir grandes variações do centro causaria perda do conhecimento adquirido anteriormente.

A dispersão da regra mais influente é ajustada com:

$$\left(\sigma_{jl}^k\right)^2 = \frac{k_l - 1}{k_l}\left(\sigma_{jl}^{k-1}\right)^2 + \frac{1}{k_l}\left(z_j^k - c_{jl}^k\right)^2 + \left(c_{jl}^k - c_{jl}^{k-1}\right)^2 \tag{3.41}$$

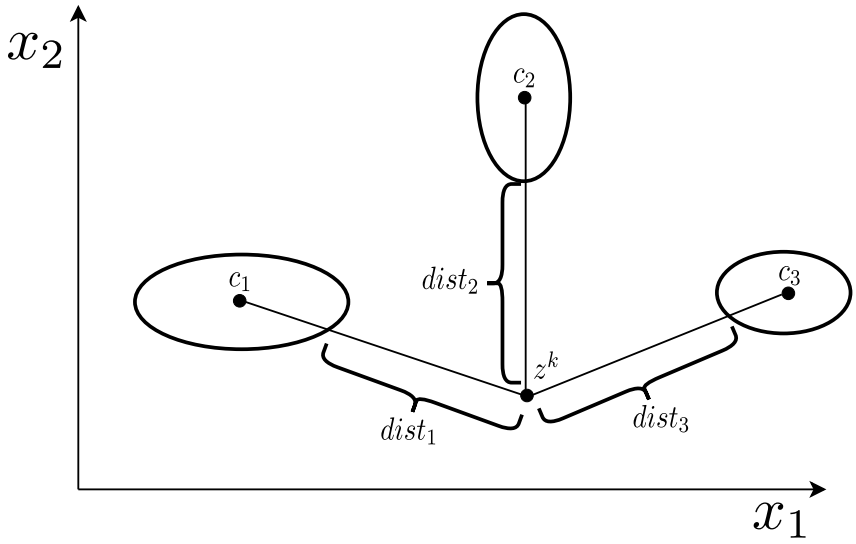


Figura 3.8 – Distância entre z^k e as fronteiras dos clusters

Caso $dist_l > \rho$, o algoritmo cria um novo cluster com centro em z^k. Uma particularidade do FLEXFIS é que os clusters recém criados não são habilitados automaticamente para influenciar na saída do modelo. Este procedimento é realizado para tornar o algoritmo mais robusto a pontos defectivos (e.g. outlier, ruído). Após a criação de um novo cluster, conta-se a quantidade de pontos que este cluster incorpora nos próximos passos. Somente quando o contador atingir o valor 10, esta regra será considerada para gerar a saída do modelo. Esta abordagem é baseada na premissa de que pontos defectivos estão geralmente isolados no espaço, sendo pouco provável que muitos pontos apareçam ao redor dele. Desta forma, o contador de uma regra gerada por um ponto defeituoso provavelmente não chegará ao valor 10, e a influência desta regra será ignorada pelo modelo.

A saída do modelo, a exemplo de outras modalidades de algoritmos nebulosos, é formada pela combinação linear entre as saídas individuais das regras:

$$\hat{y}^k = \sum_{i=1}^{R} \psi_i \left(\theta_i\right)^T \bar{x}^k \tag{3.42}$$

$$\psi_i = \frac{exp\left[\frac{-1}{2}\sum_{j=1}^{n}\left(x_j - c_{ji}\right)^2/\sigma_{ji}^2\right]}{\sum_{l=1}^{R} exp\left[\frac{-1}{2}\sum_{j=1}^{n}\left(x_j - c_{jl}\right)^2/\sigma_{jl}^2\right]} \tag{3.43}$$

onde ψ_i é o nível de ativação normalizado, e o nível de ativação de uma regra é obtido pela função Gaussiana $exp\left[\frac{-1}{2}\sum_{j=1}^{n}\left(x_j - c_{ji}\right)^2/\sigma_{ji}^2\right]$.

Assim como explicado no parágrafo anterior, as regras com contador k_i inferior ao valor 10 não são utilizadas para produzir a saída, sendo portanto ignoradas em (3.42) e (3.43).

3.3.4 Outras propostas na literatura

Um algoritmo denominado DENFIS (*Dynamic Evolving Neural-Fuzzy Inference System*) foi publicado em (KASABOV; SONG, 2002). Esse método baseou-se em outra abordagem evolutiva publicada anteriormente pelo mesmo autor, chamada de eFuNN (*evolving Fuzzy Neural Networks*) (KASABOV, 1998). O DENFIS é um sistema de inferência neuro nebuloso, sendo que os neurônios desta rede são representados por regras nebulosas e suas respectivas funções de pertinência no espaço de entrada. Estas funções são Gaussianas, com dispersões idênticas em todas as direções e, portanto, as regiões do espaço que atribuem o mesmo nível de pertinência tem formato esférico. O tamanho máximo da zona de influência de uma regra é definido pelo usuário, com um parâmetro de aprendizado. Apesar de utilizar funções de pertinência Gaussianas na etapa de clusterização, o DENFIS não adota estas mesmas funções para gerar a saída. Alternativamente, cada regra é associada a uma função de pertinência triangular para gerar o nível de ativação na saída, a exemplo daquelas ilustradas na Figura 3.1a. Além disso, apenas o subgrupo das m regras mais influentes é usado para gerar a saída, sendo m um valor definido pelo usuário.

Um algoritmo chamado SAFIS - (*Sequential Adaptive Fuzzy Inference System*) foi publicado em (RONG *et al.*, 2006). Este modelo, a exemplo do DENFIS, também usa regras Gaussianas com dispersões idênticas em todas as direções para clusterizar dados no espaço de entrada. O SAFIS foi inspirado em uma rede neural com funções de base radial (GAP-RBF - *Growing and Pruning Radial Base Function* (HUANG *et al.*, 2004)). O algoritmo de aprendizado utiliza dois critérios para adicionar regras à base: Significância estatística trazida pelo ponto mais recente ao modelo e distância Euclidiana entre este novo ponto e o centro de regra mais próximo. Esse mesmo critério de significância estatística também é aplicado para avaliar a importância das regras já existentes na base, identificando aquelas pouco influentes, que são excluídas para simplificação da estrutura. O modelo atualiza os parâmetros das regras (i.e. centro, dispersão e consequentes) utilizando uma versão estendida do filtro de Kalman. Um fato interessante deste algoritmo é que os consequentes não são formados por funções afins dos vetores de entrada, mas por simples escalares. Desta forma, o SAFIS é um modelo nebuloso funcional de ordem zero.

Um algoritmo similar ao SAFIS foi proposto em (RUBIO, 2009). Este algoritmo recebeu o nome SOFMLS (*Self-Organizing Fuzzy Modified Least-Square*), e também é um modelo de ordem zero. Uma particularidade do SOFMLS, no entanto, é o cálculo do nível de pertinência do dado mais recente para as regras da base, realizado da seguinte maneira:

$$\gamma_i = \frac{1}{n}\sum_{j=1}^{n}\frac{\left(x_j^k - c_i^k\right)}{\sigma_i^k} \tag{3.44}$$

$$\tau_i^k = exp(-\gamma_i^2) \tag{3.45}$$

onde c_i^k e σ_i^k são o centro e a dispersão da regra i, respectivamente, no instante k, e τ_i^k é o nível de pertinência do ponto x^k para a regra i. É interessante notar que o nível de pertinência é calculado como a exponencial da média das distâncias unidimensionais, diferentemente de outros algoritmos (e.g. eTS, ePL...) que utilizam o produto entre os níveis de pertinência em cada dimensão. O SOFMLS também propõe uma técnica de quadrados mínimos modificada, a fim de garantir maior estabilidade ao modelo. Esta técnica é utilizada para ajustar o centro, a dispersão e o consequente das regras.

O trabalho (DOVŽAN *et al.*, 2012) apresentou o algoritmo evolutivo eFuMo (*Evolving Fuzzy Model*). Esta abordagem apresentou novidades com relação a algumas publicações citadas anteriormente, como a capacidade de construir clusters elipsoidais em qualquer direção do espaço. Para construir estas regras, o algoritmo de agrupamento baseia-se na técnica Gustafson-Kessel evolutiva (FILEV; GEORGIEVA, 2010), (DOVŽAN; ŠKRJANC, 2010), que não demanda inversão de matrizes. O centro, dispersão e orientação das elipses são estimados recursivamente. O algoritmo pode criar, excluir, fundir e até mesmo dividir clusters. A divisão de clusters é uma contribuição interessante deste trabalho, e é executada monitorando-se a proporção do erro individual de cada regra no erro geral presente na saída do modelo. Se esta proporção individual supera um limiar pré estabelecido, esta regra é dividida em duas.

O artigo (PRATAMA *et al.*, 2014) apresentou um algoritmo denominado PANFIS (*Parsimonious Network Based on Fuzzy Inference System*), que também constrói clusters elípticos com orientações arbitrárias. A publicação apresenta duas abordagens para a construção destes clusters, sendo uma mais precisa, porém com custo computacional maior, e outra mais rápida e econômica, mas com desempenho inferior. A atualização dos centros utiliza uma versão estendida do algoritmo de mapas auto organizáveis (KOHONEN, 1982). O modelo pode fundir regras similares, sendo que esta similaridade baseia-se na proximidade entre os centros e na semelhança entre as dispersões de um par de regras. Existe também um método de exclusão de regras, que leva em conta a contribuição estatística individual de uma regra para o desempenho global do modelo. Os consequentes são atualizados utilizando-se o *Enhanced Recursive Least Squares*, que é uma atualização do método de quadrados mínimos recursivo tradicional.

3.4 Resumo

Este capítulo apresentou os modelos nebulosos evolutivos baseados em regras nebulosas. As primeiras seções detalharam os conceitos básicos deste tema, descrevendo a motivação para o seu surgimento e comparando-o com outras estratégias de inteligência computacional. Posteriormente, as demais seções exibiram exemplos de técnicas encontradas na literatura. Três modelos foram apresentados com maior profundidade, enquanto outros foram brevemente descritos. O próximo capítulo apresenta o algoritmo objeto dessa dissertação, o algoritmo nebuloso evolutivo Min-Max.

4 Algoritmo Nebuloso Evolutivo Min–Max

Este capítulo descreve o algoritmo nebuloso evolutivo Min-Max – eFMM (*evolving fuzzy Min-Max*). A Seção 4.1 traz uma breve introdução ao algoritmo proposto. A Seção 4.2 detalha a estrutura e a atualização do modelo. Já a Seção 4.3 detalha o algoritmo de aprendizado do eFMM. Finalmente, a Seção 4.4 resume os conceitos considerados neste capítulo.

4.1 Introdução

O algoritmo nebuloso evolutivo Min-Max é uma técnica capaz de processar um fluxo contínuo de dados, sem a necessidade de retreinamento, para extrair e incorporar informações de novos dados sem descartar o conhecimento previamente obtido. Por estas características, o modelo recebe a denominação de algoritmo *online*. Neste trabalho, sugere-se o eFMM para problemas de regressão.

O eFMM combina características dos algoritmos Min-Max, apresentados no Capítulo 2, com algoritmos nebulosos evolutivos, descritos no Capítulo 3. As semelhanças com os algoritmos Min-Max incluem a adoção do hiper-retângulo para definir as regras, a utilização dos operadores máximo e mínimo para atualização dos antecedentes e o uso do parâmetro θ para controlar a expansão dos hiper-retângulos, além das operações realizadas separadamente em cada dimensão. Quanto às diferenças, é importante ressaltar a ausência do teste de sobreposição e da contração no algoritmo de aprendizado do eFMM, o que simplifica o modelo e evita os erros introduzidos por estes procedimentos, assim como explicado na Seção 2.4. Além disso, o único ponto no interior do hiper-retângulo com nível de pertinência igual a 1 é o centroide, diferentemente de grande parte dos algoritmos Min-Max, onde qualquer região interna ao hiper-retângulo tem pertinência completa.

Com relação aos algoritmos nebulosos evolutivos, algumas características estruturais e de aprendizado do eFMM são também encontradas em outras técnicas, como a adoção de um ponto como centro de grupo, a capacidade de adicionar e excluir regras dinamicamente, além do gerenciamento da qualidade da base de regras. O eFMM apresenta, no entanto, alguns diferenciais importantes com relação a algumas destas técnicas, sendo elas: a estimativa automática de parâmetros, o que torna o algoritmo mais autônomo e diminui a responsabilidade do usuário; e a simplicidade das operações do algoritmo de aprendizado, que consiste na adoção das operações de máximo, mínimo, além de realizar as operações em cada dimensão separadamente, dispensando-se as normas Euclidianas. Estas características fazem com que o eFMM seja uma técnica rápida, eficiente, consideravelmente autônoma e interessante no que diz respeito à demanda de recursos computacionais e processamento de dados de alta dimensão. A Figura 4.1 resume algumas características do eFMM.

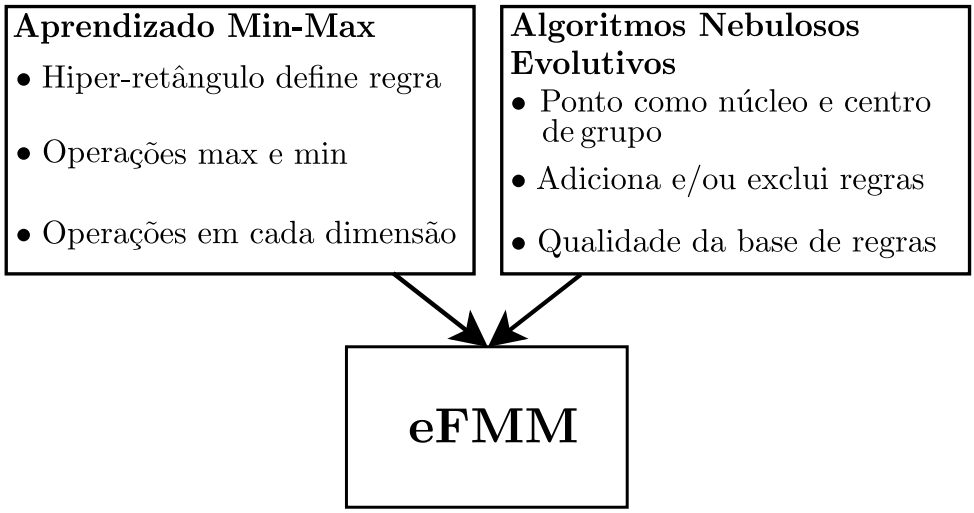


Figura 4.1 – Características do eFMM

4.2 Estrutura do modelo

O princípio do eFMM é granular o espaço de entrada utilizando hiper-retângulos. Neste trabalho, o espaço de entrada considerado é o hipercubo unitário n-dimensional I^n. O hiper-retângulo do eFMM tem o mesmo formato daquele utilizado na rede de Simpson (SIMPSON, 1992), que é uma região retangular do espaço I^n delimitada por um ponto de máximo (W) e um ponto de mínimo (V). Esta região é um hiper-retângulo n-dimensional, sendo suas fronteiras formadas por hiperplanos de dimensão $n-1$, perpendiculares a um dos eixos de coordenadas e paralelos aos demais eixos. A Figura 4.2 ilustra um hiper-retângulo no I^2. Os pontos de máximo e mínimo definem um hiper-retângulo, independentemente da dimensão. Um hiper-retângulo é formalmente definido por

$$B_i = \{I^n, V_i, W_i, b_i(x, V_i, W_i, c_i)\} \tag{4.1}$$

onde I^n é o espaço de entrada, V_i, W_i e c_i são os pontos de mínimo, máximo e centroide do hiper-retângulo i, $x \in I^n$ e b_i é a função de pertinência associada a B_i.

O eFMM assume uma modelagem baseada em regras nebulosas funcionais, com funções afins locais como consequentes. Esta abordagem é conhecida na literatura como modelo Takagi-Sugeno (ANGELOV; FILEV, 2004). Este modelo é formado por um conjunto de R regras nebulosas com o seguinte formato:

$$R_i: \; Se\ x \text{ é } B_i, \text{ então } \bar{y}_i \; = \; \theta_{0i} \; + \; \sum_{j=1}^{n} \theta_{ji} x_j \tag{4.2}$$

onde R_i é a i-ésima regra, $\bar{y}_i$ é a saída associada à regra R_i, x_j é a j-ésima componente do ponto de entrada x, θ_{ji}, $j = 1,2,...,n$, $i = 1,2,...,R$ são os parâmetros do consequente e R é o número de regras existentes.

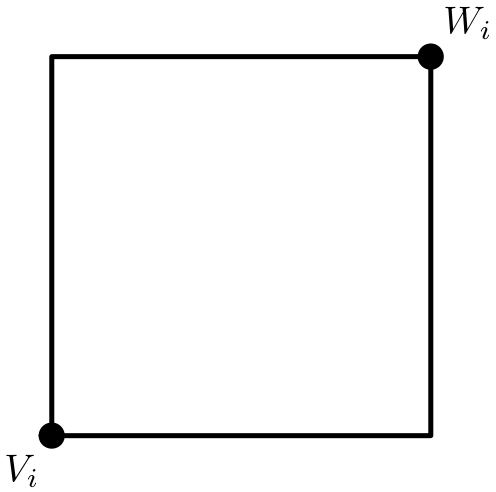


Figura 4.2 – Hiper-retângulo no I^2

O conjunto de R regras formam a saída a partir da média ponderada das funções afins locais. O peso associado a cada função é igual ao nível de ativação normalizado da regra associada a esta função. O nível de ativação de uma regra é equivalente ao nível de pertinência que esta regra atribui ao dado de entrada. O eFMM utiliza uma função de pertinência formada por agregação de funções Gaussianas unidimensionais:

$$b_i = \prod_{j=1}^{n} \bar{b}_{ji} \tag{4.3}$$

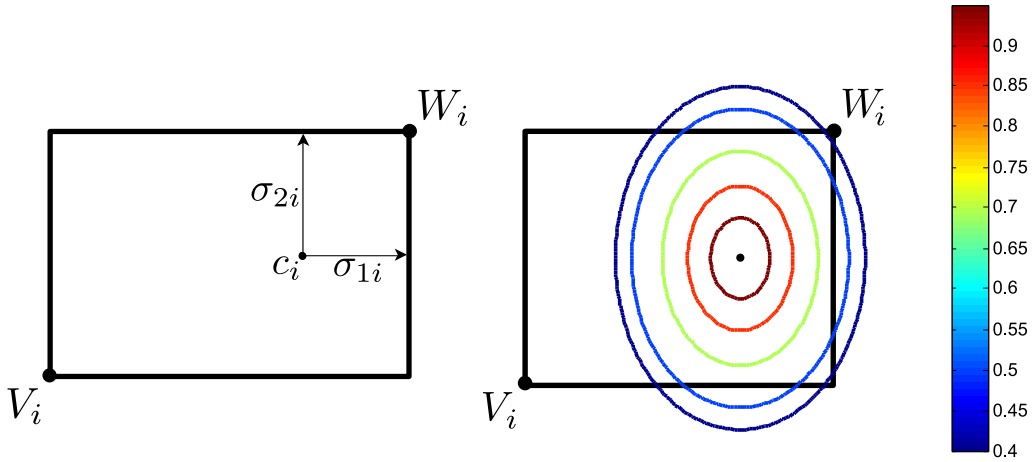
$$\sigma_{ji} = min(w_{ji} - c_{ji}, c_{ji} - v_{ji}) \tag{4.4}$$

$$\bar{b}_{ji} = exp\left(\frac{(x_j - c_{ji})^2}{2\sigma_{ji}^2}\right) \tag{4.5}$$

onde σ_{ji} e $\bar{b}_{ji}$ são a dispersão e o nível de pertinência atribuído pelo hiper-retângulo i na dimensão j, w_{ji}, v_{ji} e c_{ji} são as j-ésimas componentes dos pontos de máximo, mínimo e centroide do hiper-retângulo i, e b_i é o nível de pertinência atribuído pela regra R_i ao dado de entrada x (que é equivalente ao nível de ativação desta regra), resultado da agregação dos n níveis de pertinência unidimensionais. A Figura 4.3a ilustra um hiper-retângulo i no I^2 com as respectivas dispersões. Já a Figura 4.3b ilustra as curvas de nível de pertinência do mesmo hiper-retângulo. É possível notar que a dispersão da função Gaussiana em cada dimensão é igual à menor distância entre o centroide e a fronteira do hiper-retângulo nesta direção. Desta forma, garante-se que regiões externas ao hiper-retângulo tenham distância de pelo menos σ do centro, o que limita o nível de pertinência destas regiões.

O centroide é calculado recursivamente da seguinte forma:

$$c_i^k = \frac{M_i^k - 1}{M_i^k} c_i^{k-1} + \frac{1}{M_i^k} x^k \tag{4.6}$$



(a) Dispersões (b) Curvas de nível de pertinência

Figura 4.3 – Dispersão

onde M_i^k é o número de pontos associados à regra i no passo k.

A saída do modelo no passo k é calculada como a média ponderada das contribuições individuais de cada regra, da seguinte forma:

$$\psi_i = \frac{b_i}{\sum_{l=1}^{R} b_l} \tag{4.7}$$

$$\bar{x} = [1, x_1, x_2, \ldots, x_n] \tag{4.8}$$

$$\hat{y}^k = \sum_{i=1}^{R} \psi_i \theta_i^T \bar{x}^k \tag{4.9}$$

onde $\bar{x}$ é o vetor de entrada estendido e ψ_i é o nível de ativação normalizado da i-ésima regra.

A estrutura explicada acima é a mesma existente na primeira versão do algoritmo, o eFMR (PORTO; GOMIDE, 2018b). A versão atual, no entanto, traz algumas modificações em relação à primeira versão, de modo a incorporar: (i) o ajuste automático de um dos parâmetros de aprendizado, denotado por δ; (ii) um processo de redução de regras, que será abordado na Seção 4.2.1; e (iii) um mecanismo de avaliação da qualidade da base de regras (PORTO; GOMIDE, 2018a). Estas modificações deixam o algoritmo mais autônomo, mais adaptável em ambientes não-estacionários e reduzem a complexidade da base de regras. A seguir, a Seção 4.2.1 esclarece a atualização dos antecedentes do modelo, sendo um dos itens desta seção o ajuste do parâmetro δ. A Seção 4.2.2, por sua vez, detalha a atualização dos consequentes. A Seção 4.2.3 explica a estimativa do valor de α, que é um parâmetro utilizado no ajuste de δ, bem como no processo de redução. O último subitem desta seção, 4.2.4, apresenta a terceira novidade do eFMM com relação ao eFMR, que é a avaliação da base de regras.

4.2.1 Atualização dos antecedentes

A atualização dos antecedentes envolve o ajuste da regra mais influente no passo k. A regra mais influente é aquela que atribui o maior nível de pertinência para o dado atual, x^k. O ajuste da regra mais influente envolve a expansão do hiper-retângulo para incluir o dado x^k, o ajuste do centro, a atualização do parâmetro δ e a operação de redução. O ajuste do centro foi abordado na Seção 4.2, e é realizado pela expressão (4.6). Os outros três procedimentos são discutidos a seguir.

Expansão

Na expansão, os pontos de máximo e mínimo da regra mais influente são deslocados a fim de que o ponto x^k fique situado na fronteira do hiper-retângulo associado a esta regra. A Figura 4.4 ilustra a expansão. A expressão (4.10) resume este processo:

$$w_{ji} = max(x_j, w_{ji}), \quad v_{ji} = min(x_j, v_{ji}) \quad \text{para } j = 1,2,...,n \tag{4.10}$$

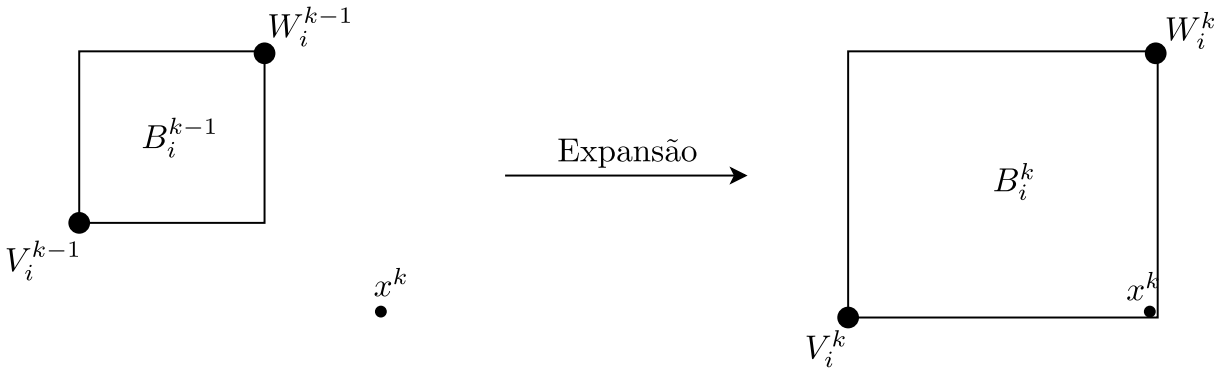


Figura 4.4 – Expansão

A expansão não modifica o formato do hiper-retângulo, sendo que os hiperplanos que o delimitam continuam sendo paralelos aos eixos de coordenadas. Caso o ponto x^k já esteja no interior ou na fronteira do hiper-retângulo a expansão não é necessária e, portanto, o hiper-retângulo mantém a sua configuração.

Parâmetro δ

O parâmetro δ define o tamanho máximo do hiper-retângulo em cada dimensão. Desta forma, o valor de δ controla o processo de expansão explicado acima. Uma vez que o

hiper-retângulo aumenta de tamanho após a expansão, é necessário verificar se ele não viola o tamanho máximo determinado por δ. Essa verificação é feita usando-se:

$$max(w_{ji}, x_j) - min(v_{ji}, x_j) \leq \delta_{ji}, \forall j = 1, 2, ..., n \tag{4.11}$$

A expansão da regra i para incluir o ponto x^k só ocorre se a Condição (4.11) for satisfeita.

Nas versões de algoritmos Min-Max encontradas na literatura, este parâmetro é comumente chamado de θ, e trata-se do principal parâmetro de aprendizado, afetando substancialmente o desempenho dos algoritmos. No eFMM, o parâmetro é referido como δ, para não ser confundido com o parâmetro θ dos consequentes. O hiper-retângulo pode ter tamanhos máximos diferentes em cada dimensão, assim como proposto em (GABRYS; BARGIELA, 2000).

O eFMM implementa um método de ajuste do parâmetro δ. Inicialmente, o usuário especifica um δ_0. Este valor se mantém como o tamanho máximo dos novos hiper-retângulos até que os seus respectivos contadores M_i atinjam um valor mínimo M_{min}, escolhido pelo usuário. Quando $M_i > M_{min}$, o valor de δ é atualizado fazendo-se:

$$d_{ji}^k = \sqrt{\frac{M_i - 1}{M_i}\left(d_{ji}^{k-1}\right)^2 + \frac{1}{M_i}\left(x_j - c_{ji}\right)^2} \tag{4.12}$$

$$\delta_{ji}^k = max\left((1-\alpha)\,\delta_{ji}^{k-1} + 2\alpha F_d d_{ji}^k,\ \ w_{ji}^k - v_{ji}^k\right) \tag{4.13}$$

onde d_{ji}^k é a dispersão dos pontos incluídos pela regra i na dimensão j e α é um parâmetro que influencia na velocidade de ajuste no modelo e será detalhado na Seção 4.2.3. O valor escolhido para F_d é 2, para que a regra i cubra a zona de 2σ (HASTIE *et al.*, 2009), pois na prática, os níveis de pertinência fora desta zona podem ser desconsiderados (ANGELOV, 2010). O fator 2 no lado esquerdo de α em (4.13) é aplicado porque o tamanho máximo deve permitir que o hiper-retângulo se expanda em ambos os sentidos de cada direção (i.e. à esquerda e à direita do centro, abaixo e acima do centro, etc). A Figura 4.5 ilustra o hiper-retângulo B_i juntamente com a sua dispersão (elipse de linha tracejada, com comprimento horizontal d_i) e o tamanho máximo $2F_d d_i$ (retângulo de linha tracejada). Como dito anteriormente, o tamanho máximo é determinado inicialmente pelo valor δ_0, definido pelo usuário, e converge gradativamente para $2F_d d_{ji}$ pela expressão (4.13).

O operador *max* na expressão (4.13) tem como objetivo impedir que δ_{ji}^k se torne menor do que o comprimento atual do hiper-retângulo na dimensão j. Como δ_{ji}^k é ajustado independentemente desse comprimento (definido por $w_{ji}^k - v_{ji}^k$), em tese, é possível que o valor de δ_{ji}^k diminuísse até se tornar menor do que $w_{ji}^k - v_{ji}^k$, impedindo que a regra B_i inclua novos pontos. Para contornar esse problema, a operação *max* é aplicada para garantir que o valor de δ_{ji}^k seja sempre maior ou igual a $w_{ji}^k - v_{ji}^k$.

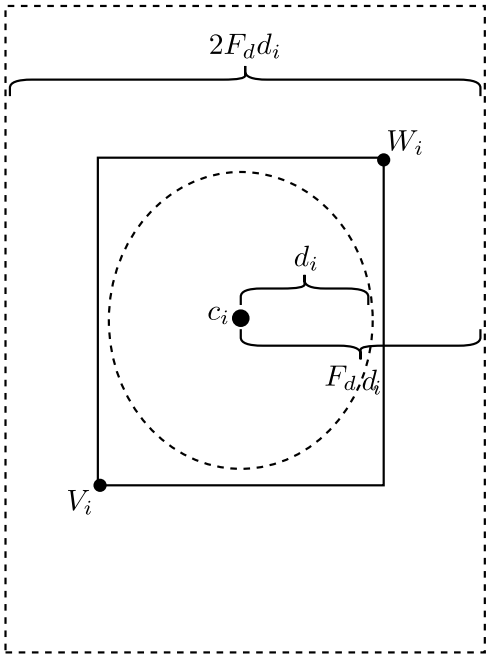


Figura 4.5 – Dispersão da regra i (d_i), região $F_d d_i$ e tamanho máximo ($2F_d d_i$).

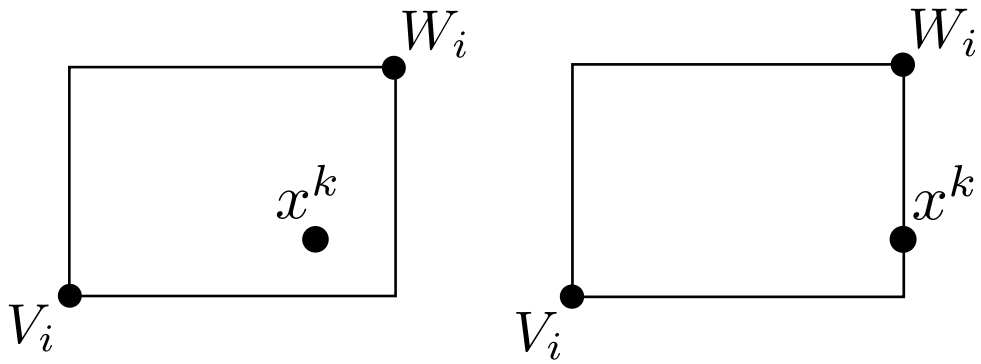
O parâmetro M_{min} determina o número mínimo de pontos associados à regra i para que o parâmetro δ_i comece a ser ajustado. Este parâmetro também é utilizado em (FILEV; GEORGIEVA, 2010), com o objetivo garantir o aprendizado dos parâmetros da matriz de covariância do modelo apresentado naquele trabalho. No artigo citado, o valor sugerido para este parâmetro é $M_{min} = n(n+1)/2$. No eFMM, no entanto, foi constatado que este valor é demasiadamente alto para os conjuntos de dados testados. Alternativamente, o valor $3,5(n+1)$ foi determinado empiricamente, sendo adequado para os conjuntos de dados testados neste trabalho. É possível que existam valores melhores para este parâmetro, e essa investigação fica como sugestão para trabalhos futuros.

Redução

A redução tem o objetivo de diminuir o tamanho dos hiper-retângulos. Essa redução é importante em algumas situações, pois regras muito grandes podem não satisfazer a condição de expansão, sendo impossibilitadas de incluir novos pontos. A operação de redução é realizada em cada componente separadamente. Assim como acontece na expansão, antes de se realizar a redução, é necessário verificar se o hiper-retângulo B_i satisfaz uma condição:

$$\textbf{SE } \left(v_{ji}^k < x_j^k \textbf{ E } x_j^k < w_{ji}^k\right),$$
$$\textbf{ENTÃO: } \begin{cases} w_{ji}^k = (1-\alpha)\, w_{ji}^{k-1} + \alpha\left(c_{ji}^k + \sigma_{ji}^k\right),\ v_{ji}^k = v_{ji}^{k-1} & \text{se } \left(w_{ji}^k - c_{ji}^k\right) > \left(c_{ji}^k - v_{ji}^k\right) \\ v_{ji}^k = (1-\alpha)\, v_{ji}^{k-1} + \alpha\left(c_{ji}^k - \sigma_{ji}^k\right),\ w_{ji}^k = w_{ji}^{k-1} & \text{se } \left(w_{ji}^k - c_{ji}^k\right) < \left(c_{ji}^k - v_{ji}^k\right) \end{cases} \tag{4.14}$$

A condição descrita em (4.14) avalia a disposição da componente x_j^k dentro do hiper-retângulo B_i. A Figura 4.6 ilustra duas situações, sendo que na primeira a condição para a redução é satisfeita, e na segunda não. A figura mostra um cenário onde a condição é avaliada no eixo horizontal. Se a componente x_j^k estiver na fronteira de B_i, assim como ilustra a Figura 4.6b, então a redução não é executada. Caso contrário, o hiper-retângulo B_i será reduzido nesta dimensão. Como a condição de redução é avaliada independentemente em cada dimensão, é possível que o hiper-retângulo B_i seja reduzido em apenas algumas componentes, enquanto em outras ele permaneça inalterado. Além disso, a condição descrita em (4.14) garante que, se o processo de expansão modificou o hiper-retângulo B_i na dimensão j no passo k, então a redução não modificará B_i nessa mesma dimensão no passo k. Isto evita que B_i seja expandido e logo depois reduzido na mesma dimensão e no mesmo passo.



(a) Condição satisfeita (b) Condição não satisfeita

Figura 4.6 – Condição de redução

Caso a condição seja satisfeita, o ponto de máximo ou o ponto de mínimo do hiper-retângulo é deslocado para diminuir o comprimento ao longo da dimensão j. Assim como descreve a expressão (4.14), a redução ocorre apenas no maior comprimento entre o centro e a fronteira do hiper-retângulo na dimensão j (i.e. entre c_{ji}^k e v_{ji}^k ou entre c_{ji}^k e w_{ji}^k). A Figura 4.7 ilustra o hiper-retângulo B_i passando pela redução no eixo horizontal. O eixo horizontal será considerado como a dimensão de índice 1, portanto $j = 1$ nesta imagem. Como a distância entre o centro e a fronteira esquerda de B_i é menor do que a distância entre o centro e a fronteira direita (i.e. $c_{1i}^k - v_{1i}^k < w_{1i}^k - c_{1i}^k$), então o ponto W_i é deslocado para a esquerda ao longo do eixo horizontal para diminuir a distância entre w_{1i}^k e c_{1i}^k. O valor de σ_{ji}^k, assim como descrito em (4.4), é igual à distância entre c_{ji} e a fronteira mais próxima do hiper-retângulo ao longo da dimensão j. Portanto, na Figura 4.7, $\sigma_{1i}^k = c_{1i}^k - v_{1i}^{k-1}$. O tamanho da redução depende, mais uma vez, do parâmetro α, que representa a compatibilidade entre o modelo afim local e a relação funcional entre os dados de entrada e saída. Este parâmetro é detalhado na Seção 4.2.3.

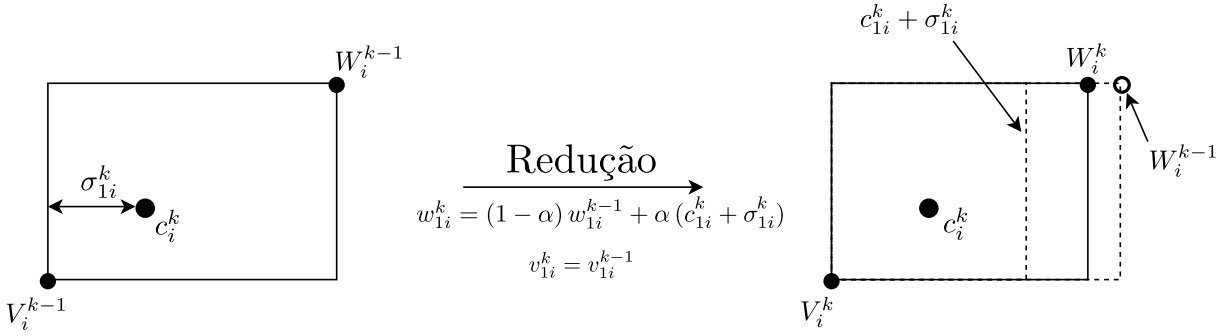


Figura 4.7 – Redução

4.2.2 Atualização dos consequentes

A atualização dos consequentes envolve o ajuste dos parâmetros da função afim de saída a cada passo. O ajuste destes parâmetros pode ser local ou global. No ajuste local, apenas os consequentes da regra mais influente são ajustados, enquanto no ajuste global todos os consequentes são ajustados a cada passo. Como o eFMM utiliza o ajuste local, esta seção detalhará este tipo de abordagem. Para mais detalhes sobre ajustes globais e locais e suas diferenças, ver (LUGHOFER, 2011), (LIMA, 2008), (MACIEL, 2012), (ANGELOV, 2010).

O ajuste local de consequentes pode ser realizado por mais de uma técnica. Estas técnicas são geralmente recursivas, e alguns exemplos de abordagens são os quadrados mínimos (ASTRÖM; WITTENMARK, 1996), quadrados mínimos com *forgetting factor* (LJUNG, 1999), com regularização (BENESTY *et al.*, 2011), ou ainda os quadrados mínimos com ponderação nebulosa - fwRLS (*fuzzily weighted recursive least squares*) (LUGHOFER, 2011). O eFMM utiliza os quadrados mínimos com *forgetting factor*, que é uma extensão dos quadrados mínimos tradicionais para atenuar o peso de dados antigos no cálculo dos consequentes. Esta seção apresentará a formulação dos quadrados mínimos tradicionais e depois explicará as modificações trazidas pela inserção do *forgetting factor*.

No eFMM, os consequentes são formados por funções afins, isto é:

$$\bar{y} = \theta_0 + \sum_{j=1}^{n} \theta_j x_j \tag{4.15}$$

onde $\bar{y}$ é a saída de uma regra qualquer e θ_j é o j-ésimo parâmetro da função afim associada a essa regra, sendo $j = 0, 1, 2, \ldots, n$. No decorrer desta seção, o índice i será omitido por questão de simplificação.

Considere que $x \in [0,1]^n$ e $y \in [0,1]$ são um ponto de entrada e a saída desejada, respectivamente, associados a uma regra qualquer da base de regras. Diz-se que os pontos x e y são associados a uma regra quando esta regra é a que atribui o maior nível de pertinência a estes pontos, dentre todas as regras existentes na base. Atribui-se a k o número total de observações passadas associadas à regra em questão. Desta forma, o objetivo da atualização dos consequentes é minimizar a diferença quadrática entre as saídas desejadas e as estimadas pela

regra:

$$\min_{\theta} \sum_{p=1}^{k} (\bar{y}^p - y^p)^2 \tag{4.16}$$

Sendo $\theta = [\theta_0,\ \theta_1,\ \theta_2, ..., \theta_n]^T$ e $\bar{x} = [1, x_1, x_2, ..., x_n]^T$, é possível obter uma solução em batelada para os k pontos:

$$\bar{X}^k = \begin{bmatrix} (\bar{x}^1)^T \\ (\bar{x}^2)^T \\ \cdots \\ (\bar{x}^k)^T \end{bmatrix} \qquad Y^k = \begin{bmatrix} y^1 \\ y^2 \\ \cdots \\ y^k \end{bmatrix}$$

$$\theta^k = \left(\left(\bar{X}^k\right)^T \bar{X}^k \right)^{-1} \left(\bar{X}^k\right)^T Y^k \tag{4.17}$$

Por questão de conveniência, será introduzida a matriz $P^k = \left(\left(\bar{X}^k\right)^T \bar{X}^k \right)^{-1}$ e o vetor $q^k = \left(\bar{X}^k\right)^T Y^k$, onde $P^k \in \mathbb{R}^{n+1 \times n+1}$ e $q^k \in \mathbb{R}^{n+1}$, de modo que a equação acima pode ser escrita como:

$$\theta^k = P^k q^k \tag{4.18}$$

Se um novo par ordenado x^{k+1}, y^{k+1} for recebido, seria possível adicionar esses dados nas matrizes $\bar{X}^k$ e Y^k, obtendo portanto $\bar{X}^{k+1}$ e Y^{k+1}, utilizar (4.17) novamente e obter um novo estimador θ^{k+1}. No entanto, essa abordagem é computacionalmente ineficiente, pois exige a inversão da matriz $\left(\left(\bar{X}^k\right)^T \bar{X}^k \right)$ em todos os passos, sendo que a matriz $\bar{X}$ cresce a cada novo ponto recebido.

Alternativamente, é possível encontrar uma relação entre os estimadores θ^k e θ^{k+1} e realizar o cálculo recursivamente. Esta técnica recebe o nome de quadrados mínimos recursivo (ASTRÖM; WITTENMARK, 1996), (LJUNG, 1999). Para demonstrá-la, assuma que um novo par de pontos de entrada e saída é recebido:

$$\bar{X}^{k+1} = \begin{bmatrix} \bar{X}^k \\ \left(\bar{x}^{k+1}\right)^T \end{bmatrix} \qquad Y^{k+1} = \begin{bmatrix} Y^k \\ y^{k+1} \end{bmatrix} \tag{4.19}$$

Aplicando a solução dada por (4.17) em (4.19), tem-se:

$$\theta^{k+1} = \left(\left(\bar{X}^{k+1} \right)^T \bar{X}^{k+1} \right)^{-1} \left(\bar{X}^{k+1} \right)^T Y^{k+1}$$
$$= \left(\begin{bmatrix} \bar{X}^k \\ \left(\bar{x}^{k+1}\right)^T \end{bmatrix}^T \begin{bmatrix} \bar{X}^k \\ \left(\bar{x}^{k+1}\right)^T \end{bmatrix} \right)^{-1} \begin{bmatrix} \bar{X}^k \\ \left(\bar{x}^{k+1}\right)^T \end{bmatrix}^T \begin{bmatrix} Y^k \\ y^{k+1} \end{bmatrix}$$

$$\theta^{k+1} = \left(\left(\bar{X}^k \right)^T \bar{X}^k + \bar{x}^{k+1} \left(\bar{x}^{k+1} \right)^T \right)^{-1} \left(\left(\bar{X}^k \right)^T Y^k + \bar{x}^{k+1} y^{k+1} \right) = P^{k+1} q^{k+1} \tag{4.20}$$

Comparando as expressões (4.18) e (4.20), nota-se a recursividade das matrizes P e q:

$$\left(P^{k+1} \right)^{-1} = \left(P^k \right)^{-1} + \bar{x}^{k+1} \left(\bar{x}^{k+1} \right)^T \tag{4.21}$$
$$P^{k+1} = \left(\left(P^k \right)^{-1} + \bar{x}^{k+1} \left(\bar{x}^{k+1} \right)^T \right)^{-1} \tag{4.22}$$
$$q^{k+1} = q^k + \bar{x}^{k+1} y^{k+1} \tag{4.23}$$

Utilizando o lema da matriz inversa (ASTRÖM; WITTENMARK, 1996):

$$(A + BCD)^{-1} = A^{-1} - A^{-1}B(C^{-1} + DA^{-1}B)^{-1}DA^{-1} \tag{4.24}$$

e comparando o lado esquerdo do lema com o lado direito da equação 4.22, tem-se $A = (P^k)^{-1}$, $B = \bar{x}^{k+1}$, $C = E$, onde E é a matriz identidade, e $D = (\bar{x}^{k+1})^T$. Usando o lado direito do lema e substituindo os valores na equação:

$$P^{k+1} = P^k - \frac{P^k \bar{x}^{k+1} (\bar{x}^{k+1})^T P^k}{1 + (\bar{x}^{k+1})^T P^k \bar{x}^{k+1}} \tag{4.25}$$

Para obter a expressão recursiva para θ^{k+1}, multiplica-se a equação acima por q^{k+1} pela direita:

$$P^{k+1} q^{k+1} = \left(P^k - \frac{P^k \bar{x}^{k+1} (\bar{x}^{k+1})^T P^k}{1 + (\bar{x}^{k+1})^T P^k \bar{x}^{k+1}} \right) \left(q^k + \bar{x}^{k+1} y^{k+1} \right)$$
$$\theta^{k+1} = \theta^k - \left(\frac{P^k \bar{x}^{k+1} (\bar{x}^{k+1})^T P^k}{1 + (\bar{x}^{k+1})^T P^k \bar{x}^{k+1}} \right) q^k + P^{k+1} \bar{x}^{k+1} y^{k+1} \tag{4.26}$$

Para desenvolver (4.26), utiliza-se (4.25), multiplicando-a por $\bar{x}^{k+1}$ pela direita:

$$
\begin{aligned}
P^{k+1}\bar{x}^{k+1} &= P^{k}\bar{x}^{k+1} - \frac{P^{k}\bar{x}^{k+1}(\bar{x}^{k+1})^{T}P^{k}}{1+(\bar{x}^{k+1})^{T}P^{k}\bar{x}^{k+1}}\bar{x}^{k+1} \\
P^{k+1}\bar{x}^{k+1}(1+(\bar{x}^{k+1})^{T}P^{k}\bar{x}^{k+1}) &= P^{k}\bar{x}^{k+1}(1+(\bar{x}^{k+1})^{T}P^{k}\bar{x}^{k+1}) - P^{k}\bar{x}^{k+1}(\bar{x}^{k+1})^{T}P^{k}\bar{x}^{k+1} \\
P^{k+1}\bar{x}^{k+1}(1+(\bar{x}^{k+1})^{T}P^{k}\bar{x}^{k+1}) &= P^{k}\bar{x}^{k+1} \\
P^{k+1}\bar{x}^{k+1} &= \frac{P^{k}\bar{x}^{k+1}}{1+(\bar{x}^{k+1})^{T}P^{k}\bar{x}^{k+1}} = K^{k+1}
\end{aligned} \tag{4.27}
$$

O vetor $K^{k+1} \in \mathbb{R}^{n+1}$ é conhecido como ganho ou fator de ponderação (MATHWORKS, 2015), (ASTRÖM; WITTENMARK, 1996). Utilizando (4.27) em (4.26):

$$
\begin{aligned}
\theta^{k+1} &= \theta^{k} - P^{k+1}\bar{x}^{k+1}(\bar{x}^{k+1})^{T}P^{k}q^{k} + P^{k+1}\bar{x}^{k+1}y^{k+1} \\
\theta^{k+1} &= \theta^{k} + P^{k+1}\bar{x}^{k+1}\left(y^{k+1} - (\bar{x}^{k+1})^{T}\theta^{k}\right) \\
\theta^{k+1} &= \theta^{k} + K^{k+1}\left(y^{k+1} - (\bar{x}^{k+1})^{T}\theta^{k}\right)
\end{aligned} \tag{4.28}
$$

E assim, obtém-se as expressões para os quadrados mínimos recursivos, sendo que a ordem de execução das equações a cada passo é (4.27), (4.28) e (4.25).

Para inicializar o algoritmo, utiliza-se o primeiro par de pontos x^1 e y^1 para obter θ^1 e $\bar{y}^1$:

$$
\begin{aligned}
\theta^{1} &= [y^{1}, 0_1, 0_2, ..., 0_n]^{T} \\
\bar{y}^{1} &= \left[1,\ \left(x^{1}\right)^{T}\right]\theta^{1}
\end{aligned} \tag{4.29}
$$

A estratégia para inicialização da matriz P^0 é

$$
P = \omega E \tag{4.30}
$$

onde $E \in \mathbb{R}^{n+1\times n+1}$ é a matriz identidade e ω é um valor alto, como por exemplo $\omega = [100,\ 10000]$.

Nos quadrados mínimos tradicionais todos os pontos tem o mesmo peso no cálculo dos parâmetros. No entanto, em sistemas variantes no tempo, deseja-se atenuar a influência de dados antigos, pois eles podem ser incompatíveis com a relação funcional entre dados de entrada e saída mais recentes. Para lidar mais adequadamente com sistemas variantes no tempo, a expressão (4.16) é modificada para (LJUNG, 1999):

$$
\min_{\theta} \sum_{p=1}^{k} \gamma^{k-p}(\bar{y}^{p} - y^{p})^{2} \tag{4.31}
$$

O parâmetro $\gamma \in [0,1]$, denominado *forgetting factor*, determina o peso dos pontos no cálculo dos quadrados mínimos. Considerando k o índice do ponto mais recente e p o índice de um ponto anterior, o peso de x^k no cálculo dos parâmetros é dado por γ^{k-p}. Desta forma, o peso de pontos antigos decai exponencialmente, sendo que valores de γ mais próximos de 1 definem um decaimento mais lento.

Com a inserção do parâmetro γ, as equações recursivas para os quadrados mínimos são:

$$K^k = \frac{P^{k-1}\bar{x}^k}{\gamma + \left(\bar{x}^k\right)^T P^{k-1}\bar{x}^k} \tag{4.32}$$

$$\theta^k = \theta^{k-1} + K^k\left(y^k - \bar{y}^k\right) \tag{4.33}$$

$$P^k = \frac{1}{\gamma}\left(E - K^k\left(\bar{x}^k\right)^T\right)P^{k-1} \tag{4.34}$$

4.2.3 Cálculo do parâmetro α

O parâmetro α é utilizado em algumas versões de algoritmos nebulosos evolutivos, onde é comumente chamado de *learning rate*. Ele recebe esta denominação por controlar a velocidade de ajuste do modelo, influenciando, por exemplo, no ritmo de deslocamento dos centros das regras na direção de novos dados. Em muitos modelos, no entanto, a tarefa de determinar um valor adequado para α é atribuída ao usuário. Esta tarefa não costuma ser trivial, pois o mesmo valor de α pode acelerar ou retardar o aprendizado para distribuições de dados diferentes. Dadas estas justificativas, o eFMM emprega um método para estimar o valor de α por meio da definição de outros parâmetros:

$$\alpha^k = \left(max\left(1 - \frac{|\bar{y}^k - y^k|}{max_{erro}},\ 0\right)\right)^t max_\alpha \tag{4.35}$$

Para compreender o papel dos parâmetros de (4.35), é conveniente observar a Figura 4.8. Esta figura ilustra o valor de α em função de $|\bar{y}^k - y^k|$ para $max_{erro} = 0,3$, $max_\alpha = 0,03$ e $t = 10$. O valor atribuído a max_{erro} determina o valor máximo da diferença $|\bar{y}^k - y^k|$ que produz $\alpha > 0$, enquanto max_α determina o valor máximo de α, que ocorre quando $\bar{y}^k = y^k$. A justificativa para esta abordagem é que, quanto menor for a diferença $|\bar{y}^k - y^k|$, maior será a compatibilidade entre a função afim da regra mais influente e a relação funcional local entre os pontos de entrada e saída. Desta forma, quando a diferença $|\bar{y}^k - y^k|$ for baixa, o valor de α é alto, para acelerar o ajuste da regra mais influente. Por outro lado, se a diferença for alta, isso implica baixa compatibilidade, e o valor atribuído a α é baixo. Por último, o valor de t ajusta o decaimento da função.

Acredita-se que a relevância desta abordagem para a estimativa do valor de α é a possibilidade do algoritmo modificar o valor deste parâmetro baseando-se na compatibilidade

entre o modelo local e a saída desejada, o que se opõe a outras abordagens da literatura em que o valor de α é fornecido pelo usuário e mantido constante durante toda a execução do algoritmo. Além disso, é possível que esta método seja melhorado, permitindo que todos os parâmetros da expressão (4.35) sejam atualizados automaticamente usando-se os dados de entrada.

Durante a elaboração deste trabalho, determinou-se empiricamente uma combinação de parâmetros que produziu resultados satisfatórios para os cinco conjuntos de dados considerados no Capítulo 5: $max_{erro} = 0,3$, $max_{\alpha} = 0,03$ e $t = 10$. Estes valores foram adotados em todas as simulações realizadas neste trabalho.

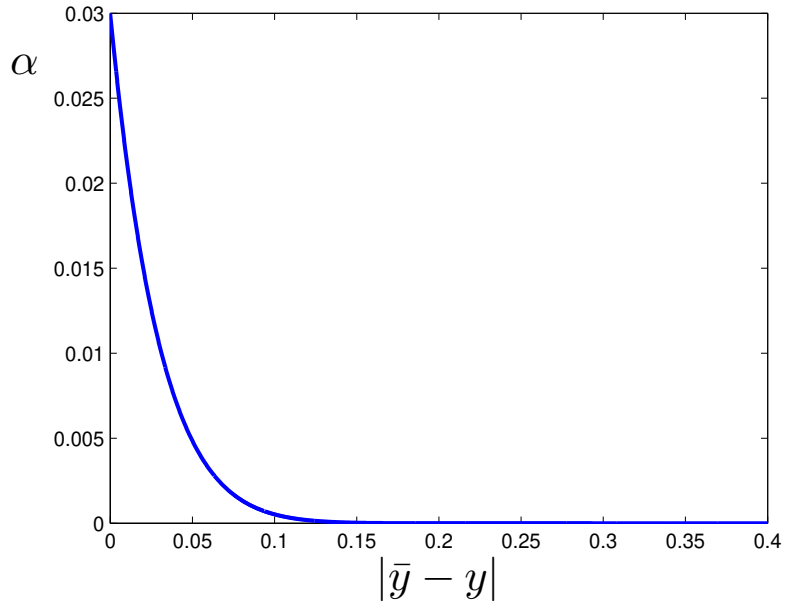


Figura 4.8 – Parâmetro α

4.2.4 Avaliação da qualidade da base de regras

Devido à natureza geralmente não estacionária dos processos geradores de dados, é possível que regras anteriormente representativas tornem-se obsoletas. Além disso, o comportamento dinâmico do algoritmo pode fazer com que dois centros de regras distintas fiquem muito próximos um do outro, tornando as respectivas regras redundantes.

Um algoritmo robusto deve lidar com as questões citadas acima em tempo real, identificando e corrigindo-as para manter a base de regras concisa e atualizada. O eFMM utiliza um mecanismo de exclusão de regras obsoletas e outro de fusão de regras redundantes.

Exclusão de regras

Assim como apresentado na Seção 3.3.1, o trabalho (ANGELOV, 2010) apresentou algumas métricas de qualidade de regras, sendo elas o suporte, a idade, a utilidade, a zona de influência e a densidade local. Quando o índice de utilidade é escolhido, as regras que possuem

um valor de utilidade abaixo de um limiar pré-definido são excluídas, assim como indicado pela Condição (3.27). Essa abordagem deixa para o usuário a tarefa de determinar um limiar adequado, o que geralmente não é um trabalho trivial.

O eFMM também usa a utilidade como métrica para a qualidade das regras, mas adota um critério diferente para determinar se uma regra é obsoleta. Esta abordagem foi inspirada pela técnica de exclusão do algoritmo eFuMo (DOVŽAN *et al.*, 2012), com a diferença de que, no eFuMo, as métricas de idade e suporte foram usadas para medir a qualidade das regras no lugar da métrica de utilidade. O critério utilizado pelo eFMM é:

$$\begin{gathered}\textbf{SE } U_i^k < \varepsilon\, \bar{U}^k, \textbf{ ENTÃO } R = R - 1,\ excluir\{B_i\} \\ \bar{U}^k = \text{ média } (U_1^k, U_2^k, ..., U_R^k)\end{gathered} \tag{4.36}$$

onde U_i^k é a utilidade da regra i no passo k e $\varepsilon \in [0{,}05,\ 0{,}5]$.

Apesar da manutenção do parâmetro ε, e da sua determinação por parte do usuário, esse valor não é usado diretamente no critério de exclusão. Emprega-se a média das utilidades de todas as regras na determinação do limiar de exclusão, o que torna este limiar mais compatível com a configuração atual da base de regras e facilita a escolha do usuário. É difícil saber se um limiar $\varepsilon = 0{,}2$ no Critério (3.27) é rigoroso ou tolerante com relação à preservação de regras pouco utilizadas, mas é mais fácil perceber que excluir regras com valor de utilidade abaixo de 90% da média é um critério rigoroso.

Seria interessante, no entanto, que o usuário não tivesse a responsabilidade de escolher este parâmetro, sendo o critério totalmente autônomo. Essa ideia fica como sugestão para trabalhos futuros.

Fusão de regras

O eFMM também tem um método para encontrar regras redundantes. Esse método é automático, e utiliza os pontos de máximo, mínimo e centroide para classificar um par de regras em redundantes ou não. Fundir regras similares é importante porque o algoritmo modifica a posição das regras de acordo com o fluxo de dados de entrada. Se um par de regras é suficientemente próximo, então é possível que este par seja substituído por uma única regra sem perda de desempenho do modelo. Isto é desejável para evitar que a base de regras seja demasiadamente complexa.

Para lidar com as questões citadas acima, o eFMM decide se as regras i e l são redundantes da seguinte forma:

$$\text{condição 1} : v_{ji} \leq v_{jl} \text{ E } w_{ji} \geq w_{jl}, \text{ para } j = 1,2,...,n \tag{4.37}$$

$$\text{condição 2} : v_{jl} \leq v_{ji} \text{ E } w_{jl} \geq w_{ji}, \text{ para } j = 1,2,...,n \tag{4.38}$$

$$\text{condição 3} : v_{ji} \leq c_{jl} \leq w_{ji} \text{ E } v_{jl} \leq c_{ji} \leq w_{jl} \text{ E } vol_{il} < vol_i + vol_l \tag{4.39}$$

$$vol_l = \prod_{j=1}^{n} (w_{jl} - v_{jl}),\ vol_i = \prod_{j=1}^{n} (w_{ji} - v_{ji}) \tag{4.40}$$

$$vol_{il} = \prod_{j=1}^{n} (max(w_{ji}, w_{jl}) - min(v_{ji}, v_{jl})) \tag{4.41}$$

$$\textbf{SE condição 1 OU condição 2 OU condição 3} \tag{4.42}$$

$$\textbf{ENTÃO } fundir\{B_i, B_l\}$$

onde vol_i, vol_l e vol_{il} são os volumes dos hiper-retângulos i,l e il, sendo que este último é formado pela união dos hiper-retângulos i e l. Mais detalhes sobre estes volumes serão fornecidos a seguir.

As regras i e l são fundidas se pelo menos uma das três condições descritas acima for verdadeira. As condições 1 e 2 dizem que se o hiper-retângulo l situa-se no interior do hiper-retângulo i, ou o hiper-retângulo i no interior do hiper-retângulo l, então estas duas regras devem se unir. A Figura 4.9 ilustra essa situação, onde o hiper-retângulo i inclui o hiper-retângulo l. Sobreposições parciais entre pares de regras são possíveis, e até mesmo desejáveis em algumas situações, mas sobreposições completas podem criar contradições se os dois centros estiverem próximos e os consequentes não forem similares.

A condição 3 avalia uma condição de fusão onde nenhuma das regras inclui a outra, mas há uma sobreposição considerável entre ambas. Para efetuar este teste, é necessário calcular os três volumes citados anteriormente, usando as expressões (4.40) e (4.41). O volume vol_i é obtido pelo produto dos n comprimentos do hiper-retângulo i. Cada um destes comprimentos é a distância entre os pontos de máximo e mínimo na dimensão j. O valor de vol_{il}, no entanto, é o volume do hiper-retângulo formado pela união entre i e l. Os pontos de máximo e mínimo deste hiper-retângulo são $W_{il} = max(W_i, W_l),\ V_{il} = min(V_i, V_l)$. A Figura 4.10 mostra um exemplo dos hiper-retângulos i, l e il, sendo que este último é delimitado pela linha tracejada. No primeiro caso desta imagem, a Condição 3 é satisfeita, o que significa que as regras devem ser fundidas. Por outro lado, a condição para a fusão das regras não é satisfeita no segundo caso.

A razão para utilizar o hiper-retângulo il é que seu volume aumenta se as regras i e l têm dispersões diferentes. Esta propriedade pode ser observada na Figura 4.10, onde no primeiro caso a maior parte do volume de il pertence a ambas as regras i e l. No segundo caso da mesma figura, isso não acontece.

Se uma das três condições descritas acima é satisfeita, então a seguinte operação de fusão é aplicada nas regras i e l:

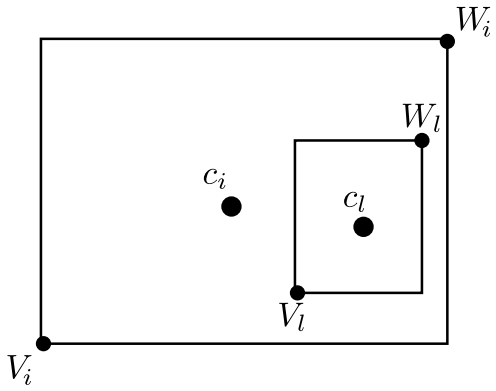


Figura 4.9 – Hiper-retângulo i incluindo hiper-retângulo l.

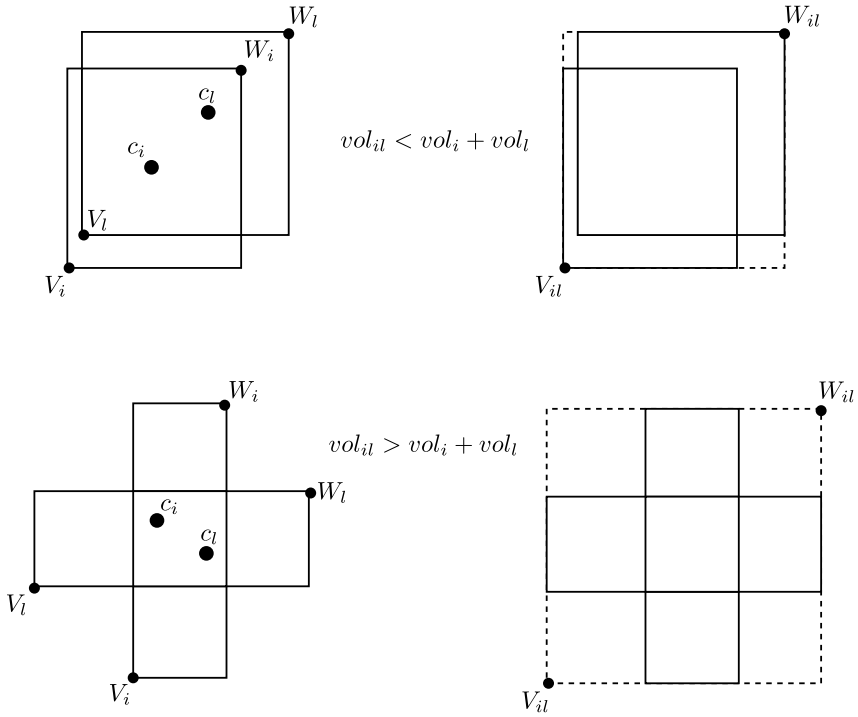


Figura 4.10 – Condição de fusão satisfeita, e condição não satisfeita.

$$p_i = \frac{vol_i}{vol_i + vol_l} \tag{4.43}$$

$$w_{ji} = p_i w_{ji} + (1 - p_i) w_{jl},\ v_{ji} = p_i v_{ji} + (1 - p_i) v_{jl} \tag{4.44}$$

$$c_{ji} = p_i c_{ji} + (1 - p_i) c_{jl},\ \theta_{ji} = p_i \theta_{ji} + (1 - p_i) \theta_{jl} \tag{4.45}$$

$$R = R - 1,\ excluir\{B_l\} \tag{4.46}$$

Quando duas regras são agrupadas, a regra resultante é uma combinação convexa das duas originais. Os pesos dessa combinação são os volumes normalizados, e $p_l = 1 - p_i$. A justificativa para esta abordagem é que o volume é um indicativo da consistência de uma regra, pois regras recém criadas têm volumes nulos, e estes tendem a aumentar quando a regra em questão adquire novos pontos. Esta técnica evita que o processo de fusão descarte o conheci-

mento armazenado na estrutura de regras bem estabelecidas quando estas são unidas a regras menores.

4.3 Algoritmo de aprendizado

O eFMM é um algoritmo *online*, sendo portanto capaz de processar um fluxo contínuo de dados em tempo real sem a necessidade de retreinamento ou armazenamento de dados antigos. A cada passo, o modelo atualiza os antecedentes e consequentes, além de produzir a saída. A atualização dos antecedentes consiste em criar, modificar, fundir ou excluir regras. Por outro lado, a atualização dos consequentes envolve o ajuste dos parâmetros da função afim da regra mais influente.

Novas regras são criadas quando nenhuma existente é capaz de se expandir para incorporar uma nova amostra. O parâmetro δ é o responsável por limitar a capacidade de expansão de uma regra. Caso alguma regra possa se expandir, os seus pontos de máximo e mínimo são deslocados a fim de incluir o novo ponto. Após a inclusão, o centro da regra é deslocado em direção ao novo ponto, em uma quantidade dependente do contador M_i.

Ao inicializar o algoritmo, o usuário determina os valores dos parâmetros M_{min}, δ_0 e ε. A base de regras encontra-se inicialmente vazia, e o primeiro dado recebido torna-se o ponto de máximo, mínimo e centro da primeira regra, fazendo-se $V_1 = W_1 = c_1 = x^1$. O contador é inicializado como $M_1 = 1$.

Sempre que um novo dado é recebido, calcula-se o seu nível de pertinência para todas as regras (que é equivalente ao nível de ativação das regras):

$$b_i = \prod_{j=1}^{n} \bar{b}_{ji}, \text{ para } j = 1,2,...,n \text{ e } i = 1,2,...,R \tag{4.47}$$

$$\bar{b}_{ji} = exp\left(\frac{(x_j - c_{ji})^2}{2\sigma_{ji}^2}\right) \tag{4.48}$$

$$\sigma_{ji} = min(w_{ji} - c_{ji}, c_{ji} - v_{ji})$$

A próxima etapa é identificar a regra i com maior nível de ativação e testar a sua condição de expansão:

$$i = \underset{m}{\text{argmax}}\{b_m\},\ m = 1,2,...,R$$

$$max(w_{ji}, x_j) - min(v_{ji}, x_j) \leq \delta_{ji}, \forall j = 1,2,...,n \tag{4.49}$$

Se a Condição (4.49) for satisfeita, a regra i se expande para incluir o ponto x^k:

$$w_{ji}^k = max(x_j, w_{ji}^{k-1}), \;\; v_{ji}^k = min(x_j, v_{ji}^{k-1}) \;\; \text{para } j = 1,2,...,n \tag{4.50}$$

Após incluir o dado atual, é necessário ajustar o contador, o centro, o tamanho máximo, os pontos de máximo e mínimo e os parâmetros dos consequentes da regra i:

$$M_i^k = M_i^{k-1} + 1 \tag{4.51}$$

$$c_i^k = \frac{M_i^k - 1}{M_i^k} c_i^{k-1} + \frac{1}{M_i^k} x^k \tag{4.52}$$

$$d_{ji}^k = \sqrt{\frac{M_i - 1}{M_i}\left(d_{ji}^{k-1}\right)^2 + \frac{1}{M_i}\left(x_j - c_{ji}\right)^2} \tag{4.53}$$

$$\delta_{ji}^k = \begin{cases} \delta_0, & \text{se } M_i < M_{min} \\ max\left((1-\alpha)\,\delta_{ji}^{k-1} + 2\alpha F_d d_{ji}^k,\;\; w_{ji}^k - v_{ji}^k\right), & \text{se } M_i \geq M_{min} \end{cases} \tag{4.54}$$

SE $\left(v_{ji}^k < x_j^k \;\; \textbf{E} \;\; x_j^k < w_{ji}^k\right)$,

$$\textbf{ENTÃO:} \begin{cases} w_{ji}^k = (1-\alpha)\,w_{ji}^{k-1} + \alpha\left(c_{ji}^k + \sigma_{ji}^k\right),\; v_{ji}^k = v_{ji}^{k-1} & \text{se } \left(w_{ji}^k - c_{ji}^k\right) > \left(c_{ji}^k - v_{ji}^k\right) \\ v_{ji}^k = (1-\alpha)\,v_{ji}^{k-1} + \alpha\left(c_{ji}^k - \sigma_{ji}^k\right),\; w_{ji}^k = w_{ji}^{k-1} & \text{se } \left(w_{ji}^k - c_{ji}^k\right) < \left(c_{ji}^k - v_{ji}^k\right) \end{cases} \tag{4.55}$$

$$K^k = \frac{P_i^{k-1}}{\gamma + (\bar{x})^T P_i^{k-1} \bar{x}} \bar{x}, \;\; \bar{x} = [1, x_1, x_2, ..., x_n]^T \tag{4.56}$$

$$\theta_i^k = \theta_i^{k-1} + K^k (y^k - \bar{y}_i^k) \tag{4.57}$$

$$P_i^k = \frac{1}{\gamma}(E - K^k (\bar{x}^k)^T) P^{k-1} \tag{4.58}$$

Se a Condição (4.49) não for satisfeita pela regra i, então as outras regras com nível de ativação $b_i > 0$ são testadas, em ordem decrescente de ativação, uma de cada vez. Se para uma das regras a Condição (4.49) for válida, então ela se expande para incluir x^k por meio do processo detalhado nas expressões de (4.50) a (4.58).

Caso nenhuma regra com nível de ativação não-nulo ($b_i > 0$) satisfaça a Condição (4.49), então o algoritmo identifica se há regras com nível de ativação zero (i.e. $b_i = 0$). Este procedimento é necessário porque regras recém criadas tem zona de influência nula, e portanto, apresentam nível de ativação zero até adquirir o primeiro ponto.

Se existirem regras com nível de ativação zero, calcula-se as distâncias dos seus respectivos centros até o ponto atual x^k:

$$dist_l = \sum_{j=1}^{n} |c_{jl} - x_j^k| \tag{4.59}$$

Após este cálculo, as distâncias $dist_l$ são dispostas em ordem crescente, e as regras correspondentes são testadas, uma de cada vez, pela Condição (4.49). Se uma dessas regras satisfizer a condição, então esta regra é expandida, e o teste é interrompido. Desta maneira, garante-se que se alguma regra é capaz de se expandir para incluir o dado atual, então o algoritmo não criará uma nova regra. Além disso, essa abordagem evita que o usuário precise determinar uma zona de influência inicial σ_0 para as regras recém criadas, assim como acontecia na primeira versão do algoritmo (PORTO; GOMIDE, 2018b).

Se nenhuma regra é capaz de atender à Condição (4.49), então uma nova regra é criada:

$$R = R+1,\ W_R = V_R = c_R = x^k \tag{4.60}$$

$$P_R = \omega E,\ \theta_R = [y^k,\ 0,\ 0,\ \ldots\ 0]^T,\ \delta_{jR} = \delta_0 \text{ para } j = 1,2,\ldots,n \tag{4.61}$$

Após o processamento do ponto x^k, a base de regras é avaliada a fim de identificar regras obsoletas

$$U_i^k = \frac{\sum_{l=1}^{k} \psi_i^l}{k - I^{i*}},\ \bar{U}^k = \textit{média}\{U_i^k\},\ i = 1,2,\ldots,R \tag{4.62}$$

$$\textbf{SE } U_i^k \leq \varepsilon \bar{U}^k, \textbf{ ENTÃO } \textit{excluir}\{B_i\} \tag{4.63}$$

ou redundantes:

$$\textbf{SE (4.37) OU (4.38) OU (4.39), ENTÃO} : vol_l = \prod_{j=1}^{n} (w_{jl} - v_{jl}),\ vol_i = \prod_{j=1}^{n} (w_{ji} - v_{ji}) \tag{4.64}$$

$$p_i = \frac{vol_i}{vol_i + vol_l} \tag{4.65}$$

$$w_{ji} = p_i w_{ji} + (1-p_i) w_{jl},\ \ v_{ji} = p_i v_{ji} + (1-p_i) v_{jl},\ j = 1,2,\ldots,n \tag{4.66}$$

$$c_{ji} = p_i c_{ji} + (1-p_i) c_{jl} \tag{4.67}$$

$$\theta_{ji} = p_i \theta_{ji} + (1-p_i) \theta_{li} \tag{4.68}$$

$$R = R - 1,\ \textit{excluir}\{B_l\} \tag{4.69}$$

A saída do modelo é calculada a cada instante k, utilizando-se (4.9). O algoritmo calcula a saída primeiramente, e depois utiliza a saída desejada (y^k) para ajustar o modelo. O algoritmo 4.1 e a Figura 4.11 resumem o aprendizado do eFMM.

4.1 Algoritmo de aprendizado eFMM

1: Escolher os parâmetros $M_{min} = M_0(n+1)$, $M_0 \in [2,3.5]$[1,2] $\delta_0 \in]0,1]$ e $\varepsilon \in [0.05,0.5]$ [1,2]
2: **Para** $k = 1,2,3,...$ **fazer**
3: Ler dado de entrada x^k
4: **Para** $i = 1,2,...,R$ **fazer**
5: Calcular o nível de ativação da regra i usando (4.48) e (4.47).
6: **Fim Para**
7: Gerar a saída do sistema usando (4.9).
8: Encontrar a regra i com maior nível de ativação que ainda não foi testada pela Condição
9: (4.49).
10: Verificar se a regra i satisfaz a Condição (4.49):
11: **Se** Condição for satisfeita **Então**
12: Atualizar a regra i utilizando as expressões (4.50) até (4.58).
13: **Senão**
14: Se ainda existem regras não testadas, encontrar a próxima delas com maior ativa-
15: ção, armazenar o seu índice na variável i e ir para o passo 10
16: **Fim Se**
17: **Se** Nenhuma regra satisfazer a Condição (4.49) **Então**
18: Verificar se existe alguma regra com nível de ativação zero (i.e. $b_i = 0$).
19: **Se** Existir **Então**
20: Calcular as distâncias elemento-a-elemento entre o ponto atual e os centros das
21: regras com nível de ativação zero, utilizando (4.59).
22: Ordenar as distâncias da etapa anterior em ordem crescente.
23: Encontrar a regra i com a menor distância e que ainda não foi testada pela Con-
24: dição (4.49).
25: Verificar se a regra i satisfaz a Condição (4.49).
26: **Se** Condição for satisfeita **Então**
27: Atualizar a regra i utilizando as expressões (4.50) até (4.58).
28: **Senão**
29: Se ainda existirem regras não testadas, ir para o passo 23.
30: **Fim Se**
31: **Fim Se**
32: **Fim Se**
33: **Se** Nenhuma regra existente na base satisfazer a Condição (4.49) **Então**
34: Criar uma nova regra usando (4.60) e (4.61).
35: **Fim Se**
36: Calcular o índice de utilidade e identificar regras obsoletas com (4.62) e (4.63).
37: Verificar se há pares de regras redundantes usando (4.64), e se positivo, fundir as regras
38: redundantes usando (4.66) até (4.69).
39: **Fim Para**

[1] A fim de se evitar ambiguidades, optou-se por utilizar um ponto para separar as partes decimal e inteira dos valores sugeridos para os parâmetros de aprendizado.
[2] Os intervalos de valores sugeridos foram obtidos empiricamente por meio da observação dos resultados das simulações realizadas neste trabalho.

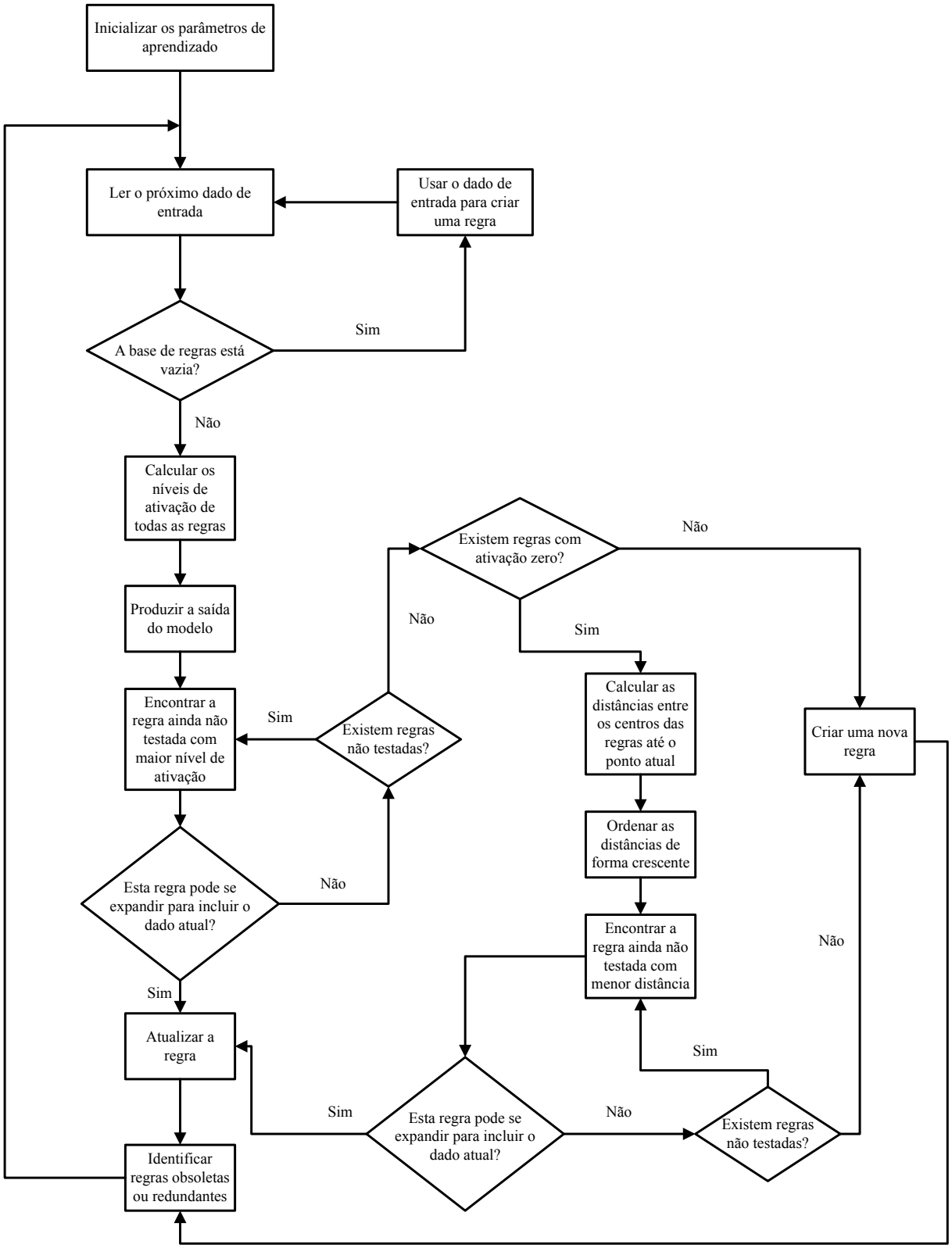


Figura 4.11 – Algoritmo eFMM

4.4 Resumo

Este capítulo apresentou o algoritmo nebuloso evolutivo Min-Max. As semelhanças e diferenças entre o modelo proposto e as demais técnicas Min-Max foram discutidas. O capítulo descreveu a estrutura e algoritmo de aprendizado do modelo proposto. O modelo atual também foi comparado com a sua primeira versão, o eFMR, sendo que as principais atualizações são o gerenciamento da base de regras, o ajuste automático do parâmetro δ e a operação de redução. O próximo capítulo apresenta os experimentos computacionais efetuados com o algoritmo proposto, comparando os resultados com outras propostas da literatura.

5 Resultados de Simulação

5.1 Introdução

Este capítulo apresenta os experimentos computacionais realizados com o algoritmo proposto nesta dissertação. O desempenho do eFMM é comparado com outras técnicas relevantes da literatura. As métricas de performance utilizadas são a raiz do erro quadrático médio (*Root Mean Squared Error* - RMSE) e o índice de erro não-dimensional (*Non-dimensional Error Index* - NDEI), cujas expressões são:

$$RMSE = \left(\frac{1}{N} \sum_{k=1}^{N} \left(\hat{y}^k - y^k \right)^2 \right)^{\frac{1}{2}} \tag{5.1}$$

$$NDEI = \frac{RMSE}{std(y)} \tag{5.2}$$

onde N é o número de dados de teste, y^k e $\hat{y}^k$ são as saídas desejadas e as do modelo, respectivamente, e $std(y)$ é o desvio padrão da saída desejada.

Em cada simulação, escolheu-se os parâmetros que produziram bons resultados com relação aos índices de erro e complexidade da base de regras. Evitou-se, no entanto, variar demasiadamente os parâmetros em diferentes simulações, salvo casos de relevante ganho de desempenho. O parâmetro M_{min}, por exemplo, foi mantido em $3,5(n+1)$ em todas as simulações, enquanto o ε foi mantido em $0,05$ em quase todos os testes, com exceção de uma simulação, onde foi modificado para $\varepsilon = 0,5$. Os valores testados para o tamanho máximo inicial dos hiper-retângulos são $\theta_0 = [0,1,\ 0,2,\ \ldots\ ,1]$.

Além da análise de desempenho, esse capítulo também compara o tempo de execução do eFMM com o de outras três técnicas da literatura, para verificar se o eFMM é um método rápido. Para que os tempos de execução fossem adequadamente comparados, todas as simulações foram realizadas na mesma máquina, dotada de um processador Intel Core 2 Duo®, 2GB de memória RAM, sistema operacional Windows® 7 e no software MATLAB®. Os algoritmos comparados são o *evolving participatory learning* - ePL (LIMA *et al.*, 2010), *enhanced evolving participatory learning* - ePL+ (MACIEL *et al.*, 2012), e o *evolving Takagi–Sugeno based on local least squares support vector machine* - eTS-LS-SVM (KOMIJANI *et al.*, 2011). Este último algoritmo consiste em um método Takagi Sugeno evolutivo que substitui os modelos afins locais, usados no eTS tradicional, por máquinas de vetores suporte. Essas máquinas são então usadas para construir os consequentes do modelo. O eTS-LS-SVM também utiliza matrizes de dispersão não diagonais e um método de exclusão de regras, o que confere certa complexidade ao modelo. Nas tabelas que apresentam os tempos de execução dos algoritmos,

as técnicas ePL e ePL+ recebem um asterisco (i.e. ePL*, ePL+*). Este asterisco indica que as simulações realizadas para calcular o tempo de execução não são as mesmas que produziram os resultados de desempenho de previsão destes dois algoritmos, medidos com as métricas RMSE e NDEI. Os intervalos de tempo de execução foram calculados em simulações realizadas durante a elaboração desta dissertação, enquanto os desempenhos de previsão foram obtidos de outros trabalhos da literatura. Para se obter os valores de tempo de execução foram realizadas 20 simulações diferentes com cada uma das 3 técnicas (ePL, ePL+ e eTS-LS-SVM), e em cada simulação testou-se uma combinação diferente de parâmetros de aprendizado. Ao fim das 20 simulações, registrou-se o tempo de execução referente à simulação que produziu o melhor desempenho de previsão. Com relação à origem dos códigos de programação das três técnicas, o algoritmo ePL foi implementado durante a elaboração desta dissertação, o ePL+ foi gentilmente cedido pelo Dr. Leandro Maciel e o eTS-LS-SVM foi obtido da internet (KOMIJANI *et al.*, 2013).

Além dos resultados e tempos de execução, este capítulo também apresenta uma análise da estacionariedade das séries temporais utilizadas nas simulações, a fim de verificar se o modelo produz resultados relevantes independentemente da natureza estatística da série. A estacionariedade de uma série temporal é caracterizada pela manutenção de propriedades estatísticas constantes ao longo do tempo. Existe mais de um tipo de estacionariedade, mas este trabalho considera a estacionariedade fraca, ou estacionariedade de segunda ordem. Ela é caracterizada por (BUENO, 2015):

1. Média constante
2. Variância constante
3. Autocovariância dependente apenas do atraso temporal: $\gamma_{t,\tau} = E\left[(y^t - \mu)(y^{t-\tau} - \mu)\right] = \gamma_\tau$

onde τ é um atraso temporal na série, $E[.]$ denota a esperança estatística (média), e $\gamma_{t,\tau}$ é a autocovariância como função do índice temporal t e do atraso temporal τ. O item 3 diz que a autocovariância da série y é constante ao longo do tempo, sendo dependente apenas do atraso temporal empregado.

Existem diversos testes para análise de estacionariedade na literatura. Em sua maioria, estes métodos aplicam testes de hipótese para avaliar a variação das propriedades estatísticas da série ao longo do tempo. No entanto, constatou-se que eles podem rejeitar a hipótese nula incorretamente (o que é conhecido como erro do tipo 1). Por este motivo, este trabalho utiliza quatro testes de estacionariedade diferentes, a fim de se obter resultados mais confiáveis. O teste *Priestley-Subba Rao* (PSR) (PRIESTLEY; RAO, 1969) estima a função de densidade espectral da série, e analisa o quão constante é esta função ao longo do tempo. O teste Kwiatkowski–Phillips–Schmidt–Shin (KPSS) (KWIATKOWSKI *et al.*, 1992) analisa se uma série é

estacionária em torno de uma média ou de uma tendência linear. Este método utiliza um teste de hipótese, onde a hipótese nula é de estacionariedade, e a hipótese alternativa é de não estacionariedade da série. O método Augmented Dickey Fuller (ADF) (FULLER, 1995) realiza um teste de hipótese onde a hipótese nula é que a série tem uma raiz unitária (sendo portanto não estacionária). A hipótese alternativa é de estacionariedade. O teste *Haar Wavelet* (HWTOS) (NASON, 2013) estima os coeficientes wavelet de Haar da série temporal analisada. Posteriormente, o método realiza múltiplos testes de hipótese, um para cada coeficiente de Haar, para constatar se eles são significativamente maiores do que zero, o que indica não estacionariedade. A hipótese nula é de estacionariedade, mas se pelo menos um dos testes rejeitar a hipótese nula, a série é considerada não estacionária. Em todos os testes de hipótese, o limiar de rejeição da hipótese nula foi configurado em $\alpha = 0,05$. Os testes foram feitos na linguagem de programação R, em bibliotecas de testes estatísticos encontradas na literatura.

A seguir, a Seção 5.2 apresenta os experimentos computacionais, sendo que cada subseção analisa os resultados em um conjunto de dados específico. Por fim, a Seção 5.3 resume os resultados e análises apresentados neste capítulo.

5.2 Experimentos computacionais

S&P-500

Esta seção analisa o desempenho do eFMM na previsão do índice *Standard and Poor's* (S&P-500). O índice S&P-500 é um indicador de 505 ações emitidas por 500 grandes empresas com capitalização de mercado de pelo menos U$6,1 bilhões. Este índice é visto como o principal indicador do mercado de ações americano, sendo um reflexo do desempenho das corporações com grandes capitais (INVESTOPEDIA, 2017). O conjunto de dados contém valores diários do índice para um período de 60 anos, e pode ser encontrado no site Yahoo! Finance (YAHOO!, 2017). Este período está compreendido entre 3 de Janeiro de 1950 e 12 de Março de 2009. Os quatro testes de estacionariedade (PSR, KPSS, ADF e HWTOS) apresentaram o mesmo resultado, apontando que a série é não estacionária. Este é o mesmo conjunto de dados utilizado em (PRATAMA *et al.*, 2014), para que o resultado do eFMM pudesse ser adequadamente comparado com os publicados no trabalho citado. O eFMM é confrontado com outros algoritmos, sendo eles o PANFIS (*Parsimonious Network based on Fuzzy Inference System*) (PRATAMA *et al.*, 2014), DENFIS (*Dynamic Evolving Neural-Fuzzy Inference System*) (KASABOV; SONG, 2002), eTS (*evolving Takagi-Sugeno*) (ANGELOV; FILEV, 2004), ANFIS (*Adaptive Network Based on Fuzzy Inference System*) (JANG, 1993), Regressão Linear (LR), EFuNN (*Evolving Fuzzy Neural Network*) (KASABOV, 1998), e SeroFAM (*Self organizing Fuzzy Associative Machine.*) (TAN; QUEK, 2010). Os desempenhos destas técnicas foram

retirados de (PRATAMA *et al.*, 2014). O modelo adotado para este problema é:

$$\hat{y}^k = f(y^{k-5}, y^{k-4}, y^{k-3}, y^{k-2}, y^{k-1}) \tag{5.3}$$

onde $\hat{y}$ é a saída do modelo.

O conjunto de dados foi normalizado no intervalo $[0,1]$. Os valores dos parâmetros do eFMM foram $\gamma = 0,99$, $M_{min} = 3,5(n+1)$, $\varepsilon = 0,05$ e $\delta_0 = 0,9$.

O desempenho do eFMM não foi muito sensível ao parâmetro δ_0. Esta constatação foi feita após realizadas 10 simulações, com $\delta_0 = [0,1,\ 0,2,\ \ldots\ ,1]$. O índice NDEI apresentou baixa oscilação nestes testes, ficando entre 0,0161 e 0,0165.

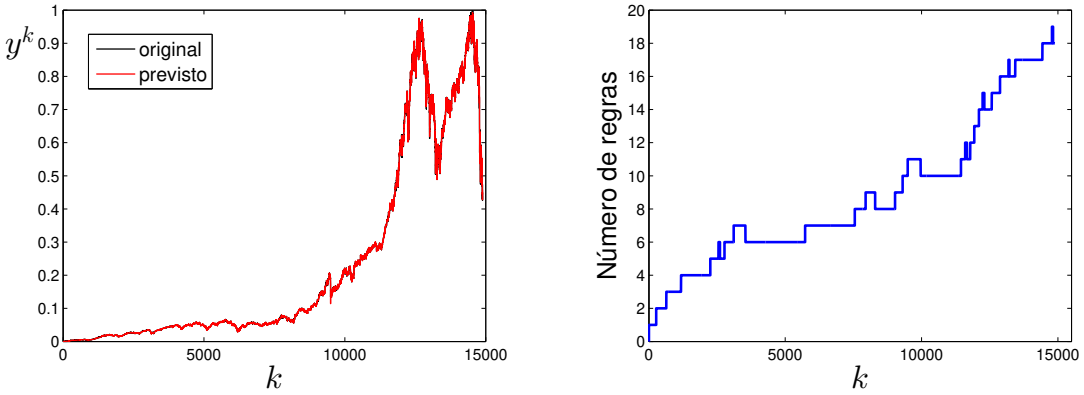
A Tabela 5.1 compara o desempenho do eFMM com os de outras técnicas. Estes desempenhos são quantificados pelo índice NDEI. A Tabela 5.1 exibe o número de regras existentes na base ao final da simulação. A Figura 5.1a ilustra o resultado da previsão e a Figura 5.1b mostra o número de regras ao longo do teste. O número de regras na base variou entre 1 e 19 ao longo do experimento, com valor médio de 8,74. Esse valor foi obtido por uma média ponderada do número de regras na base, onde o peso é a proporção de passos em que o modelo manteve um número determinado de regras na base, com relação ao número total de passos do experimento. Apesar do eFMM não superar todos os demais algoritmos, ele teve desempenho competitivo com propostas mais complexas, como o PANFIS, que utiliza matrizes de covariância não diagonais e mapas auto organizáveis para construir os antecedentes, além do *enhanced least squares* para ajustar os consequentes.

O número de regras consideravelmente alto do eFMM (i.e. 18) pode ser explicado pela baixa dispersão de dados consecutivos. A primeira regra formada pelo algoritmo incluiu todos os pontos de índice 1 até 268. A dispersão destes dados nas 5 dimensões de entrada é cerca de $8\text{x}10^{-4}$. Esta pequena dispersão fez o tamanho máximo dos hiper-retângulos diminuir, o que causou um aumento no número de regras. Para confirmar esta tese, 5 novos testes foram feitos, mas desta vez embaralhando-se os dados. Nestas novas simulações, o número de regras oscilou entre 1 e 3, enquanto o NDEI ficou entre 0,0171 e 0,0185. Houve exclusão de regras para $k = 2607$ e 13736, enquanto pares de regras foram fundidas para $k = 3541, 8298, 9967, 11666, 12303, 13223$ e 14823.

A Tabela 5.2 mostra os tempos de execução do eFMM e dos três outros algoritmos testados. Dentre as quatro técnicas comparadas, o ePL é a que apresenta o algoritmo de aprendizado mais simples, pelo fato de não excluir regras e não ajustar parâmetros de aprendizado. Provavelmente, este é o motivo deste algoritmo apresentar um tempo de execução relativamente mais baixo que os demais nesta simulação. O ePL+ incorpora duas características à primeira versão do ePL: um método de adaptação da zona de influência das regras e outro de exclusão de regras. Essas duas funcionalidades elevaram consideravelmente o tempo de execução do algoritmo, assim como mostra a Tabela 5.2. O eFMM possui um algoritmo de aprendizado com

Modelo	Número de Regras	NDEI
PANFIS	4	0,014
LR	-	0,157
DENFIS	6	0,02
ANFIS	32	0,015
EFuNN	114	0,154
SeroFAM	29	0,027
eTS	14	0,015
Simpl_eTS	7	0,045
eFMM	18	0,016

Tabela 5.1 – Previsão do índice S&P-500



(a) Conjunto de dados S&P-500 (b) Número de regras

Figura 5.1 – S&P-500

complexidade similar ao do ePL+, devido aos métodos de gerenciamento da base de regras e atualização de parâmetros existentes nos dois algoritmos. Essa similaridade resultou em tempos de execução parecidos para este conjunto de dados. O eTS-LS-SVM, em contrapartida, apresentou tempo de execução relativamente mais alto quando comparado às outras três técnicas, provavelmente devido ao uso de matrizes de dispersão não-diagonais e máquinas de vetores-suporte no lugar das funções afins locais.

Modelo	Tempo total de execução	Tempo médio por passo
eFMM	19,13s	1,28ms
ePL*	3,77s	253,51μs
ePL+*	19,62s	1,32ms
eTS-LS-SVM	62,53s	4,20ms

Tabela 5.2 – Tempo de execução

Conjunto de dados de alta dimensão

Esta seção avalia o desempenho do eFMM em um problema de identificação de sistema de alta dimensão. Esta simulação tem o propósito de testar como o algoritmo se comporta em espaços com maior dimensionalidade.

O sistema não linear a ser identificado é:

$$y^k = \frac{\sum_{p=1}^{m} y^{k-p}}{1+\sum_{p=1}^{m}\left(y^{k-p}\right)^2} + u^{k-1} \tag{5.4}$$

onde $u^k = sin(2\pi k/20)$, e $y^j = 0$ para $j = 1,...,m$ e $m = 10$.

O objetivo é prever a saída atual baseando-se nas saídas e entradas passadas. O formato do modelo para este conjunto de dados é:

$$\hat{y}^k = f(y^{k-1}, y^{k-2}, ..., y^{k-10}, u^{k-1}) \tag{5.5}$$

onde $\hat{y}$ é a saída do modelo.

O experimento foi feito da seguinte maneira: Os primeiros 3000 pontos foram usados para o treinamento do modelo e os próximos 300 pontos para teste, mantendo-se a estrutura e os parâmetros do modelo fixos após a fase de treinamento. O desempenho do eFMM foi medido pelo RMSE e comparado com técnicas relevantes da literatura. Esta simulação foi baseada na que foi publicada em (LEMOS *et al.*, 2011), de onde todos os resultados comparados com o eFMM foram retirados. Com relação aos testes de estacionariedade, três métodos indicaram que a série é estacionária (ADF, KPSS e HWTOS), enquanto um teste apontou não-estacionariedade (PSR). Desta forma, acredita-se na possibilidade do teste PSR ter rejeitado a hipótese nula incorretamente, o que implica a estacionariedade da série.

É importante ressaltar que, no experimento realizado em (LEMOS *et al.*, 2011), os dados não foram normalizados no intervalo $[0,1]$. Além disso, a métrica de erro RMSE é dependente da magnitude dos dados. Desta forma, há duas possibilidades para que o eFMM seja adequadamente comparado com as demais técnicas: a série pode ser normalizada no intervalo $[0,1]$, inserida no algoritmo e posteriormente reescalada no intervalo original antes de se calcular o RMSE, ou os valores originais podem ser inseridos diretamente no modelo. Com relação à segunda abordagem, uma das restrições associadas à rede Min-Max de Simpson é que a magnitude dos dados deve estar limitada ao intervalo $[0,1]$, pois a função de pertinência proposta por Simpson gera valores adequados de pertinência somente para dados de entrada contidos neste intervalo (ver Seção 2.2). No entanto, o eFMM substitui a função de pertinência de Simpson por uma função Gaussiana (ver Seção 4.2), o que, em tese, dispensa a restrição de que os dados estejam limitados ao hipercubo unitário I^n. No entanto, os parâmetros da expressão (4.35), usada para estimar o valor do parâmetro α (ver Seção 4.2.3), foram dimensionados considerando-se

a premissa de que os dados estão limitados ao intervalo $[0,1]$. A fim de se investigar o quanto a violação a esta restrição prejudica o desempenho de previsão, e para manter a máxima fidelidade às condições de teste adotadas em (LEMOS *et al.*, 2011), resolveu-se inserir os dados diretamente no modelo, sem nenhum tipo de normalização. Assim como mostra a Tabela 5.3, o eFMM produziu resultados satisfatórios mesmo com a violação da condição de que os dados estejam no intervalo $[0,1]$, mas acredita-se que para conjuntos de dados com magnitude mais elevada esta violação resulte em desempenhos de previsão inadequados. Essa questão é abordada novamente neste capítulo, mais especificamente na seção referente ao conjunto de dados MackeyGlass, onde um experimento é realizado para investigar o desempenho do eFMM ao processar dados não normalizados.

Os parâmetros do eFMM nesta simulação são: $\gamma = 0,95$, $M_{min} = 3,5(n+1)$, $\varepsilon = 0,05$ e $\delta_0 = 0,8$. A Tabela 5.3 compara os desempenhos do eFMM com o xTS (*eXtended Takagi-Sugeno*, Parâmetros: $\omega = 750$) (ANGELOV; ZHOU, 2006), FLEXFIS (*Flexible Fuzzy Inference System*) (LUGHOFER, 2008), eTS (Parâmetros: $r = 0,6$ e $\Omega = 750$), e eMG (*Multivariable Gaussian Evolving Fuzzy*, Parâmetros: $\lambda = 0,05$, $\omega = 25$ e $\alpha = 0,01$) (LEMOS *et al.*, 2011). A Figura 5.2a mostra o desempenho nos dados de teste e a Figura 5.2b ilustra o número de regras durante o treinamento. O número de regras na base variou entre 1 e 21, com valor médio de 17,3 regras. O eFMM precisou de cerca de 630 pontos para aprender a estrutura do sistema, e o número de regras manteve-se inalterado no restante da simulação. Não houve nenhuma fusão de regras neste teste, e regras foram deletadas para $k = 416,417$ e 505. Os resultados sugerem que o eFMM tem desempenho de previsão superior às demais técnicas, apesar da maior complexidade do modelo.

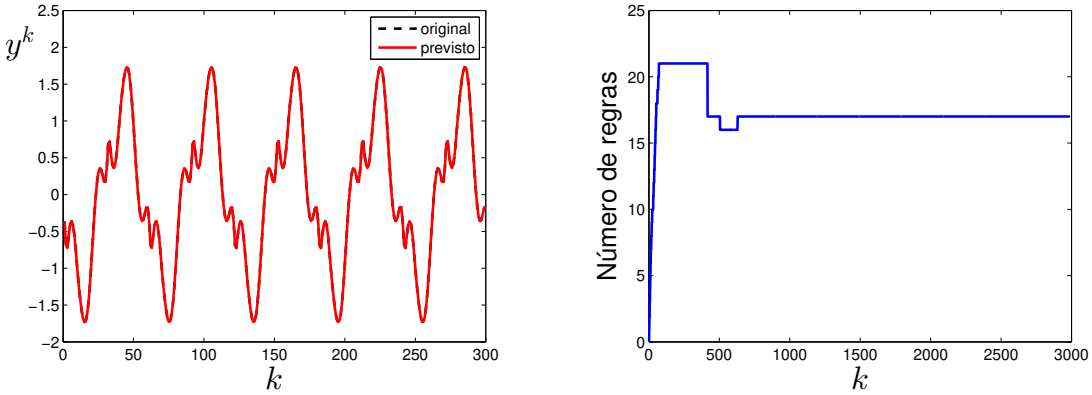
Modelo	Número de regras	RMSE
xTS	9	0,0331
FLEXFIS	15	0,0085
eTS	14	0,0075
eMG	13	0,0050
eFMM	17	5,3684 x 10^{-7}

Tabela 5.3 – Desempenho no conjunto de dados de alta dimensão

A Tabela 5.4 mostra os tempos de execução de três algoritmos evolutivos para este conjunto de dados. Nesta simulação, a diferença entre os tempos de execução foi menor quando comparada à simulação com o conjunto de dados S&P-500. Além disso, os tempos médios por passo do eFMM e do ePL foram superiores para o conjunto de dados de alta dimensão, o que indica que a dimensionalidade do problema aumenta o tempo médio de execução desses dois algoritmos. Essa propriedade não foi observada para o eTS-LS-SVM.

Modelo	Tempo total de execução	Tempo médio por passo
eFMM	6,38s	1,94ms
ePL*	4,62s	1,40ms
eTS-LS-SVM	11,11s	3,38ms

Tabela 5.4 – Tempo de execução



(a) Previsão nos dados de teste (b) Número de regras

Figura 5.2 – Dados de alta dimensão

Previsão de carga de curto prazo

A previsão de demanda de carga é de grande importância para a operação de sistemas de energia elétrica, pois diversos processos de tomada de decisão, como o planejamento de operação do sistema, análise de segurança e decisões de mercado são criticamente influenciados por valores futuros de demanda de carga. Neste cenário, um erro significativo de previsão pode resultar em perdas econômicas, violação de restrições de segurança, além de falhas operacionais do sistema. O problema de previsão de carga pode ser classificado em longo, médio e curto prazos. A previsão de longo prazo é utilizada para o planejamento de expansão da capacidade do sistema. Já a previsão de médio prazo é importante para organizar o estoque de combustível, operações de manutenção e alterações de cronograma. A previsão de curto prazo, em contrapartida, é geralmente usada para o planejamento diário do sistema elétrico e para o gerenciamento de demanda (RAHMAN; HAZIM, 1993).

Particularmente, no contexto de planejamento a curto prazo de sistemas hidrotérmicos, a previsão de carga é importante para elaborar o cronograma de operação do dia seguinte, pois erros na previsão de carga podem levar a sérias consequências, afetando a eficiência e a segurança do sistema (e.g. aumento de custos e fornecimento de energia insuficiente para a demanda existente).

O objetivo da previsão de curto prazo é estimar as cargas para as 24 horas do próximo dia, uma etapa à frente. A eficiência do eFMM é examinada utilizando-se dados de carga para uma importante companhia elétrica localizada na região sudeste do Brasil.

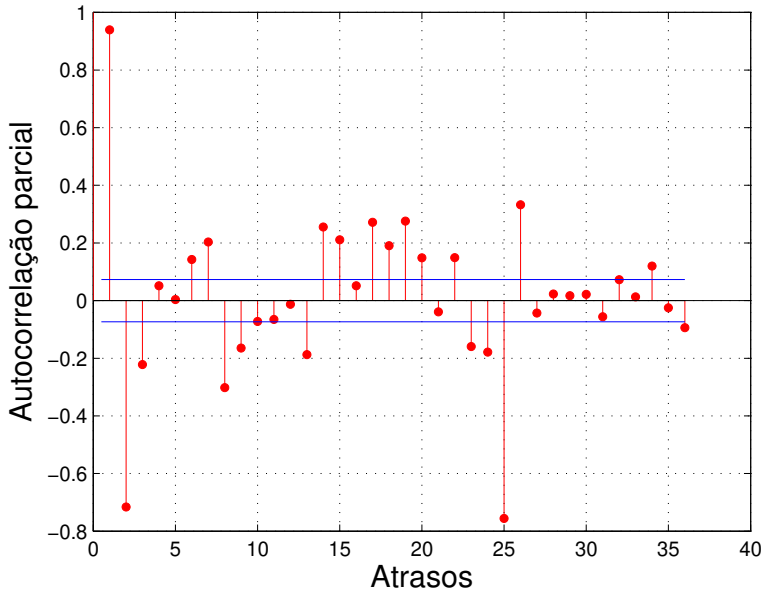


Figura 5.3 – Função de autocorrelação parcial

O conjunto de dados de carga usado neste experimento é expresso na unidade kilowatts-hora (kWh), e corresponde a 31 dias do mês de agosto do ano 2000. Os dados foram normalizados no intervalo $[0,1]$ para preservação de privacidade. Os testes de estacionariedade apontaram que a série é estacionária, sendo que os quatro métodos apresentaram resultados equivalentes para este conjunto de dados.

O eFMM foi aplicado como um estimador de um passo à frente, cuja tarefa era prever a demanda de carga atual utilizando dados passados consecutivos. Este trabalho adota o mesmo modelo de previsão utilizado em (LEMOS *et al.*, 2011), onde duas amostras passadas da série são usadas para prever a saída atual, a fim de se comparar os resultados adequadamente:

$$\hat{y}^k = f(y^{k-1}, y^{k-2}) \tag{5.6}$$

A justificativa para esta escolha dada em (LEMOS *et al.*, 2011) é que a função de autocorrelação parcial para até 36 atrasos da série sugeriu o uso de duas amostras passadas como entrada. A Figura 5.3 ilustra o gráfico desta função, onde é possível ver que as séries com um e dois atrasos têm alta correlação com a original, apesar de também existirem correlações relevantes para outros valores de atraso (i.e. ultrapassam a linha horizontal azul, sendo portanto significativamente maiores do que zero).

O experimento foi conduzido da seguinte forma: Os dados horários para os primeiros 30 dias foram inseridos na entrada do eFMM (718 observações) e os dados horários do último dia (24 observações) foram usados para testar o desempenho do modelo treinado, mantendo-se a sua estrutura e parâmetros fixos na fase de teste. A Tabela 5.5 compara os resultados do eFMM com outras técnicas evolutivas ou de estrutura fixa, usando-se as métricas RMSE

e NDEI. Os resultados das técnicas competidoras foram obtidos de (LEMOS *et al.*, 2011). A rede neural de múltiplas camadas (MLP) (DUDA *et al.*, 2000) tem uma camada oculta com 5 neurônios, treinada com o algoritmo *backpropagation*, e a rede ANFIS tem três conjuntos nebulosos para cada variável de entrada e sete regras nebulosas. A MLP utilizou o seguinte esquema de inicialização: Pesos iniciais com valores pequenos e atribuídos aleatoriamente, $\alpha = 0,9$ como parâmetro de momento, número máximo de épocas igual a 1000 e parâmetro de aprendizado adaptativo inicializado em $\eta = 0,01$. O ANFIS tem número máximo de épocas igual a 1000, $\eta_i = 0,01$ como tamanho do passo, $s_d = 0,9$ como razão de decrescimento de passo, e $s_i = 1,1$ como razão de crescimento de passo. Os parâmetros do eTS foram definidos em $r = 0,4$ e $\Omega = 750$. O xTS tem o parâmetro $\Omega = 750$. Já o ePL foi configurado com $\tau = 0,3$, $r = 0,25$ e $\lambda = 0,35$. O eMG tem parâmetros $\lambda = 0,05$, $\Sigma_{init} = 10^{-2}E$, onde E é a matriz identidade, e $\alpha = 0,01$. A Tabela 5.5 tem dois resultados para o eMG, sendo um para o tamanho de janela $\omega = 20$ e outro para $\omega = 25$. Estes dois resultados foram registrados para comparar o eMG e o eFMM com diferentes números de regras.

Os parâmetros escolhidos para o eFMM foram $\gamma = 0,95$, $M_{min} = 3,5(n+1)$ e $\varepsilon = 0,05$. Três resultados de simulação são apresentados, para $\delta_0 = 0,1$, $0,2$, e $0,5$. O resultado para $\delta_0 = 0,1$ é consideravelmente superior às demais técnicas, apesar da maior complexidade do modelo. Diferentes valores de δ_0 foram testados para avaliar o eFMM com uma estrutura mais simples, e o modelo proposto superou o eMG e o xTS com o mesmo número de regras. A Figura 5.4a ilustra o resultado da previsão nos dados de teste para $\delta_0 = 0,1$, enquanto a Figura 5.4b exibe o número de regras durante o treinamento. O número de regras variou entre 1 e 40, com valor médio de 30,87. Regras foram excluídas para $k = 355, 474, 559, 565$ e 601. Não houve fusão de regras nesta simulação.

Modelo	Número de regras/neurônios	RMSE	NDEI
xTS	4	0,0574	0,2135
MLP	5	0,0552	0,2053
ANFIS	9	0,0541	0,2012
eTS	6	0,0527	0,1960
ePL	5	0,0508	0,1889
eMG ($\omega = 20$)	5	0,0459	0,1707
eMG ($\omega = 25$)	4	0,0524	0,1948
eFMM ($\delta_0 = 0,1$)	40	0,0271	0,1008
eFMM ($\delta_0 = 0,2$)	13	0,0438	0,1628
eFMM ($\delta_0 = 0,5$)	4	0,0487	0,1810

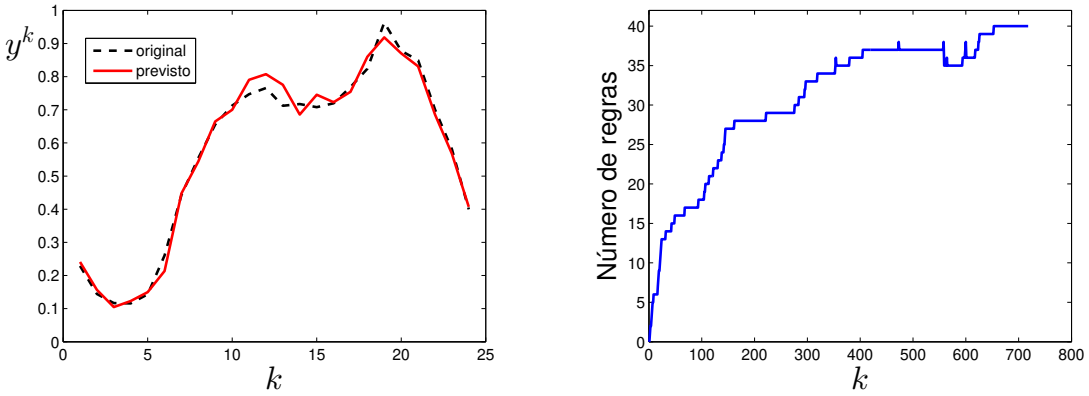
Tabela 5.5 – Previsão de carga de curto prazo

A Tabela 5.6 mostra os tempos de execução de quatro métodos evolutivos para este conjunto de dados. A tabela registra três tempos de execução para o eFMM, um para cada valor do parâmetro δ_0. A diferença nesses três intervalos de tempo mostra a influência da complexidade da base de regras no tempo de execução do algoritmo. Na simulação onde $\delta_0 = 0,5$ o

eFMM foi mais rápido do que ePL+ e eTS-LS-SVM, sendo superado somente pelo ePL. Na simulação com $\delta_0 = 0,1$, em contrapartida, a maior complexidade da base de regras resultou em um tempo de execução cerca de quatro vezes maior.

Modelo	Tempo total de execução	Tempo médio por passo
eFMM $\delta_0 = 0,1$	2,41s	3,27ms
eFMM $\delta_0 = 0,2$	1,06s	1,42ms
eFMM $\delta_0 = 0,5$	0,56s	748,87μs
ePL*	0,26s	350,12μs
ePL+*	1,14s	1,54ms
eTS-LS-SVM	3,79s	5,10ms

Tabela 5.6 – Tempo de execução



(a) Previsão de carga de curto prazo

(b) Número de regras

Figura 5.4 – Previsão de carga

Mackey–Glass

A série Mackey-Glass é amplamente utilizada na literatura de previsão de séries temporais, a fim de se medir o desempenho de sistemas nebulosos, neurais e híbridos. Essa série é criada a partir da seguinte equação diferencial:

$$\frac{dx(t)}{dt} = \frac{0,2x(t-\tau)}{1+x^{10}(t-\tau)} - 0,1x(t) \tag{5.7}$$

onde $x(0) = 1,2$, $\tau = 17$ e x(t) = 0 para $t < 0$.

Para obter dados nos valores inteiros de t, utilizou-se o método Runge-Kutta para obter a solução numérica da equação diferencial, com um tamanho de passo igual a 0,1. O objetivo é prever o valor de x^{k+85} usando o vetor de entrada $[x^{k-18}, x^{k-12}, x^{k-6}, x^k]$, sendo que cada valor de k corresponde a um valor inteiro de t. A simulação foi realizada da seguinte

maneira: Um total de 3000 pontos de treinamento foram produzidos ($k = 201, 202, ..., 3200$) e inseridos no modelo para evolução da base de regras. Posteriormente, 500 pontos de teste ($k = 5001, ..., 5500$) foram usados para avaliar o desempenho com o NDEI. Essas condições de teste foram as mesmas empregadas em (LUGHOFER, 2011), (LEMOS *et al.*, 2011), e (MACIEL *et al.*, 2012), de onde os resultados das demais técnicas foram retirados. É importante ressaltar que, para se obter os vetores de entrada associados às saídas $x^{201}, x^{202}, ..., x^{3200}$ é necessário coletar os valores a partir de x^{98} até x^{3115}. Isto acontece porque o vetor de entrada associado a x^{201} é $[x^{98}, x^{104}, x^{110}, x^{116}]$, enquanto o vetor $[x^{3097}, x^{3103}, x^{3109}, x^{3115}]$ é usado pelo modelo para prever a saída x^{3200}. Os dados não foram normalizados nesta simulação, sendo que alguns valores de entrada e saída não estão contidos no intervalo $[0, 1]$. Uma análise do desempenho do algoritmo para dados não normalizados será conduzida a seguir.

No que diz respeito aos testes de estacionariedade, três métodos apontaram que a série é estacionária (ADF, KPSS e HWTOS), enquanto o quarto método indicou que a série é não estacionária (PSR). Mais uma vez, acredita-se que a série seja estacionária, devido à decisão majoritária dos métodos empregados.

Os resultados de ePL, ePL+ (Parâmetros: $\beta = \tau = 0,16$, $\alpha = 0,01$, $\lambda = 0,84$, $\pi = 0,5$, $\Omega = 1000$ e $\varepsilon = 0,1$) e eTS+ (Parâmetros: $\pi = 0,5$, $\Omega = 1000$ e $\varepsilon = 0,1$) foram retirados de (MACIEL *et al.*, 2012). Os desempenhos de eMG (Parâmetros: $\lambda = 0,05$, $\omega = 20$, $\Sigma_{init} = 10^{-3}E_4$ e $\alpha = 0,01$), xTS (Parâmetro: $\Omega = 750$) e eTS (Parâmetros: $r = 0,4$ e $\Omega = 750$) foram retirados de (LEMOS *et al.*, 2011) e os desempenhos de DENFIS e FLEXFIS foram retirados de (LUGHOFER, 2011). Os parâmetros não informados não foram encontrados nas respectivas referências.

Os parâmetros escolhidos para o eFMM nesta simulação foram: $\varepsilon = 0,5$, $M_{min} = 3,5(n+1)$, $\delta_0 = 0,9$ e $\gamma = 0,8$. O valor de ε é relativamente mais alto nesta simulação com relação às demais, pois percebeu-se que um limiar mais rigoroso para exclusão de regras produz resultados melhores neste conjunto de dados em particular. A Tabela 5.7 compara os desempenhos das diferentes abordagens e a Figura 5.5a ilustra o resultado da previsão. A Figura 5.5b, em contrapartida, mostra o número de regras da base durante a simulação. Observando-se esta figura, é possível ver que regras foram excluídas logo depois da criação de outras regras. Isto acontece porque regras recém criadas tem valor de utilidade igual a 1, elevando o valor da utilidade média na expressão (4.63), fazendo com que a condição de exclusão seja satisfeita mais facilmente (i.e. $\varepsilon U_i < \bar{U}^k$). Apesar da Tabela 5.7 indicar que o eFMM terminou a simulação com apenas 1 regra, a Figura 5.5b mostra que 5 regras foram criadas durante a simulação. O número de regras se manteve constante na fase de teste.

Os resultados indicam que o eFMM tem desempenho superior com menor quantidade de regras, mesmo contabilizando-se todas as regras criadas durante o experimento. Nesta simulação, regras foram excluídas para $k = 187, 488, 689$, e 1294. Não houve fusão de regras.

A Tabela 5.8 ilustra os tempos de execução de quatro algoritmos evolutivos. Para

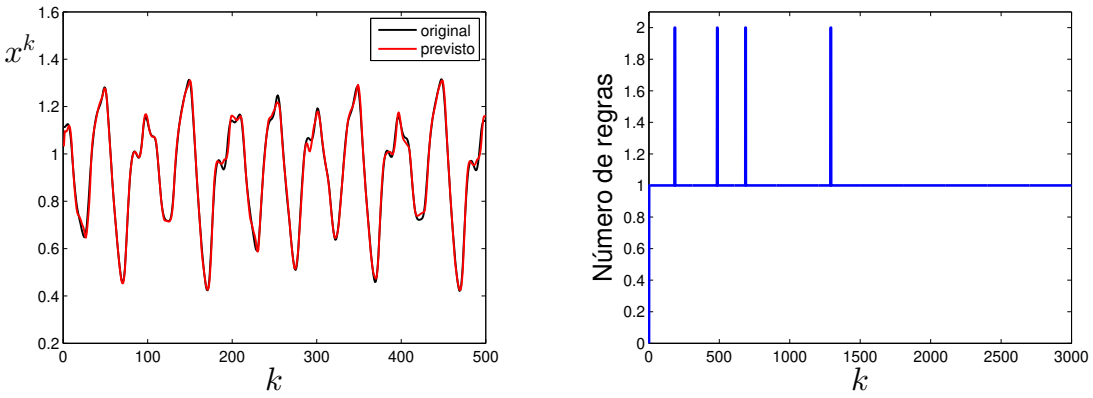
Modelo	Número de regras	NDEI
eTS	9	0,372
xTS	10	0,331
DENFIS	58	0,276
FLEXFIS	89	0,157
eTS+	8	0,438
eMG	58	0,139
ePL	8	0,311
ePL+	7	0,307
eFMM	1	0,068

Tabela 5.7 – Desempenho na série MackeyGlass

este conjunto de dados em particular, o eFMM foi a técnica mais rápida. Este fator se deve, provavelmente, à baixa complexidade na base de regras durante a simulação, o que resultou em um tempo de processamento por passo menor. O ePL+ foi mais rápido do que o ePL nesta simulação, o que é um fato curioso, devido à maior complexidade do ePL+ com relação a sua primeira versão. Analisando-se os resultados de ambos os métodos, constatou-se que o ePL+ teve um número menor de regras durante esta simulação (máximo de 3 regras) quando comparado ao ePL (máximo de 9 regras). A menor complexidade da base de regras, provavelmente, foi o fator que conferiu ao ePL+ um tempo menor de execução nesta simulação quando comparado ao ePL.

Modelo	Tempo total de execução	Tempo médio por passo
eFMM	1,87s	533,13μs
ePL*	3,57s	1,02ms
ePL+*	3,06s	873,61μs
eTS-LS-SVM	12,21s	3,49ms

Tabela 5.8 – Tempo de execução



(a) Resultado de previsão (b) Número de regras

Figura 5.5 – MackeyGlass

Além do desempenho de previsão e tempo de processamento, analisou-se também a interferência da utilização de dados não normalizados no intervalo $[0,1]$, para este conjunto de dados em particular, no desempenho do algoritmo. Em alguns experimentos deste capítulo o eFMM foi usado para processar dados não normalizados, o que viola a restrição imposta pelas redes de Simpson, mas isso não deteriorou o desempenho do algoritmo. Assim sendo, esse experimento foi executado para se averiguar, pelo menos para o caso específico da série MackeyGlass, se a amplitude dos dados realmente prejudica o desempenho de previsão. O experimento foi conduzido da seguinte maneira: Primeiramente, normalizou-se os dados no intervalo $[0,1]$. Depois, foram feitas 20 simulações, sendo que em cada simulação o conjunto de dados foi multiplicado por um fator A_r diferente antes de ser processado pelo eFMM. Os valores para este fator são $A_r = 1,2,3,...,20$. Desta forma, em cada simulação o conjunto de dados está limitado ao intervalo $[0,A_r]$, sendo que o valor de A_r é incrementado em uma unidade entre simulações consecutivas. A Figura 5.6 ilustra o desempenho de simulação medido pelo índice NDEI em função do fator A_r. Assim como explicado anteriormente, o índice NDEI não depende da amplitude dos dados de entrada e, portanto, a variação deste índice em função do fator A_r é resultado da incompatibilidade da amplitude dos dados com a amplitude apropriada (i.e. o eFMM foi projetado para processar dados contidos no I^n).

Analisando-se a Figura 5.6, é possível concluir que a ampliação da magnitude dos dados afeta negativamente o desempenho de previsão, apesar da variação irregular do NDEI, principalmente, para $A_r = 3$. Quando $A_r = 1$, o desempenho do eFMM é próximo àquele relatado na Tabela 5.8, e quando $A_r > 14$ o eFMM passa a ter desempenho de previsão inferior a todas as demais técnicas descritas nessa mesma tabela, com exceção do eTS+. Apesar desse experimento ter avaliado apenas a série MackeyGlass, ele sugere que o desempenho do eFMM foi satisfatório em experimentos com dados não normalizados porque as amplitudes dos dados não são suficientemente elevadas a ponto de prejudicar substancialmente o desempenho de previsão. Como conclusão, o eFMM deve ser usado, preferencialmente, para processar conjuntos de dados contidos no hipercubo unitário I^n.

Identificação de sistema não-linear

Esta seção aplica o eFMM em um problema de identificação de sistema não linear. Esse sistema é descrito por:

$$y^k = \frac{y^{k-1}y^{k-2}\left(y^{k-1}-0,5\right)}{1+\left(y^{k-1}\right)^2+\left(y^{k-2}\right)^2}+u^{k-1} \tag{5.8}$$

onde $u^k = sen\left(2\pi k/25\right)$ e $y^0 = y^1 = 0$.

O objetivo é prever a saída atual a partir de entradas e saídas passadas, da seguinte maneira:

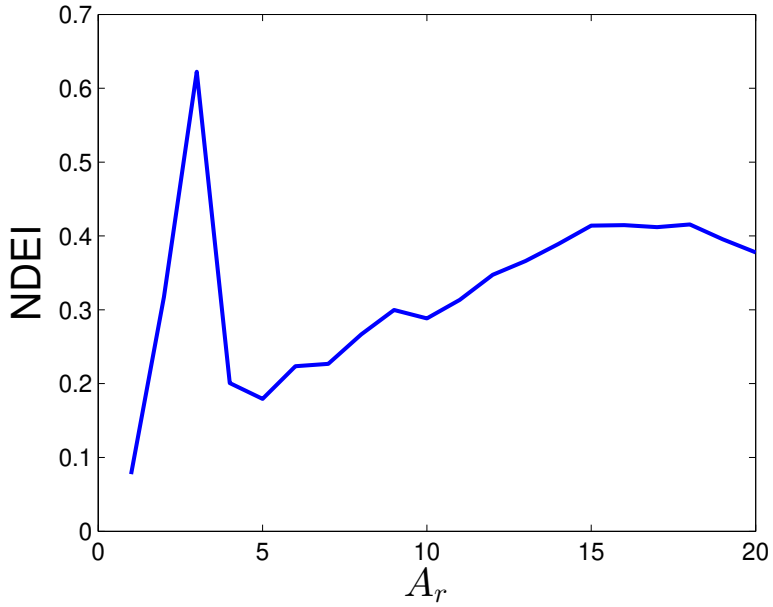


Figura 5.6 – Desempenho de previsão em função de A_r

$$\hat{y}^k = f\left(y^{k-1}, y^{k-2}, u^{k-1}\right) \tag{5.9}$$

onde $\hat{y}^k$ é a saída do modelo.

O experimento foi conduzido da seguinte forma: 5000 pontos foram gerados para treinamento do modelo, e mais 200 pontos para teste, mantendo-se a estrutura e parâmetros fixos após a fase de treinamento. Os resultados foram medidos com o RMSE e comparados com outras técnicas da literatura. Este é o mesmo cenário de simulação presente em (LEMOS *et al.*, 2011), de onde os resultados das demais técnicas foram retirados. Mais uma vez, os valores de entrada e saída não estão necessariamente contidos no intervalo $[0,1]$, mas isso não deteriorou o desempenho do algoritmo. Os métodos ADF, PSR, KPSS e HWTOS apontaram a estacionariedade da série por unanimidade.

As técnicas comparadas com o eFMM foram o eMG (Parâmetros: $\lambda = 0,05$, $\omega = 40$, $\Sigma_{init} = 10^{-1}E_3$ e $\alpha = 0,01$), o xTS (Parâmetros: $\Omega = 750$), o SAFIS (*Sequential Adaptive Fuzzy Inference System*) (RONG *et al.*, 2006), o SOFMLS (*Online Self-Organizing Fuzzy Modified Least-Square*) (RUBIO, 2009), e o FLEXFIS. Os parâmetros escolhidos para o eFMM nesta simulação foram: $M_{min} = 3,5(n+1)$, $\gamma = 0,99$ e $\varepsilon = 0,05$. A Tabela 5.9 compara os desempenhos dos modelos. Esta tabela apresenta dois resultados para o eFMM, com $\delta_0 = 0,1$ e $\delta_0 = 0,6$, para mostrar como o algoritmo se comporta com diferentes complexidades de base de regras. A Figura 5.7a exibe o desempenho de previsão para $\delta_0 = 0,6$, e a Figura 5.7b ilustra o número de regras durante essa mesma simulação. Esta última figura mostra apenas um trecho do treinamento, pois o eFMM aprendeu a estrutura do sistema nos primeiros 24 passos, sendo que a base de regras se manteve inalterada no restante da simulação. Não houve fusão ou ex-

clusão de regras neste experimento. O eFMM teve desempenho consideravelmente superior às demais técnicas nesta simulação, apesar da maior complexidade da base de regras.

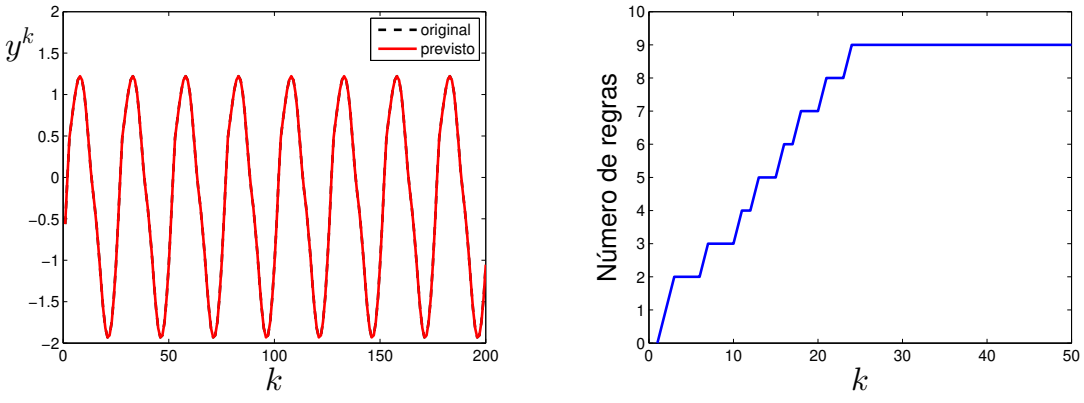
Modelo	Número de regras	RMSE
SAFIS	17	0,0221
SOFMLS	5	0,0201
FLEXFIS	8	0,0171
xTS	5	0,0063
eMG	5	0,0058
eFMM ($\delta_0 = 0,6$)	9	$2,6253\text{x}10^{-4}$
eFMM ($\delta_0 = 0,1$)	25	$2,2721\text{x}10^{-11}$

Tabela 5.9 – Desempenho na identificação de sistema não linear

A Tabela 5.10 mostra os tempos de execução de quatro métodos evolutivos. A exemplo do que foi observado na maioria das simulações, o ePL foi a técnica mais rápida e o eTS-LS-SVM foi a técnica mais lenta. A tabela apresenta dois resultados para o eFMM, sendo que na simulação com $\delta_0 = 0,1$ a maior complexidade na base de regras resultou em um tempo de execução mais do que duas vezes maior com relação à simulação com $\delta_0 = 0,6$.

Modelo	Tempo total de execução	Tempo médio por passo
eFMM ($\delta_0 = 0,1$)	13,77s	2,65ms
eFMM ($\delta_0 = 0,6$)	6,43s	1,24ms
ePL*	2,44s	469,94μs
ePL+*	8,46s	1,63ms
eTS-LS-SVM	16,67s	3,21ms

Tabela 5.10 – Tempo de execução



(a) Resultado de previsão (b) Número de regras

Figura 5.7 – Identificação de sistema não linear

5.3 Resumo

Este capítulo apresentou alguns resultados da implementação do algoritmo eFMM aplicado em problemas de previsão de séries temporais. O modelo foi testado em cinco problemas diferentes, tendo resultado superior às demais técnicas em quatro deles, e obtendo desempenho competitivo com métodos mais complexos no primeiro conjunto de dados. Em contrapartida, o eFMM criou mais regras do que os demais métodos, em alguns casos. Uma possível explicação para este fato é que, para competir com métodos que constroem regras em direções arbitrárias (e.g. PANFIS, eMG) o eFMM, que cria regras com direções paralelas aos eixos de coordenadas, precisa criar um número maior de regras.

Além do desempenho de previsão do eFMM, este capítulo também mostrou o resultado de quatro testes de estacionariedade, aplicados nos cinco conjunto de dados. Estes testes indicaram que, das cinco séries temporais, quatro são estacionárias, e uma é não estacionária, apesar destes resultados não serem unânimes em todos os casos. Estes testes indicam que o eFMM pode ser empregado na previsão de séries estacionárias ou não. O capítulo também comparou o tempo de execução do eFMM com o de outras três técnicas da literatura.

6 Conclusão

Os métodos evolutivos são alternativas interessantes em problemas de processamento e previsão em tempo real, onde os dados são gradativamente adquiridos pelo algoritmo, que deve se adaptar de forma rápida e contínua a novas informações. Esta adaptação envolve o aprendizado tanto da estrutura quanto dos parâmetros que caracterizam o modelo. Neste contexto, este trabalho apresentou uma contribuição à literatura dos métodos evolutivos, chamada de algoritmo nebuloso evolutivo Min-Max. Esta técnica utiliza regras nebulosas com um formato peculiar, que permite a implementação de um algoritmo de clusterização rápido e eficiente. A utilização deste formato de regra em um algoritmo incremental de regressão é a maior contribuição deste trabalho.

As regras nebulosas citadas anteriormente, referidas neste trabalho como hiper-retângulos, foram inicialmente propostas por Simpson em uma rede de classificação e outra de clusterização, intituladas de redes Min-Max. Diversos trabalhos propuseram atualizações ao método original, enriquecendo a literatura de aprendizado Min-Max. Esta dissertação dedicou um capítulo à introdução dos conceitos básicos desta técnica, além de uma revisão bibliográfica da literatura existente. O referido capítulo citou técnicas Min-Max de classificação, clusterização e regressão, apesar da escassez de trabalhos dedicados a esta última aplicação.

Diferentemente das técnicas Min-Max, os métodos nebulosos evolutivos são amplamente utilizados em regressão. Após a publicação do algoritmo Takagi-Sugeno evolutivo, diversos trabalhos propuseram técnicas correlatas para tarefas de previsão em tempo real. Este texto abordou a revisão bibliográfica destas técnicas em um capítulo à parte, introduzindo os conceitos de antecedente e consequente, além de apresentar diversos algoritmos para construir e atualizar essas duas partes do modelo evolutivo. Com relação aos antecedentes, as funções de pertinência associadas às regras são, geralmente, esféricas ou elipsoidais, sendo que estas últimas podem ser com direções arbitrárias ou paralelas aos eixos de coordenadas. Já as técnicas de atualização dos consequentes envolvem principalmente os quadrados mínimos recursivos e suas variações.

Além da atualização em tempo real dos antecedentes e consequentes, outros interesses recentes na literatura de métodos evolutivos são o gerenciamento da base de regras e o ajuste automático de parâmetros de aprendizado. Como resultado, diversas técnicas de exclusão e fusão de regras foram propostas, tornando os modelos mais concisos e permitindo ganhos de desempenho. Neste contexto, o método evolutivo apresentado neste trabalho incorporou técnicas para o gerenciamento da base de regras, além de propor uma forma de ajustar o tamanho máximo dos hiper-retângulos. Esta última característica é uma contribuição interessante deste trabalho, pois não foram encontradas técnicas Min-Max que atualizam este parâmetro de ma-

neira *online*, pelo menos até o presente momento.

Uma das motivações dos métodos evolutivos é a criação de algoritmos capazes de processar dados sem nenhum conhecimento *a priori* do processo gerador das amostras. Partindo-se desta premissa, é plausível concluir que os métodos devem ser testados em conjuntos de dados com características diversas, a fim de fortalecer a tese de que este método seria capaz de processar fluxos de dados de qualquer natureza. Desta forma, este trabalho aplicou o eFMM em séries temporais diversificadas no sentido do tipo de aplicação (i.e. séries econômicas, previsão de demanda de carga elétrica e identificação de sistemas não lineares), das características estatísticas (i.e. estacionárias e não estacionárias), e da dimensionalidade (i.e. dimensões de entrada variando entre 2 e 11).

Abordando mais especificamente os resultados dos experimentos computacionais, constatou-se que o eFMM obteve desempenho superior ou similar às demais técnicas. Em contrapartida, o algoritmo criou mais regras do que os demais na maioria dos testes. Em um dos experimentos, esse fato foi consequência da baixa dispersão dos dados de entrada. Em outros, no entanto, acredita-se na hipótese do eFMM necessitar de mais regras para ser competitivo com técnicas que criam regras em direções arbitrárias. Em contrapartida, o tempo de processamento do eFMM mostrou-se competitivo quando comparado com outros métodos da literatura. Os tempos de execução mostraram que o eFMM é mais rápido do que duas das três técnicas comparadas, perdendo apenas para uma técnica mais simples computacionalmente (ePL) e sendo mais rápido do que uma técnica de complexidade similar (ePL+) e uma de complexidade maior (eTS-LS-SVM).

Apesar das contribuições deste trabalho, o eFMM ainda pode ser melhorado em alguns aspectos. Um deles é a diminuição da influência do tamanho máximo inicial dos hiperretângulos no desempenho de previsão. Apesar do eFMM ajustar este parâmetro, o valor inicial fornecido pelo usuário ainda se mostrou consideravelmente influente nos resultados de simulação. Outra questão que pode ser abordada por trabalhos futuros é a adaptação *online* dos demais parâmetros de aprendizado, sendo eles o ε e o M_{min}.

Referências

ANGELOV, P. *Evolving Rule-Based Models: A Tool for Design of Flexible Adaptive Systems*. Berlin, Heidelberg: Physica-Verlag, 2002.

ANGELOV, P. Evolving fuzzy systems. *Scholarpedia*, v. 3, n. 2, p. 6274, 2008.

ANGELOV, P. Evolving Takagi-Sugeno fuzzy systems from streaming data (eTS+). In: ANGELOV, P.; FILEV, D. P.; KASABOV, N. (Ed.). *Evolving Intelligent Systems: Methodology and Applications*. Hoboken, NJ, USA: Wiley & IEEE Press, 2010. p. 21–50.

ANGELOV, P.; FILEV, D. An approach to online identification of Takagi-Sugeno fuzzy models. *IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics)*, v. 34, n. 1, p. 484–498, 2004.

ANGELOV, P.; FILEV, D. Simpl_eTS: a simplified method for learning evolving Takagi-Sugeno fuzzy models. *The 14th IEEE International Conference on Fuzzy Systems*, Reno, NV, USA, p. 1068–1073, 2005.

ANGELOV, P.; ZHOU, X. Evolving fuzzy systems from data streams in real-time. *International Symposium on Evolving Fuzzy Systems*, Ambleside, UK, p. 29–35, 2006.

ASTRÖM, K. J.; WITTENMARK, B. *Computer-Controlled Systems: Theory and Design*. 3. ed. Upper Saddle River, NJ, USA: Prentice Hall, 1996.

BARGIELA, A.; PEDRYCZ, W.; TANAKA, M. An inclusion/exclusion fuzzy hyperbox classifier. *International Journal of Knowledge-based and Intelligent Engineering Systems*, IOS Press, v. 8, n. 2, p. 91–98, 2004.

BARUAH, R. D.; ANGELOV, P. Evolving fuzzy systems for data streams: a survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, Wiley-Blackwell, v. 1, n. 6, p. 461–476, 2011.

BENESTY, J.; PALEOLOGU, C.; CIOCHINA, S. Regularization of the RLS algorithm. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E94-A, n. 8, p. 1628–1629, 2011.

BUENO, R. de L. *Econometria de Séries Temporais*. 2. ed. São Paulo: Cengage Learning, 2015.

BURGES, C. J. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, v. 2, n. 2, p. 121–167, 1998.

CHIU, S. L. Fuzzy model identification based on cluster estimation. *Journal of Intelligent & Fuzzy Systems*, IOS Press, v. 2, n. 3, p. 267–278, 1994.

DITZLER, G.; ROVERI, M.; ALIPPI, C.; POLIKAR, R. Learning in nonstationary environments: A survey. *IEEE Computational Intelligence Magazine*, v. 10, n. 4, p. 12–25, 2015.

DOVŽAN, D.; LOGAR, V.; ŠKRJANC, I. Solving the sales prediction problem with fuzzy evolving methods. *IEEE International Conference on Fuzzy Systems*, Brisbane, QLD, Australia, p. 1–8, 2012.

DOVŽAN, D.; ŠKRJANC, I. Recursive clustering based on a gustafson–kessel algorithm. *Evolving Systems*, Springer Nature, v. 2, n. 1, p. 15–24, 2010.

DUDA, R. O.; HART, P. E.; STORK, D. G. *Pattern Classification*. 2. ed. New York, NY: Wiley-Interscience, 2000.

EYKHOFF, P. *System Identification: Parameter and State Estimation*. London, UK: Wiley-Interscience, 1974.

FILEV, D.; GEORGIEVA, O. An extended version of the gustafson-kessel algorithm for evolving data stream clustering. In: ANGELOV, P.; FILEV, D. P.; KASABOV, N. (Ed.). *Evolving Intelligent Systems: Methodology and Applications*. Hoboken, NJ, USA: Wiley & IEEE Press, 2010. p. 273–299.

FULLER, W. A. *Introduction to Statistical Time Series*. 2. ed. New York, NY: Wiley-Interscience, 1995.

GABRYS, B.; BARGIELA, A. General fuzzy min–max neural network for clustering and classification. *IEEE Transactions on Neural Networks*, v. 11, n. 3, p. 769–783, 2000.

GAMA, J. A survey on learning from data streams: current and future trends. *Progress in Artificial Intelligence*, Springer Nature, v. 1, n. 1, p. 45–55, 2012.

GAMA, J.; ZLIOBAITE, I.; BIFET, A.; PECHENIZKIY, M.; BOUCHACHIA, A. A survey on concept drift adaptation. *ACM Computing Surveys*, v. 46, n. 4, p. 44:1–44:37, 2014.

GRAY, R. Vector quantization. *IEEE ASSP Magazine*, v. 1, n. 2, p. 4–29, 1984.

HASTIE, T.; TIBSHIRANI, R.; FRIEDMAN, J. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. 2. ed. New York, NY: Springer New York, 2009.

HAYKIN, S. *Neural Networks: A Comprehensive Foundation*. 2. ed. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1998.

HUANG, G.-B.; SARATCHANDRAN, P.; SUNDARARAJAN, N. An efficient sequential learning algorithm for growing and pruning rbf (gap-rbf) networks. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, v. 34, n. 6, p. 2284–2292, 2004.

INVESTOPEDIA. *Standard & Poor's 500 Index*. 2017. Disponível em: <https://www.investopedia.com/terms/s/sp500.asp>. Acesso em: 25 de Janeiro de 2018.

JAMES, G.; WITTEN, D.; HASTIE, T.; TIBSHIRANI, R. *An Introduction to Statistical Learning: with Applications in R*. New York, NY: Springer, 2017.

JANG, J. S. R. Anfis: adaptive-network-based fuzzy inference system. *IEEE Transactions on Systems, Man, and Cybernetics*, v. 23, n. 3, p. 665–685, 1993.

JOHANSSON, M. *Piecewise Linear Control Systems: A Computational Approach*. Berlin: Springer Berlin Heidelberg, 2002.

KASABOV, N. K. Evolving fuzzy neural networks — Algorithms, applications and biological motivation. In: YAMAKAWA, T.; MATSUMOTO, G. (Ed.). *Methodologies for the Conception, Design and Application of Soft Computing*. Fukuoka, Japão: World Scientific, 1998. p. 271–274.

KASABOV, N. K.; SONG, Q. DENFIS: dynamic evolving neural-fuzzy inference system and its application for time-series prediction. *IEEE Transactions on Fuzzy Systems*, v. 10, n. 2, p. 144–154, 2002.

KOHONEN, T. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, Springer-Verlag, v. 43, n. 1, p. 59–69, 1982.

KOHONEN, T. *Self-Organization and Associative Memory*. Berlin: Springer Berlin Heidelberg, 1989.

KOMIJANI, M.; LUCAS, C.; ARAABI, B. N.; KALHOR, A. Introducing evolving Takagi–Sugeno method based on local least squares support vector machine models. *Evolving Systems*, Springer Nature, v. 3, n. 2, p. 81–93, 2011.

KOMIJANI, M.; LUCAS, C.; ARAABI, B. N.; KALHOR, A. *Matlab code for the paper Introducing evolving Takagi–Sugeno method based on local least squares support vector machine models*. 2013. Disponível em: <https://www.researchgate.net/publication/258770344_Matlab_code_for_the_paper_Introducing_evolving_Takagi-Sugeno_method_based_on_local_least_squares_support_vector_machine_models>. Acesso em: 20 de Maio de 2018.

KWIATKOWSKI, D.; PHILLIPS, P. C.; SCHMIDT, P.; SHIN, Y. Testing the null hypothesis of stationarity against the alternative of a unit root: How sure are we that economic time series have a unit root? *Journal of Econometrics*, Elsevier, v. 54, n. 1, p. 159 – 178, 1992.

LEMOS, A.; CAMINHAS, W.; GOMIDE, F. Multivariable gaussian evolving fuzzy modeling system. *IEEE Transactions on Fuzzy Systems*, v. 19, n. 1, p. 91–104, 2011.

LIMA, E. *Modelagem Fuzzy Funcional Evolutiva Participativa*. Dissertação (Mestrado) — Faculdade de Engenharia Elétrica e Computação, Universidade Estadual de Campinas, Campinas - SP, 2008.

LIMA, E.; HELL, M.; BALLINI, R.; GOMIDE, F. Evolving fuzzy modeling using participatory learning. In: ANGELOV, P.; FILEV, D. P.; KASABOV, N. (Ed.). *Evolving Intelligent Systems*. Hoboken, NJ, USA: Wiley & IEEE Press, 2010. p. 67–86.

LJUNG, L. *System Identification: Theory for the User*. 2. ed. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1999.

LUGHOFER, E. FLEXFIS: A robust incremental learning approach for evolving Takagi–Sugeno fuzzy models. *IEEE Transactions on Fuzzy Systems*, v. 16, n. 6, p. 1393–1410, 2008.

LUGHOFER, E. *Evolving Fuzzy Systems: Methodologies, Advanced Concepts and Applications*. 1. ed. Berlin: Springer-Verlag Berlin Heidelberg, 2011.

MA, D.; LIU, J.; WANG, Z. The pattern classification based on fuzzy min-max neural network with new algorithm. In: WANG, J.; YEN, G. G.; POLYCARPOU, M. M. (Ed.). *Advances in Neural Networks*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. p. 1–9.

MACIEL, L.; GOMIDE, F.; BALLINI, R. An enhanced approach for evolving participatory learning fuzzy modeling. *IEEE Conference on Evolving and Adaptive Intelligent Systems*, Madrid, Spain, p. 23–28, 2012.

MACIEL, L. dos S. *Estimação e Previsão da Estrutura a Termo das Taxas de Juros Usando Técnicas de Inteligência Computacional*. Dissertação (Mestrado) — Faculdade de Engenharia Elétrica e Computação, Universidade Estadual de Campinas, Campinas - SP, 2012.

MASCIOLI, F. F.; MARTINELLI, G. A constructive approach to neuro-fuzzy networks. *Signal Processing*, Elsevier, v. 64, n. 3, p. 347–358, 1998.

MATHWORKS. *Recursive Algorithms for Online Parameter Estimation*. 2015. Disponível em: <https://www.mathworks.com/help/ident/ug/recursive-algorithms-for-online-estimation.html>. Acesso em: 31 de Outubro de 2017.

MOHAMMED, M. F.; LIM, C. P. An enhanced fuzzy min–max neural network for pattern classification. *IEEE Transactions on Neural Networks and Learning Systems*, v. 26, n. 3, p. 417–429, 2015.

MOHAMMED, M. F.; LIM, C. P. A new hyperbox selection rule and a pruning strategy for the enhanced fuzzy min–max neural network. *Neural Networks*, Elsevier, v. 86, n. Supplement C, p. 69 – 79, 2017.

MUTHUKRISHNAN, S. Data streams: Algorithms and applications. *Foundations and Trends in Theoretical Computer Science*, v. 1, n. 2, p. 117–236, 2005.

NANDEDKAR, A. V.; BISWAS, P. K. A fuzzy min–max neural network classifier with compensatory neuron architecture. *IEEE Transactions on Neural Networks*, v. 18, n. 1, p. 42–54, 2007.

NASON, G. A test for second-order stationarity and approximate confidence intervals for localized autocovariances for locally stationary time series. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, v. 75, n. 5, p. 879–904, 2013.

PEDRYCZ, W.; GOMIDE, F. *Fuzzy Systems Engineering: Toward Human-Centric Computing*. Hoboken, New Jersey: John Wiley & Sons, Inc., 2007.

PORTO, A.; GOMIDE, F. Evolving granular fuzzy min-max modeling. In: BARRETO, G. A.; COELHO, R. (Ed.). *Fuzzy Information Processing*. Cham: Springer International Publishing, 2018. p. 37–48.

PORTO, A.; GOMIDE, F. Evolving granular fuzzy min-max regression. In: MELIN, P.; CASTILLO, O.; KACPRZYK, J.; REFORMAT, M.; MELEK, W. (Ed.). *Fuzzy Logic in Intelligent System Design: Theory and Applications*. Cham: Springer International Publishing, 2018. p. 162–171.

PRATAMA, M.; ANAVATTI, S. G.; ANGELOV, P. P.; LUGHOFER, E. PANFIS: A novel incremental learning machine. *IEEE Transactions on Neural Networks and Learning Systems*, v. 25, n. 1, p. 55–68, 2014.

PRIESTLEY, M. B.; RAO, T. S. A test for non-stationarity of time-series. *Journal of the Royal Statistical Society. Series B (Methodological)*, v. 31, n. 1, p. 140–149, 1969.

RAHMAN, S.; HAZIM, O. A generalized knowledge-based short-term load-forecasting technique. *IEEE Transactions on Power Systems*, v. 8, n. 2, p. 508–514, 1993.

RIZZI, A.; PANELLA, M.; MASCIOLI, F. F. Adaptive resolution min-max classifiers. *IEEE Transactions on Neural Networks*, v. 13, n. 2, p. 402–414, 2002.

RONG, H. J.; SUNDARARAJAN, N.; HUANG, G.-B.; SARATCHANDRAN, P. Sequential adaptive fuzzy inference system (SAFIS) for nonlinear system identification and prediction. *Fuzzy Sets and Systems*, Elsevier, v. 157, n. 9, p. 1260–1275, 2006.

RUBIO, J. J. SOFMLS: Online self-organizing fuzzy modified least-squares network. *IEEE Transactions on Fuzzy Systems*, v. 17, n. 6, p. 1296–1309, 2009.

SEERA, M.; LIM, C. P.; LOO, C. K.; SINGH, H. A modified fuzzy min–max neural network for data clustering and its application to power quality monitoring. *Applied Soft Computing*, Elsevier, v. 28, p. 19–29, 2015.

SHINDE, S. V.; KULKARNI, U. V.; CHAUDHARY, A. N. Extracting the classification rules from general fuzzy min-max neural network. *International Journal of Computer Applications*, v. 121, n. 23, p. 1–7, 2015.

SIMPSON, P. Fuzzy min–max neural networks. i. classification. *IEEE Transactions on Neural Networks*, Institute of Electrical and Electronics Engineers (IEEE), v. 3, n. 5, p. 776–786, 1992.

SIMPSON, P. Fuzzy min–max neural networks - part 2: Clustering. *IEEE Transactions on Fuzzy Systems*, Institute of Electrical and Electronics Engineers (IEEE), v. 1, n. 1, p. 32, 1993.

TAGLIAFERRI, R.; ELEUTERI, A.; MENEGANTI, M.; BARONE, F. Fuzzy min–max neural networks: from classification to regression. *Soft Computing*, Springer Nature, v. 5, n. 1, p. 69–76, 2001.

TAKAGI, T.; SUGENO, M. Fuzzy identification of systems and its applications to modeling and control. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-15, n. 1, p. 116–132, 1985.

TAN, J.; QUEK, C. A BCM theory of meta-plasticity for online self-reorganizing fuzzy-associative learning. *IEEE Transactions on Neural Networks*, v. 21, n. 6, p. 985–1003, 2010.

TSYMBAL, A. The problem of concept drift: Definitions and related work. *Department of Computer Science, Trinity College*, Dublin, Ireland, 2004.

VAPNIK, V. N. *Statistical Learning Theory*. Hoboken, New Jersey: Wiley-Interscience, 1998.

YAGER, R. R. A model of participatory learning. *IEEE Transactions on Systems, Man, and Cybernetics*, v. 20, n. 5, p. 1229–1234, 1990.

YAGER, R. R.; FILEV, D. P. Generation of fuzzy rules by mountain clustering. *Journal of Intelligent & Fuzzy Systems*, IOS Press, v. 2, n. 3, p. 209–219, 1994.

YAHOO! *Historical Data of S&P-500 Index*. 2017. Disponível em: <https://finance.yahoo.com/quote/%5EGSPC/history?p=%5EGSPC>. Acesso em: 15 de Dezembro de 2017.

ZADEH, L. Fuzzy sets. *Information and Control*, Elsevier, v. 8, n. 3, p. 338–353, 1965.

ZHANG, H.; LIU, J.; MA, D.; WANG, Z. Data-core-based fuzzy min–max neural network for pattern classification. *IEEE Transactions on Neural Networks*, v. 22, n. 12, p. 2339–2352, 2011.