Tiago Mazzutti

MODELO *INCREMENTAL NEURO-FUZZY GAUSSIAN MIXTURE NETWORK* (INFGMN)

Tese submetida ao Programa de Pós-Graduação em Ciência da Computação para a obtenção do Grau de Doutor em Ciência da Computação.
Orientador: Profº Dr. Mauro Roisenberg
Co-orientador: Profº Dr. Paulo José de Freitas Filho

Florianópolis
2018

Ficha de identificação da obra elaborada pelo autor, através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Mazzutti, Tiago
Modelo Incremental Neuro-Fuzzy Gaussian Mixture Network (INFGMN) / Tiago Mazzutti ; orientador, Mauro Roisenberg, coorientador, Paulo José de Freitas Filho, 2018.
111 p.

Tese (doutorado) - Universidade Federal de Santa Catarina, Centro Tecnológico, Programa de Pós Graduação em Ciência da Computação, Florianópolis, 2018.

Inclui referências.

1. Ciência da Computação. 2. Aprendizagem Incremental. 3. Mamdani-Larsen Fuzzy. 4. Dilemas Estabilidade-Plasticidade e Acuracia Interpretabilidade. 5. Sistema NeuroFuzzy. I. Roisenberg, Mauro . II. de Freitas Filho, Paulo José . III. Universidade Federal de Santa Catarina. Programa de Pós-Graduação em Ciência da Computação. IV. Título.

TIAGO MAZZUTTI

MODELO *INCREMENTAL NEURO-FUZZY GAUSSIAN MIXTURE NETWORK* (INFGMN)

Esta tese foi julgada adequada para obtenção do Título de “Doutor em Ciência da Computação", e aprovada em sua forma final pelo Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Santa Catarina.

Florianópolis, 12 de dezembro de 2018

Profº José Luís Almada Güntzel, Dr.
Coordenador do Curso

Banca Examinadora:

Profº Mauro Roisenberg, Dr.
Orientador

Profº Paulo José de Freitas Filho, Dr.
Co-orientador

Profº Fernando Antonio Campos Gomide, Dr.
Universidade Estadual de Campinas
(participação por video-conferência)

Profº Milton Roberto Heinen, Dr.
Universidade Federal do Pampa
(participação por video-conferência)

Profª Silvia Modesto Nassar, Dra.
Universidade Federal de Santa Catarina

Dedico este trabalho à minha família, em especial a minha esposa Samantha e minhas filhas Clara e Lívia que foram compreensivas e estiverem sempre presentes me apoiando e me amparando no momentos difíceis.

AGRADECIMENTOS

A realização desta tese de doutorado contou com importantes apoios e incentivos sem os quais não se teria tornado uma realidade e aos quais estarei eternamente grato.

Primeiramente, a minha esposa, Samantha Lemke Gonzalez, pelo apoio, amor, compreensão e, principalmente, paciência dedicada a mim. Nós sabemos que nesse período do doutorado ocorreram grandes emoções! E você esteve presente em todos esses momentos. As minha filhas, Clara Lemke Mazzutti e Lívia Lemke Mazzutti, primeiro peço desculpas por não estar presente integralmente com vocês e agradeço por demonstrarem seu amor através de risadas, sorrisos e abraços.

Aos professores que, de diversas e distintas formas, se constituem em incentivos para a construção da tese.

Ao Professor Doutor Mauro Roisenberg e ao Professor Doutor Paulo José de Freitas Filho, pelas suas orientações, total apoio, disponibilidade, pelos saberes que transmitiram, pelas opiniões e críticas, total colaboração no solucionar de dúvidas e problemas que foram surgindo ao longo da realização deste trabalho e por todas as palavras de incentivo.

Aos componentes da banca de qualificação do projeto de tese e defesa, professores comprometidos em apontar os limites e as possibilidades que escolhemos traçar por meio da pesquisa: Fernando Antonio Campos Gomide (Unicamp), Heloisa de Arruda Camargo (UFSCar), Milton Roberto Heinen (Unipampa) e Silvia Modesto Nassar (UFSC).

E, por fim, registro os agradecimentos institucionais.

À Universidade Federal de Santa Catarina (UFSC), ao seu Programa de Pós-Graduação - Doutorado em Ciência da Computação - e sua correspondente secretaria, por mostrarem-se prestativos e solidários na orientação das questões estruturais do curso, bem como na organização documental.

Ao Instituto Federal Catarinense, *campus* Concórdia, por possibilitar a participação no Programa de Incentivo à Qualificação Profissional - PIQFIC (afastamento parcial), bem como no do Afastamento Integral.

Agradeço à Petrobras pelo suporte financeiro e pela oportunidade através do projeto que deu origem à minha tese. Agradeço também aos meus colegas de projeto Mariana e Rodolfo e também aos professores responsáveis Paulo, Silvia e Mauro pela experiência, sugestões e contribuições durante a execução desse trabalho.

Concórdia (SC), dezembro de 2018
Tiago Mazzutti

RESUMO

MAZZUTTI, Tiago. Modelo *Incremental Neuro-Fuzzy Gaussian Mixture Network* (INFGMN). Florianópolis, 2019. 115 f. Tese (Doutorado) Programa de Pós-Graduação em Ciência da Computação, UFSC, Florianópolis – SC.

Acurácia e interpretabilidade representam dois objetivos complementares em qualquer técnica de aprendizagem de máquina. Equilibrar entre esses critérios é um grande desafio ao se considerar um modelo que aprenda e se adapte incrementalmente à dinâmica do problema em questão, e ainda trate o problema de dados faltantes devido à perdas ou falhas na coleta dos mesmos. O objetivo não é apenas maximizar a interpretabilidade, mas também garantir alta acurácia. Neste contexto, há pouca pesquisa sobre a aprendizagem incremental utilizando modelos fuzzy do tipo Mamdani-Larsen (ML). Esta tese apresenta uma nova proposta de sistema neurofuzzy (NFS) - com capacidade de aprendizagem incremental - chamada INFGMN (*Incremental Neuro-Fuzzy Gaussian Mixture Network*), que permite obter modelos incrementais interpretáveis e acurados. As principais características da INFGMN são: (i) aprende incrementalmente usando uma única varredura sobre os dados de treinamento; (ii) pode produzir boas estimativas com base em poucos dados de treinamento, mesmo em caso de dados faltantes, cujo o mecanismo de falta seja completamente aleatório (MCAR) ou aleatório (MAR); (iii) o processo de aprendizagem (possivelmente com a realização de imputação de dados) pode prosseguir perpetuamente à medida que os novos dados de treinamento chegam (não há fases separadas para aprendizagem (*learning*) e utilização (*recalling*)); (iv) pode lidar com o dilema Estabilidade-Plasticidade e não é afetada pela interferência catastrófica ao passo que regras são adicionadas ou retiradas sempre que necessário. Desse modo, o processo de imputação é realizado sempre com uma rede atualizada e adaptada aos dados processados até o momento; (v) a base de regras fuzzy é definida automaticamente e de forma incremental; e (vi) mantém uma base de regras fuzzy do tipo ML que permite fornecer uma boa relação custo-benefício entre acurácia e interpretabilidade, e diferentemente de outras redes neurofuzzy, não necessita que os dados faltantes sejam imputados previamente. O desempenho do modelo INFGMN pode meio de várias aplicações referência e os resultados apresentados em termos de acurácia e interpretabilidade são promissores.

PALAVRAS-CHAVE: Aprendizagem Incremental, Mamdani-Larsen Fuzzy, Dilemas *Estabilidade-Plasticidade* e *Acuracia-Interpretabilidade*, Sistema NeuroFuzzy, Imputação.

ABSTRACT

MAZZUTTI, Tiago. INFGMN - Incremental Neuro-Fuzzy Gaussian Mixture Network. Florianópolis, *2019*. 115 f. *Thesis (Doctoral) Post-Graduate in Computer Scienci*, UFSC, Florianópolis – SC.

Accuracy and interpretability are contradictory objectives that conflict in all machine learning techniques. Balancing among these criteria becomes even more challenging when considering a model that learns and adapts itself incrementally to the dynamics of the problem in question, and additionally treat incomplete data due to losses or failures in the data collection process. The goal is not only to maximize interpretability, but also to ensure high accuracy. In this context, there is little research on incremental learning using Mandani-Larsen (ML) fuzzy models. This thesis presents a novel proposal for a Neuro-Fuzzy System (NFS) with an incremental learning capability, the Incremental Neuro-Fuzzy Gaussian Mixture Network (INFGMN), that attempts to generate incremental models that are highly interpretable and precise. The principal characteristics of the INFGMN are as follows: (i) the INFGMN learns incrementally using a single sweep of the training data; (ii) it is capable of producing reasonable estimates based on few training data, even in the presence of missing data; (iii) the learning process can proceed in perpetuity as new training data become available (learning and recalling phases are not separate, possibly with imputation of missing data); (iv) the INFGMN can deal with the Stability-Plasticity dilemma and is unaffected by catastrophic interference (rules are added or removed whenever necessary). In this way, the imputation process is always carried out with an up-to-date network adapted to the data seen so far; (v) the fuzzy rule base is defined automatically and incrementally; and (vi) the INFGMN maintains an ML-type fuzzy rule base that attempts to provide the best trade-off between accuracy and interpretability, thereby dealing with the Accuracy-Interpretability dilemma and unlike other neuro-fuzzy networks, the INFGMN does not require that the missing data be prefilled before the training and use of the network (this is done during its use and adaptively). The INFGMN's performance in terms of learning and modelling is assessed using a variety of benchmark applications and the results are promising.

KEYWORDS: *Incremental Learning, Mamdani-Larsen Fuzzy, Stability-Plasticity and Acuracy-Interpretability Dilemmas, Neuro-Fuzzy System, Imputation.*

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ALGORITMOS

LISTA DE ABREVIATURAS E SIGLAS

ANFIS	*Adaptive-Network-based Fuzzy Inference*
CBR	*Case Based Reasoning*
CCAI	*Credal Classification with Adaptive Imputation*
DENFIS	*Dynamic Evolving Neurofuzzy Inference System*
ECoS	*Evolving Connectionist System*
eFSM	*Evolving Neural-fuzzy Semantic Memory*
EFuNN	*Evolving Fuzzy Neural Network*
EM	*Expectation-Maximization*
eTS	*evolving Takagi-Sugeno*
FBeM	*Fuzzy set Based evolving Modeling*
FCMI	*Fuzzy c-means Imputation*
FIS	*Fuzzy Inference Systems*
FM	*Fuzzy Modeling*
FRBS	*Fuzzy Rule Based Systems*
GFM	*Generalized Fuzzy Model*
GMM	*Gaussian Mixture Model*
GRAANN	*General Regression Auto Associative Neural Network*
IGMM	*Incremental Gaussian Misture Model*
INFGMNI	*Incremental Neurofuzzy Gaussian Mixture Network Imputation*
KNNI	*K-nearest Neighbors Imputation*
LFM	*Linguistic Fuzzy Modeling*
LOCF	*Last Observation Carried Forward*
MA	*Mean Average*
MAPE	*Mean Absolute Percentage Error*
MAR	*Missing at random*
MCAR	*Missing completely at random*
MIMO	*Multi Input Multiple Output*
MISO	*Multi Input Single Output*
ML	Mamdani-Larsen
NDEI	*Non-Dimensional Error Index*
NFS	*Neurofuzzy System*
NMAR	*Not missing at random*
PDF	*Probability Density Function*
PFM	*Precise Fuzzy Modeling*

PSOAANN	*Particle Swarm Optimization Trained Auto Associative Neural Network*
PSOAAWNN	*Particle Swarm Optimization Trained Auto Associative Wavelet Neural Network*
RBFAANN	*Radial Basis Function Auto Associative Neural Network*
RMSE	*Root Mean Squared Error*
S&P-500	*Standard and Poor's 500*
SaFIN	*Self-adaptive Fuzzy Inference Network*
SeroFAM	*Self-reorganizing Fuzzy Associative Machine*
SISO	*Single Input Single Output*
SOMI	*Self-Organizing Map Imputation*
SPLAFIS	*Sequential Probabilistic Learning for Adaptive Fuzzy Inference System*
TS	Takagi-Sugeno

LISTA DE SÍMBOLOS

δ	fração definida pelo usuário
$\Longrightarrow$	operador de implicação fuzzy
λ^k	k-ésima função de pertinência fuzzy na entrada
$\wedge$	operador de conjunção fuzzy
$\vee$	operador de disjunção fuzzy
$\mathbb{R}$	conjunto dos números reais
$\mathcal{A}, \mathcal{B}$	conjuntos fuzzy
$\mathcal{G}_j$	a j-ésima componente de mistura Gaussiana
$\mathcal{IV}_i$	a i-ésima variável linguística fuzzy de entrada
$\mathcal{M}$	mecanismo de dados faltantes
$\mathcal{N}^D$	distribuição normal D-dimensional
$\mathcal{OV}_o$	a o-ésima variável linguística fuzzy de saída
$\mathcal{R}^k$	k-ésima regra fuzzy
$\mathcal{X}$	vetor de dados de entrada
$\mathcal{X}^{(t)}$, $J^{(t)}$ e $K^{(t)}$	o estado destes vetores no tempo t
$\mathcal{X}_d$	valor da d-ésima dimensão na entrada
$\mathcal{Y}_m$	o valor da m-ésima dimensão na saída
μ_j	j-ésimo vetor de médias de um GMM
μ	matriz de médias
ϕ^k	k-ésima função de pertinência fuzzy na saída
σ_{ini}	valor usado para inicializar a matriz de covariância de uma nova componente de mistura
σ_{mul}	valor usado para aumentar ou diminuir o tamanho (*spread*) das funções de pertinência Gaussianas
τ_{nov}	parâmetro de limiar de verosimilhança (ou novidade)
R	Percentual de dados faltantes
$\widetilde{\mathcal{X}}$	vetor de dados faltantes
ξ	parâmtros desconhecidos
b_k	centróide da k-ésima regra fuzzy
C	matriz de covariâncias
C_j	j-ésima matriz de covariância de um GMM
D	número de dimensões do vetor de dados
I	matriz identidade
J	número de componentes em um GMM
M	número de dimensões do vetor de saída

n	número total de valores faltantes
O_{num}	número de características (variáveis) no vetor de entrada
p	proporções de mistura
$p(j)$	j-ésima proporção de mistura ou probabilidade à *priori*
sp_j	j-ésimo acumulador de probabilidade à posteriori
sp_{min}	mínimo de relevância/utilidade a ser demonstrado
t_j	j-ésimo acumulador de tempo de vida
t_{max}	tempo máximo para demonstrar relevância/utilidade
U	conjunto universo
$v_k,\ w_k,\ ou\ \mathcal{W}_k$	peso da k-ésima regra fuzzy

SUMÁRIO

1 INTRODUÇÃO E CONTEXTUALIZAÇÃO

Acurácia e interpretabilidade representam dois objetivos complementares em qualquer técnica de inteligência artificial. Em um cenário ideal, seria interessante maximizar ambos os critérios, mas como esses objetivos são opostos, isso geralmente não é possível. Desta forma, muitos pesquisadores têm se concentrado em melhorar o equilíbrio entre acurácia e interpretabilidade, dependendo da natureza (pré-requisitos) do modelo. Em geral, é dada prioridade a um desses objetivos (CASILLAS et al., 2003; CASILLAS, 2003; ALCALÁ et al., 2006; GACTO; ALCALÁ; HERRERA, 2011; LAPA; CPALKA; RUTKOWSKI, 2018).

Inicialmente, muitas técnicas de inteligência artificial foram empregadas por especialistas humanos para gerar modelos a partir de seus conhecimentos especializados (ZADEH, 1973, 1975; TAKAGI; SUGENO, 1985). Por exemplo, em um Sistema Baseado em Regras Fuzzy (FRBS, do inglês *Fuzzy Rule Based Systems*), uma das tarefas mais importantes é a derivação do seu conhecimento (sua base de regras fuzzy). Isso pode ser feito manualmente, gerando-se bases de regras fuzzy rigidamente fixadas que normalmente não podem ser adaptadas ou ajustadas para um melhor desempenho após a fase inicial de projeto. Posteriormente, como alternativa à derivação "manual"(construção manual) do modelo, os pesquisadores trabalharam na construção direta e no ajuste de um sistema fuzzy a partir de dados numéricos de treinamento, como meio de atenuar o problema da aquisição do conhecimento. Uma abordagem popular é o uso de redes neurais para derivar a estrutura de um sistema neural fuzzy ou Sistema Neurofuzzy (NFS, do inglês *Neurofuzzy System*) (BUCKLEY; HAYASHI, 1994; LIN; LEE, 1996). O exemplo mais conhecido na literatura e a *Adaptive-Network-based Fuzzy Inference* (ANFIS) (JANG, 1993).

Ao se focar em técnicas de aprendizagem de máquina que extraem seus modelos a partir de dados, o conhecimento extraído deve ser compreensível a um ser humano, ou seja, não se deseja que este modelo seja do tipo *black-box*. No caso de FRBS, quando construídos a partir do conhecimento especialista, estes apresentam um modelo bem compreensível com acurácia satisfatória. Mas quando a estrutura do sistema fuzzy é derivado com o uso de redes neurais, a maioria dos métodos se concentram em melhorar a acurácia do modelo (GUILLAUME; MAGDALENA, 2006; SHIHABUDHEEN; PILLAI, 2018). Para Casillas et al. (2003) e Casillas (2003), o desafio está em como combinar conhecimento extraído dos dados com o conhecimento especializado, de modo a gerar sistemas compactos e robustos, com equilíbrio entre acurácia e interpretabilidade, que geralmente é chamado de dilema Acurácia-

Interpretabilidade (*Accuracy-Interpretabiliity*).

Uma vez que os procedimentos de treinamento ou geração de regras usados pela maioria desses NFS derivados a partir de redes neurais, pressupõe que as características dos processos subjacentes sendo modelados não mudam com o tempo, eles são aplicáveis apenas a ambientes estáticos. Na maioria dos casos, abordagens em bateladas (*batch mode*) ou pseudo-incrementais são empregadas para aprender e refinar os modelos gerados e, portanto, a maioria dos NFS não são adequados para a modelagem de processos mais complexos que variam no tempo em ambientes dinâmicos e que muitas podem apresentar dados faltantes devido à perdas ou falhas nos mecanismos de coleta em tempo real (por exemplo, em uma aplicação industrial, alguns dados podem estar faltando devido a falhas mecânicas/eletrônicas durante o processo de aquisição de dados) (SHIHABUDHEEN; PILLAI, 2018).

A adaptação ao ambiente em constante mudança geralmente requer uma fase de re-treino para construir um novo modelo fuzzy a partir do conjunto atualizado de dados de treinamento. No entanto, o processo de aprendizagem de novas informações pode potencialmente modificar o modelo fuzzy. Tais mudanças, podem afetar o conhecimento originalmente aprendido (que ainda é válido), fazendo com que este seja esquecido ou substituído. Sendo assim, estes NFS podem cair em um problema conhecido como o dilema Estabilidade-Plasticidade (*Stability-Plasticity*) (veja Grossberg (1982), Carpenter e Grossberg (1988) ou Mermillod, Bugaiska e Bonin (2013) para obter mais informações sobre este dilema).

Mais recentemente foram propostos novos NFS, baseados na noção de aprendizagem incremental. Exemplos são: a rede *Evolving Fuzzy Neural Network* (EFuNN) e o *Dynamic Evolving Neurofuzzy Inference System* (DENFIS) baseados no *Evolving Connectionist System* (ECoS), que podem ser encontrados em Kasabov (2001) e Kasabov e Song (2002), respectivamente, a *evolving Takagi-Sugeno* (eTS) que foi uma contribuição de Angelov e Filev (2004), a *Self-reorganizing Fuzzy Associative Machine* (SeroFAM) estudada por Tan e Quek (2010), a rede *Evolving Neural-fuzzy Semantic Memory* (eFSM) proposta por Tung e Quek (2010), a *Self-adaptive Fuzzy Inference Network* (SaFIN) criada por Tung, Quek e Guan (2011), o *Fuzzy set Based evolving Modeling* (FBeM) que foi primeiramente proposto por Leite e Gomide (2012) e posteriormente modificado por Leite et al. (2011) e a *Sequential Probabilistic Learning for Adaptive Fuzzy Inference System* (SPLAFIS) desenvolvida por Oentaryo et al. (2014).

1.1 FORMULAÇÃO DO PROBLEMA

A aprendizagem incremental é um método de aprendizagem de máquina, no qual os novos dados de entrada são usados para ampliar o conhecimento de um modelo existente, ou seja, para treinar mais o modelo. Ele representa uma técnica dinâmica de aprendizado supervisionado e aprendizado não supervisionado que pode ser aplicado quando os dados de treinamento se tornam disponíveis gradualmente ao longo do tempo ou seu tamanho está fora dos limites de memória do sistema (CARPENTER et al., 1992; FERRER-TROYANO; AGUILAR-RUIZ; RIQUELME, 2005; LAMIREL et al., 2010).

Na aprendizagem incremental, assume-se que os padrões de dados são amostrados individualmente. A aprendizagem estrutural (procedimento de geração de regras) e a aprendizagem de parâmetros (adaptação de parâmetros) são feitas de forma incremental, com base apenas na amostra de dados de treinamento corrente, que pode apresentar lacunas devido a falhas na coleta dos mesmos. Assim, o modelo gerado pode evoluir e adaptar seus conhecimentos dependendo da dinâmica do ambiente subjacente. No entanto, como apontado por Tung e Quek (2010), além dos modelos já bem estabelecidos como Falcon-ART (LIN; LIN, 1997), EFuNN, SeroFAM, eFSM e SaFIN, a maioria dos NFS incrementais propostos na literatura baseiam-se em FRBS do tipo Takagi-Sugeno (TS), e não possuem mecanismos para tratar dados faltantes.

Embora FRBS do tipo Mamdani-Larsen (ML) sejam mais interpretáveis que FRBS do tipo TS (TUNG; QUEK, 2009; RIID; RÜSTERN, 2014), há pouca investigação sobre NFS do tipo ML com capacidade de aprendizagem incremental (TUNG; QUEK, 2010). Além disso, como apontado por Tung e Quek (2010), as redes Falcon-ART e EFuNN têm sérias deficiências. A primeira não têm um mecanismo de remoção de regras. Isso pode levar a uma estrutura contendo muitas regras desatualizadas, degradando o nível da interpretabilidade da base de regras fuzzy resultante. Na EFuNN, um procedimento separado (geralmente *offline*) é necessário para identificar os conjuntos fuzzy predefinidos do sistema de regras incremental. As redes SeroFAM, eFSM e SaFIN apresentaram melhorias em relação as suas predecessoras em termos de aprendizagem incremental, lidando melhor com os dilemas Estabilidade-Plasticidade e Acurácia-Interpretabilidade. No entanto, ainda existe bastante espaço para melhorias adicionais, principalmente no que diz respeito ao ajuste fino de parâmetros a fim de manter o equilíbrio entre a acurácia e interpretabilidade.

Considerando-se aplicações do mundo real que podem apresentar dados faltantes (tuplas de dados que apresentam lacunas), é de extrema

importância que os FRBS possuam a capacidade de tratar problemas com dados faltantes, sejam eles problemas de classificação, de regressão e/ou de previsão em séries temporais. O tratamento inadequado dos dados faltantes pode causar grandes erros e/ou falsos resultados. No entanto, nenhum dos FRBS do tipo ML ou TS listados aqui tratam este problema.

Portanto, dadas aplicações do mundo real, a pergunta fundamental considerada nesta tese é: seria possível a criação de uma rede neurofuzzy do tipo ML com capacidade de aprendizagem incremental, que possibilite, por meio de ajustes de parâmetros, a obtenção de equilíbrio entre acurácia e interpretabilidade para o modelo gerado e que seja robusta a problemas com dados faltantes? Tanto quanto pôde-se observar na literatura, não existe nenhum trabalho do tipo ML que permita tal ajuste de parâmetros e que possua capacidade de aprendizagem incremental.

1.2 OBJETIVOS

Objetivo Geral

O objetivo deste trabalho é propor um novo modelo de rede neurofuzzy do tipo ML com capacidade de aprendizagem incremental, que possibilite obter, por meio de ajustes de parâmetros, modelos com acurácia e interpretabilidade balanceados, e que sejam robustos à problemas que apresentem dados faltantes. O modelo também deve permitir permitir aprendizagem, imputação dos dados faltantes e uso contínuo em aplicações de tempo real.

Objetivos Específicos

- Investigar e propor um mecanismo que permita balancear acurácia e interpretabilidade, durante a geração do modelo incremental;
- Investigar técnicas técnicas de imputação e propor um método para tal, tornando os modelos neurofuzzy gerados aptos a lidar com vetores que possuam características incompletas;
- Propor um novo modelo de rede neurofuzzy do tipo ML incremental;
- Avaliar o modelo proposto.

1.3 JUSTIFICATIVA E MOTIVAÇÃO

Atualmente, um grande número de técnicas de aprendizagem de máquina estão disponíveis. Embora a maioria dessas técnicas forneça modelos preditivos precisos e suficientemente flexíveis para serem empregados em uma ampla gama de aplicações, estes são considerados de tipo "caixa preta" (*black-box*), isto é, seu comportamento não pode ser facilmente explicado em termos de sua estrutura ou ainda, de forma mais simples, o sistema tem dificuldade em fornecer uma explicação adequada para como ele chegou a uma solução. Quando se trata da adoção de técnicas de aprendizagem de máquina, isso cria um problema, já que em pesquisa científica, medicina, engenharia e em outras áreas, é preciso haver um rastro entre a entrada e a saída; uma validação para além do fato de que um "sistema inteligente" forneceu uma resposta (DAYHOFF; DELEO, 2001; OLDEN; JACKSON, 2002; SHWARTZ-ZIV; TISHBY, 2017; JAIN; KANDEL; TEODORESCU, 2017; ADLER et al., 2018).

Neste sentido, Alonso, Castiello e Mencar (2015) destacam algumas razões pelas quais o problema da "caixa preta" deve ser considerado: a *Integração*, a *Interação/Validação* e a *Confiabilidade/Explicação*. Em relação a *Integração* em uma técnica de aprendizagem de máquina, o conhecimento adquirido deve ser confirmado e relacionado sem esforço com o conhecimento do domínio do problema de um especialista humano. Já a *Interação/Validação* preocupa-se com o fato de que a linguagem usada para comunicar o conhecimento extraído deve permitir a interação entre o usuário e o modelo, bem como deve ser facilmente validado em relação conhecimentos de senso comum e específicos do domínio do problema. Alguns pesquisadores, como Pazzani e Kibler (1992), Oursston e Mooney (1994), Towell e Shavlik (1994) e Garcez, Broda e Gabbay (2012) utilizam técnicas de aprendizado de máquina para refinar teorias que são "aproximadamente corretas" em uma determinada área. Para completar o processo de refinamento de uma teoria, é importante que o usuário seja capaz de comunicar de forma compreensível as mudanças que devem ser aplicadas ao modelo durante o processo de aprendizagem. Finalmente, a *Confiabilidade/Explicação* é importante para adoção de modelos interpretáveis, devido à sua capacidade de convencer os usuários finais sobre a confiabilidade do modelo. Trabalhos como Wolberg, Street e Mangasarian (1994), Karim e Zhou (2015), Umamaheswari, Sultana e Fatima (2016) e Laugel et al. (2018) podem ser consultados para obter mais informações sobre a importância da interpretabilidade para confiabilidade em técnicas de modelagem indutiva para extração de conhecimento a partir de dados.

Portanto, para se lidar com o problema da "caixa preta", a dispo-

nibilidade de um sistema que forneça uma explicação dos mapeamentos de entrada e saída - de uma rede neural artificial por exemplo - na forma de regras seria útil. Para tal, a extração de regras é uma técnica que tenta elucidar para o usuário como o sistema chegou à sua decisão na forma de regras SE/ENTÃO (SETIONO; LIU, 1995; LU; SETIONO; LIU, 2017). Como exemplo prático tem-se os NFS, que são sistemas que integram as técnicas de aprendizado automático de redes neurais artificiais com a capacidade de representação próxima do raciocínio humano oferecido por sistemas Fuzzy. Assim, há uma fase de treinamento, no qual são apresentados os padrões de treino com entrada/saída e como resultado obtem-se um modelo linguístico que representa o conhecimento extraído dos dados na forma de um sistema de inferência fuzzy (JANG; SUN, 1995; KAR; DAS; GHOSH, 2014; RAJAB; SHARMA, 2018).

Segundo Alonso, Castiello e Mencar (2015), para se lidar com o problema da "caixa preta", chegando-se a soluções mais interpretáveis, é preciso identificar recursos que possam ajudar a tratar este problema. Dentre esses recursos, Alonso, Castiello e Mencar destacam que o mais importante é a ferramenta para a representação do conhecimento e o mais apropriado é a adoção de FRBS do tipo ML, pois estes permitem formulações de conceitos baseados em termos linguísticos, típicos do pensamento humano abstrato.

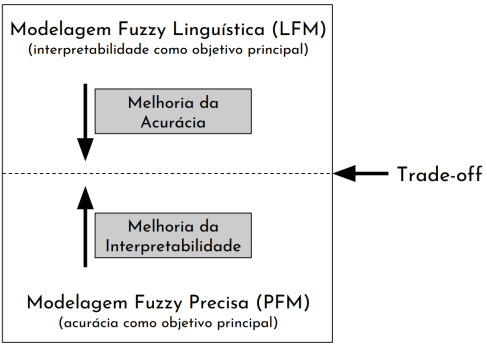

Figura 1.1 - Acurácia x Interpretabilidade *Trade-off*. Baseada em Alcalá et al. (2006)

Independentemente da abordagem, há um método muito comum na literatura (veja Alcalá et al. (2006), por exemplo) para alcançar o equilíbrio desejado entre acurácia e interpretabilidade (Figura 1.1):

i) Primeiramente, o objetivo principal (acurácia ou interpretabilidade) é decidido definindo-se a estrutura de modelo específica a ser utilizada, isto é, a abordagem de Modelagem Fuzzy (FM, do inglês

Fuzzy Modeling), Modelagem Fuzzy Precisa (PFM, do inglês *Precise Fuzzy Modeling*) ou Modelagem Fuzzy Linguística (LFM, do inglês *Linguistic Fuzzy Modeling*) (esses conceitos são melhor discutidos na Seção **2.1.7**).

ii) Em seguida, os componentes de modelagem (a estrutura do modelo e/ou o processo de modelagem) são melhorados através de diferentes mecanismos para compensar a diferença inicial entre os dois requisitos. Assim, propõem-se aprimoramentos de acurácia para LFM, enquanto melhorias de interpretabilidade são propostas para PFM.

Adicionalmente, existem duas abordagens básicas para o refinamento do modelo: (1) estender o design do modelo (criando novas partições fuzzy e/ou aprendendo novas regras); (2) estender a estrutura das regras (modificadores linguísticos, regras ponderadas, regras com exceção, regras padrão, etc.). Quanto mais flexível for o processo de modelagem, maior acurácia poderá ser alcançada, embora o processo se torne mais custoso. Por outro lado, o uso de restrições fortes em favor da interpretabilidade pode ter o custo de reduzir a acurácia. Portanto, encontrar um equilíbrio (*trade-off*) entre acurácia e interpretabilidade é uma das tarefas mais difíceis na FM (ALONSO; MAGDALENA; GONZÁLEZ-RODRÍGUEZ, 2009).

Neste contexto, este trabalho propõe uma nova abordagem para construção e utilização de redes neurofuzzy com capacidade de aprendizagem incremental, que possam ser usadas para a execução de tarefas em tempo real, que podem possivelmente apresentar vetores de dados com características incompletas. Essa proposta foi inspirada a partir de uma revisão sistemática na literatura utilizando as bases IEEE, Elsevier e Springer sobre essa temática, focando em novas pesquisas na área que visam aumentar a interpretabilidade das redes neurofuzzy sem a perda de acurácia. De forma análoga, inspira-se também em Modelos de Mistura de Gaussianas (GMM, do inglês *Gaussian Mixture Model*) incrementais, nos quais a topologia e os pesos da rede neural são definidos automaticamente e de forma incremental. A terceira fonte de inspiração, vem de resultados obtidos por Azeem, Hanmandlu e Ahmad (2000) e Gan, Hanmandlu e Tan (2005), que estabeleceram uma equivalência entre GMM e sistemas fuzzy do tipo ML.

A revisão da literatura é realizada e mostrada no Capítulo 2 e busca evidenciar o ineditismo desta tese. Ele é configurado pela abordagem que emprega técnicas de aprendizagem incremental para a geração de um modelo de rede neurofuzzy do tipo ML, que trata de forma simplificada (dois parâmetros de configuração, que são apresentados no

Capítulo 4) a opção pela acurácia, interpretabilidade ou ambas, e possa ser usado em aplicações de tempo real, que possivelmente apresentem vetores de dados com características incompletas.

1.4 CONTRIBUIÇÕES

As principais contribuições desta tese são:

i) um novo modelo de rede neurofuzzy do tipo ML, que pode aprender de forma incremental (tempo real) usando uma única varredura sobre os dados de treinamento, ou seja, cada padrão de treinamento pode ser imediatamente usado e descartado;

ii) um modelo de rede neurofuzzy do tipo ML que pode produzir boas estimativas com base em poucos dados de treinamento, mesmo com dados faltantes (tuplas de dados incompletas);

iii) o modelo proposto pode lidar com o dilema Estabilidade-Plasticidade (*Stability-Plasticity*) e não é afetada pela interferência catastrófica, de modo que novas regras fuzzy são adicionadas ou retiradas sempre que necessário. Desse modo, o processo de imputação e/ou previsão é sempre realizado com uma rede atualizada e adaptada aos dados vistos até o momento;

iv) a base de regras fuzzy do tipo ML é definida automaticamente e de forma incremental (novas regras são adicionadas sempre que for necessário);

v) o modelo proposto mantém uma base de regras fuzzy do tipo ML que procura fornecer a melhor relação custo-benefício entre acurácia e interpretabilidade, lidando desta forma com o dilema Acurácia-Interpretabilidade (*Accuracy-Interpretability*), e diferentemente de outras redes neurofuzzy, a rede neurofuzzy INFGMN não necessita que os dados faltantes sejam previamente imputados (isto é feito durante o seu uso e de forma adaptativa); e

vi) por fim, a rede INFGMN, por meio de seu método de imputação adaptativo INFGMNI, pode ser utilizada como um mecanismo de imputação de dados faltantes para serem utilizados em outras técnicas de aprendizagem de máquina.

1.5 ORGANIZAÇÃO DO TRABALHO

Nos capítulos que se seguem, esta tese está organizada da seguinte forma:

- O Capítulo 2, faz uma revisão bibliográfica, apresentando conceitos relacionados a Sistemas de Inferência Fuzzy (FIS do inglês *Fuzzy Inference Systems*), redes neurofuzzy (juntamente com as principais propostas existentes na literatura) e a dinâmica do mecanismo de dados faltantes;

- O GMM, o algoritmo *Expectation-Maximization* (EM) e as condições para a equivalência entre um GMM e um modelo fuzzy do tipo ML, juntamente com todos os conceitos relacionados, são apresentados no Capítulo 3. Para finalizar, este capítulo descreve também o *Incremental Gaussian Misture Model* (IGMM), que pode ser visto como uma contrapartida incremental do algoritmo EM;

- No Capítulo 4, a arquitetura, modos de operação, parâmetros de configuração da rede INFGMN (proposta de pesquisa desta tese) e seu método adaptativo de imputação (INFGMNI) são apresentados;

- Para avaliar o desempenho de aprendizagem e de modelagem da rede INFGMN e do seu método de imputação de dados INFGMNI foram utilizadas várias aplicações *benchmark*. Portanto, o Capítulo 5 apresenta os principais resultados e discussões desta avaliação, bem como da proposta desta tese como um todo; e

- Finalmente, o Capítulo 6 apresenta as principais conclusões deste trabalho, bem como um apanhado sobre os possíveis trabalhos futuros.

2 ESTADO DA ARTE

2.1 SISTEMAS DE INFERÊNCIA FUZZY

Sistemas de Inferência Fuzzy (FIS do inglês *Fuzzy Inference Systems*) são modelos computacionais utilizados com o objetivo de mapear determinadas entradas para uma saída fazendo uso da teoria de Conjuntos Fuzzy, da Lógica Fuzzy e de um conjunto de regras do tipo *IF-THEN* (ZADEH, 1973). De acordo com Tanscheit (2004), a teoria de Conjuntos Fuzzy e os conceitos de Lógica Fuzzy fornecem uma base matemática sólida para modelar a imprecisão de sistemas do mundo real. Por exemplo, pode-se considerar a representação da temperatura ambiente, que na teoria de conjuntos seria representada de maneira binária como "quente" ou "frio", pode ser representada na teoria de conjuntos fuzzy com diferentes variações entre "quente" e "frio", que é uma representação mais próxima de como o mundo real funciona.

2.1.1 Conjuntos Fuzzy

Conforme introduzido por Zadeh (1965), a teoria de Conjuntos Fuzzy é uma extensão da teoria dos conjuntos clássicos, com o objetivo de construir uma base matemática sólida para o tratamento de informações imprecisas ou aproximadas. Diferentemente da teoria clássica de conjuntos, na teoria fuzzy de Zadeh usa funções de pertinência mais gerais, que podem assumir valores no intervalo $[0,\ 1]$. Assim, para um determinado conjunto fuzzy $\mathcal{A}$ em um universo U, pode-se representar a pertinência de elementos de U a $\mathcal{A}$ por uma função característica $\lambda(x)$, tal que

$$\lambda(x) :\ U\ \rightarrow\ [0,\ 1], \tag{2.1}$$

onde $\lambda(x)$ é uma função que indica o grau de pertinência de x a $\mathcal{A}$. Existem muitos tipos de funções de pertinência que podem ser empregadas. As mais comuns são funções triangulares, trapezoidais ou Gaussianas (as vezes podem se chamadas de funções de base radial).

2.1.2 Variáveis Linguísticas

De acordo com Zadeh (1965), uma Variável Linguística é uma variável cujos valores são palavras ou sentenças em uma linguagem natural ou artificial. Intuitivamente, uma variável linguística é um substantivo, enquanto seus valores são adjetivos, representados por conjuntos fuzzy. Bassanezi e Barros (2006) define uma variável linguística X no universo U como uma variável cujos valores assumidos são subconjuntos fuzzy de

U (KLIR; YUAN, 1995). Por exemplo, a estatura de uma pessoa pode ser considerada uma Variável Linguística, podendo assumir valores como "baixa", "média" ou "alta". Esses valores são conjuntos fuzzy, representados por funções de pertinência, conforme representado na Figura 2.1 por funções de pertinência triangulares e trapezoidais.

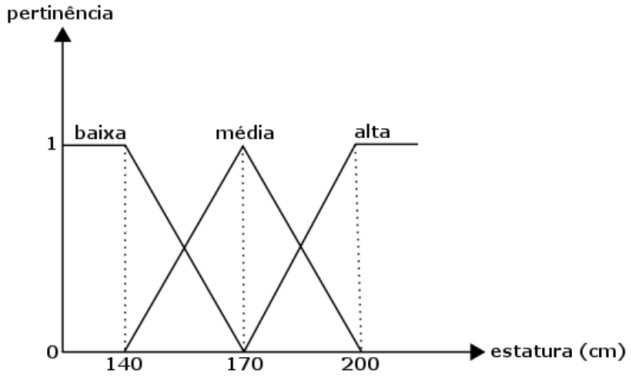

Figura 2.1 - Funções de pertinência para a Variável Linguística estatura

2.1.3 Sistemas Fuzzy

Sistemas fuzzy utilizam conjuntos e regras fuzzy para construir modelos de processos físicos e de processamento de informação. Esses modelos computacionais podem ser usados para tomar decisões parecidas com as decisões humanas (TANSCHEIT, 2004).

Ao utilizar conjuntos fuzzy pode-se construir regras do tipo ***IF-THEN***, como segue:

$$Regra = \begin{cases} \boldsymbol{IF}\ (temperatura\ \boldsymbol{is}\ fria)\ \vee\ (temperatura\ \boldsymbol{is}\ muito\ fria) \\ \boldsymbol{THEN}\ ar\ condicionado\ deve\ aumentar. \end{cases}$$

2.1.4 Modelo Mamdani-Larsen

Mamdani (1974) e Larsen (1980) propuseram um FIS que é atualmente conhecido como Mamdani-Larsen (ML) Fuzzy (LEE, 2004). Neste, as regras fuzzy são descritas por meios de variáveis linguísticas, cujos valores são palavras ou sentenças em uma linguagem natural ou artificial. Essas regras fuzzy, capturam as relações entre as variáveis linguísticas e fornecem um mecanismo para vincular as descrições linguísticas dos sistemas com suas realizações computacionais (PEDRYCZ;

GOMIDE, 2007). A regra

$$\boldsymbol{IF}\ pressão\ \boldsymbol{is}\ alta\ \boldsymbol{THEN}\ volume\ \boldsymbol{is}\ pequeno\ (0.5)$$

é um exemplo de regra fuzzy do tipo ML, onde *alta* e *pequeno* são variáveis linguísticas definidas por funções de pertinência e (0.5) é o peso dado a esta regra.

O modelo ML usa conjunção multiplicativa (ou conjunção probabilística (GUPTA; QI, 1991; LIU; GENG; ZHANG, 2005)) e implicação multiplicativa (ou implicação de Larsen (LEE, 2004)). Supondo-se regras fuzzy da seguinte forma:

$$\mathcal{R}^k:\ \boldsymbol{IF}\ x\ \boldsymbol{is}\ \mathcal{A}_1^k\ \wedge\ y\ \boldsymbol{is}\ \mathcal{A}_2^k\ \boldsymbol{THEN}\ z\ \boldsymbol{is}\ \mathcal{B}^k, k = 1, 2, \ldots, K$$

e $x = x_0$ e $y = y_0$, então o grau de ativação α_k é definido por

$$\alpha_k = \lambda_{\mathcal{A}_1^k}(x_0) \wedge \lambda_{\mathcal{A}_2^k}(y_0) = \lambda_{\mathcal{A}_1^k}(x_0) \cdot \lambda_{\mathcal{A}_2^k}(y_0)$$

e o conjunto fuzzy resultante $\mathcal{B}^{*k}$ para a regra $\mathcal{R}^k$ é definido por

$$\lambda_{\mathcal{B}^{*k}}(z) = \alpha_k \cdot \lambda_{\mathcal{B}^k}(z)$$

e portanto, a avaliação de todas a regras produz como resultado agregado

$$\lambda_{\mathcal{B}^*}(z) = \bigvee_{k=1}^{K} [\alpha_k \cdot \lambda_{\mathcal{B}^k}(z)].$$

2.1.5 Modelo Takagi-Sugeno

O modelo Takagi-Sugeno (TS) foi proposto por Takagi e Sugeno (1985). Neste modelo o consequente das regras fuzzy é uma função das variáveis do antecedente. Um exemplo de regra fuzzy do tipo Takagi-Sugeno é dado por:

$$\boldsymbol{IF}\ x\ \boldsymbol{is}\ A\ \wedge\ y\ \boldsymbol{is}\ B\ \boldsymbol{THEN}\ z = f(x, y).$$

Nesse tipo de regra fuzzy, o consequente de cada regra é uma função que descreve a saída em função das entradas.

2.1.6 Defuzzificação

A defuzzificação é a etapa onde a saída fuzzy é traduzida em um valor numérico. Leekwijck e Kerre (1999) apresentam vários métodos de defuzzificação, sendo os mais comuns:

- **Média dos máximos**: o valor de saída é igual a elementos que correspondem aos maiores valores de pertinência do conjunto fuzzy de saída;
- **Média ponderada dos máximos**: o valor de saída média dos n elementos que correspondem aos maiores valores de pertinência do conjunto fuzzy de saída ponderado pelos graus de pertinência.
- **Centróide ou centro de gravidade**: o valor de saída será o centro de gravidade do conjunto de saída do sistema fuzzy;
- **Critério máximo ou mínimo**: o valor de saída será o valor máximo (ou mínimo) de ativação do conjunto de saída do sistema fuzzy.

2.1.7 Sistema Fuzzy Baseado em Regras

Considerando FRBS em geral, a Modelagem Fuzzy (FM, do inglês *Fuzzy Modeling*) pode ser dividida em Modelagem Fuzzy Precisa (PFM, do inglês *Precise Fuzzy Modeling*) ou Modelagem Fuzzy Linguística (LFM, do inglês *Linguistic Fuzzy Modeling*), dependendo dos requisitos da modelagem do sistema. O objetivo principal da PFM é obter um modelo fuzzy com boa acurácia. Um modelo PFM é desenvolvido através de FRBS do tipo TS e usa variáveis com conjuntos fuzzy associados, mas sem propriamente definir um significado para esses conjuntos.

Por outro lado, na LFM, o objetivo principal é obter um modelo fuzzy com boa interpretabilidade e este é desenvolvido principalmente através de FRBS linguísticos to tipo ML. Os FRBS linguísticos baseiam-se em regras linguísticas, nas quais o antecedente e o consequente das regras fuzzy fazem uso de variáveis linguísticas, que são compostas de termos e conjuntos fuzzy associados que definem o seu significado. Neste trabalho, o foco será mantido na LFM (ALCALÁ et al., 2006; ALONSO; CASTIELLO; MENCAR, 2015).

Para Alonso, Castiello e Mencar (2015), uma característica distinta da LFM, é a representação do conhecimento em dois níveis. No "baixo nível", a representação consiste na definição formal dos conjuntos fuzzy, em termos de suas funções de pertinência, bem como funções de agregação usadas para o processo de inferência. Este nível de representação define a semântica do modelo fuzzy baseado em regras ao mesmo tempo que determina o comportamento do modelo, isto é, o mapeamento entre a entrada e a saída para a qual ele é responsável. Já no "alto nível", o conhecimento é representado sob a forma de regras. Elas definem uma estrutura formal, onde estão envolvidas variáveis linguísticas, que são reciprocamente conectadas por alguns operadores formais, como *AND* e

OR, por exemplo. As variáveis linguísticas correspondem às entradas e saídas do modelo. Os valores simbólicos que assumem estão relacionados aos termos linguísticos, que por sua vez são mapeados para os conjuntos fuzzy definidos no nível baixo. Os operadores formais também são mapeados para as funções de agregação. Este mapeamento fornece uma transição de interpretação que é bastante comum no contexto da matemática: a semântica é atribuída a uma estrutura formal mapeando os símbolos (termos linguísticos e operadores) para objetos (conjuntos fuzzy e funções de agregação) (MAMDANI; ASSILIAN, 1975; ZADEH, 1975).

2.2 REDES NEUROFUZZY

O termo Neurofuzzy se refere à combinação de Redes Neurais Artificiais com Sistemas de Inferência Fuzzy (KOSKO, 1992). Nesses sistemas são incorporadas técnicas de aprendizado automático de redes neurais artificiais com a capacidade de representação próxima do raciocínio humano oferecido por sistemas Fuzzy. Assim, há uma fase de treinamento, no qual são apresentados os padrões de treino com entrada/saída e como resultado obtem-se um modelo linguístico que representa o conhecimento extraído dos dados na forma de um sistema de inferência fuzzy (JANG; SUN, 1995; KAR; DAS; GHOSH, 2014).

De acordo com Jang e Sun (1995), devido a sua natureza híbrida, as redes neurofuzzy herdam as características de redes neurais e sistemas de inferência fuzzy, ou seja, além do aprendizado automático a partir de dados que fornece conhecimento implícito interpretável por seres humanos, novas regras podem ser inseridas no sistema de inferência a partir do conhecimento explícito fornecido por um especialista. As regras geradas a partir dos dados também podem ser otimizadas utilizando o conhecimento do especialista. Portanto, a principal motivação para o uso de redes neurofuzzy é a facilidade de sua manipulação e entendimento do conhecimento implícito da rede, ou seja, sua interpretabilidade.

2.2.1 ANFIS

A rede *Adaptative Neural Fuzzy Inference System* (ANFIS), foi proposta por Jang (1993). Se trata de uma rede neural generalizada, composta por uma rede *feed-foward* com um procedimento de aprendizado por descida de gradiente. Após o treinamento da rede, a mesma é utilizada para contruir um FIS do tipo Takagi-Sugeno (TS) (TAKAGI; SUGENO, 1985).

O modelo ANFIS é composto por cinco camadas de neurônios, como exibido na Figura 2.2. Cada uma dessas camadas possui diferentes

tipos de neurônios, que podem ser adaptativos ou fixos. Os neurônios adaptativos podem ser adicionados e modificados, enquanto os fixos podem apenas ter modificações nos pesos de suas conexões neurais.

A primeira camada é composta por neurônios adaptativos, cada um representa uma função de pertinência fuzzy para cada entrada. A segunda camada é composta por neurônios com a função "produto", cuja saída é o produto de todos os sinais de entrada. A terceira camada é uma camada de normalização. Sua saída é a soma de todos os sinais de entrada, ponderada por $\overline{\omega}_i$ e em seguida é normalizada. A quarta camada representa a parte consequente das regras fuzzy, onde cada neurônio é do tipo adaptativo e sua saída é o resultado de uma função que leva como parâmetro o sinal de entrada da terceira camada de rede e as entradas de treino da própria rede. Para finalizar, a quinta camada faz o cálculo da soma de todos os sinais de entrada ponderados por $\overline{\omega}_i * f_i$.

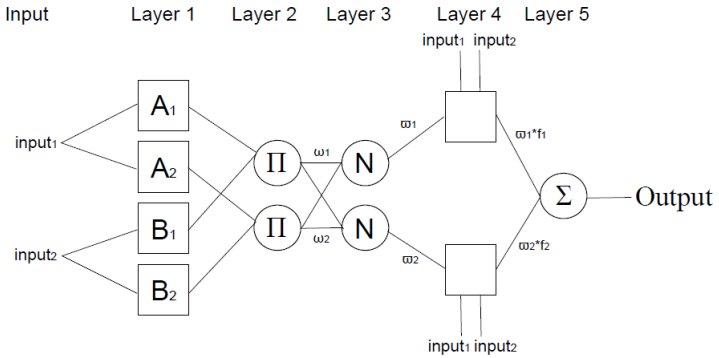

Figura 2.2 - Arquitetura da Rede ANFIS. Adaptada de Jang (1993).

Apesar de ser uma proposta de rede neural antiga, pesquisas recentemente elaboradas por Ocak e Ertunc (2013), Chang e Wang (2013) e Karaboga e Kaya (2018) atribuiram bons resultados à rede ANFIS. A grande desvantagem da ANFIS é o fato de que ela utiliza um sistema de inferência fuzzy do tipo TS. TS possui uma interpretabilidade mais baixa quando comparado com modelos do tipo ML (JASSBI et al., 2006, 2007).

2.3 REDES NEUROFUZZY INCREMENTAIS

Redes Neurofuzzy Incrementais podem ser definidas como sistemas auto-organizáveis, que podem adaptar seus parâmetros e também sua estrutura *online* incrementalmente durante o seu uso. De acordo com Pedrycz (1991), essas redes neurais combinam elementos das teorias de

conjuntos fuzzy e de neurais, gerando modelos que integram o tratamento da incerteza e a interpretabilidade provida por sistemas fuzzy e a habilidade de aprendizado proporcionada por redes neurais. Modelos fuzzy incrementais híbridos foram propostos em Jang (1993), Kim e Kasabov (1999), Kasabov (2001) e Kasabov e Song (2002) e porteriormente, foram definidos como redes neurofuzzy incrementais por Pedrycz e Gomide (2007). Em Pratama, Pedrycz e Webb (2018) foi proposta a construção incremental de um sistema neurofuzzy para aprendizado contínuo de fluxos de dados não-estacionários. Esses modelos possuem estrutura multicamada, de onde é possível extrair regras fuzzy a partir de sua estrutura. Algoritmos de treinamento permitem adaptar os pesos da rede, assim como o número de neurônios em determinadas camadas. A seguir são apresentados alguns exemplos.

2.3.1 ECoS

Sistemas Conexionistas Evolutivos (ECoS) (do inglês *Evolving Connectionist Systems*), é uma família de redes neurais artificiais, que foi proposta por Kasabov (1998). Evolutivo no contexto do ECoS não está relacionado a evolutivo no contexto de algoritmos genéticos. No caso do ECoS, evolutivo se refere ao fato de esses sistemas mudarem ao longo do tempo, ou seja, a estrutura de rede do ECoS é dinâmica. A Figura 2.3 mostra a arquitetura básica de um sistema ECoS.

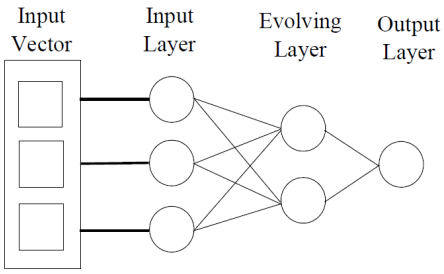

Figura 2.3 - Arquitetura básica da família ECoS. Adaptada de Kasabov (1998).

A maior característica é o algoritmo de aprendizagem que modifica a estrutura da rede enquanto os padrões de treinamento são apresentados à rede. Kasabov (1998) apresenta os 3 princípios básicos que todos os sistemas dessa família seguem:

- O algoritmo de aprendizado é em batelada (*batch mode*). Isto é, na primeira passagem do exemplo de treinamento para a rede, ela já deve aprender o dado.

- O aprendizado é incremental, sendo que arquitetura geral de um sistema ECoS e do algoritmo de aprendizado possibilitam que uma rede ECoS acomode novos dados sem perder a capacidade adquirida com os dados apresentados em treinamentos anteriores.

- Os neurônios são adicionados quando dados são apresentados. Inicialmente alguns dados são armazenados, porém, quando novos dados são apresentados para a rede, os neurônios existentes podem ser modificados ou novos neurônios podem ser adicionados. Caso um neurônio adicionado não seja modificado é possível extrair qual dado foi responsável pela sua origem.

2.3.1.1 DENFIS

O *Dynamic Evolving Neural-Fuzzy Inference System* (DENFIS) é uma aplicação especial da família ECoS, que foi proposta por Kasabov e Song (2002). Existem dois tipos de DENFIS, um com algoritmo de aprendizado *Online* e o outro utiliza um algoritmo de aprendizado *Offline*. Ambos os tipos utilizam o sistema de inferência fuzzy do tipo TS.

O mecanismo básico utilizado pelo DENFIS para gerar regras é um algoritmo de clusterização chamado *Evolving Clustering Method* (ECM). Os *clusters* são utilizados para particionar o espaço de entrada dos dados, depois uma rede neural artificial do tipo MLP, com o algoritmo *backpropagation*, é utilizada para achar os valores ótimos para a parte consequente das regras fuzzy. A parte antecedente das regras fuzzy é determinada pelos centros dos *clusters*.

2.3.1.2 EFuNN

Evolving Fuzzy Neural Networks (EFuNN) é outra aplicação especial de ECoS desenvolvida por Kasabov (1998). É basicamente uma rede neural *feed-forward* de 5 camadas. A primeira camada de neurônios é a camada de entrada. A segunda camada é a camada de condições. Cada neurônio na camada de condições representa uma função de pertinência fuzzy para uma entrada. A terceira camada de neurônios é a camada evolutiva, onde regras fuzzy são adicionadas e armazenadas. A quarta camada é a camada de ação e os neurônios dessa camada representam as funções de pertinência fuzzy ligadas aos neurônios de saída. A quinta camada é a camada de saída, onde o valor *crisp* é calculado a partir da saída fuzzy da quarta camada utilizando um método de defuzzificação.

Regras fuzzy do tipo ML podem ser extraídas de uma EFuNN já treinada utilizando um algoritmo específico, descrito em Kasabov e Song (2002). Bons resultados mostrando rápida performance e baixo erro já

foram atribuídas à EFuNN na literatura em Chang, Fan e Lin (2011) e Almomani et al. (2012). Entretanto, Petrovic-Lazarevic e Zhang (2009) relata o fato de que algumas variáveis, como número e tipo de funções de pertinência para cada variável de saída, são calculadas automaticamente e não podem ser modificadas e os parâmetros ótimos para o aprendizado são difíceis de serem configurados.

2.3.2 Outros Modelos de Redes Neurofuzzy Incrementais

O HyFIS (KIM; KASABOV, 1999) é um NFS conexicionista que gera regras fuzzy do tipo ML, o que pode garantir um certo nível de interpretabilidade. No Entanto, conforme foi apontando por Kothamasu e Huang (2007), HyFIS usa uma estratégia de defuzzificação que restringe as funções de pertinência de saída a assumirem uma forma funcional Gaussiana. Embora isso não prejudique sua capacidade de gerar boas soluções, não é possível que um especialista de domínio interaja com o modelo em todas as situações (por exemplo, quando as funções de pertinência nos consequentes não sejam Gaussianas).

Uma abordagem para identificação de modelos incrementais foi proposta em Angelov e Filev (2004). Esta baseia-se em um algoritmo incremental de agrupamento subtrativo, que atualiza recursivamente a estrutura de uma base de regras do tipo TS e seus parâmetros através de uma combinação de aprendizado supervisionado e não-supervisionado. A base de regras evolui continuamente adicionando novas regras com maior poder de sumarização e modificando as regras e parâmetros existentes. A criação de novas regras ou a modificação das regras existentes é realizada pela avaliação recursiva do potencial das novas amostras. Os parâmetros do consequente das regras são atualizados com o algoritmo recursivo de mínimos quadrados. Esse modelo evolutivo é chamado de eTS (evolving Takagi-Sugeno).

SeroFAM (TAN; QUEK, 2010) é um NFS com cinco camadas em linha que inicia com zero neurônios, adota uma abordagem de aprendizagem em batelada e usa um algoritmo de agrupamento fuzzy auto-organizado para definir *clusters* de associação. O SeroFAM usa o método do tipo ML generalizado para geração de regras fuzzy. Este modelo representa regras criando um mapeamento entre uma região fuzzy de entrada e uma região fuzzy de saída, o que proporciona um bom grau de interpretabilidade. No SeroFAM, a força sináptica é ajustada durante o processo BCM (BIENENSTOCK; COOPER; MUNRO, 1981) de acordo com a composição da saída real em dado instante de tempo. Embora o SeroFAM utilize regras fuzzy do tipo ML, o número de regras obtido é grande, o que de certa forma prejudica a interpretabilidade dos modelos

gerados, conforme foi relatado por Jacob et al. (2012).

O eFSM (TUNG; QUEK, 2010) e o SaFIN (TUNG; QUEK; GUAN, 2011) são modelos auto-organizadores que asseguram a aprendizagem semi-incremental e incremental, respectivamente. Ambos modelos são do tipo ML. O dilema Estabilidade-Plasticidade é considerado nesses modelos, de forma que o conhecimento prévio e as novas informações são integrados entre si. A adaptação é feita nas partes conseqüentes e antecedentes das regras de forma independente. Portanto, a estrutura de aprendizagem inclui a poda das regras inconsistentes ou idênticas e exclusão de regras órfãs. O eFSM usa o critério de cobertura uniforme (isto é, um limiar) durante a aprendizagem estrutural. As partições racionais dependem muito da distribuição dos dados de saída. Assim, de acordo com Ahmed e Isa (2015), o limiar provoca partições incertas neste modelo. No SaFIN, a abordagem de agrupamento aborda de forma eficaz o dilema Estabilidade-Plasticidade. SaFIN integra novos e antigos conhecimentos, de modo que um conjunto distinto é formado simplesmente pela média das variações dos vizinhos à esquerda e à direita. Portanto, SaFIN também usa o critério de cobertura uniforme de cada *cluster* porque o centro de um agrupamento permanece constante. Assim, a adaptação usando uma função de incerteza dinâmica é necessária para alcançar uma base de regras compacta, consistente e eficaz (TUNG; QUEK; GUAN, 2011).

2.4 MECANISMOS DE RACIOCÍNIO NA OCORRÊNCIA DE DADOS FALTANTES

Hoje em dia, dados são gerados em quase todos os lugares: redes de sensores, saúde, finanças, pesquisas de opinião sobre qualquer tópico, entre muitos outros. Muitos desses dados podem estar faltando por diversos motivos. Em uma aplicação industrial, alguns dados podem estar faltando devido a falhas mecânicas/eletrônicas durante o processo de aquisição de dados. Em um diagnóstico médico, alguns testes não podem ser feitos porque o hospital não possui o equipamento médico necessário, ou alguns exames médicos podem não ser apropriados para certos pacientes (SCHAFER, 1997; BUUREN, 2018). Neste trabalho, o termo dados faltantes define tuplas de dados incompletos.

A questão de dados faltantes já foi tratada extensivamente na literatura de análise estatística. Trabalhos como os de Schafer (1997), Allison (2001), Little e Rubin (2014), Galán et al. (2017) apresentam alguns mecanismos para lidar com esse problema. Para estes autores, a capacidade de manipular os dados faltantes tornou-se um requisito fundamental para problemas de classificação, de regressão e de previsão de séries

temporais, pois o tratamento inadequado dos dados faltantes pode causar grandes erros e/ou falsos resultados.

Para Little e Rubin (2014), a maneira mais apropriada de lidar com dados faltantes depende, na maioria das vezes, de como eles se tornaram faltantes. O mecanismo de dados faltantes caracteriza-se pela distribuição condicional de $\mathcal{M}$ dado $\mathcal{X}$,

$$p(\mathcal{M}|\mathcal{X}, \widetilde{\mathcal{X}}, \xi) \tag{2.2}$$

onde, $\mathcal{X}$ é o conjunto de entrada observado (padrões completos), $\widetilde{\mathcal{X}}$ é o conjunto de entradas desconhecidas (padrões incompletos) e ξ indica os parâmetros desconhecidos que definem o mecanismo de dados faltantes. Os autores definem ainda vários tipos únicos de mecanismos de dados faltantes (BUUREN, 2018):

- ***Missing completely at random*** **(MCAR)**: uma situação MCAR ocorre quando a probabilidade de falta de uma variável é independente da própria variável e de quaisquer outras influências externas;

- ***Missing at random*** **(MAR)**: a falta é independente das variáveis, mas o padrão de falta de dados é rastreável ou previsível a partir de outras variáveis;

- ***Not missing at random*** **(NMAR)**: o padrão de falta de dados não é aleatório e depende da variável que apresenta dados faltantes.

De acordo com Schafer (SCHAFER, 1997), quando os dados são MCAR ou MAR, o mecanismo de dados em falta é denominado ignorável. Mecanismos ignoráveis são importantes, porque quando eles ocorrem, um pesquisador pode ignorar os motivos da falta de dados na análise dos mesmos e, assim, simplificar os métodos utilizados para analisar e tratar os dados faltantes. Por esta razão, a maioria das pesquisas (SCHAFER, 1997; BATISTA; MONARD et al., 2002; LI et al., 2004; LUENGO; SÁEZ; HERRERA, 2012; FESSANT; MIDENET, 2002; LIU et al., 2016, 2015; ANKAIAH; RAVI, 2011) abordam os casos em que os dados faltantes são do tipo MAR ou MCAR. Neste sentido, todos os experimentos realizados neste trabalho utilizam dados cujo mecanismo de falta é MCAR ou MAR.

Na literatura é possível encontrar várias pesquisas que abordam o problema de dados faltantes, cujo mecanismo de dados em falta é ignorável. Para problemas de classificação, é possível encontrar descrições de métodos de aprendizagem de máquinas utilizados no processo de imputação, como por exemplo o *K-nearest Neighbors Imputation* (KNNI)

(BATISTA; MONARD et al., 2002), o *Fuzzy c-means (FCM) Imputation* (FCMI) (LI et al., 2004; LUENGO; SÁEZ; HERRERA, 2012), *Self-Organizing Map Imputation* (SOMI) (FESSANT; MIDENET, 2002), e o *Credal Classification with Adaptive Imputation* (CCAI) (LIU et al., 2016) que é uma extensão do método PCC apresentado em Liu et al. (2015).

Do mesmo modo, para problemas de regressão podem ser listados alguns métodos como o *General Regression Auto Associative Neural Network* (GRAANN), *Radial Basis Function Auto Associative Neural Network* (RBFAANN), *Particle Swarm Optimization Trained Auto Associative Wavelet Neural Network* (PSOAAWNN) e *Particle Swarm Optimization Trained Auto Associative Neural Network* (PSOAANN) propostos recentemente por (RAVI; KRISHNA, 2014) e o K-means + MLP proposto por Ankaiah e Ravi (2011).

Além disso, existem também métodos mais tradicionais como o *Mean Imputation* (que substitui os valores faltantes pelo valor médio das outras observações), *Mean Average* (MA) *Imputation* (que substitui o valor em falta pela média móvel ponderada, usando um tamanho de janela semi-adaptativo para garantir que todos os dados faltantes sejam substituídos) e *Last Observation Carried Forward* (LOCF) *Imputation* (que substitui cada valor faltante com o valor atual mais recente antes dele), que são empregados para imputação em séries temporais e para o quais, podem ser encontradas implementações em diversas linguagens, como é o caso da biblioteca *imputeTS*[1] da linguagem de programação R (MORITZ; BARTZ-BEIELSTEIN, 2015).

[1] https://cran.r-project.org/web/packages/imputeTS/index.html

3 FUNDAMENTAÇÃO TEÓRICA

Este Capítulo abrange os conceitos básicos necessários para descrever a presente proposta de pesquisa. Na Seção 3.1 é apresentado o GMM, juntamente com conceitos relacionados. Na sequência, a Seção 3.2 descreve brevemente o algoritmo *Expectation Maximization* (EM) e na Seção 3.3, são apresentadas as condições para a equivalência entre um GMM e um modelo fuzzy do tipo ML. Para finalizar, a Seção 3.4 descreve o modelo incremental IGMM.

3.1 MODELO DE MISTURA DE GAUSSIANAS - GMM

Uma Gaussiana (Figura 3.1) é uma função simétrica característica com formato do sino assintótica para $(-\infty; +\infty)$, que possui uma média μ_j e um desvio padrão $\sigma_j = \sqrt{C_j}$, onde $f(x)$ é definida na Equação 3.2 (WEISSTEIN, 2002).

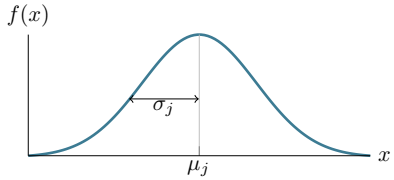

Figura 3.1 - Função Gaussiana 2-Dimensional

Um modelo de mistura (do inglês *Mixture Model*) é um modelo probabilístico que assume que os dados envolvidos pertencem a uma mistura de distribuições. A Figura 3.2 apresenta um modelo de mistura com três componentes.

De acordo com Reynolds (2015), um modelo de mistura de Gaussianas (GMM, do inglês *Gaussian Mixture Model*) é uma Função de Densidade Probabilidade (PDF, do inglês *Probability Density Function*) paramétrica, sendo representada como a soma ponderada da função densidade probabilidade das suas J Gaussianas componentes, podendo ser representada pela equação:

$$G(x; p, \mu, C) = \sum_{j=1}^{J} p(j)\mathcal{N}^D(x; \mu_j, C_j), \tag{3.1}$$

onde $p(1), p(2), ..., p(j), ..., p(J)$ são as proporções de mistura (ou probabilidades à *priori*) de cada componente Gaussiana, $C_1, C_2, ..., C_j, ..., C_J$

Figura 3.2 - Modelo de Mistura

são as matrizes de covariâncias de dimensão D x D, $\mu_1, \mu_2, ..., \mu_j, ..., \mu_J$ são os vetores de médias de dimensão D e $\mathcal{N}^D(x; \mu_j, C_j)$ é a função densidade multivariada normal da j-ésima componente

$$\mathcal{N}^D(x; \mu_j, C_j) = (2\pi)^{-D/2}|C_j|^{-1/2} \times exp\left\{-\frac{1}{2}\left(x-\mu_j\right)^{'} C_j^{-1}\left(x-\mu_j\right)\right\}. \tag{3.2}$$

Além disso, p deve satisfazer as seguintes restrições

$$0 \leqslant p(j) \leqslant 1, \forall j \tag{3.3}$$

e

$$\sum_{j=1}^{J} p(j) = 1. \tag{3.4}$$

As Figuras 3.3 e 3.4 são exemplos de modelo de mistura de Gaussianas hipotéticos para dados com dimensões 1 e 2D, respectivamente. Na Figura 3.3, a curva em vermelho representa a mistura das três Gaussianas componentes. Já na Figura 3.4, cada componente está marcada com um sinal $+$ no ponto representativo da média do componente de densidade. A linha que contorna todas as componentes representa o modelo de mistura resultante.

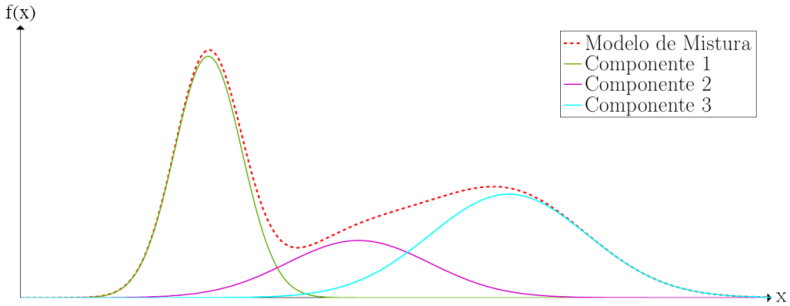

Figura 3.3 - Modelo de Mistura de Gaussianas para Dados 1D

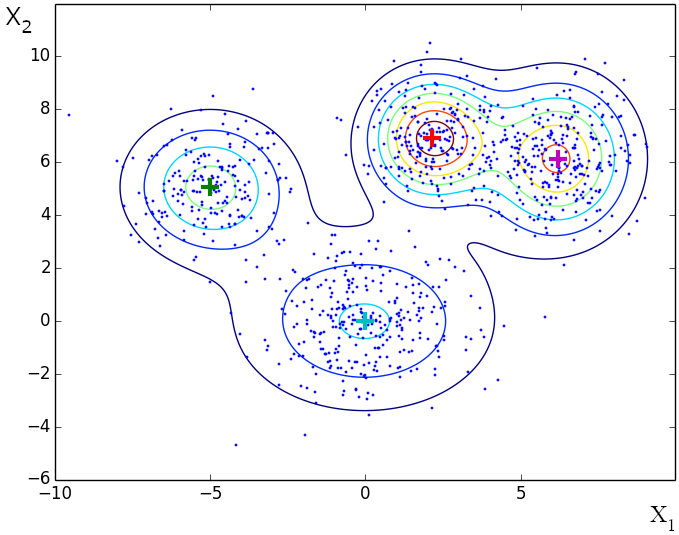

Figura 3.4 - Modelo de Mistura de Gaussianas para Dados 2D

3.2 O ALGORITMO *EXPECTATION-MAXIMIZATION* (EM)

O algoritmo EM é um procedimento iterativo para encontrar estimativas (locais) de Máxima Verossimilhança (*Maximum Likelihood*) para os parâmetros de um modelo estatístico, nos casos em que as equações não podem ser resolvidas diretamente. Normalmente, esses modelos envolvem variáveis latentes, além de parâmetros desconhecidos e observações de dados conhecidas (essas observações de dados podem conter lacunas) (DEMPSTER; LAIRD; RUBIN, 1977). É uma maneira iterativa de aproximar a função de Máxima Verossimilhança. Na estimativa da Máxima Verossimilhança ($\mathcal{L}(\theta, x)$), deseja-se encontrar os parâmetros ($\hat{\theta}$) do modelo para o qual os dados observados mais se aproximem, conforme Equação 3.5,

$$\hat{\theta} = \{\arg\max_{\theta \in \Theta} \mathcal{L}(\theta, x)\} \tag{3.5}$$

A cada iteração do algoritmo EM ocorrem duas etapas: A *E-step* e a *M-step*. Na *E-step*, é gerada uma estimativa para os dados latentes, considerando os dados observados e a estimativa corrente dos parâmetros do modelo. Na *M-step*, a função de verossimilhança é maximizada, assumindo-se que os dados ausentes são conhecidos. Os dados estimados durante a *E-step* são usados no lugar dos dados latentes. A convergência é garantida, uma vez que o algoritmo garante o aumento da verossimilhança a cada iteração (DEMPSTER; LAIRD; RUBIN, 1977; MCLACHLAN; KRISHNAN, 2007).

Resumindo, os passos básicos para o algoritmo EM são:

1. Uma estimativa inicial é feita para os parâmetros do modelo e uma distribuição de probabilidade é criada.

2. Na *E-step*, é gerada uma estimativa para os dados latentes, considerando os dados observados e a estimativa corrente dos parâmetros do modelo.

3. A distribuição de probabilidade é ajustada usando os dados estimados durante a *E-step* no lugar dos dados latentes (*M-step*).

4. Os passos 2 a 4 são repetidos até que a estabilidade (isto é, uma distribuição que não muda do *E-Step* para o *M-step*) seja alcançada.

Na Figura 3.5 observa-se um exemplo de execução do EM para um conjunto de dados 1D. De cima para baixo observa-se a convergência do algoritmo a medida que as etapas *E-step* e *M-step* vão sendo executadas a cada iteração. A linha azul pontilhada representa a probabilidade

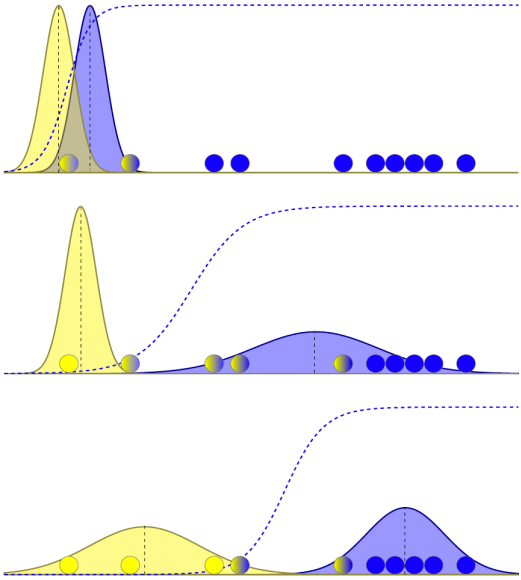

Figura 3.5 - Exemplo 1D do Algoritmo *Expectation Maximization*

à *posteriori* dos pontos de dados pertenceram a Gaussiana azul. Neste caso, quanto menor a probabilidade à *posteriori* do ponto de dado em questão, mais provável que ele pertença a Gaussiana amarela e, por outro lado, quanto maior a probabilidade à *posteriori* do ponto de dado, mais provável que ele pertença a Gaussiana azul.

3.3 A EQUIVALÊNCIA ENTRE MODELOS GMM E FUZZY DO TIPO ML

Como pretende-se demonstrar, este trabalho utiliza a equivalência entre GMM e fuzzy do tipo ML, para propor uma nova rede neurofuzzy do tipo ML com capacidade de aprendizagem incremental. Desse modo, descrevemos nesta seção a base teórica para o modelo proposto.

Adicinalmente ao que foi apresentado na Seção 2.1, modelos fuzzy aditivos encontrados na literatura podem ser geralmente dividido em 3 tipos gerais, ou seja, o Mamdani-Larsen (ML), o modelo de Takagi-Sugeno (TS) e, mais recentemente, o Modelo Fuzzy Generalizado (GFM, do inglês *Generalized Fuzzy Model*) proposto em Gan, Hanmandlu e Tan (2005).

Em Gan, Hanmandlu e Tan (2005) foi demonstrado como um

GMM pode ser traduzido para um sistema fuzzy aditivo. Os autores comprovaram a equivalência matemática entre a média condicional de um GMM e a saída defuzzificada de um GFM. Eles também estenderam seu trabalho aos modelos fuzzy do tipo ML e TS, que, conforme apresentado pelos autores, são casos especiais do GFM (MAMDANI, 1977; LARSEN, 1980; EVERITT, 1981; TAKAGI; SUGENO, 1985).

3.3.1 GFM versus ML

Considerando que cada regra fuzzy é baseada no vetor $x = [x_1, x_2, \ldots, x_D]$, e mapeia subconjuntos difusos no espaço de entrada $\mathcal{A}^k \subset \mathbb{R}^D$ para um subconjunto fuzzy no espaço de saída $\mathcal{B}^k \subset \mathbb{R}^D$ e, considerando que $\mathcal{B}^k(b_k, v_k)$ é um conjunto fuzzy com peso v_k e centróide b_k, então apesar de terem a mesma forma na parte *IF* (conforme pode ser visualizado nas Equações 3.6 e 3.7), os modelos fuzzy ML e TS são distinguidos em como a parte *THEN* de suas regras difusas é definido.

Do mesmo modo, o modelo GFM proposto por Azeem, Hanmandlu e Ahmad (2000), como uma combinação de modelos ML e TS, distingue-se do ML e TS pela parte *THEN* das regras fuzzy, sendo na realidade uma combinação destes dois modelos. Enquanto ML fuzzy tem um centróide estático em $\mathcal{B}^k(b_k, v_k)$, GFM tem um centróide variável $f^k(x)$ em $\mathcal{B}^k(f^k(x), v_k)$, como nas Equações 3.6, 3.7 e 3.8,

$$\begin{aligned} ML : \mathcal{R}^k : \; & IF\ x_1\ is\ \mathcal{A}_1^k \wedge\ x_2\ is\ \mathcal{A}_2^k \wedge \cdots \wedge \\ & x_D\ is\ \mathcal{A}_D^k\ THEN\ y\ is\ \mathcal{B}^k(b_k, v_k) \end{aligned} \tag{3.6}$$

$$\begin{aligned} TS : \mathcal{R}^k : \; & IF\ x_1\ is\ \mathcal{A}_1^k \wedge\ x_2\ is\ \mathcal{A}_2^k \wedge \cdots \wedge \\ & x_D\ is\ \mathcal{A}_D^k\ THEN\ y\ is\ f^k(x) \end{aligned} \tag{3.7}$$

$$\begin{aligned} GFM : \mathcal{R}^k : \; & IF\ x_1\ is\ \mathcal{A}_1^k \wedge\ x_2\ is\ \mathcal{A}_2^k \wedge \cdots \wedge \\ & x_D\ is\ \mathcal{A}_D^k\ THEN\ y\ is\ \mathcal{B}^k(f^k(x), v_k) \end{aligned} \tag{3.8}$$

onde, $\mathcal{A}^k$ são subconjuntos fuzzy no espaço de entrada $\mathbb{R}^D$, $\mathcal{B}^k$ é um subconjunto fuzzy no espaço de saída $\mathbb{R}$, $\wedge$ é um operador de conjunção fuzzy e v_k é o peso da regra fuzzy.

Sendo $\phi^k(y)$ a função de pertinência de $\mathcal{B}^k$, então v_k e b_k são computados pelas Equações 3.9 e 3.10, respectivamente, que são fórmulas para a área e o centróide de $\phi^k(y)$ (AZEEM; HANMANDLU; AHMAD, 2000; GAN; HANMANDLU; TAN, 2005).

$$v_k = \int_y \phi^k(y) dy \tag{3.9}$$

$$b_k = \frac{\int_y y\phi^k(y)dy}{\int_y \phi^k(y)dy} \tag{3.10}$$

Dada a k-ésima regra, seu grau de ativação pode ser obtida tomando-se a conjunção fuzzy ($\wedge$) dos graus de pertinência da entrada, isto é

$$\lambda^k = \lambda_1^k(x_1) \wedge \lambda_2^k(x_2) \wedge \cdots \wedge \lambda_D^k(x_D), \tag{3.11}$$

onde $\lambda_d^k(x_d)$ é a função de pertinência do conjunto $\mathcal{A}_d^k$ (AZEEM; HANMANDLU; AHMAD, 2000; GAN; HANMANDLU; TAN, 2005).

O operador $\implies$ é usado para conjuntos fuzzy do espaço de entrada $\mathcal{A}^k \subset \mathbb{R}^D$ para um conjunto fuzzy no espaço de saída $\mathcal{B}^k \subset \mathbb{R}$) e o conjunto fuzzy resultante $\mathcal{B}^{*k}$ define uma relação de pertinência na forma

$$\phi^{*k}(y) = \lambda^k(x^*) \implies \phi^k(y). \tag{3.12}$$

O resultado da inferência $\mathcal{B}^o$ é obtido agregando a contribuição de cada regra individual usando um operador de agregação como, por exemplo, uma s-norma $\vee$, isto é, determina-se $\mathcal{B}^o$ usando:

$$\mathcal{B}^o = \mathcal{B}^{*1} \vee \mathcal{B}^{*2} \vee \cdots \vee \mathcal{B}^{*K}, \tag{3.13}$$

onde $\mathcal{B}^o$ é o conjunto fuzzy espaço de saída. Depois disso, o valor de saída defuzzificado y^o pode ser obtido por um método defuzzificação usando a função de pertinência resultante $\phi^o(y)$ de $\mathcal{B}^o \subset \mathbb{R}$ no espaço de saída (AZEEM; HANMANDLU; AHMAD, 2000; GAN; HANMANDLU; TAN, 2005).

Como no trabalho de Gan, Hanmandlu e Tan (2005), este trabalho concentra-se apenas na classe específica de modelos fuzzy que empregam a conjunção e implicação multiplicativas e operadores de disjunção aditiva. Dito isto, as fórmulas apresentadas por Laviolette et al. (1995) para obter as saídas defuzzificadas correspondentes aos modelos ML e GFM são, respectivamente

$$y^o = \sum_{k=1}^{K} \frac{\lambda^k(x).v_k}{\sum_{k'=1}^{K} \lambda^{k'}(x).v_{k'}}.b_k \tag{3.14}$$

$$y^o = \sum_{k=1}^{K} \frac{\lambda^k(x).v_k}{\sum_{k'=1}^{K} \lambda^{k'}(x).v_{k'}} f^k(x). \tag{3.15}$$

3.3.2 De GMM para GFM

Como dito anteriormente, Gan, Hanmandlu e Tan (2005) demonstraram como um GMM pode ser traduzido para um GFM. Aqui, considerando que um sistema Entradas Múltiplas e Saída Única (MISO, do

inglês *Multi Input Single Output*) - que é mais geral do que um sistema Entrada Única e Saída Única (SISO, do inglês *Single Input Single Output*) e que um sistema Entradas Múltiplas e Saídas Múltiplas (MIMO, do inglês *Multi Input Multiple Output*) pode ser decomposto em vários sistemas MISO - transcrevem-se os aspectos mais importantes e condições sob as quais a equivalência GMM $\equiv$ ML se mantém para um sistema MISO. Uma prova completa pode ser encontrada no artigo de Gan, Hanmandlu e Tan (2005).

Sejam x e y os vetores de entrada e saída de um sistema MISO, respectivamente. A forma geral da mistura Gaussiana para o caso MISO pode ser reescrita como

$$G(x,y) = \sum_{j=1}^{J} p(j)\mathcal{N}^{D+1}\left(\begin{bmatrix} x \\ y \end{bmatrix}; \begin{bmatrix} \mu_{jx} \\ \mu_{jy} \end{bmatrix}, C_j\right) \tag{3.16}$$

onde $C_j = \begin{bmatrix} \{\sigma_{jid}\}_{D\times D} & \{\sigma_{jd(D+1)}\}_{D\times 1} \\ \{\sigma_{j(D+1)d}\}_{1\times D} & \sigma_{j(D+1)(D+1)} \end{bmatrix}$ para i, d = 1, 2, ..., D, μ_{jx} = $\begin{bmatrix}\mu_{j1} \cdots \mu_{jD}\end{bmatrix}^{'}$ e $\mu_{jy} = \mu_{j(D+1)}$. Além disso, C_j é uma matriz simétrica onde $\sigma_{jdi} = \sigma_{jid}$.

Gan, Hanmandlu e Tan (2005) derivaram a PDF marginal de x como

$$\begin{aligned} G(x) &= \int_{-\infty}^{+\infty} G(x,y)dy \\ &= \sum_{j=1}^{J} \int_{-\infty}^{+\infty} p(j)\mathcal{N}^{D+1}\left(\begin{bmatrix} x \\ y \end{bmatrix}; \begin{bmatrix} \mu_{jx} \\ \mu_{jy} \end{bmatrix}, C_j\right) dy \\ &= \sum_{j=1}^{J} p(j)\mathcal{N}^{D}(x; \mu_{jx}, \sigma_{jxx}), \end{aligned} \tag{3.17}$$

onde σ_{jxx} é um menor (*minor*) da matriz C_j, depois de eliminar a $(D+1)$-ésima linha e $(D+1)$-ésima coluna (veja o Apêndice B em Gan, Hanmandlu e Tan (2005)). Da mesma forma, a PDF condicional é dada por

$$G(y|x) = \frac{\sum_{j=1}^{J} p(j)\mathcal{N}^{D+1}\left(\begin{bmatrix} x \\ y \end{bmatrix}; \begin{bmatrix} \mu_{jx} \\ \mu_{jy} \end{bmatrix}, C_j\right)}{\sum_{j'=1}^{J} p(j')\mathcal{N}^{D}(x; \mu_{j'x}, \sigma_{j'xx})}. \tag{3.18}$$

Adicionalmente, Gan, Hanmandlu e Tan (2005) derivaram a equação

$$E(y|x) = \left(\mu_{jy} - \frac{\left[x - \mu_{jx}\right]' \sigma_j^{xy}}{\sigma_j^{yy}} \right) \times \frac{\sum_{j=1}^{J} p(j) \prod_{d=1}^{D} \mathcal{N}^D(x_d; \mu_{jd}, \sigma_{jdd})}{\sum_{j=1}^{J} p(j') \prod_{d=1}^{D} \mathcal{N}^D(x_d; \mu_{j'd}, \sigma_{j'dd})} \tag{3.19}$$

que pode ser usada para calcular a esperança matemática de y condicionada a x, onde σ_{jxx} é a variância de x (uma partição da matriz de covariância), C_j, enquanto σ_j^{xy} e σ_j^{yy} (variância de y) são partições de seu inverso.

Agora, de acordo com Gan, Hanmandlu e Tan (2005), as Equações 3.19 e 3.15 são equivalentes nas seguintes condições.

Condições I:

i) O número de regras, K, na base de regras do GFM é igual ao número de componentes, J, no GMM, isto é,

$$K = J \tag{3.20}$$

for $k = j = 1, 2, ..., J$.

ii) O peso de cada regra, v_k, é dado pela probabilidade de $p(j)$ à *priori*, correspondente no modelo de misturas, isto é,

$$v_k = p(j). \tag{3.21}$$

iii) A função de regressão na parte *THEN* das regras do GFM é linear, e é uma função de todas as variáveis de entrada, dada por

$$f^k(x) = \mu_{jy} - \frac{\left[x - \mu_{jx}\right]' \sigma_j^{xy}}{\sigma_j^{yy}}. \tag{3.22}$$

iv) A função de pertinência de cada uma das variáveis da parte *IF* é uma função Gaussiana, ou seja,

$$\lambda^k(x_d) = \mathcal{N}^1(x_d; \mu_{jd}, \sigma_{jdd}), \tag{3.23}$$

for $d = 1, 2, ..., D$.

v) O sistema fuzzy é aditivo, com conjunção e implicação multiplicativas.

Estas são as condições apresentadas por Gan, Hanmandlu e Tan (2005) sob as quais existe uma equivalência matemática entre as saídas de um GFM e um GMM. Isto significa que para qualquer sistema MISO cuja distribuição conjunta de probabilidade de entrada-saída é conhecida por ser uma mistura de Gaussianas, existe - sob as condições estabelecidas em *Condições I* - um sistema GFM que modela de forma equivalente a sua saída esperada.

3.3.3 De GFM para ML

Azeem, Hanmandlu e Ahmad (2000) demonstraram que um GFM pode ser reduzido a um modelo fuzzy do tipo ML. Posteriormente, Gan, Hanmandlu e Tan (2005) derivaram as PDF que podem ser usadas para acomodar este caso especial, mostrando que um GMM também pode ser reduzido a um modelo ML.

A fórmula de defuzzificação do modelo ML representada na Equação 3.14 é igual à fórmula do GFM quando a função de regressão na Equação 3.15 é reduzida a uma constante, isto é, $f^k = b_k$. Pode-se obter uma constante eliminando-se todas as variáveis (x). Isso pode ser feito configurando-se σ_j^{xy} para ser um vetor nulo na Equação 3.22. Desse modo, σ_j^{xy} torna-se zero quando σ_{jxy} em C_j torna-se zero (veja a prova em Gan, Hanmandlu e Tan (2005)). Então, $f^k = \mu_{jy}$ e a equação

$$f^k(x) = b_k = \mu_{jy} \tag{3.24}$$

é utilizada para substituir a condição iii) em *Condições I* no caso especial de um modelo ML. Todas as demais condições em *Condições I*, permanecem inalteradas. Todas as derivações de equações e uma prova completa podem ser encontradas em Gan, Hanmandlu e Tan (2005).

3.4 INCREMENTAL GAUSSIAN MIXTURE MODEL - IGMM

Sob o ponto de vista da aprendizagem de máquina, um GMM é uma ferramenta de modelagem estatística que pode ser usada em problemas supervisionados e não supervisionados de classificação e regressão. Em problemas de classificação não supervisionados, as classes são aprendidas a partir da similaridade dos padrões de dados. Essa tarefa pode ser vista como um problema de categorização ou agrupamento que visa encontrar os agrupamentos naturais nos dados multidimensionais (um vetor de características D-dimensional) (ARANDJELOVIC; CIPOLLA, 2006; KRISTAN; SKOCAJ; LEONARDIS, 2008). A referência a este tipo de

modelo probabilístico é o algoritmo EM, que segue a abordagem de distribuição de mistura de Gaussianas para fazer uma modelagem probabilística (DEMPSTER; LAIRD; RUBIN, 1977; TAN et al., 2006). Como no EM, o modelo IGMM também adota uma distribuição de mistura de Gaussianas para modelagem. No entanto, tal como foi apresentado em Engel e Heinen (2010), o IGMM pode ser expandido dinamicamente para acomodar novos padrões de dados aprendidos a partir de um fluxo de dados, onde cada padrão de dados está instantaneamente disponível para o processo de aprendizagem (FISHER, 1987; GENNARI; LANGLEY; FISHER, 1989). Dessa forma, o modelo IGMM pode ser visto como uma versão incremental do algoritmo EM, com a diferença de que não há necessidade do conjunto de treinamento completo (*batch-mode*) e nem que o número de componentes seja estabelecido e fixado no início do processo de aprendizagem (TAN et al., 2006).

De acordo com Engel e Heinen (2010), o IGMM assume que a PDF dos dados de entrada $G(x)$ pode ser modelada por uma combinação linear de componentes de funções de densidade correspondentes a processos probabilísticos independentes, sob a forma de um modelo de mistura, conforme estabelecido na Equação 3.1.

As probabilidades à *priori*, $p(j)$, são ajustadas e as funções de densidade das componentes são normalizadas para satisfazer as restrições das Equações 3.3 e 3.4 e, adicionalmente:

$$\int_{-\infty}^{+\infty} p(x|j)dx = 1, \tag{3.25}$$

onde $p(x|j)$ é a probabilidade de que o vetor observado, $x = (x_1, ..., x_i, ..., x_D)$, pertença ao j-ésimo componente da mistura. Essa probabilidade é computada por uma Gaussiana multivariada, com média μ_j e matriz de covariância C_j, de acordo com a Equação 3.2.

Um ponto x no espaço de entrada, não é reconhecido como pertencente a um componente de mistura, j, se sua probabilidade $p(x|j)$ for inferior a uma fração do valor máximo da função de verossimilhança (*likelihood*), especificada através do parâmetro de limiar de verosimilhança (ou novidade) τ_{nov}, que é um parâmetro de configuração especificado pelo usuário (esta é uma característica importante para lidar com o dilema Estabilidade-Plasticidade). Quando x é rejeitado por todos as componentes de funções de densidade, conforme representado pela Equação 3.26, isso significa que ele contém novas informações e uma nova componente é adicionada ao modelo, ajustando-se adequadamente os seus parâmetros.

$$p(x|j) < \frac{\tau_{nov}}{(2\pi)^{D/2}\sqrt{|C_j|}}, \forall j. \tag{3.26}$$

A nova componente criada, i, começa centrada no dado de entrada atual (x), com uma matriz linha de covariância especificada por padrão, ou seja, $u_i = x$ e $C_i = \sigma_{ini}I$, onde I denota a matriz identidade e σ_{ini} é inicializada usando-se uma fração δ definida pelo usuário a partir da variância total (por exemplo, δ = 1/100) dos atributos envolvidos na modelagem em questão. Assim,

$$\sigma_{ini}^k = \delta\left[max(k) - min(k)\right], \forall k, \tag{3.27}$$

onde $[min(k), max(k)]$ define o domínio aproximado para o atributo ou característica k do vetor de entrada.

Conforme mencionado anteriormente, o modelo IGMM segue uma versão incremental do algoritmo EM. Na etapa *estimation* as probabilidades à *posteriori* são obtidas através da aplicação do teorema de Bayes, como segue:

$$p(j|x) = \frac{p(x|j)p(j)}{\sum_{j=1}^{J} p(x|j)p(j)}. \tag{3.28}$$

As probabilidades à *posteriori* podem então ser usadas para calcular novas estimativas para os valores do novo vetor de médias μ_j^*, da nova matriz de covariância C_j^* de cada componente de densidade, e as probabilidades à *priori* $p(j)^*$ na etapa *maximization*. Engel e Heinen (2010) e Heinen, Engel e Pinto (2011) derivaram as seguintes equações recursivas para atualizar as distribuições do modelo IGMM:

$$sp_j^* = sp_j + p(j|x), \forall j, \tag{3.29}$$

$$w_j = p(j|x)/sp_j^*, \forall j, \tag{3.30}$$

$$\mu_j^{k*} = \mu_j^k + w_j(x - \mu_j^k), \forall j, \forall k, \tag{3.31}$$

$$C_j^{k*} = C_j^k - (\mu_j^{k*} - \mu_j^k)(\mu_j^{k*} - \mu_j^k)' + w_j\left[(x - \mu_j^{k*})(x - \mu_j^{k*})' - C_j^k\right], \forall j, \forall k \tag{3.32}$$

$$p(j)^* = \frac{sp_j}{\sum_{q=1}^{J^*} sp_q}, \forall j, \tag{3.33}$$

onde μ_j^{k*}, C_j^{k*} e sp_j^* referem-se aos valores de μ_j^k, C_j^k e sp_j após a atualização.

Outra característica importante do IGMM é que componentes espúrios e desnecessários são removidos automaticamente. A cada componente Gaussiana é dado algum tempo t_{max} para demonstrar sua importância para o modelo na forma de um acúmulo de suas probabilidades à *posteriori* sp_j. Formalmente, um componente j é removido sempre que $t_j > t_{max}$ e $sp_j < sp_{min}$, onde t_{max} e sp_{min} são selecionados manualmente (por exemplo, 12,0 e 7,0, respectivamente) e t_j é o acumulador de tempo de vida do j-ésimo componente (PINTO; ENGEL; HEINEN, 2012).

Em resumo, o modelo IGMM possui capacidade de aprendizado incremental não supervisionado baseado-se em um *framework* bayesiano, que cria e ajusta continuamente um GMM consistente com todos os dados que foram apresentados sequencialmente, ou seja, após a apresentação de cada ponto de dados (ou padrão de treino) o IGMM ajusta os parâmetros de cada distribuição de acordo com a Equação 3.28 e as Equações recursivas 3.29, 3.30, 3.31, 3.32 e 3.33 que são equivalentes incrementais às equações usadas em modo *batch* pelo algoritmo EM. Para tanto, o usuário deve configurar manualmente os parâmetros $\boldsymbol{\tau_{nov}}$ (que controla quando uma nova componente de mistura deve ser criada), $\boldsymbol{t_{max}}$ e $\boldsymbol{sp_{min}}$ (que permite verificar a relevância de cada componente Gaussiana j existente no modelo) e $\boldsymbol{\delta}$ (que interfere no tamanho inicial (*spread*) dos novos componentes de mistura Gaussiana adicionados ao modelo).

4 PROPOSTA DO MODELO INCREMENTAL NEUROFUZZY GAUSSIAN MIXTURE NETWORK (INFGMN)

Considerando que o que foi apresentado na Seção 1.3 e o trabalho de Alonso, Castiello e Mencar (2015), embora a interpretabilidade seja uma qualidade de FRBS linguísticos, ela não é imediatamente quantificável e a adoção de um LFM não garante a interpretabilidade por si só. Técnicas para extrair conhecimentos a partir de dados existentes comumente utilizadas atualmente, muitas vezes produzem padrões ininteligíveis. Nesses casos, a grande vantagem de um LFM é perdida e, em termos de interpretabilidade, esses modelos são comparáveis aos modelos do tipo *black-box*, como é o caso de redes neurais, por exemplo. No entanto, é necessária uma definição quantitativa tanto para avaliar a interpretabilidade dos sistemas existentes quanto para desenvolver novos sistemas neurofuzzy.

Com isso em mente, nesta tese propõe-se um novo modelo de rede neurofuzzy com capacidade de aprendizado incremental que implementa regras fuzzy do tipo ML e que procura fornecer uma melhor relação custo-benefício entre acurácia e interpretabilidade.

O modelo proposto é uma rede neurofuzzy - com capacidade de aprendizagem incremental - de cinco camadas denominada INFGMN (*Incremental Neurofuzzy Gaussian Mixture Network*), que se utiliza da equivalência entre um modelo GMM e um modelo ML - estabelecido por (GAN; HANMANDLU; TAN, 2005) e descrito na Seção 3.3 - e a versão incremental de um GMM chamado IGMM (veja a Seção 3.4), para fornecer um NFS com capacidade de aprendizagem incremental, que seja o mais interpretável possível, sem perder a sua acurácia.

Cada uma dessas cinco camadas tem um papel bem específico dentro do modelo proposto. A primeira camada simplesmente recebe e transmite os padrões de treinamento diretamente para a segunda camada, sem alterá-los. Já em sua segunda camada, a rede neurofuzzy INFGMN mantém um IGMM, que após a apresentação de cada padrão de treinamento, ajusta os parâmetros de cada distribuição Gaussiana, permanecendo consistente e atualizado de acordo com o fluxo de dados, mesmo que o treinamento seja interrompido e reiniciado mais tarde com novos dados. Na sequência, na terceira camada são gerados (atualizados e/ou removidos) os antecedentes e os consequentes de cada regra fuzzy, utilizando-se o IGMM atualizado da segunda camada e as regras de equivalência (veja Secão 3.3) que serão melhor detalhadas a seguir. Por fim, na quarta camada as regras fuzzy do tipo ML são geradas e na quinta e última camada, as regras fuzzy são avaliadas e é então estimada a saída da rede. Cada uma das cinco camadas será descrita detalhadamente na

Seção 4.1 a seguir.

Além da possibilidade da criação de modelos neurofuzzy que usam regras do tipo ML de modo *online* e continuo, o uso do IGMM traz outra grande vantagem: a rede INFGMN pode aproveitar a característica intrínseca do IGMM para fazer imputação dos dados faltantes, cujo mecanismo de falta seja do tipo MCAR ou MAR, permitindo que sejam criados modelos neurofuzzy mais robustos e que permitem realizar inferências, mesmo com dados faltantes. Na seção a seguir os detalhes técnicos da arquitetura e do funcionamento da rede INFGMN, bem como o seu mecanismo de imputação de dados faltantes, serão apresentadas e discutidos detalhadamente.

4.1 ARQUITETURA DA REDE INFGMN

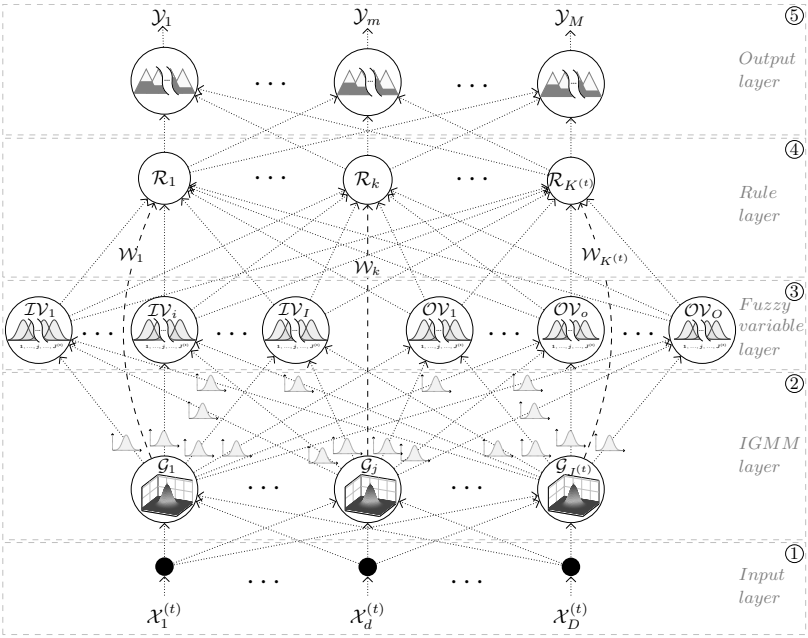

Figura 4.1 - Arquitetura da rede INFGMN

A INFGMN é uma rede neurofuzzy de cinco camadas, como mostrado na Figura 4.1. A camada ① é constituída pelos nós que representam as variáveis de entrada da rede. Na sequência, a camada ② é a camada IGMM. Na camada ③ estão representados os nós com variáveis fuzzy de entrada/saída (ou antecedentes e consequentes fuzzy). Já a camada ④

compreende os nós que representam as regras fuzzy e a camada ⑤ os nós que representam as variáveis de saída da rede. Na INFGMN, o vetor de entrada é denotado como $\mathcal{X} = (\mathcal{X}_1, ..., \mathcal{X}_d, ..., \mathcal{X}_D)$ e o vetor de saída correspondente é denotado por $\mathcal{Y} = (\mathcal{Y}_1, ..., \mathcal{Y}_m, ..., \mathcal{Y}_M)$. As notações utilizadas na Figura 4.1 são definidas da seguinte forma:

D é a dimensão do vetor de entrada;

M é a dimensão do vetor de saída;

$\mathcal{X}_d$ é a d-ésima componente do vetor de entrada;

$\mathcal{Y}_m$ é o valor da m-ésima dimensão na saída;

$\mathcal{G}_j$ é a j-ésima componente de mistura Gaussiana na camada IGMM (*IGMM layer*);

$\mathcal{IV}_i$ é a i-ésima variável linguística fuzzy de entrada na camada de variáveis fuzzy (*Fuzzy variable layer*);

$\mathcal{OV}_o$ é a o-ésima variável linguística fuzzy de saída na camada de variáveis fuzzy (*Fuzzy variable layer*);

$\mathcal{R}_k$ é a k-ésima regra fuzzy na *Rule layer*;

$\mathcal{W}_k$ é o k-ésimo peso de regra fuzzy; e

$\mathcal{X}^{(t)}$, $J^{(t)}$ e $K^{(t)}$ representam o estado dos respectivos vetores no tempo t.

Como será visto na Seção 4.2, a INFGMN possui dois modos de operação chamados *learning* e *recalling*. Para diferenciar as operações realizadas em cada modo de operação nas respectivas camadas da rede, f_L será usado para o modo de operação *learning* e f_R para o modo de operação *recalling*.

Camada ① (*Input Layer*): cada nó representa uma *variável de entrada* para a INFGMN. O vetor de dados, $\mathcal{X}^{(t)}$, é diretamente transmitido para a camada ② como descrito em

$$f^{①}(\mathcal{X}_d^{(t)}) = \mathcal{X}_d^{(t)}, \forall d \in \{1..D\} \tag{4.1}$$

onde $f^{①}(\mathcal{X}_d^{(t)})$ denota a saída do nó d para a entrada $\mathcal{X}_d$ no tempo t. Note que D não é necessariamente igual a $J^{(t)}$, ou seja, igual ao número de componentes Gaussianas no tempo t. Esta camada é ativada somente no modo de operação *learning*.

Camada ② (*IGMM layer*): esta camada cria e ajusta continuamente os parâmetros de um IGMM que permanece consistente com o fluxo de dados, isto é, após a apresentação de cada ponto de dados (padrão de treinamento), esta camada ajusta os parâmetros de cada distribuição Gaussiana, de acordo com a Equação 3.28 e as Equações recursivas 3.29, 3.30, 3.31, 3.32 e 3.33, que são equivalentes incrementais aproximados das equações de atualização em batelada utilizadas pelo algoritmo EM (veja Secção 3.4). O modo de operação *learning* nesta camada pode ser representado por

$$f^{②}(\mathcal{G}_j(\mathcal{X}^{(t-1)}; p, \mu, C), \mathcal{X}^{(t)}) = \\ \mathcal{G}_j(\mathcal{X}^{(t)}; p^*, \mu^*, C^*), \forall j \in \{1..J^{(t)}\} \tag{4.2}$$

onde, $\mathcal{G}_j(\mathcal{X}^{(t-1)}; p, \mu, C)$ é a j-ésima componente de mistura Gaussiana (representada pelas Gaussianas 3D na Figura 4.1) atualizada até o tempo $t-1$ do vetor de entrada $\mathcal{X}^{(t)}$. Já $\mathcal{G}_j(\mathcal{X}^{(t)}; p^*, \mu^*, C^*)$ é a componente de mistura Gaussiana atualizada após a apresentação do vetor $\mathcal{X}^{(t)}$, de acordo com as equações recursivas previamente mencionadas. Observe que estando no momento t não significa necessariamente que a rede INFGMN tenha visto apenas t padrões de treinamento, isto é, em cada etapa t, o vetor $\mathcal{X}$ pode conter um ou mais padrões de treinamento. Observe também que esta camada começa vazia e novas componentes de mistura Gaussiana j, com o centros $\mu = \mathcal{X}_d$ e covariâncias $C_j = \sigma_{ini} I$, são adicionadas sob demanda. Em $\sigma_{ini} I$, I denota a matriz identidade e σ_{ini} é inicializada usando-se uma fração δ definida pelo usuário a partir da variância total (por exemplo, δ = 1/100) aproximada dos atributos envolvidos na modelagem em questão, estimada a partir do intervalo de valores que estes atributos podem assumir, de acordo com a Equação 3.27. As médias (μ^*) e as covariâncias (C^*) atualizadas são transmitidas para a camada ③ e as probabilidades a *priori* (atualizadas) da mistura Gaussianas, p^*, são transmitidas diretamente para a camada ④ como o vetor de pesos $\mathcal{W}$. Isto é representado na Figura 4.1 pelas setas pontilhadas com Gaussianas 2D conectando a camada ② à camada ③ e pelas setas tracejadas com os pesos $\mathcal{W}_k$, para $k = 1, ..., K^{(t)}$, conectando a camada ② à camada ④ respectivamente. Esta camada não é ativada em modo de operação *recalling*.

Camada ③ (*Fuzzy variable layer*): após cada etapa de treinamento t, compreendendo um ou mais padrões de treino contidos em $\mathcal{X}^{(t)}$, o antecedente e o consequente de cada regra fuzzy (*IF-THEN*) do tipo ML podem ser gerados, atualizados ou removidos. Isso é feito na camada ③ da INFGMN. Como pode ser visto na Figura 4.1, esta camada é composta pelas variáveis linguísticas fuzzy de entrada (*input linguistic*

variables) $\mathcal{IV}_i$, para $i = 1, ..., I$, e pelas variáveis linguísticas fuzzy de saída (*output linguistic variables*) $\mathcal{OV}_o$, para $o = 1, ..., O$, onde cada variável linguística $\mathcal{IV}_i$ ou $\mathcal{OV}_o$ é formada por um conjunto de $\mathcal{IV}_{i_{(j)}}$ ou $\mathcal{OV}_{o_{(j)}}$ termos linguísticos fuzzy, para $j = 1, ...J^{(t)}$ (representado pelos conjuntos fuzzy com funções de pertinência Gaussianas dentro dos nós da camada). Cada termo linguístico fuzzy tem seu significado completamente representado por um conjunto fuzzy (com função de pertinência Gaussiana), com parâmetros $\mu_{i_{(j)}}$ e $C_{i_{(j)}}$ ou $\mu_{o_{(j)}}$ e $C_{o_{(j)}}$, respectivamente, que foram gerados na camada ②. Esta camada é ativada tanto para o modo de operação *learning* quanto para o modo *recalling* como mostrado em azul na Figura 4.2. Considerando o modo de operação *learning*, o conjunto de funções de pertinência fuzzy de entrada e saída são gerados/atualizados usando-se as operações definidas pelas Equações 4.3 e 4.4, onde $\mathcal{IV}_{i_{(j)}}(\mu_{i_{(j)}}, C_{i_{(j)}})$ e $\mathcal{OV}_{o_{(j)}}(\mu_{o_{(j)}}, C_{o_{(j)}})$ correspondem à $i_{(j)}$-ésima e $o_{(j)}$-ésima função de pertinência de entrada e de saída, respectivamente. Note que $D = I + O$, isto é, o "conhecimento" implícito em cada dimensão do espaço de entrada/saída é aprendido pela INFGMN através da geração de variáveis linguísticas fuzzy para representá-lo, em cada uma destas dimensões. Observe também que na transformação das componentes Gaussianas encontradas na camada ② (*IGMM Layer*) em variáveis linguísticas fuzzy, aplicam-se as *Condições* $iii)$ e $iv)$ descritas na Seção **3.3.2** e modificadas pela Equação 3.24.

$$f^{③}_{L_{(I)}}(\mu, C) = \Big\{ \Big\{ \mathcal{IV}_{i_{(j)}} \left(\mu_{i_{(j)}}, C_{i_{(j)}} \right) \mid j \in \{1..J^{(t)}\} \Big\} \mid i \in \{1..I\} \Big\} \tag{4.3}$$

$$f^{③}_{L_{(O)}}(\mu, C) = \Big\{ \Big\{ \mathcal{OV}_{o_{(j)}} \left(\mu_{o_{(j)}}, C_{o_{(j)}} \right) \mid j \in \{1..J^{(t)}\} \Big\} \mid o \in \{1..O\} \Big\} \tag{4.4}$$

Para o modo de operação *recalling*, esta camada tem a responsabilidade de fuzzificar (*fuzzify*) os dados do vetor de entrada $\mathcal{X}$ (em azul na Figura 4.2) e calcular a área das funções de pertinência fuzzy de saída, ou seja, o "significado" dos termos linguísticos fuzzy. Esta etapa está representada nas Equações 4.5 e 4.6, onde $\lambda^{(\mathcal{IV}_i)}$ é a função de pertinência fuzzy da i-ésima variável linguística de entrada, $\phi^{(\mathcal{OV}_o)}$ é a função de pertinência fuzzy da o-ésima variável linguística de saída, e $\mathcal{N}^1$ é uma distribuição normal 1-Dimensional, de acordo com a Equação 3.2 e a

Condição iv) da Seção **3.3.2**.

$$f^{③}_{R_{(I)}}(\mathcal{X},\mathcal{IV}) = \Bigg\{ \lambda^{(\mathcal{IV}_i)} : \Big\{ \mathcal{N}^1\left(\mathcal{X}_i, \mu_{i_{(j)}}, C_{i_{(j)}}\right) \mid j \in \{1..J^{(t)}\} \Big\} \mid i \in \{1..I\} \Bigg\} \tag{4.5}$$

$$f^{③}_{R_{(O)}}(\mathcal{OV}) = \Bigg\{ \phi^{(\mathcal{OV}_o)} : \Bigg\{ \int_y \mathcal{N}^1\left(y, \mu_{o_{(j)}}, C_{o_{(j)}}\right) dy \mid j \in \{1..J^{(t)}\} \Bigg\} \mid o \in \{1..O\} \Bigg\} \tag{4.6}$$

Camada ④ (*Rule layer*): uma vez que o antecedente e consequente foram gerados na camada ③, as regras fuzzy (*IF-THEN*) to tipo ML podem ser geradas, atualizadas ou removidas na camada ④ da INFGMN. Esta camada é ativada para ambos os modos de operação, *learning* e *recalling*. Durante o modo de operação *learning*, as regras fuzzy da INFGMN são geradas e/ou atualizadas de acordo com a Equação 4.7, onde o número $K^{(t)}$ de regras gerado é igual ao número $J^{(t)}$ de componentes de mistura encontrados na camada ② (*IGMM Layer*) no tempo t, de acordo com a *Condição* i) descrita na Seção **3.3.2**. O número de antecedentes da parte IF de cada regra é igual a I e o número de consequentes da parte $THEN$ de cada regra é igual a O. O peso de cada regra, $\mathcal{W}_k$, que corresponde ao grau de ativação, é dado pela probabilidade à *priori*, $p(j)$, identificado na camada ③, atendendo a *Condição* ii) apresentada na Seção **3.3.2**. Observe também que a restrição estabelecida pela Equação 3.4 é mantida, uma vez que $\sum_{k=1}^{K^{(t)}} \mathcal{W}_k = 1$.

Quando o modo de operação *recalling* é considerado para esta camada, a operação representada pela Equação 4.8 é usada para avaliar as regras fuzzy da INFGMN, da sequinte maneira: I) o conjunto de graus de ativação α_k, para $k = 1, ..., K^{(t)}$, correspondente à parte antecedente da implicação $f^{④}_{R}$ de cada regra, é calculado através da aplicação de uma operação de conjunção fuzzy - que pela *Condição* v) definida na Seção **3.3.2**, é uma conjunção multiplicativa - a cada antecedente (parte *IF*) da regra. Este grau de ativação é então multiplicada pelo peso, $\mathcal{W}_k$, da regra. Isso é feito para todas as regras fuzzy da INFGMN; II) uma vez que os graus de ativação foram computadas, então a implicação fuzzy - que pela *Condição* v) é também uma implicação multiplicativa - é calculada para a parte consequente da função $f^{④}_{R}$. Isto também é feito para todas as regras

fuzzy da INFGMN; III) tal como definido na *Condição* v), a parte fuzzy da rede INFGMN é aditiva, de modo que a agregação da parte *THEN* de todas as regras que foram ativadas é feita através de uma agregação fuzzy aditiva, resultando no conjunto fuzzy, $\mathcal{OV}^{*m}$, contendo todos os conjuntos fuzzy agregados e suas respectivas funções de agregação fuzzy, ϕ^{*m}, para todas as saídas $m = 1, ..., M$, geradas pela parte consequente da função $f_R^{④}$ definida na Equação 4.8.

$$f_L^{④}(\mathcal{IV}, \mathcal{OV}, \mathcal{W}) = \left\{ \mathcal{R}^k : IF \bigwedge_{i=1}^{I} \mathcal{X}_i \; is \; \mathcal{IV}_{i_{(k)}} \; THEN \left[\mathcal{OV}^{*m} : \bigvee_{o=1}^{O} \mathcal{Y}_m \; is \; \mathcal{OV}_{o_{(k)}} \right]_{m=1}^{M} (\mathcal{W}_k) \mid k \in \{1..K^{(t)}\}, \; K^{(t)} = J^{(t)} \; e \; M = O \right\} \tag{4.7}$$

$$f_R^{④}(\lambda^{(\mathcal{IV})}, \phi^{(\mathcal{OV}^*)}) = \left\{ \alpha_k : \mathcal{W}_k \cdot \prod_{i=1}^{I} \lambda^{(\mathcal{IV}_{i_{(k)}})} \mid k \in \{1..K^{(t)}\} \right\} \implies \left\{ \phi^{*m} : \sum_{k=1}^{K^{(t)}} \alpha_k \cdot \phi^{\left(\mathcal{OV}^{*m}_{(k)}\right)} \mid m \in \{1..M\}, \; K^{(t)} = J^{(t)} \; e \; M = O \right\} \tag{4.8}$$

Camada ⑤ (*Output layer*): esta camada é ativada somente durante o modo de operação *recalling* e é responsável por defuzzificar (*defuzzify*) a saída da rede INFGMN. Isto é feito por meio do método do centróide fuzzy (representado pelas regiões sombreadas nas funções de pertinência fuzzy dentro dos nós da camada ⑤ (*Output Layer*) na Figura 4.2), conforme descrito na Equação 4.9. O centróide fuzzy é calculado para cada um dos conjuntos fuzzy resultantes do processo de agregação, $\mathcal{OV}^*$, aplicando-se sua função de pertinência ϕ^* na região de saída obtida da camada ④.

$$f_R^{⑤}(\phi^*) = \left\{ \frac{\int_y y\phi^{*m}(y)dy}{\int_y \phi^{*m}(y)} dy \mid m \in \{1..M\} \right\} \tag{4.9}$$

4.2 OPERAÇÃO DA INFGMN

A INFGMN possui dois modos de operação, denominados *learning* e *recalling*, respectivamente.

4.2.1 Modo de Operação *learning*

No modo de operação *learning*, apenas as camadas de ① até ④ na Figura 4.1 são ativadas. No final de uma execução completa do modo

de operação *learning*, o resultado é um modelo FRBS linguístico atualizado, que procura fornecer a melhor relação custo-benefício entre acurácia e interpretabilidade, através da geração de uma base de regras do tipo ML (que possui a interpretabilidade como uma característica intrínseca), mas ao mesmo tempo gerando somente o número de regras necessárias para obtenção de uma boa acurácia. Este equílibrio, conforme será descrito na sequência, é governado pelo ajuste dos parâmetros τ_{nov}, t_{max} e sp_{min}. O número de premissas na parte *IF* de cada regra é igual ao número de variáveis de entrada I, o qual é governado pelo número de saídas O a ser considerado para o modelo atual, ou seja, o número de variáveis de entrada é $I = D - O$. Considere, por exemplo, o problema da Iris (*Iris Problem*) do conjunto de dados UCI (*UCI Dataset*[1]), que é um problema 7-dimensional, o número de saídas é $O = 3$, uma para cada classe Iris. Portanto, o número de variáveis de entrada é $I = 7 - 3$ e, consequentemente, cada parte *IF* das regras terá apenas 4 antecedentes (ou premissas).

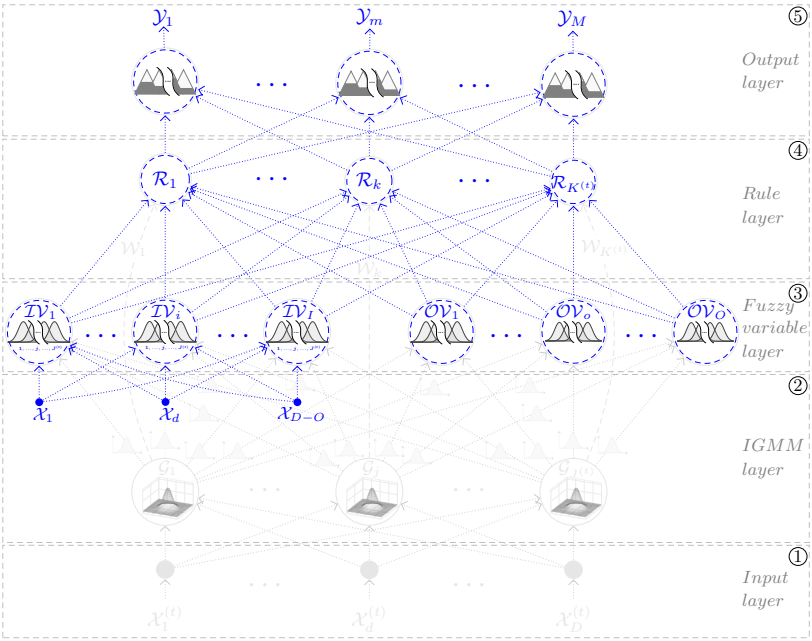

Figura 4.2 - Fluxo da informação na rede INFGMN em modo de operação *recalling*

O número total de regras geradas é governado pelo número de

[1] https://archive.ics.uci.edu/ml/about.html

componentes de mistura Gaussianas encontrados na camada ② (*IGMM Layer*) durante o modo de operação *learning*. Este número é uma consequência dos parâmetros de configuração *novidade mínima* (*minimum novelty*), τ_{nov}, e *utilidade/relevância mínima* (*minimum utility/relevance*), t_{max}/sp_{min}. A INFGMN tem 6 parâmetros de configuração, sendo τ_{nov} e t_{max}/sp_{min} os mais críticos. São eles:

> $\boldsymbol{\tau_{nov}}$ - este parâmetro escolhido pelo usuário, controla quando um novo $\mathcal{X}_d$ não é reconhecido como pertencente a qualquer componente de mistura j. Se sua probabilidade $p(\mathcal{X}_d|j)$ é menor que τ_{nov}, então ele não é considerado um membro de j. Quando $\mathcal{X}_d$ é rejeitado por todos os componentes de densidade (como representado pela Equação 3.26), isso significa que ele contém novas informações e um novo componente é adicionado ao modelo, ajustando-se adequadamente seus parâmetros. Esta é uma característica importante para lidar com o dilema Estabilidade-Plasticidade;
>
> $\boldsymbol{t_{max}}$ e $\boldsymbol{sp_{min}}$ - para cada componente Gaussiana j é dado algum tempo t_{max} para que mostre sua utilidade/relevância para o modelo em termos da acumulação de sua probabilidade à *posteriori* sp_j. Formalmente, um componente j é removido sempre que $t_j > t_{max}$ e $sp_j < sp_{min}$. Esta também é uma característica importante para lidar com o dilema Estabilidade-Plasticidade;
>
> $\boldsymbol{\delta}$ - valor definido pelo usuário (por exemplo, δ = 1/100) a ser usado como uma fração da variância total do domínio estimado das características envolvidas no problema (variáveis) e passadas no vetor de entrada. Este parâmetro interfere no tamanho inicial (*spread*) dos novos componentes de mistura Gaussiana adicionados ao modelo;
>
> $\boldsymbol{O_{num}}$ - o número de características (variáveis) no vetor de treino, que devem ser consideradas a parte de saída dos padrões de treinamento;
>
> $\boldsymbol{\sigma_{mul}}$ - este parâmetro determina o quão rápido os valores de pertinência fuzzy diminuem de 1 para 0, ou seja, pode ser usado para aumentar ou diminuir o tamanho (*spread*) das funções Gaussianas de pertinência fuzzy geradas pela INFGMN. Como consequência, este ajuste pode melhorar a interpretabilidade geral do modelo. A seleção do valor apropriado para σ_{mul} é um processo subjetivo que depende do domínio dos padrões de treinamento. Este parâmetro não é crítico e portanto o valor padrão pode ser mantido;
>
> $\boldsymbol{\sigma_{ini}}$ - é inicializado usando δ para obter uma fração da variância total do domínio estimado das características (variáveis) no vetor

de treino, de acordo com a Equação 3.27. $\sigma_{ini}I$ (onde I é a matriz de identidade) é usada para inicializar as matrizes de covariância dos novos componentes da mistura Gaussiana adicionados ao modelo.

O algoritmo apresentado na Listagem 1 sumariza os passos executados pela rede INFGMN durante sua execução em modo de operação *learning*. Este algoritmo também pode ser encontrado no Apêndice A, em código Matlab, a partir da linha 6.

Listagem 1 - Pseudocódigo Algoritmo do Modo de Operação *learning*

function INFGMNLEARNIG($\boldsymbol{X_D}$)
  /* Computa a verossimilhança para todos os neurônios da camada ②*/
  for all neuronio j **in** Camada ② **do**
    $p(\boldsymbol{X_D}|j) = (2\pi)^{-D/2}|C_j|^{-1/2} \times exp\left\{-\frac{1}{2}\left(\boldsymbol{X_D}-\mu_j\right)' C_j^{-1}\left(\boldsymbol{X_D}-\mu_j\right)\right\}$
  end for
  /* Cria um novo neurônio j na Camada ② se necessário */
  if $J < 1$ **or** $p(\boldsymbol{X_D}|j) < \frac{\tau_{nov}}{(2\pi)^{D/2}\sqrt{|C_j|}}, \forall j$ **then**
    $J^* = J + 1$; $t_j = 0$; $sp_j = 1.0$
    $\mu_j = \boldsymbol{X_D}$; $C_j = \sigma_{ini}I$
    $p(\boldsymbol{X_D}|j) = (2\pi)^{-D/2}|C_j|^{-1/2} \times exp\left\{-\frac{1}{2}\left(\boldsymbol{X_D}-\mu_j\right)' C_j^{-1}\left(\boldsymbol{X_D}-\mu_j\right)\right\}$
    $p(j) = sp_j / \sum_{q=1}^{J^*} sp_q, \forall j$
  end if
  /* Computa as probabilidades a posteriori na Camada ② */
  $p(j|\boldsymbol{X_D}) = \frac{p(\boldsymbol{X_D}|j)p(j)}{\sum_{q=1}^{J} p(\boldsymbol{X_D}|q)p(q)}, \forall j$
  /* Atualiza todos os neurônios da camada ②*/
  for all neuronio j **in** Camada ② **do**
    if $p(\boldsymbol{X_D}|j) < \frac{\tau_{nov}}{(2\pi)^{D/2}\sqrt{|C_j|}}$ **then**
      $sp_j^* = sp_j + p(j|\boldsymbol{X_D})$
      $w_j = p(j|\boldsymbol{X_D})/sp_j^*$
      $\mu_j^{k*} = \mu_j^k + w_j(\boldsymbol{X_D} - \mu_j^k), \forall k$
      $C_j^{k*} = C_j^k - (\mu_j^{k*} - \mu_j^k)(\mu_j^{k*} - \mu_j^k)' + w_j\left[(x-\mu_j^{k*})(x-\mu_j^{k*})' - C_j^k\right], \forall k$
      $p(j)^* = \frac{sp_j}{\sum_{q=1}^{J^*} sp_q}$
    end if
  end for
  /* Reinicia os acumuladores e deleta todos os neurônios que são ruídos */
  for all neuronio j **in** Camada ② **do**
    if $t_j > t_{max}$ **and** $sp_j < sp_{min}$ **then**
      delete neuronio j
    else
      $t_j = 0; sp_j = 1.0$
    end if
  end for
  /* Atualiza os $I = D - O_{num}$ neurônios (variáveis linguísticas fuzzy) da camada ③ */
  for all neuronio i **in** Camada ③ **do**
    $\mathcal{IV}_i = \left\{\mathcal{IV}_{i_{(j)}}\left(\mu_{i_{(j)}}, C_{i_{(j)}}\right) \mid j \in J^{(t)}\right\}$
  end for
  /* Atualiza os $O = O_{num}$ neurônios (variáveis linguísticas fuzzy) da camada ③ */
  for all neuronio o **in** Camada ③ **do**
    $\mathcal{OV}_o = \left\{\mathcal{OV}_{o_{(j)}}\left(\mu_{o_{(j)}}, C_{o_{(j)}}\right) \mid j \in J^{(t)}\right\}$

40: **end for**
41: /* Atualiza todos os neurônios $K^{(t)} = J^{(t)}$ (regras fuzzy) da camada ④ */
42: **for all** neuronio k **in** Camada ④ **do**
43: $\mathcal{R}^k = IF \bigwedge_{i=1}^{I} \boldsymbol{X}_i \; is \; \mathcal{IV}_{i_{(k)}} \; THEN \bigvee_{o=1}^{O} \boldsymbol{X}_{I+o} \; is \; \mathcal{OV}_{o_{(k)}}(\mathcal{W}_k)$
44: **end for**
45: **end function**

4.2.2 Modo de Operação *recalling*

Como mostrado em azul na Figura 4.2, somente as camadas de ③ até ⑤ são ativadas durante o modo de operação *recalling* da INFGMN. Durante este modo de operação, não ocorrem alterações nos parâmetros da rede. O vetor de entrada $\mathcal{X}_d$ for $d = 1, ..., I = D - O$ é apresentado a uma rede *parcialmente* treinada, diretamente para a camada ③, que segue sua execução até que $M = O$ saídas estejam disponíveis na camada ⑤ (*Output Layer*).

Os modos de operação *learning* e *recalling* não precisam ocorrer separadamente, isto é, aprendizado e utilização podem ser intercalados. De fato, mesmo após a apresentação de um único padrão de treinamento, a rede INFGMN já pode ser usada no modo *recalling* (a INFGMN pode usar imediatamente o conhecimento adquirido) e as estimativas produzidas se tornam mais precisas à medida que são apresentados mais dados de treinamento. Além disso, o processo de aprendizagem pode prosseguir perpetuamente, isto é, os parâmetros da INFGMN podem sempre ser atualizados à medida que os novos dados de treinamento estejam disponíveis.

O algoritmo apresentado na Listagem 2 sumariza a os passos executados pela rede INFGMN durante sua execução em modo de operação *recalling*. No Apêndice A, este algoritmo está implementado em Matlab entre as linhas 146 e 178.

Listagem 2 - Pseudocódigo Algoritmo do Modo de Operação *recalling*

1: **function** INFGMNRECALLING($\boldsymbol{X_I}$)
2: /* Fuzzifica I entradas na camada ③ */
3: $\lambda^{(\mathcal{IV}_i)} = \left\{ \mathcal{N}^1 \left(\mathcal{X}_i, \mu_{i_{(j)}}, C_{i_{(j)}} \right), \mid j \in J^{(t)} \right\}, \forall i \in I$
4: /* Calcula as áreas das funções de pertinência de saída na camada ③ */
5: $\phi^{(\mathcal{OV}_o)} = \left\{ \int_y \mathcal{N}^1 \left(y, \mu_{o_{(j)}}, C_{o_{(j)}} \right) dy \mid j \in J^{(t)} \right\}, \forall o \in O$
6: /* Avalia as $K^{(t)} = J^{(t)}$ regras fuzzy na camada ④ */
7: **for all** neuronio m **in** Camada ⑤ **do**
8: **for all** neuronio k **in** Camada ④ **do**
9: /* Computa o grau de ativação α_k, usando a conjunção multiplicativa */
10: $\alpha_k = \mathcal{W}_k \cdot \prod_{i=1}^{I} \lambda^{\left(\mathcal{IV}_{i_{(k)}}\right)}$

11: /* Computa a implicação multiplicativa e a agregação aditiva */
12: $\phi^{*m} = \phi^{*m} + \alpha_k \cdot \phi^{\left(\mathcal{OV}^{*m}_{(k)}\right)}$
13: **end for**
14: **end for**
15: /* Defuzzifica M saídas pelo método do centróide na camada ⑤ */
16: **for all** neuronio m **in** Camada ⑤ **do**
17: $y_m = \frac{\int_y y\phi^{*m}(y)dy}{\int_y \phi^{*m}(y)}dy$
18: **end for**
19: **end function**

4.3 O MÉTODO DE IMPUTAÇÃO INFGMN (*INFGMN IMPUTATION* (INFGMNI))

Originalmente, somente as camadas de ③ até ⑤ (representadas em azul na Figura 4.3) são ativadas durante o modo de operação *recalling* da rede neurofuzzy INFGMN. O vetor de entrada $\mathcal{X}_d$ for $d = 1, ..., I = D - O$ é apresentado a uma rede *parcialmente* treinada, diretamente para a camada ③, que segue sua execução até que $M = O$ saídas estejam disponíveis na camada ⑤ (*Output Layer*). Com as mudanças realizadas para acomodar o método de imputação da rede neurofuzzy INFGMN (INGFMNI), que estão destacadas em verde na Figura 4.3, todas as camadas da rede passam a ser ativadas em modo de operação *recalling*, lembrando que neste modo de operação, o estado e os parâmetros da rede neurofuzzy INFGMN treinada não são alterados.

Na camada ① os valores faltantes (cujo mecanismo de falta é MCAR ou MAR) são representados pelo vetor de dados faltantes $\widetilde{\mathcal{X}}$. Esta camada não sofreu alterações significativas e, portanto, a Equação 4.1 ainda é adequada à descrição do comportamento da mesma, tanto em modo de operação *learning* quanto em modo de operação *recalling* com imputação habilitada. Em ambos os casos, a camada ① transmite o vetor de entrada diretamente para a camada ② sem fazer alterações no mesmo.

Como ja foi dito na Seção 4.1, a rede neurofuzzy INFGMN faz uso de uma versão incremental de um modelo de mistura de Gaussianas (*Incremental Gaussian Mixture Model* - IGMM (ENGEL; HEINEN, 2010)) na camada ②. O IGMM pode ser visto como uma contrapartida incremental do algoritmo *Expectation Maximization* (EM). Assim, é possível utilizar uma versão aproximada do algoritmo EM para realizar o processo de imputação dos dados faltantes durante a execução do modo de operação *recalling* da rede neurofuzzy INFGMN (do ponto de vista de um IGMM, não existem diferenças entre as entradas e saídas). Desse modo, a operação $f^{②}$ descrita na Equação 4.2 passa a se chamar $f^{②}_L$ para

representar a operação desta camada em modo *learning*. Para operação em modo *recalling* (com imputação), esta camada passa a executar a seguinte operação:

$$f_R^{②}(\mathcal{X}) = \bigcup_{i=1}^{X} \left\{ \bigcup_{d=1}^{D-O} \begin{cases} \sum_{j=1}^{J^{(t)}} p(j|\mathcal{Z}^i)\overline{\mathcal{X}^i}, & \text{se } \mathcal{X}^i \in \widetilde{\mathcal{X}}^i \\ \mathcal{X}_d^i, & \text{senão} \end{cases} \right\}, \tag{4.10}$$

onde, $X = |\mathcal{X}|$ representa o número de padrões sob teste, $\mathcal{Z}^i = \{\mathcal{X}_1^i, \mathcal{X}_2^i, \cdots, \mathcal{X}_d^i, \cdots, \mathcal{X}_{D-O}^i\}$ representa o i-ésimo padrão de teste, $p(j|\mathcal{Z}^i)$ é a i-ésima probabilidade à *posteriori*, computada de acordo com a regra de Bayes e $\overline{\mathcal{X}^i}$ representa a forma geral da operação (veja (HEINEN, 2011) para a fundamentação desta operação) representada entre colchetes na Equação 4.13.

Para simplificar a explicação, considerou-se um problema MISO 3-dimensional. Então $D - O = 2$, uma vez que $O = 1$. Neste caso tem-se $\mathcal{Z} = \{\mathcal{X}_1, \mathcal{X}_2\}$. Considerando-se também que para um determinado padrão de teste i, temos $\mathcal{Z}^i = \left\{\mathcal{X}_1^i, \mathcal{X}_2^i \in \widetilde{\mathcal{X}}^i\right\}$, onde $\widetilde{\mathcal{X}}$ é o vetor de dados faltantes. Nestas condições, podemos aplicar o método de imputação INFGMNI para estimar e substituir o valor de $\mathcal{X}_2^i$ por $\hat{\mathcal{X}}_2^i$ antes de realizar o restante do método (estendido) de *recalling* da rede neurofuzzy INFGMN. No entanto, é importante lembrar que a INFGMN pode ser utilizada para problemas MIMO.

Inicialmente, cada componente j de $\mathcal{N}^{\mathcal{X}_1}$ realiza a seguinte operação:

$$\begin{aligned} p(\mathcal{X}_1^i|j) = & \frac{1}{(2\pi)^{D^{\mathcal{X}_1}/2}} \sqrt{|C_j^{\mathcal{X}_1}|} \\ & \times \exp\left\{ -\frac{1}{2} (\mathcal{X}_1^i - \mu_j^{\mathcal{X}_1})^T C_j^{\mathcal{X}_1 -1} (\mathcal{X}_1^i - \mu_j^{\mathcal{X}_1}) \right\}, \end{aligned} \tag{4.11}$$

isto é, uma distribuição Gaussiana multivariada. Cada componente j mantêm um vetor de médias $\mu_j^{\mathcal{X}}$ e uma matriz de covariâncias $C_j^{\mathcal{X}}$. Na sequência, a probabilidade à *posteriori* pode ser computada usando a regra de Bayes:

$$p(j|\mathcal{Z}^i) = \frac{p(\mathcal{X}_1^i|j)p(\mathcal{X}_2^i|j)p(j)}{\sum_{q=1}^{J^{(t)}} p(\mathcal{X}_1^i|q)p(\mathcal{X}_2^i|q)p(q)}. \tag{4.12}$$

Depois de computar a probabilidade à *posteriori* $p(j|\mathcal{Z}^i)$, $\hat{\mathcal{X}}_2^i$ pode ser

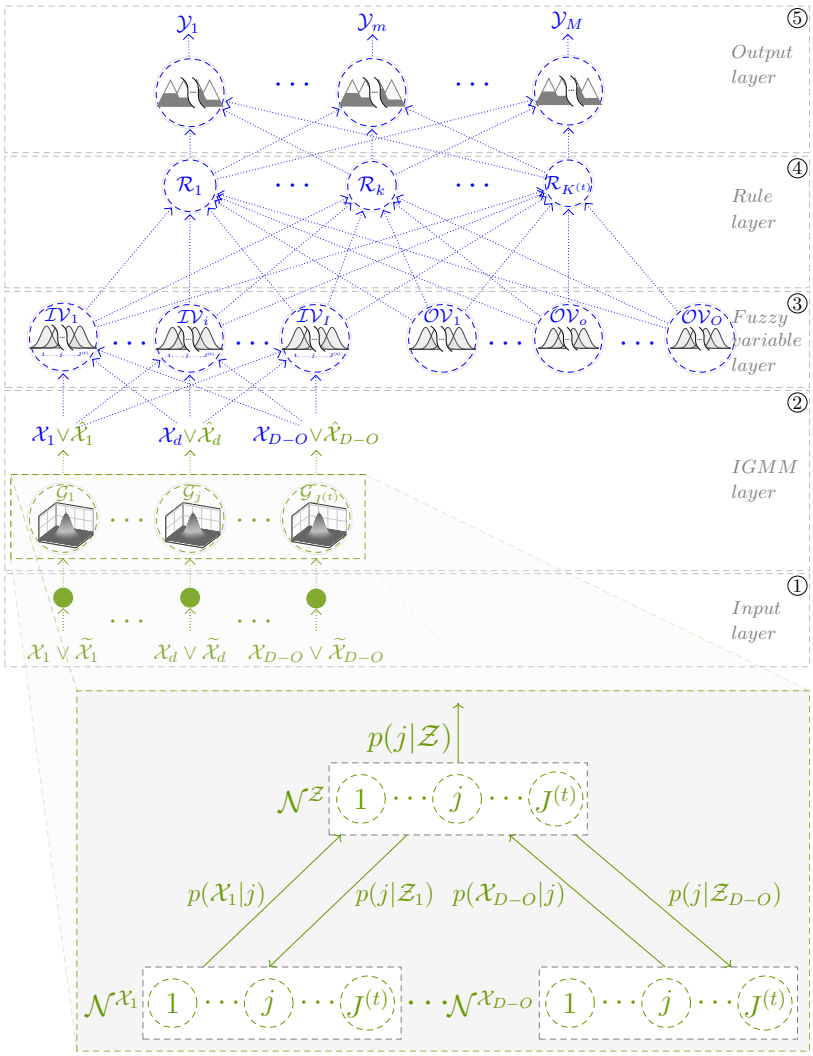

Figura 4.3 - Fluxo de informação na rede INFGMN em modo de operação *recalling* com imputação adaptiva

estimada como segue:

$$\hat{\mathcal{X}}_2^i = \sum_{j=1}^{J^{(t)}} p(j|\mathcal{Z}^i) \left[\mu_j^{\mathcal{X}_2} + C_j^{\mathcal{X}_2\mathcal{X}_1} {C_j^{\mathcal{X}_1}}^{-1} (\mathcal{X}_1^i - \mu_j^{\mathcal{X}_1}) \right], \tag{4.13}$$

onde $C_j^{\mathcal{X}_2\mathcal{X}_1}$ é a matriz de associação das componentes de mistura de $\mathcal{N}^{\mathcal{X}_1}$ e $\mathcal{N}^{\mathcal{X}_2}$.

O Algoritmo 3 sumariza o método de imputação da rede neuro-fuzzy INFGMN (INFGMNI) para o exemplo simplificado descrito nesta Seção. Uma possível implementação em código Matlab pode ser encontrado no Apêndice A, entre as linhas 180 e 212.

Listagem 3 - Método de Imputação INFGMN (INFGMNI)

```
1:  function INFGMNI(𝒵)
2:      𝒳₁ ← 𝒵(1)
3:      𝒳₂ ← 𝒵(2)
4:      for all 𝒳₂ⁱ in 𝒳₂ do
5:          if 𝒳₂ⁱ ∈ 𝒳̃₂ then
6:              for all j in 𝒩^{𝒳₁} do
7:                  p(𝒳₁ⁱ|j) ← Eq. 4.11
8:              end for
9:              p(j|𝒵ⁱ) ← Eq. 4.12
10:             𝒳₂ⁱ ← 𝒳̂₂ⁱ ← Eq. 4.13
11:         end if
12:     end for
13: end function
```

5 RESULTADOS E DISCUSSÕES

Esta seção avalia os desempenho de aprendizagem e modelagem da rede INFGMN proposta e de seu método adaptativo de imputação (INFGMNI), principalmente em termos do dilema Acurácia-Interpretabilidade, realizando-se três experimentos com aplicações *benchmark* que possuem dados completos: (1) identificação *online* de um sistema dinâmico não-linear com características variáveis no tempo; (2) um conjunto de 3 problemas utilizados para benchmark e disponíveis na base *Machine Learning repository* do UCI[1]; e (3) previsão das séries temporais do índice de mercado de ações *Standard and Poor's 500* (S&P-500), e três experimentos com conjuntos de dados reais que apresentam dados faltantes, para problemas de (1) classificação, (2) regressão e (3) uma série temporal.

Como medida de interpretabilidade considerou-se a complexidade do modelo gerado, isto é, o número de variáveis presentes na premissa das regras fuzzy geradas, bem como o número total de regras fuzzy geradas. Para medir a acurácia, foram utilizadas as seguintes métricas: o *Root Mean Squared Error* (RMSE), conforme Equação 5.1

$$RMSE(\mathbb{X}) = \sqrt{\frac{1}{X}\sum_{t=1}^{X}\left[x_{(t)} - \hat{x}_{(t)}\right]^2}, \tag{5.1}$$

onde $\mathbb{X}$ representa o conjunto de dados (tuplas), $X = |\mathbb{X}|$ é o número total de tuplas, $x_{(t)}$ desejada (esperada) para tupla t (ou momento t), conforme representado na Equação 5.6, e $\hat{x}_{(t)}$ é a resposta dada pela rede neurofuzzy INFGMN no momento t; o *Non-Dimensional Error Index (NDEI)*, definido por

$$NDEI(\mathbb{X}) = \frac{RMSE(\mathbb{X})}{\sigma(\mathbb{X})}, \tag{5.2}$$

que é o resultado da divisão do RMSE pelo desvio padrão da série de dados; e o *Mean Absolute Percentage Error* (MAPE) (FLORES, 1986), conforme Equação 5.3

$$MAPE = \frac{100}{n}\sum_{i=1}^{n}\left|\frac{x_i - \hat{x}_i}{x_i}\right|, \tag{5.3}$$

onde, x é o valor real, $\hat{x}$ é o valor previsto e n é o número total de valores faltantes.

[1] https://archive.ics.uci.edu/ml/datasets.html

Todos os experimentos foram conduzidos no ambiente Matlab rodando em um iMac 3.2GHz com um processador Intel Core i5 quad-core e 8GB de memória RAM DDR3 1867MHz.

5.1 EXPERIMENTOS COM APLICAÇÕES *BENCHMARK* QUE POSSUEM DADOS COMPLETOS

5.1.1 Experimento 1 - Identificação *Online* de um Sistema Dinâmico Não-linear com Características Variáveis no Tempo

Este experimento foi inicialmente proposto por Juang e Lin (1998) e adaptado por Tung e Quek (2010) para demonstrar a capacidade do modelo eFSM em evoluir (adaptar-se progressivamente) a sua estrutura computacional (base de regras) para descrever as características subjacentes de uma estrutura não-linear dinâmica com propriedades que variam no tempo. O sistema não-linear é definido pela equação expressa em

$$y(t+1) = \frac{y(t)}{1-y^2(t)} + u^3(t) + f(t), \tag{5.4}$$

onde $u(t) = sin(2\pi t/100)$, $y(t)$ e $f(t)$ são respectivamente sinal de entrada, saída e uma perturbação a ser introduzida no sistema. As condições iniciais $(u(0), y(0))$ são dadas como $(0, 0)$ para $u(t) \in [-1, 0; 1, 0]$ e $y(t) \in [-1, 5; 1, 5]$. O objetivo é aproximar $y(t+1)$ quando dado $(u(t), y(t))$ levando em conta a perturbação $f(t)$ que é definida como

$$f(t) = \begin{cases} 0, & 1 \leqslant t \leqslant 1000 \text{ and } t \geqslant 2001 \\ 1,0, & 1001 \leqslant t \leqslant 2000. \end{cases} \tag{5.5}$$

A Figura 5.1 apresenta o desempenho da modelagem incremental da rede eFSM e a Figura 5.2 apresenta o desempenho da modelagem incremental para a rede INFGMN proposta neste trabalho, levando em conta uma simulação com duração $t \in [1, 3000]$.

Como pode ser observado na Figura 5.2 (a) e 5.2 (b), há uma correspondência quase perfeita entre as saídas computadas pela INFGMN e as saídas desejadas do sistema, com a INFGMN atingindo um RMSE de $\pm 0,06$ para o conjunto de dados de teste (Figura 5.2 (d)), com picos de erro de no máximo $\pm 0,07$, enquanto a eFSM atingiu um RMSE de $\pm 0,1$ (Figura 5.1 (d)) mas com picos de erro que vão até $\pm 3,0$ e com uma correspondência aproximada entre as saídas calculadas e desejadas (Figura 5.1 (a) e 5.1 (b)).

Considerando o número de regras (Figura 5.1 (c) e Figura 5.2 (c)), a rede eFSM resultante obteve uma média de ± 21 regras, com flutuações

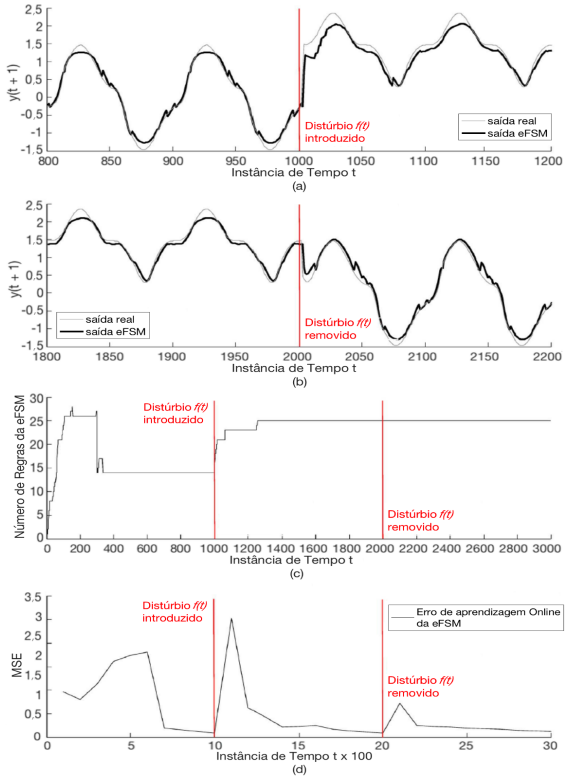

Figura 5.1 - Identificação *online* de um sistema não-linear com propriedades que variam no tempo usando eFSM. (a) Saída da rede eFSM quando o distúrbio $f(t)$ é introduzido no tempo $t = 1000$. (b) Saída da rede eFSM quando o distúrbio $f(t)$ é removido no tempo $t = 2000$. (c) Número de regras fuzzy identificadas pela eFSM durante o experimento. (d) Erro de aprendizagem *online* da eFSM. Adaptada de Tung e Quek (2010).

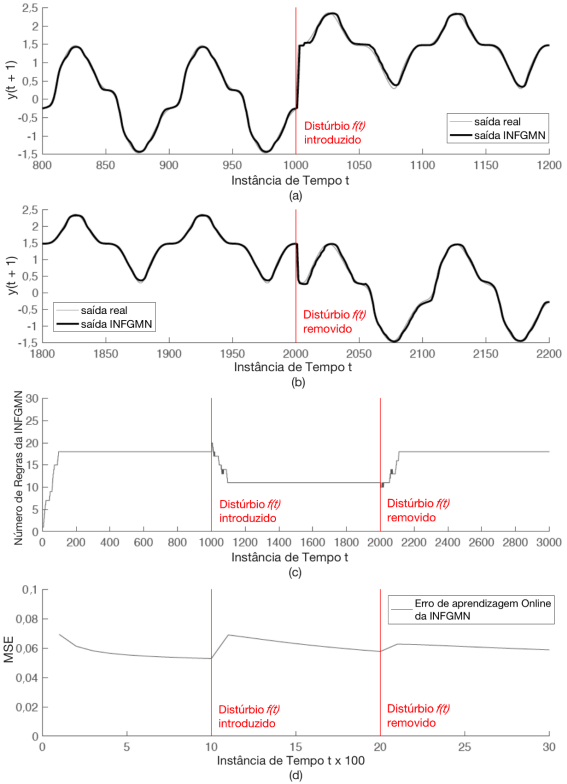

Figura 5.2 - Identificação *online* de um sistema não-linear com propriedades que variam no tempo usando INFGMN. (a) Saída da rede INFGMN quando o distúrbio $f(t)$ é introduzido no tempo $t = 1000$. (b) Saída da rede INFGMN quando o distúrbio $f(t)$ é removido no tempo $t = 2000$. (c) Número de regras fuzzy identificadas pela INFGMN durante o experimento. (d) Erro de aprendizagem *online* da INFGMN.

no número de regras fuzzy identificadas durante ≈ 400 passos no início do experimento e na inserção da perturbação em $t = 1000$. Observe que a eFSM não conseguiu detectar a remoção da perturbação no instante $t = 2000$, pelo menos não em termos de número de regras. Isso ocorreu porque as regras mais antigas e desatualizadas não foram removidas mesmo depois de o sistema não-linear retornar às suas características originais, a partir do tempo $t = 2000$. Para a rede INFGMN proposta, o número médio de regras foi de $\pm 15,5$ e as flutuações no número de regras fuzzy identificadas ocorreram somente durante ≈ 100 passos no início do experimento e na inserção ($t = 1000$) e remoção ($t = 2000$) da perturbação. Claramente, a INFGMN foi mais rápida e suave para aprender o modelo de dados subjacente, principalmente no início do experimento. Como consequência, os picos de RMSE na inserção e remoção dos distúrbios foram menores, como pode ser comparado na Figura 5.1 (d) e na Figura 5.2 (d) (observe que as escalas estão diferentes). Isso torna a rede INFGMN mais confiável.

Um ponto importante a se observar nos resultados deste experimento é que a rede INFGMN foi capaz de aprender ao longo de toda a sua vida, e sendo assim, ela e é aplicável a problemas complexos do mundo real com características dinâmicas ou variáveis no tempo. A INFGMN aborda efetivamente o dilema Estabilidade-Plasticidade, frequentemente encontrado no estudo e modelagem de redes neurofuzzy.

A Figura 5.3 apresenta os conjuntos fuzzy derivados para o problema descrito acima, considerando a entrada $y(t)$ no tempo $t = 1400$ (com perturbação nos dados) e $t = 2400$ (sem perturbação nos dados). Como pode ser observado, os conjuntos fuzzy derivados são ajustados aos deslocamentos de intervalo da variável $y(t)$ causados pela remoção e reintrodução da perturbação $f(t)$. Esta é outra evidência de que INFGMN efetivamente aborda o dilema Estabilidade-Plasticidade. Outra coisa importante a ser observada nesta figura, é que os conjuntos fuzzy derivados pela INFGMN são ordenados e possuem significados semânticos distintamente claros, e que podem ser melhorados pela adição de rótulos (*labels*) fuzzy individuais. Observe também que no caso dos conjuntos fuzzy para $y(t = 2400)$ há espaço para melhorias adicionais, substituindo-se os conjuntos fuzzy que estão altamente sobrepostos por um único conjunto fuzzy equivalente. Isso será abordado em trabalhos futuros.

Levando em conta o dilema Acurácia-Interpretabilidade, pode-se dizer que INFGMN é uma abordagem melhor (pelo menos para problemas dinâmicos não-lineares com características variáveis no Tempo). Como visto, a INFGMN gerou um modelo muito mais acurado e ao

mesmo tempo um modelo com mais interpretabilidade que o modelo eFSM, já que a complexidade das regras geradas é aproximadamente a mesma, mas o número de regras foi menor.

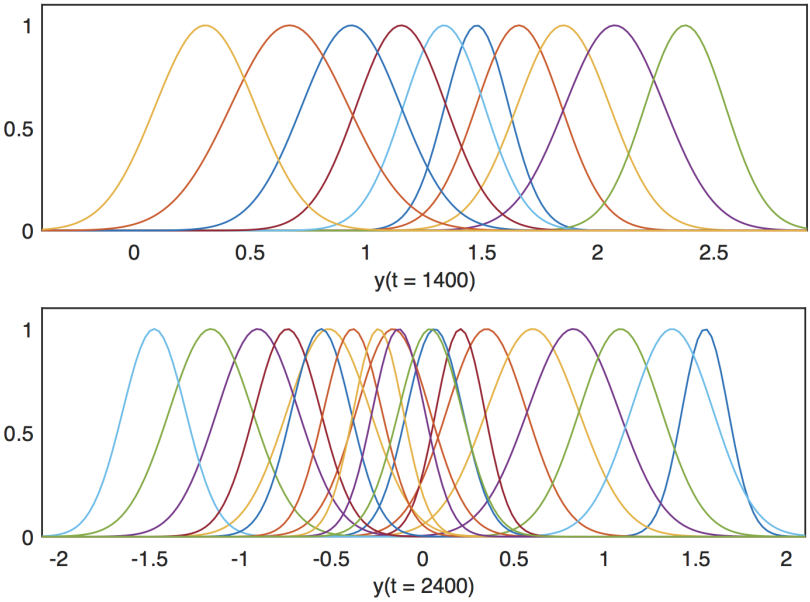

Figura 5.3 - A distribuição das funções de pertinência geradas pela INFGMN para $y(t = 1400)$ e $y(t = 2400)$, para o problema Identificação *Online* de um Sistema Dinâmico Não-linear com Características Variáveis no Tempo (Experimento 1).

5.1.2 Experimento 2 - Conjunto de dados UCI

Para avaliar as habilidades da rede INFGMN em problemas de classificação e regressão, três conjuntos de dados da vida real do conjunto de dados UCI, a saber: 1) os dados do *Servo*; 2) os dados do *Wine*; e 3) os dados do *Iris*, foram utilizados. Tanto os dados do *Wine* quanto os dados do *Iris* são problemas de classificação, enquanto os dados do *Servo* compõe um problema de regressão. Mais informações sobre esses três conjuntos de dados podem ser encontradas em Lichman (2013).

Seguindo o que foi proposto por Tung, Quek e Guan (2011), nas tarefas de regressão do *Servo* e classificação do *Wine*, os primeiros 60% dos dados são utilizados como conjunto de treino, enquanto os restantes 40% são utilizados para testes. Já para classificação do *Iris*, é é feita uma validação tripla com 34% dos dados usados para treinamento e 66% dos

Tabela 5.1 - Resultados experimentais consolidados para o conjunto de dados UCI.

			Servo		Wine		Iris		
Modelo	Tipo	MA	#Regras (*)	RMSE (*)	#Regras (*)	TA% (Rank)	#Regras (*)	TA% (σ) (*)	$\langle * \rangle$
k-NN	N.A.	NI	N.A.	1,022 (7)	N.A.	89,6 (6)	N.A.	94,65 ($\pm$ 1,60) (5)	6
MLP	N.A.	NI	N.A.	0,712 (3)	N.A.	82,54 (5)	N.A.	93,98 ($\pm$ 3,53) (6)	4,6
HyFIS	ML	I	84 (4)	0,742 (5)	96 (4)	94,37 (4)	13,3 (3)	95,36 ($\pm$ 3,17) (4)	4
DENFIS	TS	SI	8 (1)	0,734 (4)	15 (2)	98,59 (3)	12 (2)	97,01 ($\pm$ 0,95) (2)	2,3
EFuNN	ML	SI	75 (3)	0,804 (6)	100 (5)	100 (1)	28 (5)	92,63 ($\pm$ 2,98) (7)	4,5
SaFIN	ML	SI	12 (2)	0,575 (2)	93 (3)	100 (1)	13,7 (4)	96,34 ($\pm$ 0,55) (3)	2,5
INFGMN	**ML**	**I**	**8 (1)**	**0,380 (1)**	**7 (1)**	**100 (1)**	**3,33 (1)**	**97,11 ($\pm$ 0.,38) (1)**	**1**

MA = Modo de Aprendizagem, TA = Taxa de Acerto, I = Incremental, SI = Semi-Incremental, NI = Não-Incremental;
(*) Ranking parcial para o item em questão;
(σ) Desvio padrão;
$\langle * \rangle$ Ranking final médio.

dados como conjunto de testes. As medidas de *benchmarking* são a acurácia das respostas dos modelos para os dados de teste (calculada como RMSE para o problema de regressão e taxa de acerto (TA%) de classificação para as tarefas de classificação) e a interpretabilidade das redes geradas (indicada pelo tamanho da base de regras). A rede INFGMN é comparada com os seguintes modelos: k-NN (COVER; HART, 1967), MLP; HyFIS (KIM; KASABOV, 1999), DENFIS (KASABOV; SONG, 2002), EFuNN (KASABOV, 2001) e SaFIN (TUNG; QUEK; GUAN, 2011). Os resultados para estes modelos foram extraídos de Tung, Quek e Guan (2011).

Os resultados experimentais consolidados para o conjunto de dados UCI são mostrados na Tabela 5.1. Para as tarefas do *Servo* e do *Wine*, os resultados dos testes são listados, enquanto que para a classificação do *Iris* são apresentados os desempenhos médios. Os desempenhos dos modelos são classificados de acordo com suas acurácias de teste e número de regras geradas (interpretabilidade)[2]. Como pode ser observado, a rede INFGMN é a que possui o melhor desempenho quando os resultados dos modelos apresentados na Tabela 5.1 são classificados de acordo com suas acurácias e interpretabilidade para os dados teste, considerando o tamanho da base de regras gerada, para as três tarefas. A INFGMN gera uma quantidade menor de regras em comparação com os demais modelos de *benchmarking*, mas ao mesmo tempo, a acurácia nos testes é a melhor. Em média, a INFGMN atinge o topo do *ranking* (1), tornando-a a rede com melhor desempenho neste conjunto de experimentos. Considerando apenas os modelos do tipo ML e modelos com modo de aprendizagem (MA) *Incremental* e *Semi-Incremental*, o número médio de regras geradas dos modelos de *benchmarking* foi de 57,2, contra uma média de 6,1

[2]Os modelos k-NN e MLP não são classificados pelo tamanho de sua base de regras porque as informações não são aplicáveis ou não disponíveis.

regras geradas pela INFGMN. Estes resultados demonstram que a rede INFGMN é capaz de proporcionar desempenhos de modelagem superiores em relação às acurácias, mantendo um bom equilíbrio em termos de interpretabilidade, considerando-se a complexidade dos modelos gerados.

$\mathcal{R}^1$: IF sepal-length $is\ \mathcal{IV}_{1_{(1)}}(0,50; 6,08)$ $\wedge$ sepal-width $is\ \mathcal{IV}_{2_{(1)}}(0,27; 2,80)$ $\wedge$
petal-length $is\ \mathcal{IV}_{3_{(1)}}(0,53; 4,38)$ $\wedge$ petal-width $is\ \mathcal{IV}_{4_{(1)}}(0,21; 1,35)$
$THEN$ $(y\ is\ Setosa(\mathcal{OV}_{1_{(1)}}(0,04; 0,00)))$ $(y\ is\ Versicolour(\mathcal{OV}_{2_{(1)}}(0,04; 1,01)))$
$(y\ is\ Virginica(\mathcal{OV}_{3_{(1)}}(0,04; 0,00)))$ $WITH$ $\mathcal{W}_1(0,33)$

$\mathcal{R}^2$: IF sepal-length $is\ \mathcal{IV}_{1_{(2)}}(0,33; 5,04)$ $\wedge$ sepal-width $is\ \mathcal{IV}_{2_{(2)}}(0,29; 3,48)$ $\wedge$
petal-length $is\ \mathcal{IV}_{3_{(2)}}(0,31; 1,41)$ $\wedge$ petal-width $is\ \mathcal{IV}_{4_{(2)}}(0,16; 0,26)$
$THEN$ $(y\ is\ Setosa(\mathcal{OV}_{1_{(2)}}(0,04; 1,01)))$ $(y\ is\ Versicolour(\mathcal{OV}_{2_{(2)}}(0,04; -0,01)))$
$(y\ is\ Virginica(\mathcal{OV}_{3_{(2)}}(0,04; -0,01)))$ $WITH$ $\mathcal{W}_2(0,33)$

$\mathcal{R}^3$: IF sepal-length $is\ \mathcal{IV}_{1_{(3)}}(0,69; 6,66)$ $\wedge$ sepal-width $is\ \mathcal{IV}_{2_{(3)}}(0,33; 3,01)$ $\wedge$
petal-length $is\ \mathcal{IV}_{3_{(3)}}(0,55; 5,65)$ $\wedge$ petal-width $is\ \mathcal{IV}_{4_{(3)}}(0,27; 1,98)$
$THEN$ $(y\ is\ Setosa(\mathcal{OV}_{1_{(3)}}(0,04; -0,01)))$ $(y\ is\ Versicolour(\mathcal{OV}_{2_{(3)}}(0,04; 0,00)))$
$(y\ is\ Virginica(\mathcal{OV}_{3_{(3)}}(0,04; 1,01)))$ $WITH$ $\mathcal{W}_3(0,33)$

Figura 5.4 - Regras fuzzy do tipo ML extraídas da INFGMN para o conjunto de dados *Iris* do UCI.

Para dar um exemplo, a Figura 5.4 mostra um total de 3 regras fuzzy do tipo ML, que foram identificadas pela INFGMN com base nos dados de treinamento dados do conjunto de dados *Iris*. Pode-se facilmente compreender o conjunto de regras fuzzy extraídas pela INFGMN e verificar a validade dessas regras nesta tarefa de classificação. As regras seguem a mesma sintaxe usada pela Equação 4.7, com $I = 4$, $O = 3$ e $K = J = 3$. Os termos $\mathcal{IV}_{i_{(j)}}(\mu_{i_{(j)}}, C_{i_{(j)}})$ e $\mathcal{OV}_{o_{(j)}}(\mu_{o_{(j)}}, C_{o_{(j)}})$ correspondem à $i_{(j)}$-ésima e $o_{(j)}$-ésima funções Gaussianas de pertinência de entrada e saída, respectivamente, encontradas pela INFGMN durante a etapa de treinamento. Note que, como os termos são ordenados e possuem significados semânticos distintos e claros, eles podem ser substituídos por termos linguísticos fuzzy, por exemplo: "IF sepal-length $is\ SHORT$..." or "IF sepal-length $is\ AVERAGE$", e assim por diante. Pode-se notar que a IFNGMN forneceu um modelo fuzzy acurado e interpretável para o problema *Iris* (e para o conjunto de dados UCI em geral, como mostrado na Tabela 5.1) e que os conhecimentos adquiridos poderiam ser validados e relacionados com o conhecimento do domínio

do problema de um especialista humano.

Como o problema *Iris* é um "problema de classificação", a saída deve ser uma classe que pertence a {1 = *Setosa*, 2 = *Versicolour*, 3 = *Virginica*}. Na INFGMN, os termos $\mathcal{OV}_{o_{(j)}}(\mu_{o_{(j)}}, C_{o_{(j)}})$ das variáveis de saída são ordenados de tal forma que permitem a determinação da classe resultante com base em sub-intervalos do domínio de saída. Portanto, para o problema *Iris*, uma saída que cai no intervalo $(-\infty; 1,5)$ é considerada como sendo da classe *Setosa*, $[1,5; 2,5)$ classe *Versicolour* e $[2,5; +\infty)$ classe *Virginica*, respectivamente.

Uma observação final para este experimento é que a INFGMN apresentou um desempenho bom em tarefas de classificação, como pode ser observado na Tabela 5.1. Para o problema *Iris*, por exemplo, com uma média de apenas 3,33 regras, a INFGMN conseguiu classificar corretamente em 97,11% dos casos, usando uma validação cruzada tripla, com 34% dos dados como treinamento e 66% dos dados como conjunto de testes.

5.1.3 Experimento 3 - Previsão das Séries Temporais do Índice S&P-500

Seguindo os trabalhos de Tan e Quek (2010) e Oentaryo et al. (2014), este experimento estuda a capacidade de auto-reorganização da INFGMN usando dados de séries temporais financeiras do mundo real com base no índice de mercado S&P-500. No total, existem 14.893 padrões de dados, que foram coletados diariamente durante 60 anos (de 3 de janeiro de 1950 a março 12, 2009), e que atualmente estão disponíveis no site *Yahoo! Finance* no *ticker symbol* "GSPC"[3]. A Figura 5.5 mostra o comportamento variante no tempo dos padrões de dados (índice de ações), como uma distribuição não uniforme no intervalo de [16,66; 1565,15]. Como em Tan e Quek (2010) e Oentaryo et al. (2014), o interesse aqui está nas mudanças de trajetória dos valores dos índices, especialmente após os anos 80, quando a volatilidade crescente nas diferenças diárias é mais perceptível (veja a segunda metade da Figura 5.5).

O experimento consiste em uma simulação *online* da previsão diária do índice S&P-500, com cinco entradas definidas como $[y(t-4), y(t-3), y(t-2), y(t-1), y(t)]$, onde $y(t)$ é o valor absoluto do índice S&P-500 no tempo t, e a saída, $y(t+1)$, a ser prevista pelo modelo com capacidade de aprendizagem incremental. O resultado apresentado pela INFGMN para esta tarefa escalonada é comparado aos resultados dos seguintes modelos: EFuNN (KASABOV, 2001), DENFIS (KASABOV; SONG, 2002), ANFIS (JANG, 1993), eTS (ANGELOV; FILEV,

[3] https://finance.yahoo.com/quote/%5EGSPC/

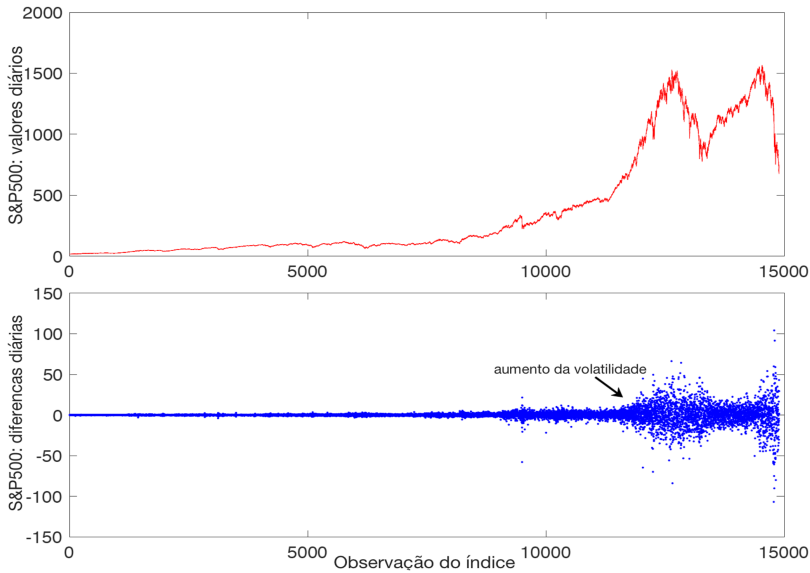

Figura 5.5 - Índice S&P-500 diário de 3 Janeiro 1950 a 12 Março 2009.

2004), SeroFAM (TAN; QUEK, 2010) e SPLAFIS (OENTARYO et al., 2014).

Tabela 5.2 - Resultados consolidados para os dados do índice S&P-500.

Modelo	Tipo	Modo de Aprendizagem	#Regras	NDEI
EFuNN	ML	Semi-Incremental	114,3	0,154
DENFIS	TS	Semi-Incremental	6,0	0,020
ANFIS	TS	Não-Incremental	32,0	0,015
eTS	TS	Incremental	14,0	0,015
SeroFAM	ML	Incremental	29,9	0,027
SPLAFIS	TS	Incremental	10,2	0,015
INFGMN	**ML**	**Incremental**	**7,2**	**0,023**

Da Tabela 5.2, pode-se concluir que a INFGMN produz performances competitivas em termos de acurácia e bons resultados em termos de interpretabilidade. Ela obteve um NDEI relativamente baixo = 0,023, e apresentou bons resultados com uma média de 7,2 regras fuzzy. Embora a SeroFAM forneça NDEI comparável neste estudo, o tamanho de sua base de regras é aproximadamente quatro vezes maior que o da INFGMN, o que implica menor interpretabilidade. Além disso, podemos

ver que a INFGMN oferece um desempenho competitivo em termos do teste NDEI, comparável ao de DENFIS, ANFIS, eTS e SPLAFIS que são modelos NFS do tipo TS. Entretanto, exceto para o DENFIS, a INFGMN produz menos regras do que os outros três. Embora o DENFIS gere um número de regras menor do que a INFGMN, ele não possui um sistema de aprendizado totalmente incremental. Em contraste, INFGMN é capaz de treinar e prever recursivamente a qualquer momento, sem a necessidade de conhecimento prévio do conjunto de dados completo. Estes resultados mostram a capacidade da INFGMN em acompanhar de perto as principais mudanças de trajetórias em ambientes com variação temporal. Isto pode ser ainda mais justificado pela comparação dos valores de índice reais e previstos do índice S&P-500 na Figura 5.6, i.e., a INFGMN acompanhou muito bem as mudanças no regime de valor do índice, incluindo os dois picos em torno do ano 2000 e 2007 e o vale em torno de 2003. Claramente a partir dos resultados, é possível afirmar que a INFGMN foi capaz de lidar com o dilema Estabilidade-Plasticidade.

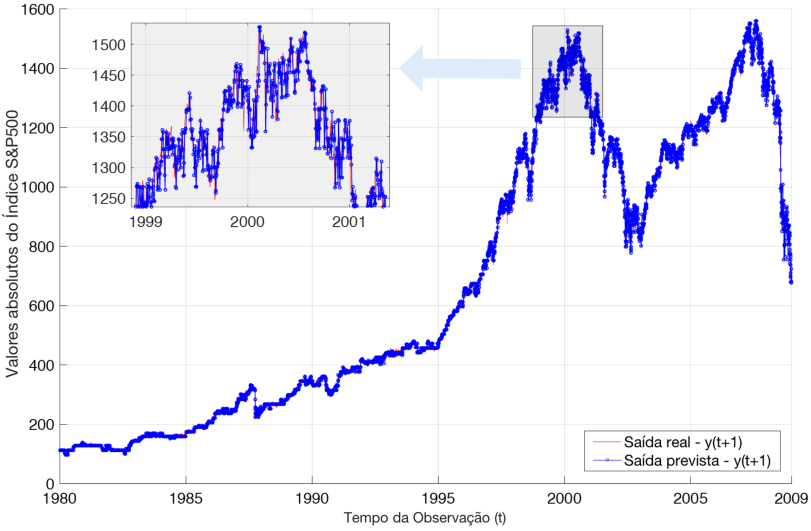

Figura 5.6 - S&P-500: saída prevista versus saída real. Baseada em Tan e Quek (2010).

5.2.1 Experimento 4 - Problemas de Classificação com Dados Faltantes

Neste experimento o método de imputação (INFGMNI) da rede neurofuzzy INFGMN foi comparado com os métodos de imputação KNN (KNNI) (BATISTA; MONARD et al., 2002), FCM (FCMI) (LUENGO; SÁEZ; HERRERA, 2012; LI et al., 2004), SOM (SOMI) (FESSANT; MIDENET, 2002) e o método CCAI (LIU et al., 2016). As informações básicas dos conjuntos de dados reais utilizados neste experimento são dadas na Tabela 5.3. No caso do conjunto de dados *Hepatitis*, muitos padrões já continham valores faltantes. Os padrões com valores ausentes são considerados amostras de teste, e os outros são usados como amostras de treinamento. Não há valores faltantes nos demais conjuntos de dados, e desta forma, presume-se que n valores estejam faltando de maneira completamente aleatória (MCAR) em todas as dimensões de cada amostra de teste, conforme a metodologia descrita em (LIU et al., 2016). Para estes 4 conjuntos de dados, foi realizada uma validação cruzada dupla[4], levando em consideração a vantagem de os conjuntos de treinamento e teste serem grandes, sendo que amostra é usada para treinar e testar em cada validação. A validação cruzada dupla foi repetida 10 vezes e a taxa média de erro Re dos diferentes métodos são apresentadas na Tabela 5.4. A taxa de erro denotada por Re é calculada por $Re = Ne/T$, onde Ne é o número de erros de classificação e T é o número de objetos sob teste. Particularmente, o resultado de classificação relatado para KNNI representa o valor médio, com K variando de 5 a 15. Os dados consolidados apresentados na Tabela 5.4 para os métodos KNNI, FCMI, SOMI e CCAI foram extraídos de (LIU et al., 2016), considerando-se para tal os melhores resultados para ambos os métodos e para ambos os classificadores, EK-NN e ENN, conforme foram utilizados em (LIU et al., 2016). A coluna CCAI da Tabela 5.4, apresenta o melhor dos resultados relatados em (LIU et al., 2016) para os métodos PCC (LIU et al., 2015), SCCAI e CCAI. Estes dados foram agrupados desta forma devido a limitação de espaço e ao fato de SCCAI e CCAI serem extensões do método PCC.

Pode-se verificar na Tabela 5.4 que o resultado de classificação dado pelo método INFGMNI sempre produz a menor taxa de erro em comparação ao método CCAI e demais métodos mais tradicionais FCMI,

[4]Mais precisamente, as amostras em cada classe são aleatoriamente designadas para dois conjuntos S_1 e S_2 com tamanhos iguais. Em seguida, cada modelo é treinado com S_1 e testado com S_2, e reciprocamente.

Tabela 5.3 - Informações básicas dos conjuntos de dados usados no Experimento 4 - Problemas de Classificação com Dados Faltantes.

Nome	Classes	Atributos	Instâncias
Breast	2	9	699
Hepatitis	2	19	155
Iris	3	4	150
Seeds	3	7	210
Wine	3	13	178

Tabela 5.4 - Resultados para problemas de classificação considerando-se diferentes conjuntos de dados (taxas de erro em %).

Conj. de dados	n	FCMI	KNNI	SOMI	CCAI	INFGMNI
Hepatitis	N.A.	25,33	26,67	25,33	23,67	**12,80**
Breast	3	3,81	3,95	3,51	3,66	**3,17**
	6	6,18	8,20	5,93	6,15	**5,20**
	7	11,42	11,54	12,45	9,66	**7,28**
Iris	1	6,89	4,89	5,00	5,33	**4,40**
	2	13,89	11,33	12,67	12,00	**7,40**
Seeds	2	15,24	11,19	10,20	9,52	**5,57**
	4	17,14	11,98	12,59	10,48	**7,23**
Wine	3	26,97	26,97	27,53	6,97	**2,80**
	7	33,24	26,22	31,30	7,87	**6,06**

n = número total de valores faltantes, N.A. = Não se Aplica.

KNNI e SOMI. Este resultado indica que o método de imputação empregado pela INFGMN pode ser bastante efetivo para problemas de classificação com dados faltantes, com a vantagem de que a INFGMN pode lidar com o dilema Estabilidade-Plasticidade, não sendo afetada pelo problema da interferência catastrófica. Desse modo, seu processo de aprendizagem e por consequência imputação pode continuar perpetuamente à medida que novos dados de treinamento se tornam disponíveis, ou seja, imputar, aprender e relembrar não precisam acontecer separadamente. Como uma informação adicional em relação ao método INFGMNI, o tempo médio gasto com treino foi de 0,19s e com teste foi de 0,22s em cada uma das validações, ou seja, levando-se em conta o tamanho médio dos conjuntos de dados em questão, obtêm-se um tempo médio de 15ms para cada chamada (imputação, treino e teste) da rede neurofuzzy INFGMN.

5.2.2 Experimento 5 - Problemas de Regressão com Dados Faltantes

Neste experimento a eficácia para a imputação de dados faltantes do método INFGMNI proposto foi testada em alguns conjuntos de dados para problemas de regressão, apresentados na Tabela 5.5. Inicialmente, esses conjuntos de dados não têm valores faltantes, então, como em (GAUTAM; RAVI, 2015), o experimento foi realizado excluindo-se aleatoriamente alguns valores dos conjuntos de dados originais. Cada conjunto de dados é dividido em 10 partes iguais e então 9 partes agrupadas e usadas para treinamento e a décima parte - com 10% dos valores (células) excluídos aleatoriamente (assegurando-se que pelo menos uma célula de cada registro seja excluída) - é usada para teste. Esse procedimento é aplicado para todas as partes, selecionando uma parte para teste de cada vez, até que todas sejam selecionadas. Assim, tem-se 10 validações cruzadas. Para todos os conjuntos de dados, a Tabela 5.6 apresenta os valores médios e desvios padrão com base na medida de erro MAPE, conforme Equação 5.3.

Tabela 5.5 - Informações básicas dos conjuntos de dados usados no Experimento 5.

Nome	Atributos	Instâncias
Boston housing	13	506
Forest fires	10	516
Auto mpg	7	392
Body fat	14	252

O desempenho do método INFGMNI proposto foi comparado com vários métodos híbridos *online* e *offline* para imputação, como *Particle Swarm Optimization Trained Auto Associative Neural Network* (PSOAANN), *Particle Swarm Optimization Trained Auto Associative Wavelet Neural Network* (PSOAAWNN), *Radial Basis Function Auto Associative Neural Network* (RBFAANN), *General Regression Auto Associative Neural Network* (GRAANN), que foram propostos em (RAVI; KRISHNA, 2014), K-means + MLP proposto por (ANKAIAH; RAVI, 2011) e como referência foi utilizado também *Mean Imputation*.

Os resultados consolidados para PSOAANN, PSOAAWNN, RBFAANN, GRAANN, K-means + MLP e *Mean Imputation* para todos os conjuntos dados foram extraídos de (GAUTAM; RAVI, 2015). Nota-se novamente que, para os conjuntos de dados testados, a rede neurofuzzy INFGMN através de seu método de imputação adaptativo INFGMNI obteve um melhor desempenho em todos os conjuntos de dados utilizados no experimento. A melhoria de desempenho variou de 14,7 a 59,8% em

Tabela 5.6 - Problemas de Regressão com dados faltantes, com média e desvio padrão dos valores MAPE para 10 repetições.

Conj. dados	GRAANN (*)	PSOAANN (*)	PSOAAWNN (*)	RBFAANN (*)	MI (*)	KMLPI (*)	INFGMNI (*)
Boston housing	15,38 (2,4)	24,61 (5,9)	30,94 (7,5)	98,87 (32,7)	37,77 (10,3)	21,01 (4,1)	**8,98 (3,4)**
Forest fires	18,47 (2,0)	22,69 (5,9)	26,62 (5,4)	59,24 (19,1)	24,72 (6,8)	26,61 (5,2)	**17,97 (3,6)**
Auto mpg	15,54 (3,7)	37,59 (10,1)	38,16 (13,5)	62,53 (15,2)	59,70 (14,3)	23,75 (4,5)	**6,24 (1,3)**
Body fat	4,61 (2,0)	7,61 (4,5)	9,21 (4,0)	25,40 (14,7)	11,61 (7,1)	7,83 (1,6)	**3,93 (1,9)**

MI - *Mean Imputation*, KMLPI - *K-means* + MLP *Imputation*;
(*) Desvio padrão.

relação ao melhor dos resultados apresentados pelos demais métodos. Para o caso do conjunto de dados *Auto mpg*, por exemplo, a rede neurofuzzy INFGMN obteve uma melhora de ≈60% em relação ao método GRAANN e de ≈990% de melhora em relação ao método RBFAANN. Esses resultados indicam que a INFGMN pode ser uma excelente alternativa para problemas de regressão com dados faltantes, operando em modo *online* (adaptativo) e/ou *offline*.

5.2.3 Experimento 6 - Serie Temporal Univariada

No terceiro e último experimento, a intensão foi avaliar o desempenho da rede neurofuzzy INFGMN na previsão de séries temporais univariadas com dados faltantes. Para tanto, a serie temporal univariada chamada *Air Passengers* foi selecionada. Esta série temporal representa o número total de passageiros aéreos das companhias aéreas internacionais entre 1949 e 1960, em milhares, perfazendo um total de 144 observações, conforme ilustrado na Figura 5.7.

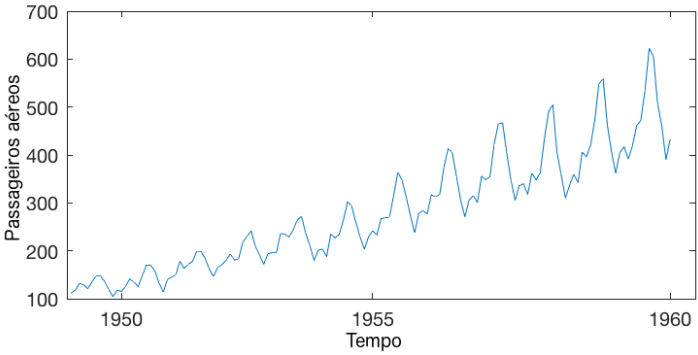

Figura 5.7 - Série de dados de passageiros aéreos.

Para modelar essa serie temporal com o uso da rede neurofuzzy INFGMN, foram usados os atrasos (lags) $x_{(t)}$, $x_{(t-10)}$ e $x_{(t-11)}$ para prever $x_{(t+1)}$, criando-se 132 tuplas seguindo o template

$$[x_{(t-12)},\ x_{(t-11)},\ x_{(t-1)}] \rightarrow x_{(t)}. \tag{5.6}$$

A Medida de erro utilizada foi o RMSE, confome Equação 5.1.

O método de imputação INFGMNI foi comparado com *Mean Imputation* (que substitui os valores faltantes pelo valor médio das outras observações), *MA Imputation* (que substitui o valor em falta pela média móvel ponderada, usando um tamanho de janela semi-adaptativo para garantir que todos os dados faltantes sejam substituídos) e *Last Observation Carried Forward* - LOCF *Imputation* (que substitui cada valor faltante com o valor atual mais recente antes dele). Esses métodos de imputação estão disponíveis na biblioteca *imputeTS* da linguagem de programação R (MORITZ; BARTZ-BEIELSTEIN, 2015), com os nomes $na.mean$, $na.ma$ e $na.locf$, sendo que os mesmos foram empregados em suas configurações padrão.

A dinâmica do experimento se deu da seguinte forma:

- Criar uma série temporal com taxa R% de dados faltantes ao acaso.
- Para os métodos de imputação Mean, LOCF ou MA:
  - 1 - Reconstruir a série temporal de usando um método de imputação (*Mean*, LOCF ou MA).
  - 2 - Criar 132 tuplas seguindo o template representado na Equação 5.6.
  - 3 - Para cada tupla t, usar a rede neurofuzzy INFGMN em modo *online* para:
    - 3.1 - Estimar (recall) $x(t)$ a partir de $[x_{(t-12)}, x_{(t-11)}, x_{(t-1)}]$;
    - 3.2 - Computar o erro, de acordo com a Equação 5.1, utilizando o valor real esperado ($x_{(t)}$) e o o valor previsto ($\hat{x}_{(t)}$);
    - 3.3 - Treinar a rede apresentando a tupla t.
- Para os método de imputação da rede neurofuzzy INFGMN (INFGMNI):
  - 1 - Criar 132 tuplas seguindo o template representado na Equação 5.6.
  - 2 - Para cada tupla t, usar a rede neurofuzzy INFGMN em modo *online* para:

2.1 - Imputar os valores faltantes na tupla t;

2.2 - Estimar (recall) $x(t)$ a partir de $[x_{(t-12)}, x_{(t-11)}, x_{(t-1)}]$;

2.3 - Computar o erro, de acordo com a Equação 5.1, utilizando o valor real esperado ($x_{(t)}$) e o o valor previsto ($\hat{x}_{(t)}$);

2.4 - Treinar a rede apresentando a tupla t imputada.

- Para a rede neurofuzzy INFGMN sem imputação (INFGMN*):

1 - Criar 132 tuplas seguindo o template representado na Equação 5.6.

2 - Para cada tupla t, usar a rede neurofuzzy INFGMN em modo *online* para:

2.1 - Estimar (recall) todos os valores faltantes e também $\hat{x}_{(t)}$ a partir dos valores presentes na tupla t;

2.2 - Computar o erro, de acordo com a Equação 5.1, utilizando o valor real esperado ($x_{(t)}$) e o o valor previsto ($\hat{x}_{(t)}$);

2.3 - Treinar a rede apresentando a tupla t preenchida com os valores estimados.

A Tabela 5.7 apresenta os resultados para este experimentos. Mais uma vez, para este caso específico da série temporal univariada *Air Passengers*, a rede neurofuzzy INFGMN, com seu método de imputação INFGMNI apresentou melhor desempenho. Note que para o caso do método de imputação proposto (INFGMNI), o processo é realizado apenas com o uso do conhecimento adquirido pela rede neurofuzzy INFGMN até o momento, operando em modo *online*, podendo prosseguir intercalando imputação, *recall* e treino perpetuamente.

Outro fato interessante evidenciado pelo experimento foi que, mesmo sem realizar o processo de imputação, a rede neurofuzzy INFGMN foi capaz de realizar boas previsões (coluna INFGMN* na Tabela 5.7), utilizando somente a informação disponível em cada tupla.

Em termos de tempo de execução, o método de imputação INFGMNI é relativamente rápido. Neste experimento, nos casos em que 25% dos dados eram faltantes, o tempo médio para um ciclo completo de imputação, *recall* e treino foi de 11,7ms e o tempo médio para os casos com 25% dos dados eram faltantes sem realização de imputação foi 11,5ms, o que da uma diferença de apenas 0,2ms e demonstra que o mecanismo de imputação não causa sobrecarga à rede neurofuzzy INFGMN.

Tabela 5.7 - Previsão de serie temporal univariada, com média e desvio padrão dos valores RMSE para 20 repetições - Serie *Air Passengers*

R%	Mean	LOCF	MA	INFGMNI	INFGMN*
5	30,01 (2,62)	25,72 (1,03)	25,25 (1,37)	**24,47 (0,93)**	24,87 (1,25)
10	32,53 (2,51)	26,41 (1,56)	25,99 (1,34)	**25,15 (1,19)**	25,69 (1,48)
15	34,92 (3,29)	27,40 (1,83)	27,12 (1,67)	**26,01 (1,57)**	27,13 (1,96)
20	36,44 (3,35)	27,87 (2,13)	28,16 (1,52)	**26,88 (1,48)**	28,43 (2,32)
25	40,92 (3,06)	29,45 (2,88)	28,57 (1,47)	**28,24 (2,17)**	28,98 (2,01)

* INFGMN sem imputação, R = Percentual de dados faltantes.

5.3 DISCUSSÃO

Conforme demonstrado nas Seções 5.1 e 5.2, a rede INFGMN pode produzir estimativas razoáveis com base em poucos dados de treinamento, ou mesmo em caso dados faltantes (tuplas incompletas), aprendendo de forma incremental e usando uma varredura única dos dados de treinamento (cada padrão de treinamento pode ser usado e descartado imediatamente). Uma vez que a rede INFGMN pode lidar com o dilema Estabilidade-Plasticidade e não é afetada pelo problema da interferência catastrófica, seu processo de aprendizagem pode continuar perpetuamente à medida que novos dados de treinamento se tornam disponíveis, ou seja, aprendizagem e utilização não precisam ser realizadas separadamente. A INFGMN define uma base de regras fuzzy de tipo ML de forma automática e incremental (novas regras são adicionadas sempre que necessário), tentando fornecer o melhor *trade-off* entre acurácia e interpretabilidade, permitindo que ela lide muito bem com o dilema Acurácia-Interpretalidade.

Vários experimentos foram conduzidos para validar o modelo proposto. O primeiro baseou-se na identificação *online* de um sistema dinâmico não-linear com características que variam ao longo do tempo, usando dados sintéticos definidos pelas Equações 5.4 e 5.5. Os resultados deste experimento demonstraram que a rede INFGMN conseguiu um desempenho muito bom em termos de acurácia e interpretabilidade, aprendendo o modelo de dados subjacente de forma rápida e suave, apresentando um resultado mais confiável do que sua concorrente eFSM.

O segundo experimento foi selecionado para avaliar a capacidade da rede INFGMN em lidar com os problemas de classificação e regressão. Neste exeprimento foram usados três conjuntos de dados do mundo real do repositório UCI, a saber: 1) o conjunto de dados *Servo*; 2) o con-

junto de dados *Wine*; e 3) o conjunto de dados *Iris*. Mais uma vez, os resultados apresentados são promissores. A rede INFGMN demonstrou melhor desempenho em termos de acurácia e interpretabilidade quando comparada aos modelos de *benchmark*. Em média, a INFGMN liderou o ranking (1) e alcançou o melhor desempenho neste conjunto de experimentos. Considerando-se apenas os modelos do tipo ML, com modos de aprendizagem incremental ou semi-incremental, o número médio de regras geradas pelos modelos de *benchmarking* foi de 57,2, em contrapartida a uma média de 6,1 regras que foram geradas pela rede INFGMN. Esses resultados demonstram que a rede INFGMN é capaz de obter um desempenho de modelagem superior em termos de acurácia, mantendo um bom equilíbrio em relação à interpretabilidade (considerando a complexidade dos modelos gerados).

Uma observação final com relação ao segundo experimento é que a rede INFGMN produz boas estimativas, independentemente da ordem em que os dados são apresentados. Portanto, a rede INFGMN pode ser usada tanto para problemas *online* (tempo real), onde os dados geralmente chegam de forma ordenada, como também para modelagem de sistemas *offline*, onde os dados podem se apresentados de forma não ordenada e aleatória.

Na sequência, para estudar a capacidade de auto-reorganização da rede INFGMN, o terceiro experimento usou dados de séries temporais financeiras do mundo real, baseadas no índice *Standard and Poor's 500* (S&P-500). Para tal, foi realizada uma simulação *online* de previsão dos valores diários do índice. Os resultados obtidos pela rede INFGMN foram competitivos em termos de acurácia e em termos de interpretabilidade, provando que a rede é capaz de aprender e prever recursivamente qualquer ponto do intervalo de dados, sem conhecimento prévio do conjunto de dados completo. Claramente, com base nesses resultados, a rede INFGMN conseguiu lidar com o dilema Estabilidade-Plasticidade.

Considerando-se a extensão da rede neurofuzzy INFGMN pela adição do método de imputação adaptativo INFGMNI, tornando-a apta a lidar com problemas com dados faltantes, cujos os mecanismos de falta sejam do tipo MCAR ou MAR, a rede INFGMN pode produzir estimativas razoáveis com base em poucos dados de treinamento, aprendendo incrementalmente usando uma única varredura sobre os dados de treinamento, mesmo nos casos em que ocorram tuplas incompletas de dados . Assim, é possível afirmar que a rede neurofuzzy INFGMN pode servir como um mecanismo de imputação adaptativo.

Para avaliar a capacidade da rede INFGMN em lidar com problemas que não possuem dados completos, foram realizados vários experimentos. O Experimento 4, avaliou o desempenho da rede neuro-

fuzzy INFGMN para problemas de classificação com dados faltantes. Os resultados deste experimento demonstraram que a INFGMN obteve um desempenho muito bom em termos de estimativa dos dados faltantes, fornecendo um resultado mais confiável que os demais modelos de *benckmark*. Claramente, a INFGMN foi capaz de aprender e fazer estimativas de qualidade, mesmo na ocorrência de dados faltantes, o que a torna aplicável a problemas mais complexos do mundo real com características dinâmicas ou variáveis no tempo, onde possa aplicar sua capacidade de aprendizagem e imputação adaptativas. Isso efetivamente aborda o problema de dados faltantes, comumente encontrado no estudo e modelagem de sistemas neurofuzzy.

No quinto experimento, a eficácia de imputação de dados faltantes pelo método INFGMNI proposto, foi testada em alguns conjuntos de dados para problemas de regressão, a saber: 1) *Boston housing*, 2) *Forest fires*, 3) *Auto mpg* e 4) *Body fat*. Mais uma vez, os resultados foram promissores. A INFGMN foi a rede que apresentou melhor desempenho em termos de acurácia na ocorrência de dados faltantes. A rede INFGMN por meio de seu método de imputação adaptativo INFGMNI obteve o melhor desempenho para todos os conjuntos de dados utilizados neste experimento. A melhoria de desempenho variou de 14,7 a 59,8% em relação ao melhor dos resultados apresentados pelos demais métodos. Para o caso do conjunto de dados *Auto mpg*, por exemplo, a rede neurofuzzy INFGMN obteve uma melhora de $\approx$60% em relação ao método GRAANN e de $\approx$990% de melhora em relação ao método RBFAANN. Esses resultados indicam que a INFGMN pode ser uma excelente alternativa para problemas de regressão com dados faltantes, operando em modo *online* (adaptativo) e/ou *offline*.

No sexto e último experimento, a intenção foi avaliar o desempenho da rede NFGMN na previsão de séries temporais univariadas com dados faltantes. Usando a serie temporal univariada chamada *Air Passengers*, realizou-se uma simulação *online* da previsão mensal do número total de passageiros aéreos das companhias aéreas internacionais entre 1949 e 1960, em milhares. Mais uma vez, para este caso específico da serie temporal *Air Passengers*, a rede neurofuzzy INFGMN, com seu método de imputação INFGMNI apresentou melhor desempenho. Neste experimento, o método INFGMNI realizou o processo de imputação apenas com o uso do conhecimento adquirido pela rede neurofuzzy INFGMN até o momento, operando em modo *online*, podendo prosseguir intercalando treino, imputação e previsão perpetuamente.

Embora sejam necessários mais experimentos com problemas reais de regressão, classificação e previsão de séries temporais (multivariadas, inclusive), os resultados aqui apresentados para os experimentos

realizados são bastante promissores, indicando que a rede neurofuzzy INFGMN pode ser uma excelente alternativa para trabalhar com conjuntos de dados faltantes, tanto em modo *offline* quanto *online* (adaptativo).

Em relação ao tempo de execução, embora o cálculo das PDF requeira a inversão de várias matrizes de covariância em cada iteração, o desempenho computacional da rede INFGMN é melhor do que os outros modelos que usam GMM. Diferentes estímulos de entrada/saída são processados em neurônios distintos (componentes Gaussianas na camada *IGMM layer* e, portanto, as matrizes de covariância são divididas em matrizes de covariância menores associadas a cada estímulo de entrada/-saída. Além disso, em problemas de grandes dimensões, o número de neurônios pode ser aumentado para reduzir ainda mais o tamanho das respectivas matrizes, o que possibilita estabelecer um compromisso entre acurácia e desempenho computacional, o que é muito útil em aplicações em tempo real. Considerando-se as camadas *Fuzzy variable layer* e *Rule layer*, o tempo de execução da rede INFGMN é relativamente rápido. Por exemplo, o tempo médio para um total de 100 execuções completas de cada experimento, para o conjunto de dados UCI (**5.1.2**), foi de de 4,7ms para o problema *Iris*, 17,2ms para o problema *Wine* e 6,5ms para o problema *Servo*. Para as tarefas *online* apresentadas nos experimentos **5.1.1** e **5.1.3**, a rede INFGMN teve um tempo de execução médio de 3ms e 6ms, respectivamente.

6 CONSIDERAÇÕES FINAIS

6.1 CONCLUSÃO

Este trabalho apresentou a rede neuro-fuzzy INFGMN. Um NFS com capacidade de aprendizagem incremental que pode gerar modelos interpretáveis e acurados. A rede INFGMN pode produzir estimativas razoáveis com base em poucos dados de treinamento e com dados faltantes aprendendo incrementalmente usando uma única varredura sobre os dados de treinamento, ou seja, cada padrão de treinamento pode ser usado e imediatamente descartado. Este tipo de abordagem é ideal para aplicações de tempo real. Como a INFGMN pode lidar com o dilema *E*stabilidade-Plasticidade e não sofre de interferências catastróficas, seu processo de aprendizagem pode prosseguir perpetuamente, à medida que os novos dados de treinamento chegam, isto é, não há fases separadas para aprendizagem (modo de operação *learning*) e utilização (modo de operação *recalling*). A rede INFGMN define uma base de regra fuzzy do tipo ML de forma automática e incremental (novas regras são adicionadas sempre que necessário) e procura fornecer a melhor relação custo-benefício entre acurácia e interpretabilidade e graças a isso, a INFGMN também pode lidar muito bem com o dilema Acurácia-Interpretabilidade.

Para validar o modelo proposto, foram realizados vários experimentos. O primeiro foi a identificação *online* de um sistema dinâmico não linear com características que variam no tempo usando dados sintéticos definidos pelas Equações 5.4 e 5.5. Os resultados deste experimento demonstraram que a INFGMN obteve um desempenho muito bom em termos de interpretabilidade e acurácia e também foi mais rápida e suave para aprender o modelo de dados subjacente, fornecendo um resultado mais confiável. Claramente, a INFGMN foi capaz de aprender ao longo da vida e devido a isso é aplicável a problemas mais complexos do mundo real com características dinâmicas ou variáveis no tempo. Isso efetivamente aborda o dilema Estabilidade-Plasticidade, frequentemente encontrado no estudo e modelagem neuro-fuzzy.

O segundo experimento foi proposto para avaliar as habilidades da INFGMN para lidar com problemas de classificação e regressão, utilizando três conjuntos de dados da vida real do conjunto de dados UCI, a saber: 1) os dados do *Servo*; 2) os dados do *Wine*; E 3) os dados do *Iris*. Mais uma vez, os resultados foram promissores. A INFGMN foi a rede que melhor desempenho em termos de acurácia e interpretabilidade ao ser comparada com demais modelos *benchmarking*. Os resultados apresentados e discutidos no Capítulo 5 demonstram que a rede INFGMN é capaz de proporcionar desempenhos de modelagem superiores em rela-

ção à acurácia, mantendo um equilíbrio muito bom em termos de interpretabilidade, considerando-se a complexidade da rede.

O terceiro experimento estudou a capacidade de auto reorganização da INFGMN, usando dados de séries temporais financeiras do mundo real com base no índice de mercado *Standard and Poor's* 500 (S& P-500). Realizou-se uma simulação *online* da previsão diária do índice S&P-500. A INFGMN produziu resultados competitivos em termos de acurácia e bons resultados em termos de interpretabilidade, sendo capaz de treinar e prever recursivamente a qualquer momento, sem conhecimento prévio do conjunto de dados completo. Claramente a partir dos resultados, a INFGMN foi capaz de lidar com o dilema Estabilidade-Plasticidade.

Ao adicionar-se à rede INFGMN um mecanismo de imputação, ela passou a ter a capacidade de lidar com problemas que podem apresentar dados faltantes. Os três últimos experimentos apresentados no Capítulo 5 demonstraram que a rede INFGMN foi capaz de obter bons desempenhos em problemas reais de regressão, classificação e previsão de séries temporais univariadas, mesmo para conjuntos de dados faltantes, executando tanto em modo *online* quanto *offline*.

Embora ainda hajam várias áreas para melhoria do modelo proposto (conforme destacado a seguir), pode-se afirmar que os objetivos deste trabalho foram atingidos. A rede INFGMN é uma rede neuro-fuzzy do tipo ML, com capacidade de aprendizagem incremental, que permite por meio de ajustes de parâmetros a obtenção de modelos com uma boa relação custo-benefício entre acurácia e interpretabilidade, mesmo na presença, possivelmente, de vetores de dados com características incompletas. A rede também possui a capacidade de modificar a importância dada aos parâmetros de ajustes no intuito de se obter um modelo inclinado à interpretabilidade ou à acurácia, de modo que o modelo gerado esteja ajustado à natureza do problema sendo abordado. Além disso, por permitir a aprendizagem e imputação incrementais, a rede INFGMN pode ser usada continuamente em aplicações de tempo real, lidando de maneira adequada com o dilema de Estabilidade-Plasticidade, e desta forma não sofrendo com o problema da interferência catastrófica.

6.2 TRABALHOS FUTUROS

Como trabalhos futuros para esta pesquisa, propõe-se:

- Avaliar a possibilidade de alternar as entradas e saídas da rede INFGMN em tempo real, ou seja, para o que agora é uma saída se torne uma entrada ou vice-versa. Uma possível aplicação seria para conjuntos de dados para o problema inverso em tempo real.

Alguns exemplos na literatura podem ser encontrados em Tveito et al. (2010) e Aster, Borchers e Thurber (2011).

- A fim de lidar com as características do problema de detecção de anomalias, propõe-se o uso da rede neuro-fuzzy INFGMN, através da criação de um conjunto de modelos de um dado problema, possivelmente através do uso de Raciocínio Baseado em Casos (CBR, do inglês *Case Based Reasoning*)(RAJA et al., 2011; SHOKOUHI; SKALLE; AAMODT, 2014), que posteriormente, podem ser utilizados para detectar situações anômalas em tempo real (por exemplo, medições de sensores que não estejam em conformidade com os modelos criados através da INFGMN). Essa proposta vai ao encontro de abordagens definidas na literatura, como Jiang, Tseng e Su (2001) e Barbará et al. (2003), que são abordagens de *clustering* que podem detectar anomalias ou *outliers* como elementos que não pertencem, ou que estão próximos, a qualquer cluster identificado previamente a partir de um modelo gerado com base em um conjunto de dados históricos, com a diferença de que com o uso da INFGMN, os modelos podem evoluir e se adaptar a medida que forem sendo usados.

- Embora a estrutura proposta para a rede INFGMN ofereça uma poderosa metodologia de aprendizagem incremental, ainda há várias áreas para melhoria. Por exemplo, o atual algoritmo de aprendizado da INFGMN não leva em conta explicitamente a remoção e o ajuste de conjuntos fuzzy altamente sobrepostos, quando a base de regra é construída/atualizada. Várias direções de pesquisa interessantes para melhorar ainda mais a interpretabilidade da base de regras fuzzy em NFS incrementais foram propostas em Oentaryo, Pasquier e Quek (2011), Angelov (2012) e Lughofer (2013).

- Adicionalmente, para validar o modelo proposto em aplicações reais e/ou também para testar sua adequação em muitas aplicações potenciais que exigem aprendizado incremental e desempenho em tempo real, outros experimentos além dos listados anteriormente poderão ser conduzidos usando a rede INFGMN em aplicações práticas, tais como como robótica e tarefas relacionadas. Além disso, uma melhoria adicional no tempo de execução ainda poderia ser alcançada com uma pequena alteração no algoritmo do INFGMN para que ela atualize apenas apenas as camadas *Fuzzy variable layer* e *Rule layer* sob demanda. Testes adicionais seriam necessários para comprovar o ganho em desempenho.

REFERÊNCIAS

ADLER, Philip et al. Auditing black-box models for indirect influence. *Knowledge and Information Systems*, Springer, v. 54, n. 1, p. 95–122, 2018.

AHMED, Md Manjur; ISA, Nor Ashidi Mat. Evolving output-context fuzzy system for effective rule base. *Expert Systems with Applications*, Elsevier, v. 42, n. 4, p. 1972–1986, 2015.

ALCALÁ, Rafael et al. Hybrid learning models to get the interpretability–accuracy trade-off in fuzzy modeling. *Soft Computing*, Springer, v. 10, n. 9, p. 717–734, 2006.

ALLISON, Paul D. Missing data: Sage university papers series on quantitative applications in the social sciences (07–136). *Thousand Oaks, CA*, 2001.

ALMOMANI, Ammar et al. Evolving fuzzy neural network for phishing emails detection. *Journal of Computer Science*, Science Publications, v. 8, n. 7, p. 1099, 2012.

ALONSO, Jose M; CASTIELLO, Ciro; MENCAR, Corrado. Interpretability of fuzzy systems: Current research trends and prospects. In: *Springer Handbook of Computational Intelligence*. [S.l.]: Springer, 2015. p. 219–237.

ALONSO, José M; MAGDALENA, Luis; GONZÁLEZ-RODRÍGUEZ, Gil. Looking for a good fuzzy system interpretability index: An experimental approach. *International Journal of Approximate Reasoning*, Elsevier, v. 51, n. 1, p. 115–134, 2009.

ANGELOV, Plamen. *Autonomous learning systems: from data streams to knowledge in real-time*. [S.l.]: John Wiley & Sons, 2012.

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

ANKAIAH, Narravula; RAVI, Vadlamani. A novel soft computing hybrid for data imputation. In: *Proceedings of the 7th international conference on data mining (DMIN)*. [S.l.: s.n.], 2011.

ARANDJELOVIC, O; CIPOLLA, R. Incremental learning of temporally coherent gaussian mixture models. In: SOCIETY OF MANUFACTURING ENGINEERS. *Proceedings of the 16th British Machine Vision Conference (BMVC'05)*. [S.l.], 2006. v. 2, p. 759–768.

ASTER, Richard C; BORCHERS, Brian; THURBER, Clifford H. *Parameter estimation and inverse problems*. [S.l.]: Academic Press, 2011.

AZEEM, Mohammad Fazle; HANMANDLU, Madasu; AHMAD, Nesar. Generalization of adaptive neuro-fuzzy inference systems. *IEEE Transactions on Neural Networks*, IEEE, v. 11, n. 6, p. 1332–1346, 2000.

BARBARÁ, Daniel et al. Bootstrapping a data mining intrusion detection system. In: ACM. *Proceedings of the 2003 ACM symposium on Applied computing*. [S.l.], 2003. p. 421–425.

BASSANEZI, RC; BARROS, LC. Tópicos de lógica fuzzy e biomatemática. *Coleção IMECC: textos didáticos*, v. 5, 2006.

BATISTA, Gustavo EAPA; MONARD, Maria Carolina et al. A study of k-nearest neighbour as an imputation method. *HIS*, v. 87, n. 251-260, p. 48, 2002.

BIENENSTOCK, Elie L; COOPER, Leon N; MUNRO, Paul W. *Theory for the development of neuron selectivity: orientation specificity and binocular interaction in visual cortex.* [S.l.], 1981.

BUCKLEY, James J; HAYASHI, Yoichi. Fuzzy neural networks: A survey. *Fuzzy sets and systems*, Elsevier, v. 66, n. 1, p. 1–13, 1994.

BUUREN, Stef Van. *Flexible imputation of missing data*. [S.l.]: Chapman and Hall/-CRC, 2018.

CARPENTER, Gail A.; GROSSBERG, Stephen. The art of adaptive pattern recognition by a self-organizing neural network. *Computer*, IEEE, v. 21, n. 3, p. 77–88, 1988.

CARPENTER, Gail A et al. Fuzzy artmap: A neural network architecture for incremental supervised learning of analog multidimensional maps. *IEEE Transactions on neural networks*, IEEE, v. 3, n. 5, p. 698–713, 1992.

CASILLAS, Jorge. *Interpretability Issues in Fuzzy Modeling*. [S.l.]: Springer Science & Business Media, 2003.

CASILLAS, Jorge et al. *Accuracy Improvements in Linguistic Fuzzy Modeling*. [S.l.]: Springer Science & Business Media, 2003.

CHANG, Fi-John; WANG, Kuo-Wei. A systematical water allocation scheme for drought mitigation. *Journal of Hydrology*, Elsevier, v. 507, p. 124–133, 2013.

CHANG, Pei-Chann; FAN, Chin-Yuan; LIN, Jyun-Jie. Monthly electricity demand forecasting based on a weighted evolving fuzzy neural network approach. *International Journal of Electrical Power & Energy Systems*, Elsevier, v. 33, n. 1, p. 17–27, 2011.

COVER, Thomas; HART, Peter. Nearest neighbor pattern classification. *IEEE transactions on information theory*, IEEE, v. 13, n. 1, p. 21–27, 1967.

DAYHOFF, Judith E; DELEO, James M. Artificial neural networks: opening the black box. *Cancer: Interdisciplinary International Journal of the American Cancer Society*, Wiley Online Library, v. 91, n. S8, p. 1615–1635, 2001.

DEMPSTER, Arthur P; LAIRD, Nan M; RUBIN, Donald B. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, JSTOR, p. 1–38, 1977.

ENGEL, Paulo Martins; HEINEN, Milton Roberto. Incremental learning of multivariate gaussian mixture models. In: SPRINGER. *Brazilian Symposium on Artificial Intelligence*. [S.l.], 2010. p. 82–91.

EVERITT, Brian S. *Finite mixture distributions*. [S.l.]: Wiley Online Library, 1981.

FERRER-TROYANO, Francisco; AGUILAR-RUIZ, Jesus S; RIQUELME, Jose C. Incremental rule learning based on example nearness from numerical data streams. In: ACM. *Proceedings of the 2005 ACM symposium on Applied computing*. [S.l.], 2005. p. 568–572.

FESSANT, Françoise; MIDENET, Sophie. Self-organising map for data imputation and correction in surveys. *Neural Computing & Applications*, Springer, v. 10, n. 4, p. 300–310, 2002.

FISHER, Douglas H. Knowledge acquisition via incremental conceptual clustering. *Machine learning*, Springer, v. 2, n. 2, p. 139–172, 1987.

FLORES, Benito E. A pragmatic view of accuracy measurement in forecasting. *Omega*, Elsevier, v. 14, n. 2, p. 93–98, 1986.

GACTO, María José; ALCALÁ, Rafael; HERRERA, Francisco. Interpretability of linguistic fuzzy rule-based systems: An overview of interpretability measures. *Information Sciences*, Elsevier, v. 181, n. 20, p. 4340–4360, 2011.

GALÁN, Celestino Ordóñez et al. Missing data imputation of questionnaires by means of genetic algorithms with different fitness functions. *Journal of Computational and Applied Mathematics*, Elsevier, v. 311, p. 704–717, 2017.

GAN, Ming-Tao; HANMANDLU, Madasu; TAN, Ai Hui. From a gaussian mixture model to additive fuzzy systems. *IEEE Transactions on Fuzzy Systems*, IEEE, v. 13, n. 3, p. 303–316, 2005.

GARCEZ, Artur S d'Avila; BRODA, Krysia; GABBAY, Dov M. *Neural-symbolic learning systems: foundations and applications*. [S.l.]: Springer Science & Business Media, 2012.

GAUTAM, Chandan; RAVI, Vadlamani. Data imputation via evolutionary computation, clustering and a neural network. *Neurocomputing*, Elsevier, v. 156, p. 134–142, 2015.

GENNARI, John H; LANGLEY, Pat; FISHER, Doug. Models of incremental concept formation. *Artificial intelligence*, Elsevier, v. 40, n. 1-3, p. 11–61, 1989.

GROSSBERG, Stephen. Studies of mind and brain: Neural principles of learning, perception, development. *Cognition, and Motor Control. Reidel, Boston*, 1982.

GUILLAUME, Serge; MAGDALENA, Luis. Expert guided integration of induced knowledge into a fuzzy knowledge base. *Soft Computing-A Fusion of Foundations, Methodologies and Applications*, Springer, v. 10, n. 9, p. 773–784, 2006.

GUPTA, Madan M; QI, J. Theory of t-norms and fuzzy inference methods. *Fuzzy sets and systems*, Elsevier, v. 40, n. 3, p. 431–450, 1991.

HEINEN, Milton Roberto. A connectionist approach for incremental function approximation and on-line tasks. 2011.

HEINEN, Milton Roberto; ENGEL, Paulo Martins; PINTO, Rafael C. Igmn: An incremental gaussian mixture network that learns instantaneously from data flows. *Proc VIII Encontro Nacional de Inteligência Artificial (ENIA2011)*, 2011.

JACOB, Biju Joseph et al. Self-reorganizing tsk fuzzy inference system with bcm theory of meta-plasticity. In: IEEE. *Neural Networks (IJCNN), The 2012 International Joint Conference on*. [S.l.], 2012. p. 1–8.

JAIN, Lakhmi C; KANDEL, Abraham; TEODORESCU, Horia-Nicolai L. *Fuzzy and neuro-fuzzy systems in medicine*. [S.l.]: CRC Press, 2017.

JANG, J-SR. Anfis: adaptive-network-based fuzzy inference system. *IEEE transactions on systems, man, and cybernetics*, IEEE, v. 23, n. 3, p. 665–685, 1993.

JANG, Jyh-Shing Roger; SUN, Chuen-Tsai. Neuro-fuzzy modeling and control. *Proceedings of the IEEE*, IEEE, v. 83, n. 3, p. 378–406, 1995.

JASSBI, JJ et al. A comparison of mandani and sugeno inference systems for a space fault detection application. In: IEEE. *Automation Congress, 2006. WAC'06. World*. [S.l.], 2006. p. 1–8.

JASSBI, Javad J et al. Transformation of a mamdani fis to first order sugeno fis. In: *FUZZ-IEEE*. [S.l.: s.n.], 2007. p. 1–6.

JIANG, Mon-Fong; TSENG, Shian-Shyong; SU, Chih-Ming. Two-phase clustering process for outliers detection. *Pattern recognition letters*, Elsevier, v. 22, n. 6, p. 691–700, 2001.

JUANG, Chia-Feng; LIN, Chin-Teng. An online self-constructing neural fuzzy inference network and its applications. *IEEE Transactions on Fuzzy Systems*, IEEE, v. 6, n. 1, p. 12–32, 1998.

KAR, Samarjit; DAS, Sujit; GHOSH, Pijush Kanti. Applications of neuro fuzzy systems: A brief review and future outline. *Applied Soft Computing*, Elsevier, v. 15, p. 243–259, 2014.

KARABOGA, Dervis; KAYA, Ebubekir. Adaptive network based fuzzy inference system (anfis) training approaches: a comprehensive survey. *Artificial Intelligence Review*, Springer, p. 1–31, 2018.

KARIM, Awudu; ZHOU, Shangbo. X-trepan: a multi class regression and adapted extraction of comprehensible decision tree in artificial neural networks. *arXiv preprint arXiv:1508.07551*, 2015.

KASABOV, Nikola. Evolving fuzzy neural networks for supervised/unsupervised online knowledge-based learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, IEEE, v. 31, n. 6, p. 902–918, 2001.

KASABOV, Nikola K. The ecos framework and the eco learning method for evolving connectionist systems. *JACIII*, v. 2, n. 6, p. 195–202, 1998.

KASABOV, Nikola K; SONG, Qun. Denfis: dynamic evolving neural-fuzzy inference system and its application for time-series prediction. *IEEE Transactions on fuzzy systems*, IEEE, v. 10, n. 2, p. 144–154, 2002.

KIM, Jaesoo; KASABOV, N. Hyfis: adaptive neuro-fuzzy inference systems and their application to nonlinear dynamical systems. *Neural Networks*, Elsevier, v. 12, n. 9, p. 1301–1319, 1999.

KLIR, George; YUAN, Bo. *Fuzzy sets and fuzzy logic*. [S.l.]: Prentice hall New Jersey, 1995.

KOSKO, Bart. Neural networks and fuzzy systems: a dynamical systems approach to machine intelligence/book and disk. *Vol. 1Prentice hall*, 1992.

KOTHAMASU, Ranganath; HUANG, Samuel H. Adaptive mamdani fuzzy model for condition-based maintenance. *Fuzzy sets and Systems*, Elsevier, v. 158, n. 24, p. 2715–2733, 2007.

KRISTAN, Matej; SKOCAJ, Danijel; LEONARDIS, Aleš. Incremental learning with gaussian mixture models. In: *Proc. Computer Vision Winter Workshop, Moravske Toplice, Slovenia*. [S.l.: s.n.], 2008. p. 25–32.

LAMIREL, Jean-Charles et al. A new incremental growing neural gas algorithm based on clusters labeling maximization: application to clustering of heterogeneous textual data. In: SPRINGER. *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*. [S.l.], 2010. p. 139–148.

LAPA, Krystian; CPALKA, Krzysztof; RUTKOWSKI, Leszek. New aspects of interpretability of fuzzy systems for nonlinear modeling. In: SPRINGER. *Advances in Data Analysis with Computational Intelligence Methods*. [S.l.], 2018. p. 225–264.

LARSEN, P Martin. Industrial applications of fuzzy logic control. *International Journal of Man-Machine Studies*, Elsevier, v. 12, n. 1, p. 3–10, 1980.

LAUGEL, Thibault et al. Comparison-based inverse classification for interpretability in machine learning. In: SPRINGER. *International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*. [S.l.], 2018. p. 100–111.

LAVIOLETTE, Michael et al. A probabilistic and statistical view of fuzzy methods. *Technometrics*, Taylor & Francis Group, v. 37, n. 3, p. 249–261, 1995.

LEE, Kwang Hyung. *First course on fuzzy theory and applications*. [S.l.]: Springer Science & Business Media, 2004.

LEEKWIJCK, Werner Van; KERRE, Etienne E. Defuzzification: criteria and classification. *Fuzzy sets and systems*, Elsevier, v. 108, n. 2, p. 159–178, 1999.

LEITE, Daniel; GOMIDE, Fernando. Evolving linguistic fuzzy models from data streams. In: *Combining Experimentation and Theory*. [S.l.]: Springer, 2012. p. 209–223.

LEITE, Daniel et al. Fuzzy granular evolving modeling for time series prediction. In: IEEE. *Fuzzy Systems (FUZZ), 2011 IEEE International Conference on*. [S.l.], 2011. p. 2794–2801.

LI, Dan et al. Towards missing data imputation: a study of fuzzy k-means clustering method. In: SPRINGER. *Rough sets and current trends in computing*. [S.l.], 2004. v. 3066, p. 573–579.

LICHMAN, M. *UCI Machine Learning Repository*. 2013. Disponível em: <http://archive.ics.uci.edu/ml>.

LIN, Cheng-Jian; LIN, Chin-Teng. An art-based fuzzy adaptive learning control network. *IEEE Transactions on Fuzzy Systems*, IEEE, v. 5, n. 4, p. 477–496, 1997.

LIN, Chin-Teng; LEE, CS. *Neural fuzzy systems: a neuro-fuzzy synergism to intelligent systems*. [S.l.]: Prentice-Hall, Inc., 1996.

LITTLE, Roderick JA; RUBIN, Donald B. *Statistical analysis with missing data*. [S.l.]: John Wiley & Sons, 2014.

LIU, Fuyu; GENG, Hongli; ZHANG, Yan-Qing. Interactive fuzzy interval reasoning for smart web shopping. *Applied Soft Computing*, Elsevier, v. 5, n. 4, p. 433–439, 2005.

LIU, Zhun-ga et al. Adaptive imputation of missing values for incomplete pattern classification. *Pattern Recognition*, Elsevier, v. 52, p. 85–95, 2016.

LIU, Zhun-Ga et al. A new incomplete pattern classification method based on evidential reasoning. *IEEE transactions on cybernetics*, IEEE, v. 45, n. 4, p. 635–646, 2015.

LU, Hongjun; SETIONO, Rudy; LIU, Huan. Neurorule: A connectionist approach to data mining. *arXiv preprint arXiv:1701.01358*, 2017.

LUENGO, Julián; SÁEZ, José A; HERRERA, Francisco. Missing data imputation for fuzzy rule-based classification systems. *Soft computing*, Springer, v. 16, n. 5, p. 863–881, 2012.

LUGHOFER, Edwin. On-line assurance of interpretability criteria in evolving fuzzy systems–achievements, new concepts and open issues. *Information Sciences*, Elsevier, v. 251, p. 22–46, 2013.

MAMDANI, Ebrahim H. Application of fuzzy algorithms for control of simple dynamic plant. In: IET. *Proceedings of the Institution of Electrical Engineers*. [S.l.], 1974. v. 121, n. 12, p. 1585–1588.

MAMDANI, Ebrahim H. Application of fuzzy logic to approximate reasoning using linguistic synthesis. *IEEE transactions on computers*, IEEE, v. 100, n. 12, p. 1182–1191, 1977.

MAMDANI, Ebrahim H; ASSILIAN, Sedrak. An experiment in linguistic synthesis with a fuzzy logic controller. *International journal of man-machine studies*, Elsevier, v. 7, n. 1, p. 1–13, 1975.

MCLACHLAN, Geoffrey; KRISHNAN, Thriyambakam. *The EM algorithm and extensions*. [S.l.]: John Wiley & Sons, 2007.

MERMILLOD, Martial; BUGAISKA, Aurélia; BONIN, Patrick. The stability-plasticity dilemma: Investigating the continuum from catastrophic forgetting to age-limited learning effects. *Frontiers in psychology*, Frontiers, v. 4, p. 504, 2013.

MORITZ, Steffen; BARTZ-BEIELSTEIN, Thomas. imputets: Time series missing value imputation. *R package version 0.4*, 2015.

OCAK, Hasan; ERTUNC, Huseyin Metin. Prediction of fetal state from the cardiotocogram recordings using adaptive neuro-fuzzy inference systems. *Neural Computing and Applications*, Springer, v. 23, n. 6, p. 1583–1589, 2013.

OENTARYO, Richard J et al. Online probabilistic learning for fuzzy inference system. *Expert Systems with Applications*, Elsevier, v. 41, n. 11, p. 5082–5096, 2014.

OENTARYO, Richard J; PASQUIER, Michel; QUEK, Chai. Rfcmac: A novel reduced localized neuro-fuzzy system approach to knowledge extraction. *Expert Systems with Applications*, Elsevier, v. 38, n. 10, p. 12066–12084, 2011.

OLDEN, Julian D; JACKSON, Donald A. Illuminating the "black box": a randomization approach for understanding variable contributions in artificial neural networks. *Ecological modelling*, Elsevier, v. 154, n. 1-2, p. 135–150, 2002.

OURSTON, Dirk; MOONEY, Raymond J. Theory refinement combining analytical and empirical methods. *Artificial Intelligence*, Elsevier, v. 66, n. 2, p. 273–309, 1994.

PAZZANI, Michael; KIBLER, Dennis. The utility of knowledge in inductive learning. *Machine learning*, Springer, v. 9, n. 1, p. 57–94, 1992.

PEDRYCZ, Witold. Neurocomputations in relational systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, IEEE, v. 13, n. 3, p. 289–297, 1991.

PEDRYCZ, Witold; GOMIDE, Fernando. *Fuzzy systems engineering: toward human-centric computing*. [S.l.]: John Wiley & Sons, 2007.

PETROVIC-LAZAREVIC, Sonja; ZHANG, Jian Ying. Neuro-fuzzy models and tobacco control. In: SPRINGER. *Proceedings of the European computing conference*. [S.l.], 2009. p. 25–31.

PINTO, Rafael C; ENGEL, Paulo M; HEINEN, Milton R. One-shot learning in the road sign problem. In: IEEE. *The 2012 International Joint Conference on Neural Networks (IJCNN)*. [S.l.], 2012. p. 1–6.

PRATAMA, Mahardhika; PEDRYCZ, Witold; WEBB, Geoffrey I. An incremental construction of deep neuro fuzzy system for continual learning of non-stationary data streams. *arXiv preprint arXiv:1808.08517*, 2018.

RAJA, Hamayun Zafar et al. Case-based reasoning: predicting real-time drilling problems and improving drilling performance. In: SOCIETY OF PETROLEUM ENGINEERS. *SPE Middle East Oil and Gas Show and Conference*. [S.l.], 2011.

RAJAB, Sharifa; SHARMA, Vinod. A review on the applications of neuro-fuzzy systems in business. *Artificial Intelligence Review*, Springer, v. 49, n. 4, p. 481–510, 2018.

RAVI, Vadlamani; KRISHNA, Mannepalli. A new online data imputation method based on general regression auto associative neural network. *Neurocomputing*, Elsevier, v. 138, p. 106–113, 2014.

REYNOLDS, Douglas. Gaussian mixture models. *Encyclopedia of Biometrics*, Springer, p. 827–832, 2015.

RIID, Andri; RÜSTERN, Ennu. Adaptability, interpretability and rule weights in fuzzy rule-based systems. *Information Sciences*, Elsevier, v. 257, p. 301–312, 2014.

SCHAFER, Joseph L. *Analysis of incomplete multivariate data*. [S.l.]: CRC press, 1997.

SETIONO, Rudy; LIU, Huan. Understanding neural networks via rule extraction. In: *IJCAI*. [S.l.: s.n.], 1995. v. 1, p. 480–485.

SHIHABUDHEEN, KV; PILLAI, GN. Recent advances in neuro-fuzzy system: A survey. *Knowledge-Based Systems*, Elsevier, v. 152, p. 136–162, 2018.

SHOKOUHI, Samad Valipour; SKALLE, Pål; AAMODT, Agnar. An overview of case-based reasoning applications in drilling engineering. *Artificial Intelligence Review*, Springer, p. 1–13, 2014.

SHWARTZ-ZIV, Ravid; TISHBY, Naftali. Opening the black box of deep neural networks via information. *arXiv preprint arXiv:1703.00810*, 2017.

TAKAGI, Tomohiro; SUGENO, Michio. Fuzzy identification of systems and its applications to modeling and control. *IEEE transactions on systems, man, and cybernetics*, IEEE, n. 1, p. 116–132, 1985.

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

TAN, Pang-Ning et al. *Introduction to data mining*. [S.l.]: Pearson Education India, 2006.

TANSCHEIT, Ricardo. Sistemas fuzzy. *Departamento de Engenharia Elétrica, Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro*, 2004.

TOWELL, Geoffrey G; SHAVLIK, Jude W. Knowledge-based artificial neural networks. *Artificial intelligence*, Elsevier, v. 70, n. 1, p. 119–165, 1994.

TUNG, Sau Wai; QUEK, Chai; GUAN, Cuntai. Safin: A self-adaptive fuzzy inference network. *IEEE Transactions on Neural Networks*, IEEE, v. 22, n. 12, p. 1928–1940, 2011.

TUNG, Whye Loon; QUEK, Chai. A mamdani-takagi-sugeno based linguistic neural-fuzzy inference system for improved interpretability-accuracy representation. In: IEEE. *Fuzzy Systems, 2009. FUZZ-IEEE 2009. IEEE International Conference on*. [S.l.], 2009. p. 367–372.

TUNG, Whye Loon; QUEK, Chai. efsm—a novel online neural-fuzzy semantic memory model. *IEEE Transactions on Neural Networks*, IEEE, v. 21, n. 1, p. 136–157, 2010.

TVEITO, Aslak et al. Parameter estimation and inverse problems. In: *Elements of Scientific Computing*. [S.l.]: Springer, 2010. p. 411–421.

UMAMAHESWARI, J; SULTANA, Jabeen; FATIMA, Ruhi. An efficient method to diagnose the treatment of breast cancer using multi-classifiers. *International Journal of Computer Science and Information Security*, LJS Publishing, v. 14, n. 4, p. 72, 2016.

WEISSTEIN, Eric W. Normal distribution. Wolfram Research, Inc., 2002.

WOLBERG, William H; STREET, W Nick; MANGASARIAN, OL. Machine learning techniques to diagnose breast cancer from image-processed nuclear features of fine needle aspirates. *Cancer letters*, Elsevier, v. 77, n. 2, p. 163–171, 1994.

ZADEH, Lotfi A. Information and control. *Fuzzy sets*, v. 8, n. 3, p. 338–353, 1965.

ZADEH, Lotfi A. Outline of a new approach to the analysis of complex systems and decision processes. *IEEE Transactions on systems, Man, and Cybernetics*, IEEE, n. 1, p. 28–44, 1973.

ZADEH, Lotfi A. The concept of a linguistic variable and its application to approximate reasoning—i. *Information sciences*, Elsevier, v. 8, n. 3, p. 199–249, 1975.

APÊNDICES

Apêndice - A: Implementação do modelo INFGMN (Matlab)

```
classdef INFGMN < handle

% inicialização dos parâmetros omitida ...

    % Método executado em modo aprendizagem
    function self = INFGMNLearning(self, X)
        X = double(X);
        if self.normalize
            X = mapminmax('apply', X', self.proportion)';
        end
        N = size(X,1);
        for i = 1:N
            x = X(i,:);
            if (any(isnan(x)))
                x = self.infgmni(x);
            end
            self.computeLikelihood(x);
            if (~self.hasAcceptableDistribution())
                self.createComponent(x);
            end
            self.computePosterior();
            self.updateComponents(x);
            self.sampleSize = self.sampleSize + 1;
            self.removeSpurious();
        end
        self.needsFisUpdate = true;
    end

    function computeLikelihood(self, x)
        if self.nc > 0
            [self.loglike, self.mahalaD] =  ...
               self.logmvnpdf(x, self.means, self.covs);
        end
    end

    function [loglike, mahalaD] = logmvnpdf(~, x, means, covs)
        [n,d] = size(x);
        k = size(means,1);
        loglike = zeros(n, k);
        mahalaD = zeros(n, k);
        logDetCov = zeros(1, k);

        for j = 1:k
            L = sqrt(covs(:,:,j));
            if  any(L < eps(max(L)) * d)
                error('Ill Conditioned Covariance.');
            end
            logDetCov(j) = sum(log(covs(:,:,j)));

            Xcentered = bsxfun(@minus, x, means(j,:));
            xRinv = bsxfun(@times, Xcentered , (1 ./ L));

            mahalaD(:,j) = sum(xRinv.^2, 2);
            loglike(:,j) = - 0.5 * ...
                (mahalaD(:,j) + logDetCov(j) + d * log(2 * pi));
```

```
56          end
57      end
58
59      function computePosterior(self)
60          logPrior = log(self.priors);
61          self.post = bsxfun(@plus, self.loglike, logPrior);
62          maxll = max(self.post, [], 2);
63          self.post = exp(bsxfun(@minus, self.post, maxll));
64          density = sum(self.post, 2);
65          self.post = bsxfun(@rdivide, self.post, density);
66          logpdf = log(density) + maxll;
67          self.dataLikelihood = self.dataLikelihood + sum(logpdf);
68      end
69
70      function h = hasAcceptableDistribution(self)
71          for j = 1:self.nc
72              if (self.mahalaD(j) <= self.maxDist)
73                  h = true;
74                  return;
75              end
76          end
77          h = false;
78      end
79
80      function createComponent(self, x)
81          self.nc = self.nc + 1;
82          self.means(self.nc,:) = x;
83          self.sps(1,self.nc) = 1;
84          self.st(1,self.nc) = 0;
85          self.sp(1,self.nc) = 0;
86          self.updatePriors();
87          if self.nc > 1
89              self.covs(:,:,self.nc) = mean(self.covs, 3);
90          else
91              self.covs(:,:,self.nc) = self.sigma;
92          end
93          [self.loglike(1,self.nc), self.mahalaD(1, self.nc)] = ...
94             self.logmvnpdf(x, ...
95             self.means(self.nc,:), self.covs(:,:,self.nc));
96      end
97
98      function updatePriors(self)
99          if ~self.uniform
100             self.priors = self.sps ./ sum(self.sps);
101         else
102             self.priors = ones(1,self.nc) ./ self.nc;
103         end
104     end
105
106     function updateComponents(self, x)
107         self.sps = self.sps + self.post;
108         self.st = self.st + 1.0;
109         self.sp = self.sp + self.post;
110         w = (self.post ./ self.sps) * self.beta;
111         for j = 1:self.nc
112             Xcentered = x - self.means(j,:);
113             deltaMU = w(j) * Xcentered;
114             self.means(j,:) = self.means(j,:) + deltaMU;
115             XcenteredNEW = x - self.means(j,:);
116             self.covs(:,:,j) = self.covs(:,:,j) - ...
117                 deltaMU.^2 + w(j) * (XcenteredNEW.^2 - ...
118                 self.covs(:,:,j)) + self.regValue;
119
```

```
120         end
121         self.updatePriors();
122     end
123
124     function removeSpurious(self)
125         for i = self.nc:-1:1
126             if (self.st(i) == self.tmax)
127                 if self.sp(i) < self.spmin
128                     self.nc = self.nc - 1;
129                     self.priors(i) = [];
130                     self.means(i,:) = [];
131                     self.covs(:,:,i) = [];
132                     self.st(i) = [];
133                     self.sps(i) = [];
134                     self.sp(i) = [];
135                     self.post(i) = [];
136                     self.mahalaD(i) = [];
137                     self.loglike(i) = [];
138                 else
139                     self.st(i) = 0;
140                     self.sp(i) = 0;
141                 end
142             end
143         end
144     end
145
146     % Método executado em modo recalling
147     function y = INFGMNrecalling(self, x)
148         varNames = x.Properties.VarNames;
149         inputIndexes = cellfun( @(var)find(strcmp(...
150             self.varNames, var)), varNames, 'UniformOutput', 1);
151         outputIndexes = 1:length(self.varNames);
152         outputIndexes(inputIndexes) = [];
153         self.updateFuzzyLayer(inputIndexes, outputIndexes);
154         if self.normalize
155             y = ones(size(x, 1), length(self.varNames));
156             y(:, nputIndexes) = double(x);
157             y = mapminmax('apply', y', self.proportion)';
158             if any(any(isnan(y(:, inputIndexes))))
159                 x_imp = nan(size(x, 1), length(self.varNames));
160                 x_imp(:, inputIndexes) = y(:, inputIndexes);
161                 x_imp = self.internal_imputation(x_imp);
162                 y(:, inputIndexes) = x_imp(:, inputIndexes);
163             end
164             y(:, outputIndexes) = evalfis(y(:, inputIndexes), self.fis);
165             y = mapminmax('reverse', y', self.proportion)';
166             y = y(:, outputIndexes);
167         else
168             y = double(x);
169             if any(any(isnan(y)))
170                 x_imp = nan(size(x, 1), length(self.varNames));
171                 x_imp(:, inputIndexes) = y;
172                 x_imp = self.infgmni(x_imp);
173                 y = x_imp(:, inputIndexes);
174             end
175             y = evalfis(y, self.fis);
176         end
177         y = mat2dataset(y , 'VarNames', self.varNames(outputIndexes));
178     end
179
180     % Método de imputação
181     function x = infgmni(self, x)
182         indexes = isnan(x);
```

```
183         if all(indexes)
184             error(['Each trainning input pattern must'...
185                 'have at least 1 non nan column.']);
186         end
187         if any(any(indexes))
188             N = size(x, 1);
189             for j = 1:N
190                 pajs = zeros(self.nc, 1);
191                 curIndexes = indexes(j, :);
192                 xm = zeros(self.nc, sum(curIndexes));
194                 for i = 1:self.nc
195                     meanA = self.means(i, ~curIndexes);
196                     meanB = self.means(i, curIndexes);
197                     covA = self.covs(1, ~curIndexes, i);
198                     xm(i,:) = meanB;
199                     ll = self.logmvnpdf(x(j, ~curIndexes), meanA, covA);
200                     pajs(i) = exp(ll) * self.priors(i);
201                 end
202                 pajs = pajs ./ sum(pajs);
203                 x(j, curIndexes) = sum(bsxfun(@times, xm, pajs));
204                 x(j, x(j, :) > self.ranges(2, :)) = ...
205                     self.ranges(2, x(j, :) > self.ranges(2, :));
206                 x(j, x(j, :) < self.ranges(1, :)) = ...
207                     self.ranges(1, x(j, :) < self.ranges(1, :));
208             end
209         end
210     end
212 end %end methods
213
214 methods (Access = private)
215
216     function updateFuzzyLayer(self, inputIndexes, outputIndexes)
217         self.needsFisUpdate = self.needsFisUpdate || ...
218             isempty(self.fis.output) || ...
219             ~isempty(setdiff([self.fis.output(:).name], ...
220             self.varNames(outputIndexes))) || ...
221             ~isempty(setdiff(self.varNames(outputIndexes),...
222             [self.fis.output(:).name]));
223         if (self.needsFisUpdate)
224             self.cleanFis();
225             self.createFuzzyVariables(inputIndexes, outputIndexes);
226             self.createRules();
227         end
228         self.needsFisUpdate = false;
229     end
230
231     function createFuzzyVariables(self, inputIndexes, outputIndexes)
232         for i = 1:length(inputIndexes)
233             range = self.ranges(:, inputIndexes(i));
234             mus = self.means(:, inputIndexes(i));
235             spreads = self.covs(:, inputIndexes(i), :) .^ self.spread;
236             self.fis = addvar(self.fis, 'input', ...
237                 self.varNames(inputIndexes(i)), [range(1), range(2)]);
238             for j = 1:self.nc
239                 mu = mus(j);
240                 spd = spreads(j);
241                 params = self.toMfParams('trimf', mu, spd, self.inMfType);
242                 if self.fis.input(i).range(1) >  params(1)
243                     self.fis.input(i).range(1) = params(1);
244                 else
245                     if self.fis.input(i).range(2) <  params(3)
246                         self.fis.input(i).range(2) = params(3);
247                     end
```

```
248                 end
249                 self.addInputMf(i, mu, spd, ['MF' num2str(j)])
250             end
251         end
252         for i = 1:length(outputIndexes)
253             range = self.ranges(:, outputIndexes(i));
254             mus = self.means(:, outputIndexes(i));
255             spreads = self.covs(:, outputIndexes(i), :)  .^ self.spread;
256             self.fis = addvar(self.fis, 'output', ...
257                 self.varNames(outputIndexes(i)), [range(1), range(2)]);
258             for j = 1:self.nc
259                 mu = mus(j);
260                 spd = spreads(j);
261                 params = self.toMfParams('trimf', mu, spd, self.outMfType);
262                 if self.fis.output(i).range(1) >  params(1)
263                     self.fis.output(i).range(1) = params(1);
264                 else
265                     if self.fis.output(i).range(2) <  params(3)
266                         self.fis.output(i).range(2) = params(3);
267                     end
268                 end
269                 self.addOutputMf(i, mu, spd, ['MF' num2str(j)])
270             end
271         end
272     end
273
274     function createRules(self)
275         numVars = length(self.varNames);
276         mfsTemplate = ones(1, numVars);
277         ruleList = ones(self.nc, numVars + 2);
278         for i = 1:self.nc
279             ruleList(i, :) = [(mfsTemplate .* i) self.priors(i) 1];
280         end
281         self.fis = addrule(self.fis, ruleList);
282     end
283
284     function cleanFis(self)
285         self.fis.input = [];
286         self.fis.output = [];
287         self.fis.rule = [];
288     end
289
290     function addInputMf(self, index, mean, spread, mfName)
291         params = self.toMfParams(self.inMfType, mean, ...
292             spread, self.defaultFISOptions.inMfType);
293         self.fis = addmf(self.fis, 'input', ...
294             index, mfName, self.inMfType, params);
295     end
296
297     function addOutputMf(self, index, mean, spread, mfName)
298         params = self.toMfParams(self.outMfType, mean, ...
299             spread, self.defaultFISOptions.outMfType);
300         self.fis = addmf(self.fis, 'output', ...
301             index, mfName, self.outMfType, params);
302     end
303
304     function params = toMfParams(~, mfType, mean, spread, defaultMfType)
305         params = [spread mean];
306         if ~strcmp(mfType, defaultMfType)
307             params = mf2mf(params, defaultMfType, mfType);
308         end
309     end
310 end
```