Modelos Locais Crescentes para Identificação de Sistemas com Outliers: Um Estudo Comparativo

Jéssyca A Bessa
Nuven-IFCE
Federal Institute of Ceará
Maranguape, Ceará, Brazil
bessa.jessyca@ifce.edu.br

Guilherme A Barreto
Dep. of Teleinformatics Engineering
Federal University of Ceará
Fortaleza, Ceará, Brazil
gbarreto@ufc.br

Darlan A Barroso
Dep. of Teleinformatics Engineering
Universidade Federal do Ceará
Fortaleza-CE, Brazil
darlan.engemec@alu.ufc.br

***Abstract*—In this paper, we compare evolving variants of the growing local models (RAN and INC-LLM) for recursive dynamical system identification. The proposed model has the following features: growing online structure, fast recursive updating rules, better memory use (no storage of covariance matrices is required) and outlier-robustness. In this regard, efficiency in performance and simplicity of implementation are the essential qualities of the proposed approach. The proposed outlier-robust versions of RAN and INC-LLM results from a synergistic amalgamation of two simple but powerful ideas. For this purpose, we combine the neuron insertion strategy and outlier-robust LMS-like parameter estimation rules. A comprehensive evaluation involving two benchmarking data sets corroborates the proposed approach's superior predictive performance in outlier-contaminated scenarios compared to original models.**

***Index Terms*—Growing Local models, System identification, Least Mean Estimate.**

I. Introdução

Uma tarefa muito importante em um processo de identificação é a de selecionar uma estrutura de modelo [1]. De maneira sucinta, estas estruturas podem ser classificadas em duas categorias: modelos globais ou modelos locais. Modelos globais são projetados para caracterizar todo o domínio do problema usando apenas uma única estrutura matemática. Estas estruturas podem ser polinomiais, racionais, e até neurais (*feedforward* ou recorrentes, uni- ou multicamadas, etc.) [2].

Modelos locais, por outro lado, particionam o domínio do problema em regiões menores de modo que um modelo estruturalmente mais simples seja adotado para cada região do espaço como os modelos baseados em TS, RBF e SOM [3], [4]. As saídas dos vários submodelos podem ser agregadas ou utilizadas individualmente para prever a saída do modelo. Para este trabalho, tem-se um particular interesse em duas abordagens locais: *local linear mapping* (LLM) e *radial basis functions networks* (RBFN). Tal interesse surge principalmente em razão da simplicidade dos modelos originais, que utilizam técnicas de quantização vetorial do espaço de entrada para encontrar as subregiões onde os modelos locais serão construídos. Para estimar os parâmetros dos submodelos é comum usar as regras LMS (*least mean squares*) e as regras RLS (*recursive least squares*) [5].

Uma desvantagem dos modelos locais supracitados é a necessidade de especificar o número de submodelos antes do treinamento. Trabalhos tais como [6], [7] e [8] abordam esta limitação de modelos locais. Uma abordagem para mitigar esta desvantagem envolve a utilização de modelos crescentes (*growing models*) nos quais os submodelos são progressivamente inseridos até que um determinado requisito de desempenho baseado em erro ou quantidade máxima permitida de submodelos seja alcançado. Esta abordagem também é adequada para lidar com a não estacionariedade, uma vez que a dinâmica do sistema pode mudar ao longo do tempo e o modelo pode adaptar sua estrutura para lidar com as novas demandas da tarefa [9].

Um modelo crescente baseado em TS, no qual novas regras *fuzzy* são inseridas, é proposto em [10]. Modelos baseados em SOM são introduzidos em [11], [12], enquanto modelos baseados na rede RBF são propostos em [13], [14] e [15]. Um modelo de crescimento baseado em LMN foi recentemente introduzido em [16] e aplicado à previsão de séries temporais.

Tem-se que destacar que um outro requisito desejável para a identificação do sistema é a robustez a *outliers*, aqui entendido em um sentido amplo como uma medição anômala. Em cenários de identificação de sistema *online*, *outliers* devem ser adequadamente tratados por métodos robustos de estimação de parâmetros [17]. Independente da estrutura do modelo (global ou local) é desejável que a estrutura seja capaz de lidar com estas amostras discrepantes, conferindo-lhe robustez em cenários não-ideais de uso.

Um dos poucos trabalhos que abordam diretamente o efeito da presença de *outliers* nos dados de estimação é o de [18], com os modelos LLM, RBF e LMN de tamanho fixo utilizando modificação das regras de aprendizado com o auxílio do arcabouço de estatística robusta conhecido como *estimação-M* [19].

Com esse intuito, realizamos uma comparação entre dois modelos lineares locais oriundos da área de redes neurais artificiais e modificamos suas regras de aprendizado com

o auxílio do arcabouço de estatística robusta conhecido como *estimação-M* [19]. Experimentos computacionais são realizados em dois conjuntos de teste (*benchmarkig*) com o objetivo de validar a hipótese de que o uso de estimadores-M aumenta consideravelmente a robustez dos modelos avaliados.

As seções abaixo estão organizadas da seguinte maneira: Na Seção II, apresenta-se o problema de identificação de sistemas. Na Seção III os modelos locais avaliados neste artigo são descritos. O arcabouço teórico da estimação-M é descrito na Seção IV. A metodologia e os conjuntos de dados utilizados são descritos na Seção V. Resultados são discutidos na Seção VI. As conclusões são realizadas na Seção VII.

II. O problema de Identificação de Sistemas

Para todos os modelos a serem apresentados nesta e nas próximas sessões,a dinâmica dos sistemas de interesse é considerada adequadamente descrita por um modelo NARX cuja formulação geral obedece à seguinte equação [1]:

$$y(t) = G\left(y(t-1),\ \cdots, y(t-L_y); u(t-1),\ \cdots, u(t-L_u)\right). \tag{1}$$

em que $L_u \geq 1$ e $L_y \geq 1$ denotam as ordens de memória de entrada e saída, respectivamente, e $n(t)$ é uma variável aleatória do tipo AWGN [1]. Qualquer que seja a escolha, o t-ésimo vetor de regressão de entrada $\mathbf{x}(t) \in \mathbb{R}^{L_u+L_y}$ é montado pela concatenação de L_y saídas observadas passadas e L_u entradas anteriores, tudo isso em um único vetor de regressores da forma

$$\mathbf{x}(t) = [y(t-1), \cdots, y(t-L_y),\ u(t-1), \cdots, u(t-L_u)]^T \tag{2}$$

Dessa forma, os dados de de entrada-saída observados devem ser utilizados para construir um modelo aproximado $\hat{G}(\cdot)$ para a função $G(\cdot)$, que é expresso por

$$\hat{y}(t) = \hat{G}\left(y(t-1),\ \cdots,\ y(t-L_y),\ \ u(t-1),\ \cdots,\ u(t-L_u)\right) \tag{3}$$

O erro no instante t é dado por

$$e(t) = y(t) - \hat{y}(t), \tag{4}$$

sendo chamado de erro de estimação ou resíduo quando são usados os dados de treinamento; ou erro de predição quando forem usados os dados de teste.

III. Modelos Locais Crescentes

Em [20], aprender incrementalmente significa que os pesos de um sistema de aprendizagem são atualizados sem degradar o conhecimento anterior. Assim, três questões precisam ser consideradas:

1) um cenário de memória limitada de tal forma que, após uma nova amostra ser aprendida, ela é descartada e não pode ser reutilizada;
2) as distribuições de entrada e saída são desconhecidas;
3) essas distribuições podem variar no tempo [21].

Em suma, busca-se um modelo que: (i) tenha estrutura online crescente (ou seja, não há necessidade de definir o número de unidades ocultas de antemão), (ii) possua regra de atualização recursiva rápida; ou seja, com poucas operações matemáticas, (iii) com poucos requisitos de memória, e (iv) resiliente a *outliers*.

Nesse sentido, tanto a acurácia no desempenho quanto a simplicidade de implementação são os principais pontos a serem considerados. O estado da arte em algoritmos de estimação recursiva robusta a *outliers* voltados para identificação de sistemas possuem alguns dos recursos listados acima, mas não todos. A maioria deles utiliza o algoritmo RLS, que requer muito mais espaço de memória do que o algoritmo LMS devido ao uso de matrizes de covariância, uma para cada neurônio da rede.

Dito isso, nesta seção, daremos destaque a dois modelos que satisfazem boa parte dos requisitos desejados e já estão bem consolidados na literatura:

1) Mapeamento Linear Local Incremental (INC-LLM) [22] que é uma versão crescente do modelo LLM (*Local Linear Mapping*).
2) Rede de Alocação de Recursos (*Resource Allocating Network* (RAN)) [23] que pode ser considerado um modelo crescente baseado na Rede de Função de Base Radial (RBFN).

A. Mapeamento Linear Local Incremental (INC-LLM)

Abordagens baseadas no modelo LLM geralmente assumem um número fixo de centros que são distribuídos no espaço de entrada por algum método de quantização vetorial. Um modelo de rede incremental para aprendizagem supervisionada foi proposto em [24]. Nessa abordagem, as informações de erro obtidas durante o treinamento são utilizadas para determinar quando e onde inserir novas unidades. A teoria da rede GNG (*growing neural gas*) [22] foi utilizada para dar um caráter incremental ao modelo LLM.

O modelo GNG descrito em [22] não é supervisionado e insere novas unidades para reduzir a distância entre o posicionamento dos vetores protótipos e o vetor de entrada. Por esta razão, o erro de distorção é acumulado localmente e novas unidades são inseridas perto da unidade com maior erro acumulado.

Isto posto, o modelo GNG foi descrito em [22]. Esse modelo é um algoritmo de quantização vetorial, mas quando usado com a regra de Hebb é capaz de preservar a topologia dos dados de entrada de modo similar à rede SOM.

Ao analisar [22], nota-se que o modelo GNG foi criado com o intuito de ser não supervisionado e insere novas unidades para reduzir o erro médio entre os vetores de entrada e os vetores protótipos. [24] descreveu como esse princípio poderia ser utilizado para o aprendizado supervisionado. Primeiro tem-se que definir qual é a saída

[1] *Additive white Gaussian noise*. Ou seja, ruído branco (sem memória) e Gaussiano de média zero e variância σ_n^2.

da rede (que não era necessário para o aprendizado não supervisionado). Então, faz-se uso da diferença entre a saída real e saída desejada para orientar as inserções de novas unidades. O problema em questão é aproximar uma função $G(\cdot) : \mathbb{R}^{L_u+L_y} \rightarrow \mathbb{R}^{n_y}$, dado pares de entrada-saída, com o caso particular onde $n_y = 1$.

Associado a cada neurônio j da rede (posicionado em $\mathbf{w}_j$ no espaço de entrada), tem-se uma saída $\hat{y}_j$ e um vetor $\mathbf{a}_j$ de dimensão $n_x \times 1$ associada a esta saída, onde $n_x = L_u + L_y$. O escalar $\hat{y}_j$ é a saída da rede para os casos em que os vetores de entrada coincidem com um dos centros, ou seja, $\mathbf{x}(t) = \mathbf{w}_j$. Para um vetor de entrada qualquer, o centro mais próximo s_1 é determinado conforme distância euclidiana e a saída $\hat{y}(t)$ da rede é calculada da seguinte forma:

$$\hat{y}(t) = \hat{y}_{s_1}(t) + \mathbf{a}_{s_1}^T(t)(\mathbf{x}(t) - \mathbf{w}_{s_1}(t)), \tag{5}$$

em que o termo $\hat{y}_{s_1}$ é uma primeira aproximação local da saída com o segundo termo promovendo uma correção de primeira ordem, à guisa de uma expansão de Taylor em torno do ponto $\hat{y}_{s_1}$.

Agora, tem-se que promover uma ligeira mudança no algoritmo GNG original para permitir o ajuste adequado dos modelos lineares locais, uma vez que se está interessado em reduzir o erro quadrático médio. Muda-se então a equação original de erro do modelo GNG para

$$\Delta e_{s_1}(t) = |y(t) - \hat{y}(t)|^2, \tag{6}$$

em que $|\cdot|$ denota o valor absoluto.

A Equação (6) significa que agora o erro é acumulado localmente em relação à função a ser aproximada. Novas unidades são inseridas quando essa aproximação for considerada ruim.

Os mapeamentos lineares locais $\mathbf{a}_j$ associados às unidades da rede são inicialmente definidos de forma aleatória. Em cada etapa de adaptação, o par de dados entrada-saída é utilizado duas vezes: (i) o vetor de entrada é utilizado para a adaptação do centro e (ii) todo o par é utilizado para melhorar o vetor de coeficientes $\mathbf{a}_{s_1}$ do centro mais próximo s_1. Isso é feito utilizando, mais uma vez, a regra LMS [25]:

$$\hat{y}_{s_1}(t+1) = \hat{y}_{s_1}(t) + \alpha e_{s_1}(t), \tag{7}$$

$$\mathbf{a}_{s_1}(t+1) = \mathbf{a}_{s_1}(t) + \alpha e_{s_1}(t)(\mathbf{x}(t) - \mathbf{w}_{s_1}), \tag{8}$$

onde $0 < \alpha \ll 1$ é a taxa de aprendizado.

Quando uma nova unidade r é inserida, um mapeamento linear local é interpolado entre os vizinhos q e f:

$$\hat{y}_r(t) = 0,5(\hat{y}_q(t) + \hat{y}_f(t)) \tag{9}$$

$$\mathbf{a}_r(t) = 0,5(\mathbf{a}_q(t) + \mathbf{a}_f(t)). \tag{10}$$

Um critério de parada deve ser definido para terminar o processo de crescimento. Isso pode ser escolhido arbitrariamente, dependendo da aplicação. Uma escolha possível é observar o desempenho da rede em um conjunto de validação durante o treinamento e parar quando esse desempenho começar a diminuir. Alternativamente, o erro no conjunto de treinamento pode ser usado ou simplesmente o número de unidades na rede se, por algum motivo, um tamanho de rede específico for desejado.

O modelo proposto por [24] se tornou um propulsor para a abordagem incremental de mapeamentos lineares locais. Alguns trabalhos encontrados na literatura foram propostos baseados no modelo apresentado [26]–[29]. Em [26], a proposta é bem semelhante à de [24], a cada iteração, a informação de realimentação do erro de aproximação do modelo atual é utilizada para tomar a decisão de inserção de novos modelos locais na área de entrada com maior erro.

B. Proposta 1: Modelo RAN-LMS

A rede de alocação de recursos (*resource allocation network*, RAN) foi proposta por [23] e é uma rede de duas camadas, que se assemelha a uma rede RBF, porém com uma estrutura crescente.

Esta rede visa construir representações locais do mapeamento de entrada-saída subjacente. As unidades ocultas da rede RAN respondem apenas a uma região local do espaço de entrada. A rede aprende alocando novas unidades e ajustando os parâmetros das unidades existentes. Se a rede funcionar mal para um determinado vetor de entrada, conforme medido pelo erro de aproximação do neurônio de saída, uma nova unidade oculta é alocada para corrigir a resposta da rede a esse vetor de entrada. Se a rede funcionar bem para um determinado vetor de entrada, os parâmetros das unidades de saída existentes serão atualizados usando a regra LMS padrão já descrita em seções anteriores.

A RAN começa com uma folha em branco: nenhum padrão foi apresentado ainda. Conforme os padrões são apresentados a ela, a rede escolhe armazenar alguns deles. Os pares de entrada-saída são apresentados e a rede identifica um padrão que não está bem representado no momento e aloca uma nova unidade oculta que memoriza essa amostra. A saída da nova unidade oculta se estende à camada de saída. Depois que a nova unidade é alocada, a saída da rede é igual à saída desejada: $\hat{y}(t) = y(t)$. Seja o índice desta nova unidade n. Uma formalização do procedimento de inserção de uma nova unidade oculta é fornecida abaixo.

1) O centro da nova unidade é definido como o $\mathbf{c}_n = \mathbf{x}(t)$.
2) O peso que conecta a nova unidade oculta à unidade de saída é definido com $w_n = y(t) - \hat{y}(t)$, em que $\hat{y}(t)$ pode ser calculado conforme a equação de saída do modelo RBF tradicional. Vale ressaltar que o modelo RAN se assemelha a um modelo crescente da rede RBF.
3) O raio da nova unidade oculta é dado por

$$\sigma_n(t) = \kappa\|\mathbf{x}(t) - \mathbf{c}_{nearest}(t)\|, \tag{11}$$

tal que $\kappa > 0$ é um fator de sobreposição e $\mathbf{c}_{nearest}$ é o centro mais próximo de $\mathbf{c}_n$.

Para decidir sobre a inserção de uma nova unidade oculta, o modelo RAN verifica duas condições de detecção de novidade.

Condição 1 - Verificar se o vetor de entrada está muito distante dos centros existentes, o que é verificado pela seguinte regra:

$$\|\mathbf{x}(t) - \mathbf{c}_{nearest}(t)\| > \delta(t), \tag{12}$$

em que $\delta(t)$ é um limiar decrescente para detecção de novidade.

Condição 2 - Verificar se o erro entre a saída desejada e a saída atual da rede é grande, ou seja,

$$|y(t) - \hat{y}(t)| > \epsilon, \tag{13}$$

em que $|\cdot|$ é o operador de valor absoluto e ϵ é a acurácia desejada para o problema de aproximação

Observação 4 - Para estabilizar o processo de crescimento da rede, o limiar de novidade começa com um valor alto $\delta_{\max}$, diminuindo com o tempo até atingir um valor mínimo $\delta_{\min}$, que é mantido constante pelo tempo restante do processo de aprendizagem. A este respeito, a função de decaimento sugerida em [23] é usada para o limiar de novidade:

$$\delta(t) = \max\left[\delta_{\max}\exp(-t/\tau), \delta_{\min}\right], \tag{14}$$

em que τ é uma constante de decaimento que define a velocidade de inserção de novas unidades ocultas e, portanto, o número de unidades ocultas que podem ser inseridas. Valores pequenos de τ levam a um rápido decaimento de $\delta(t)$, implicando em um aumento na velocidade de inserção de novas unidades ocultas.

Quando uma nova unidade não é alocada, os centros das unidades ocultas existentes e os pesos e o limiar do neurônio de saída são ajustados por meio de variantes da regra LMS desenvolvidas a seguir. **Vale ressaltar que estas equações foram desenvolvidas neste trabalho e diferem bastante das equações do modelo RAN original. Por isso, optou-se por colocar esta variante como uma proposta deste trabalho.** Para a aplicação desta regra, a função objetivo global do modelo é o erro quadrático instantâneo do neurônio de saída:

$$J_{LMS}(t) = \frac{1}{2}e^2(t) = \frac{1}{2}(y(t) - \hat{y}(t))^2, \tag{15}$$

$$= \frac{1}{2}\left(y(t) - \sum_{j=1}^{S} z_j(t)w_j(t)\right)^2, \tag{16}$$

em que $w_j(t)$ é o peso associado à j-ésima unidade oculta e $z_j(t)$ é a saída da unidade oculta associada. Portanto, para atualizar o vetor de peso da j-ésima função local, a seguinte regra de atualização é usada:

$$\mathbf{w}(t+1) = \mathbf{w}(t) - \alpha(t)\frac{\partial J_{LMS}(t)}{\partial \mathbf{w}(t)}, \tag{17}$$

$$= \mathbf{w}(t) - \alpha(t)\frac{\partial J_{LMS}(t)}{\partial e(t)}\frac{\partial e(t)}{\partial \hat{y}(t)}\frac{\partial \hat{y}(t)}{\partial \mathbf{w}(t)}, \tag{18}$$

$$= \mathbf{w}(t) + \alpha(t)e(t)\mathbf{z}(t), \tag{19}$$

em que se pode notar que a atualização dos vetores de peso dos submodelos locais é modulada pelas saídas das unidades ocultas $z_j(t)$, $j = 1, \ldots, S$. Esta modulação dá um caráter localizado ao processo de atualização dos parâmetros. Quanto maior o valor de $z_j(t)$, maior é a variação no vetor de peso $\mathbf{w}(t)$.

A atualização dos centros das unidades ocultas segue um procedimento semelhante. Para isto, utilizou-se mais uma vez a função objetivo global do modelo supracitada. Neste caso, aplicou-se a seguinte regra:

$$\mathbf{c}_j(t+1) = \mathbf{c}_j(t) - \alpha(t)\frac{\partial J_{LMS}(t)}{\partial \mathbf{c}_j(t)}, \tag{20}$$

$$= \mathbf{c}_j(t) + \frac{\alpha(t)}{\sigma_j^2}z_j(t)e(t)w_j(t)(\mathbf{x}(t) - \mathbf{c}_j(t)), \tag{21}$$

em que a Equação (21) é derivada usando a função de base gaussiana. As Equações contidas em (20) foram desenvolvidas para este trabalho buscando um melhor entendimento da atualização dos centros das unidades ocultas realizada no trabalho de [23].

Para os experimentos deste trabalho, obteve-se melhores resultados usando a regra LMS normalizada [30]. Para implementar esta variante da regra LMS, substitui-se a taxa de aprendizagem original $\alpha(t)$ com $\alpha'(t) = \alpha(t)/\|\mathbf{x}(t)\|^2$, onde $\|\mathbf{x}(t)\|^2$ é a norma quadrática do vetor de entrada atual. O modelo RAN está disposto na forma de pseudocódigo no algoritmo abaixo.

A facilidade de implementação e o baixo custo computacional são dois grandes atrativos do modelo RAN. [31] aplicaram a RAN para estimar o desgaste da ferramenta em operações de fresamento.

[32] apresentaram uma nova abordagem para a identificação de regiões do cérebro responsáveis pela doença de Alzheimer utilizando imagens de ressonância magnética. A abordagem incorporou uma versão da RAN, a SRAN (*self-adaptive resource allocation network*) [33] para a classificação da doença de Alzheimer. O classificador SRAN usa um algoritmo de aprendizagem sequencial, empregando limites auto-adaptativos para selecionar as amostras de treinamento apropriadas e descartar as amostras redundantes para evitar o excesso de treinamento.

[34] demonstraram a aplicabilidade do modelo RAN por meio de três exemplos de diversos domínios: rede de resfriamento de água, planejamento do setor de energia com restrição de carbono e rede de alocação de água. Em [35], o modelo RAN é treinado *offline* para determinar uma estrutura inicial. A partir daí, o modelo RAN inicial

Início:

Definir: κ, ϵ, τ, δ_{max}, δ_{min}, σ_j, e α.

Atribuir: $\delta_0 = \delta_{max}$, e $S = 1$.

Iteração: apresentação dos pares entrada-saída

1. Predizer: Para cada vetor de entradar $\mathbf{x}(t)$, calcular

1.1. $z_j(t) = \phi\left(d(\mathbf{x}(t), \mathbf{c}_j(t))\right); \quad j = 1, \ldots, S.$

1.2. $\hat{y}(t) = \sum_{j=1}^{S} z_j(t) w_j(t)$;

2. Calcular o erro:

2.1. $e(t) = y(t) - \hat{y}(t)$;

3. Encontrar as distâncias até o centro mais próximo:

3.1. $d_{near}(t) = \|\mathbf{x} - \mathbf{c}_{near}(t)\|$;

4. Decidir entre crescer ou atualizar:

SE $|e(t)| > \epsilon$ e $d_{near}(t) > \delta(t)$, **ENTÃO**

{ % Crescer

Aloca nova unidade $\mathbf{c}_n = \mathbf{x}(t)$ e $w_n(t) = e(t)$;

SE essa é a primeira unidade a ser alocada,

ENTÃO $\sigma_n = \kappa\delta(t)$;

SENÃO $\sigma_n = \kappa\|\mathbf{x}(t) - \mathbf{c}_{nearest}(t)\|$.

}

SENÃO

{ % Atualizar

Atualiza vetores de peso usando Equação (19).

Atualiza centro usando Equação (21).

Atualiza limiar de novidade usando Equação (14).

}

é utilizado para a previsão *online* de energia fotovoltaica e é posteriormente atualizado.

[36] utilizaram o modelo RAN para estimação dos parâmetros de um controlador proporcional integrador derivativo (PID) para veículos guiados automatizados (*automated guided vehicle*, AGV).

IV. Modelos Locais Robustos

A robustez que buscamos está relacionada a modelos que não são fortemente afetados pela presença de *outliers* nos dados de treinamento. Esta robustez é um recurso valioso para os métodos de identificação do sistema. [19] é considerado o pioneiro em estatísticas robustas. O objetivo é produzir estimadores que não sejam afetados por variações pequenas relacionadas às hipóteses dos modelos [37].

A partir do exposto, a abordagem de regressão ou identificaçao de sistemas robusta busca desenvolver estimadores que são resilientes a *outliers*. Para isto, [19] desenvolveu uma classe de estimadores com base nas estatísticas robustas conhecida como estimadores M, onde o M representa máxima verossimilhança.

Com base nesta estrutura, estimadores recursivos robustos são encontrados em [38] e [39], que introduziram modificações simples na regra LMS padrão a fim de melhorar seu desempenho em dados ricos em *outliers*. A regra resultante foi então chamada de LMM (*least mean M-estimate*) e é tão rápida quanto a regra LMS original, no sentido de que nenhuma carga computacional extra é adicionada ao processo de estimativa de parâmetros.

Os sinais $x(t)$ e $\hat{y}(t)$ são, respectivamente, a entrada e sinais de saída do modelo. O erro de estimativa é dado por $e(t) = y(t) - (\mathbf{w}^T(t-1) \cdot \mathbf{x}(t))$, onde $\mathbf{w}(t-1)$ e $\mathbf{x}(t)$ são os vetores de peso e de entrada, respectivamente. O sinal $\hat{y}(t)$ é o sinal de saída desejado. Em aplicações prátcas, $x(t)$ e $\hat{y}(t)$ podem ser corrompidos por *outliers*. Baseado em estatística robusta para estimação [19], a seguinte função objetivo para modelo é proposta [38]

$$J_p = \xi[\varrho(e(t))], \tag{22}$$

onde $\varrho(\cdot)$ é uma função de estimativa M robusta para suprimir o ruído impulsivo (*outlier*). Neste trabalho, pela simplicidade da função, utilizaremos a função Huber Modificada descrita como

$$\varrho(e(t)) = \begin{cases} e(t)^2/2, & \text{se } 0 \leq |e(t)| < \beta, \\ \kappa^2/2, & \text{caso contrário,} \end{cases} \tag{23}$$

onde β é o parâmetro de limite usado para suprimir o efeito de *outlier* quando o erro de estimativa e é muito grande. Da mesma forma do algoritmo LMS, a função custo J_p será minimizado atualizando $\mathbf{w}(t)$ na direção negativa do vetor gradiente. Portanto, o vetor gradiente $\nabla_w(J_p)$ é aproximado por

$$\nabla_w(J_p) = -q(e(t))e(t)\mathbf{x}(t), \tag{24}$$

onde $q(e) = \frac{\psi(e)}{e} = \frac{\partial\varrho(e)}{\partial e}$ e $\psi(e)$ é a função de ponderação do erro. Para a função HM:

$$q_{HM}(e) = \frac{\psi_{HM}(e)}{e} = \begin{cases} 1, & \text{se } 0 \leq |e(t)| < \beta, \\ 0, & \text{caso contrário.} \end{cases} \tag{25}$$

Finalmente, o algoritmo LMM proposto em [39] é:

$$\mathbf{w}(t+1) = \mathbf{w}(t) - \alpha\hat{\nabla}_{w_\varrho} = \mathbf{w}(t) + \alpha\psi(e(t)\mathbf{x}(t)). \tag{26}$$

Geralmente, quando $e(t)$ é menor do que β, a função de ponderação $q(e(t))$ é igual a um e, assim, a Equação 26 se torna idêntica ao algoritmo LMS. Porém, quando $e(t)$ é maior do que certos limites, $q(e(t))$ se tornará zero e impedirá a atualização do vetor de pesos. Recomenda-se usar $\beta = 1.345\ \sigma$ a fim de produzir 95% de eficiência quando os erros são normalmente distribuídos, enquanto ainda oferece proteção contra *outliers*, onde σ corresponde ao desvio padrão dos resíduos [40].

V. Metodologia

A. Erro Médio Quadrático (RMSE)

Uma escolha comum para avaliar os modelos é o Erro Médio Quadrático (RMSE - *Root Mean Squared Error*) que é calculado entre os resultados previstos pelo modelo e as

observações reais em todo o conjunto de dados de teste. Todos os modelos são numericamente avaliados por meio do Erro Médio Quadrático expresso matematicamente por $RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N} e^2(t)}$,onde $e(t) = y(t) - \hat{y}(t)$ e N é a quantidade de amostras do conjunto de dados de teste.

B. Conjuntos de Dados Avaliados

O modelos implementados neste trabalho, utilizam como amostras cinco conjuntos de dados (*benchmarking*) (Atuador Hidráulico [41] e *pH*) que estão publicamente disponíveis para download no site do repositório DaISy em *Katholieke Universiteit Leuven*. Algumas características importantes dos conjuntos de dados supracitados estão na Tabela I.

TABLE I: Características dos conjuntos de dados avaliados.

	Conjuntos Avaliados			
Qtde. Amostras	**Estimação**	**Teste**	$\hat{L}_u$	$\hat{L}_y$
Atuador Hidráulico	512	512	4	4
pH	200	800	5	5

VI. Resultados

Nesta seção, iremos reportar os resultados de uma comparação abrangente de desempenho entre os modelos:

- INC-LLM e RAN
- e suas versões robustas.

O objetivo dos experimentos é avaliar a capacidade dos modelos propostos em fornecer um modelo incremental e preciso para a identificação de sistemas dinâmicos na presença de *outliers*.

Todos os modelos avaliados foram implementados do zero utilizando a linguagem de script Octave e os códigos correspondentes estão disponíveis mediante solicitação. Algumas características importantes dos conjuntos da dados avaliados já foram resumidas na Tabela I. A acurácia de todos os modelos é avaliada pelo erro médio quadrático (RMSE).

Outliers foram introduzidos artificialmente em dados de treinamento apenas em diferentes proporções. Para tanto, seguimos o mesmo procedimento utilizado por [42]. Os resultados reportados correspondem à avaliação do modelo em cenários livres de *outliers*. A justificativa para essa abordagem é avaliar como os *outliers* afetam o processo de estimativa de parâmetros (treinamento) e seu impacto na fase de validação (teste) do modelo. Os modelos treinados foram testados sob o regime de *simulação livre*, no qual os valores de saída previstos são realimentados para o vetor de regressão de entrada.

A. Resultados no conjunto de dados Atuador Hidráulico

Definimos 100 épocas para treinar todos os modelos. O número de neurônios é inicialmente definido como $S = 5$ para todos os modelos com tamanho fixo (originais e robustos), isso também se deve ao fato de valores maiores não mostrarem desempenho expressivamente superiores. Definimos o número de modelos locais iniciais como $S = 2$ para o modelo crescente INC-LLM (original e robusto) e como $S = 1$ para os modelos crescentes RAN (original e robusto) . As taxas de aprendizagem inicial e final foram definidas como $\alpha_0 = 0,5$ e $\alpha_T = 0,001$ para todos os modelos. Quanto à escolha dos hiperparâmetros, para o modelo LLM Incremental, utilizamos: a idade máxima para remoção dos links $a_{max} = 50$; $\epsilon_b = 0,99$ e $\epsilon_n = 0,005$; número máximo de modelos $N = 10$; $\lambda = 100$. Uma grande dificuldade enfrentada pelo modelo incremental LLM é a quantidade de hiperparâmetos a ser definida antes da estimação do modelo. Para o modelo RAN, definimos $\kappa = 0,9$, $\delta_{\max} = \max(\|\mathbf{x} - \mathbf{c}_0\|)$, $\delta_{\min} = \delta_{\max}/10$, $\tau = 10$, e $\epsilon = 10^{-4}$. Perceba, que parte dos parâmetros dos dois últimos modelos foram calculados em função dos próprios dados de entrada.

TABLE II: Valores de RMSE para dois níveis de contaminação de *outliers* (Atuador Hidráulico).

	0% outliers		15% outliers	
Modelos	**RMSE**	**STD**	**RMSE**	**STD**
INC-LLM-LMS	3.14E-02	1.08E-04	2.12E-01	3.65E-04
INC-LLM-LMM	2.06E-02	1.01E-04	6.06E-02	8.04E-04
RAN-LMS	4.63E-02	2.14E-05	6.45E-01	6.34E-05
RAN-LMM	2.38E-02	1.89E-05	1.18E-01	2.17E-05

Os valores RMSE em função da porcentagem de *outliers* são mostrados na Tabela II. Como esperado, o modelos robustos tiveram um desempenho melhor do que as versões não robustas (aquelas que usam a regra LMS simples). O mocelo INC-LLM-LMM se mostrou com menor sensibilidade à presença de *outliers*. Notem que até 5% de *outliers*, a performance dos modelos é semelhante.

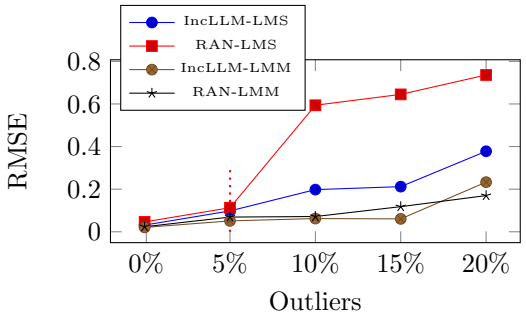

Fig. 1: Valores de RMSE em função da quantidade de *outliers* dos dados de treinamento (Atuador Hidráulico).

A evolução correspondente do número de modelos locais para os dois modelos propostos é mostrada na linha superior da Fig. 2. Esse número se estabilizou em $S = 7$ para os modelos baseados na RAN e para o INC-LLM-LMS foi até $S = 8$. Deve-se notar que o número de modelos locais cresce mais rápido para os modelos robustos, que usa a regra de aprendizado LMM. Esta regra trata automaticamente os *outliers*, evitando sua influência negativa no processo de estimativa de parâmetros.

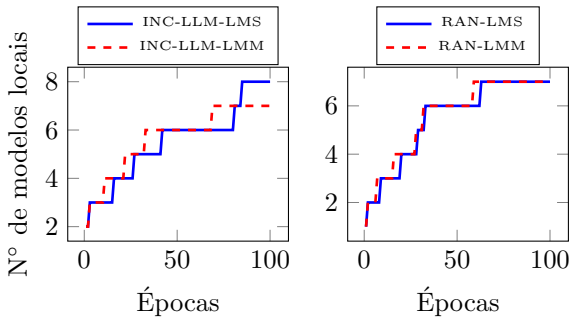

Fig. 2: Evolução do número de modelos locais em função das épocas de treinamento para modelos propostos (Atuador Hidráulico).

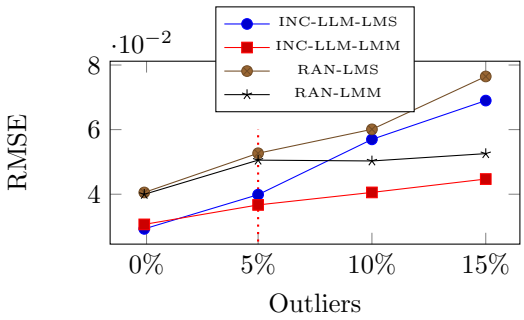

Fig. 3: Valores de RMSE em função da quantidade de *outliers* dos dados de treinamento (Conjunto de dados *pH*).

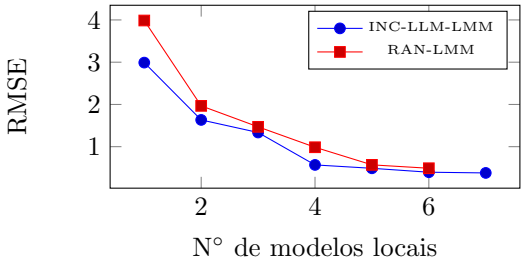

Fig. 4: RMSE no treinamento *versus* número de modelos locais com 15% outliers (Conjunto de dados *pH*)

B. Resultados no conjunto de dados pH

Para os testes com esse conjunto de dados, as taxas de aprendizagem inicial e final foram definidas como $\alpha_0 = 0,5$ e $\alpha_T = 0,001$ para todos os modelos. Para o modelo LLM Incremental, utilizamos: a idade máxima para remoção dos links $a_{max} = 50$; $\epsilon_b = 0,99$ e $\epsilon_n = 0,005$ (Equação **??**); número máximo de modelos $N = 10$; $\lambda = 100$. Para os modelos RAN , definimos $\kappa = 0,9$, $\delta_{\max} = \max(\|\mathbf{x} - \mathbf{c}_0\|)$, $\delta_{\min} = \delta_{\max}/10$, $\tau = 10$, e $\epsilon = 10^{-4}$.

Uma comparação em termos de valores RMSE médios dos três modelos que utilizam regra LMS para atualização e os três modelos robustos para o conjunto de dados *pH* é mostrado na Fig. 3 para diferentes cenários de contaminação com *outliers*. Como esperado para o cenário livre de *outliers*, os desempenhos dos modelos é praticamente o mesmo. Para este cenário, podemos destacar o modelo INC-LLM (versão original e robusta). Estes dois modelos alcançaram menores valores de RMSE.

No entanto, conforme o nível de contaminação de valores discrepantes aumenta, pode-se observar novamente que as versões robustas alcançam um desempenho melhor do que aqueles modelos que usam a regra de aprendizado baseada em LMS original. O do decaimento do RMSE (treinamento) devido a inserção de modelos é mostrado na Fig. 4. Para este conjunto de dados, o melhor desempenho foi alcançado pelo modelo INC-LLM-LMM proposto.

TABLE III: Valores de RMSE para dois níveis de contaminação de *outliers* (conjunto de dados *pH*).

	0% outliers		15% outliers	
Modelos	**RMSE**	**STD**	**RMSE**	**STD**
INC-LLM-LMS	2.93E-02	2.95E-05	6.90E-02	7.30E-05
INC-LLM-LMM	3.07E-02	6.17E-05	4.47E-02	8.36E-05
RAN-LMS	4.05E-02	2.69E-05	7.65E-02	4.16E-05
RAN-LMM	4.00E-02	8.97E-05	5.26E-02	6.37E-05

VII. Conclusão

Neste trabalho, abordamos a tarefa de identificação de sistemas dinâmicos em cenários contaminados com *outliers*. Para este fim, revisitamos uma classe de modelos de identificação conhecida como modelos locais crescentes e a questão da robustez na estimação de parâmetros (ou seja, aprendizagem) na presença de *outliers*.

Os modelos locais discutidos e avaliados utilizam a conhecida regra de Widrow-Hoff [25], também chamada de regra LMS, na estimação recursiva de seus parâmetros. Um dos objetivos deste artigo foi, em primeiro lugar, avaliar o quanto os *outliers* afetam o desempenho preditivo dos modelos locais de aproximação e, em segundo lugar, se a substituição da regra LMS original por uma versão robusta a *outliers*, chamada de regra LMM [38], é capaz de melhorar de alguma forma o desempenho dos modelos locais, proporcionando-lhes maior resiliência a outliers.

Outro objetivo foi a avaliação o desempenho de modelos locais robustos crescentes, ou seja, modelos em que a especificação *a priori* do número de neurônios não é necessária, pois neurônios vão sendo alocados com o passar do tempo, na medida que for necessário. Estas novas unidades são inseridas apenas quando exigidas pelo desempenho do modelo.

Os resultados de um estudo comparativo com dados de sistemas dinâmicos reais mostraram que a simples alteração da regra LMS por sua versão robusta proporcionou uma melhora considerável no desempenho dos modelos locais avaliados em cenários ricos em *outliers*.

Agradecimentos

Os autores agradecem ao IFCE, à UFC e ao CNPq (processo no. 309451/2015-9) pelo suporte financeiro a esta pesquisa.

References

[1] L. Ljung, "System identification," in *Signal analysis and prediction*, pp. 163–173, Springer, 1998.

[2] S. Lawrence, A. C. Tsoi, and A. D. Back, "Function approximation with neural networks and local methods: Bias, variance and smoothness," in *Australian Conference on Neural Networks*, vol. 1621, Australian National University, 1996.

[3] B. A. Foss and T. A. Johansen, "On local and fuzzy modelling," in *Third International Conference on Industrial Fuzzy Control and Intelligent Systems (IFIS'93)*, pp. 80–87, 1993.

[4] C.-C. Chuang, "Fuzzy weighted support vector regression with a fuzzy partition," *IEEE Transactions on Systems, Man, and Cybernetics - Part B*, vol. 37, no. 3, pp. 630–640, 2007.

[5] S. S. Haykin, B. Widrow, and B. Widrow, *Least-mean-square adaptive filters*, vol. 31. Wiley Online Library, 2003.

[6] R. Gao, *Local Model Network Application in Control*. PhD thesis, Dublin Institute of Technology, 2004.

[7] L. G. M. Souza, *Modelos lineares locais para identificação de sistemas dinâmicos usando redes neurais competitivas*. PhD thesis, Universidade Federal do Ceará, 2012.

[8] J. Belz, T. Munker, T. O. Heinz, G. Kampmann, and O. Nelles, "Automatic modeling with local model networks for benchmark processes," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 470–475, 2017.

[9] A. Kalhor, B. N. Araabi, and C. Lucas, "Evolving Takagi-Sugeno fuzzy model based on switching to neighboring models," *Applied Soft Computing Journal*, vol. 13, no. 2, pp. 939–946, 2013.

[10] P. P. Angelov and D. P. Filev, "An approach to online identification of takagi-sugeno fuzzy models," *IEEE Transactions on Systems, Man, and Cybernetics - part B*, vol. 34, no. 1, pp. 484–497, 2004.

[11] J. Liu, D. Djurdjanovic, K. Marko, and J. Ni, "Growing structure multiple model systems for anomaly detection and fault diagnosis," *Journal of Dynamic Systems, Measurement, and Control*, vol. 131, no. 5, pp. 1–13, 2009.

[12] B. Kiumarsi, F. L. Lewis, and D. S.Levine, "Optimal control of nonlinear discrete time-varying systems using a new neural network approximation structure," *Neurocomputing*, vol. 156, pp. 157–165, 2015.

[13] B. Fritzke, "Fast learning with incremental RBF networks," *Neural Processing Letters*, vol. 1, no. 1, pp. 2–5, 1994.

[14] K. Okamoto, S. Ozawa, and S. Abe, "A Fast Incremental Learning Algorithm of RBF Networks with Long-Term Memory," *Proceedings of the International Joint Conference on Neural Networks*, vol. 1, no. June, pp. 102–107, 2003.

[15] H. Han, Q. Chen, and J. Qiao, "Research on an online self-organizing radial basis function neural network," *Neural Computing and Applications*, vol. 19, pp. 667–676, 2010.

[16] S. Blazic and I. Skrjanc, "Incremental Fuzzy C-Regression Clustering from Streaming Data for Local-Model-Network Identification," *IEEE Transactions on Fuzzy Systems*, vol. 28, no. 4, pp. 758–767, 2020.

[17] F. R. Hampel, E. M. Ronchetti, P. J. Rousseeuw, and W. A. Stahel, *Robust statistics: the approach based on influence functions*, vol. 114. John Wiley & Sons, 2011.

[18] J. A. Bessa and G. A. Barreto, "Recursive system identification using outlier-robust local models," *IFAC-PapersOnLine*, vol. 52, no. 1, pp. 436–441, 2019.

[19] P. Huber, "Robust estimation of a location parameter," *The Annals of Mathematical Statistics*, vol. 35, no. 1, pp. 73–101, 1964.

[20] P. H. Ferreira and A. F. Araújo, "Growing self-organizing maps for nonlinear time-varying function approximation," *Neural Processing Letters*, pp. 1–26, 2020.

[21] S. Schaal and C. G. Atkeson, "Constructive incremental learning from only local information," *Neural computation*, vol. 10, no. 8, pp. 2047–2084, 1998.

[22] B. Fritzke, "A growing neural gas network learns topologies," in *Advances in Neural Information Processing Systems 7*, pp. 625–632, MIT Press, 1995.

[23] J. Platt, "A resource-allocating network for function interpolation," *Neural Computation*, vol. 3, no. 2, pp. 213–225, 1991.

[24] B. Fritzke, "Incremental learning of local linear mappings," in *Proceedings of the International Conference on Artificial Neural Networks (ICANN'95)*, pp. 217–222, 1995.

[25] B. Widrow, "Thinking about thinking: The discovery of the LMS algorithm," *IEEE Signal Processing Magazine*, vol. 22, no. 1, pp. 100–106, 2005.

[26] G. Vachkov, "Growing model algorithm for process identification based on neural-gas learning and local linear mapping," in *Fourth International Conference on Hybrid Intelligent Systems (HIS'04)*, pp. 222–227, IEEE, 2004.

[27] H. He, S. Chen, K. Li, and X. Xu, "Incremental learning from stream data," *IEEE Transactions on Neural Networks*, vol. 22, no. 12, pp. 1901–1914, 2011.

[28] P. Fuangkhon, "An incremental learning preprocessor for feed-forward neural network," *Artificial Intelligence Review*, vol. 41, no. 2, pp. 183–210, 2014.

[29] T. Lu, L. Pan, J. Jiangs, Y. Zhang, and Z. Xiong, "Dlml: Deep linear mappings learning for face super-resolution with nonlocal-patch," in *2017 IEEE International Conference on Multimedia and Expo (ICME)*, pp. 1362–1367, IEEE, 2017.

[30] B. Widrow and M. A. Lehr, "30 years of adaptive neural networks: Perceptron, madaline and backpropagation," *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1415–1442, 1990.

[31] P. S. Pai, T. Nagabhushana, and P. R. Rao, "Tool wear estimation using resource allocation network," *International Journal of Machine Tools and Manufacture*, vol. 41, no. 5, pp. 673–685, 2001.

[32] B. S. Mahanand, S. Suresh, N. Sundararajan, and M. A. Kumar, "Identification of brain regions responsible for alzheimer's disease using a self-adaptive resource allocation network," *Neural Networks*, vol. 32, pp. 313–322, 2012.

[33] S. Suresh, K. Dong, and H. J. Kim, "A sequential learning algorithm for self-adaptive resource allocation network classifier," *Neurocomputing*, vol. 73, no. 16-18, pp. 3012–3019, 2010.

[34] S. Jain and S. Bandyopadhyay, "Resource allocation network for segregated targeting problems with dedicated sources," *Industrial & Engineering Chemistry Research*, vol. 56, no. 46, pp. 13831–13843, 2017.

[35] C. Lv, T. Zhang, F. Ma, and D. Yue, "A very short-term online forecasting model for photovoltaic power based on two-stage resource allocation network," in *2018 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–6, IEEE, 2018.

[36] Q. Wang, X. Wang, and B. Sun, "Resource allocation network and PID control based on automated guided vehicles," *ACADEMIC JOURNAL OF MANUFACTURING ENGINEERING*, vol. 17, no. 2, 2019.

[37] A. M. Pires, J. A. Branco, and C. A. da Sociedade Portuguesa de Estatística, *Introdução aos métodos estatísticos robustos*. 2007.

[38] Y. Zou, S.-C. Chan, and T.-S. Ng, "Least mean M-estimate algorithms for robust adaptive filtering in impulse noise," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 47, no. 12, pp. 1564–1569, 2000.

[39] S.-C. Chan and Y. Zhou, "On the performance analysis of the least mean M-estimate and normalized least mean M-estimate algorithms with gaussian inputs and additive gaussian and contaminated Gaussian noises," *Journal of Signal Processing Systems*, vol. 60, no. 1, pp. 81–103, 2010.

[40] P. J. Rousseeuw and A. M. Leroy, *Robust regression and outlier detection*, vol. 589. John wiley & sons, 2005.

[41] J. Sjöberg, Q. Zhang, L. Ljung, A. Benveniste, B. Delyon, P.-Y. Glorennec, H. Hjalmarsson, and A. Juditsky, "Nonlinear black-box modeling in system identification: a unified overview," *Automatica*, vol. 31, no. 12, pp. 1691–1724, 1995.

[42] C. L. C. Mattos, Z. Dai, A. Damianou, G. A. Barreto, and N. D. Lawrence, "Deep recurrent gaussian processes for outlier-robust system identification," *Journal of Process Control*, vol. 60, pp. 82–94, 2017.