Pós-Graduação em Ciência da Computação

"UM MÉTODO DE APRENDIZAGEM SEQUENCIAL COM FILTRO DE KALMAN E EXTREME LEARNING MACHINE PARA PROBLEMAS DE REGRESSÃO E PREVISÃO DE SÉRIES TEMPORAIS"

Por

Jarley Palmeira Nóbrega

Tese de Doutorado

Universidade Federal de Pernambuco
posgraduacao@cin.ufpe.br
www.cin.ufpe.br/~posgraduacao

RECIFE/2015

Universidade Federal de Pernambuco
Centro de Informática
Pós-graduação em Ciência da Computação

Jarley Palmeira Nóbrega

UM MÉTODO DE APRENDIZAGEM SEQUENCIAL COM FILTRO DE KALMAN E EXTREME LEARNING MACHINE PARA PROBLEMAS DE REGRESSÃO E PREVISÃO DE SÉRIES TEMPORAIS

Trabalho apresentado ao Programa de Pós-graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Doutor em Ciência da Computação.

Orientador: *Adriano Lorena Inácio de Oliveira*

RECIFE
2015

Catalogação na fonte
Bibliotecária Jane Souto Maior, CRB4-571

N754m Nóbrega, Jarley Palmeira
Um método de aprendizagem seqüencial com filtro de Kalman e Extreme Learning Machine para problemas de regressão e previsão de séries temporais / Jarley Palmeira Nóbrega. – Recife: O Autor, 2015.
144 p.: il., fig., tab.

Orientador: Adriano Lorena Inácio de Oliveira.
Tese (Doutorado) – Universidade Federal de Pernambuco. CIn, Ciência da computação, 2015.
Inclui referências e apêndice.

1. Inteligência computacional. 2. Previsão de séries temporais. 3. Redes neurais artificiais. 4. Regressão. I. Oliveira, Adriano Lorena Inácio de (orientador). II. Título.

006.3 CDD (23. ed.) UFPE- MEI 2015-138

Tese de Doutorado apresentada por **Jarley Palmeira Nóbrega** à Pós-Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco, sob o título "**Um Método de Aprendizagem Sequencial com Filtro de Kalman e Extreme Learning Machine para Problemas de Regressão e Previsão de Séries Temporais**" orientada pelo **Prof. Adriano Lorena Inácio de Oliveira** e aprovada pela Banca Examinadora formada pelos professores:

_____

Prof. George Darmiton da Cunha Cavalcanti
Centro de Informática / UFPE

_____

Prof. Tsang Ing Ren
Centro de Informática / UFPE

_____

Profa. Renata Maria Cardoso Rodrigues de Souza
Centro de Informática / UFPE

_____

Prof. Antônio de Pádua Braga
Departamento de Engenharia Eletrônica / UFMG

_____

Prof. Hélio Magalhães de Oliveira
Departamento de Estatística / UFPE

Visto e permitida a impressão.
Recife, 24 de agosto de 2015

_____

Profa. Edna Natividade da Silva Barros
Coordenador da Pós-Graduação em Ciência da Computação do
Centro de Informática da Universidade Federal de Pernambuco.

Esse trabalho é dedicado aos meus pais, que me ensinaram o valor da educação, e à minha esposa e filhos, que foram a luz do meu caminho até aqui.

Agradecimentos

No longo caminho percorrido para a conclusão desse trabalho, não poderia deixar de fazer um agradecimento para as pessoas e entidades que tornaram a minha jornada possível.

Gostaria de agradecer ao Ministério da Ciência, Tecnologia e Inovação (MCTI), onde iniciei em 2010 os estudos que culminaram com a pesquisa apresentada nesse trabalho. Em especial, sou grato aos professores Ivon Fittipaldi e Amaro Lins por reconhecer a importância da formação acadêmica no desenvolvimento de minha carreira.

Um agradecimento especial aos professores do Centro de Informática: Paulo Adeodato, Robson Fidalgo, Francisco Carvalho e Renata Souza. Suas aulas foram fundamentais para consolidar o conhecimento usado ao longo desse trabalho.

Ao meu orientador, Adriano Lorena, sou eternamente grato por sua paciência e apoio incondicional ao longo do meu doutorado. Obrigado por ter acreditado no meu trabalho. Espero que a nossa parceria científica gere bons frutos no futuro!

Gostaria de agradecer aos meus pais, Januir e Céu, por ensinar a importância da educação em minha vida. Prometo que farei o mesmo por seus netos.

Faço uma dedicatória especial à minha irmã, Jane. A sua história me ensinou que, mesmo diante das barreiras mais difíceis da vida, não podemos desistir nunca de lutar!

Esse trabalho é dedicado às pessoas mais importantes de minha vida: minha amada esposa Ana Carolina, por seu amor e paciência. Não teria conseguido terminar esse trabalho sem o seu apoio. Aos meus filhos, Guilherme e Gabriela, por me proporcionar toda a felicidade do mundo ao vê-los crescendo. Todo o sacrifício desses anos de trabalho valeram a pena por vocês!

Por fim, agradeço a Deus pela serenidade e saúde concedida durante todo o período do meu doutorado.

Resumo

Em aplicações de aprendizagem de máquina, é comum encontrar situações onde o conjunto de entrada não está totalmente disponível no início da fase de treinamento. Uma solução conhecida para essa classe de problema é a realização do processo de aprendizagem através do fornecimento sequencial das instâncias de treinamento. Entre as abordagens mais recentes para esses métodos, encontram-se as baseadas em redes neurais do tipo *Single Layer Feedforward Network* (SLFN), com destaque para as extensões da *Extreme Learning Machine* (ELM) para aprendizagem sequencial.

A versão sequencial da ELM, chamada de *Online Sequential Extreme Learning Machine* (OS-ELM), utiliza uma solução recursiva de mínimos quadrados para atualizar os pesos de saída da rede através de uma matriz de covariância. Entretanto, a implementação da OS-ELM e suas extensões sofrem com o problema de multicolinearidade entre os elementos da matriz de covariância.

Essa tese introduz um novo método para aprendizagem sequencial com capacidade para tratar os efeitos da multicolinearidade. Chamado de *Kalman Learning Machine* (KLM), o método proposto utiliza o filtro de Kalman para a atualização sequencial dos pesos de saída de uma SLFN baseada na OS-ELM. Esse trabalho também propõe uma abordagem para a estimativa dos parâmetros do filtro, com o objetivo de diminuir a complexidade computacional do treinamento. Além disso, uma extensão do método chamada de *Extended Kalman Learning Machine* (EKLM) é apresentada, voltada para problemas onde a natureza do sistema em estudo seja não linear.

O método proposto nessa tese foi comparado com alguns dos mais recentes e efetivos métodos para o tratamento de multicolinearidade em problemas de aprendizagem sequencial. Os experimentos executados mostraram que o método proposto apresenta um desempenho melhor que a maioria dos métodos do estado da arte, quando medidos o de erro de previsão e o tempo de treinamento. Um estudo de caso foi realizado, aplicando o método proposto a um problema de previsão de séries temporais para o mercado financeiro. Os resultados confirmaram que o KLM consegue simultaneamente reduzir o erro de previsão e o tempo de treinamento, quando comparado com os demais métodos investigados nessa tese.

Palavras-chave: Redes Neurais. Aprendizagem Sequencial. Extreme Learning Machine. Filtro de Kalman. Multicolinearidade. Séries Temporais. Regressão.

Abstract

In machine learning applications, there are situations where the input dataset is not fully available at the beginning of the training phase. A well known solution for this class of problem is to perform the learning process through the sequential feed of training instances. Among most recent approaches for sequential learning, we can highlight the methods based on *Single Layer Feedforward Network* (SLFN) and the extensions of the *Extreme Learning Machine* (ELM) approach for sequential learning.

The sequential version of the ELM algorithm, named *Online Sequential Extreme Learning Machine* (OS-ELM), uses a recursive least squares solution for updating the output weights through a covariance matrix. However, the implementation of OS-ELM and its extensions suffer from the problem of multicollinearity for the hidden layer output matrix.

This thesis introduces a new method for sequential learning in which the effects of multicollinearity is handled. The proposed *Kalman Learning Machine* (KLM) updates sequentially the output weights of an OS-ELM based network by using the Kalman filter iterative procedure. In this work, in order to reduce the computational complexity of the training process, a new approach for estimating the filter parameters is presented. Moreover, an extension of the method, named *Extended Kalman Learning Machine* (EKLM), is presented for problems where the dynamics of the model are non linear.

The proposed method was evaluated by comparing the related state-of-the-art methods for sequential learning based on the original OS-ELM. The results of the experiments show that the proposed method can achieve the lowest forecast error when compared with most of their counterparts. Moreover, the KLM algorithm achieved the lowest average training time when all experiments were considered, as an evidence that the proposed method can reduce the computational complexity for the sequential learning process. A case study was performed by applying the proposed method for a problem of financial time series forecasting. The results reported confirm that the KLM algorithm can decrease the forecast error and the average training time simultaneously, when compared with other sequential learning algorithms.

Keywords: Neural Networks. Sequential Learning. Extreme Learning Machine. Kalman Filter. Multicollinearity. Time Series. Regression.

Lista de Figuras

Lista de Tabelas

Lista de Acrônimos

AUKF	Augmented Unscented Kalman Filter
CKF	Cubature Kalman Filter
EKF	Extended Kalman Filter
EKLM	Extended Kalman Learning Machine
ELM	Extreme Learning Machine
EOS-ELM	Ensemble of OS-ELM
FLS	Flexible Least Squares
FOKELM	Online Kernelized ELM with Forgetting Mechanism
FORELM	Online Regularized ELM with Forgetting Mechanism
FOS-ELM	Online Learning Algorithm with Forgetting Mechanism
GGAP-RBF	Generalized Growing and Prunning RBF
GHKF	Gauss–Hermite Kalman Filter
HOS-ELM	Hybrid OS-ELM
IELM	Incremental Extreme Learning Machine
IMM-KF	Interacting Multiple Model Kalman Filter
KB-IELM	Kernel Based Incremental ELM
KLM	Kalman Learning Machine
KNN	K-Nearest Neighbour
LMS	Least Mean Squares
LS-IELM	Least Squares Incremental ELM
MN-IELM	Minimum Norm Incremental ELM
M-RAN	Minimal Resource Allocation Network
MLP	Multi-layer Perceptron
OLS	Ordinary Least Squares
OS-ELA	Online Sequential Extreme Learning Algorithm
OS-ELM	Online Sequential Extreme Learning Machine
OS-ELMK	OS-ELM with Kernels
OS-ELM-TV	OS-ELM Time-varying

OSFuzzy-ELM Online Sequential Fuzzy ELM

PSO Particle Swarm Optimization

P2P Peer-to-Peer

POS-ELM Partitioned OS-ELM

RAN Resource Allocation Network

RAN-EKF Resource Allocation Network-Extended Kalman Filter

RBF Radial Basis Function

ReOS-ELM Regularized OS-ELM

RLS Recursive Least Squares

SAO-ELM Structure-adjustable Online ELM

SVD Singular Value Decomposition

SLFN Single Layer Feedforward Network

SVM Support Vector Machine

TOSELM Timeliness Online Sequential ELM

UCI University of California Irvine

UKF Unscented Kalman Filter

VAR Vector Autoregression

WOS-ELM Weighted Online Sequential ELM

Sumário

1 Introdução

Em aplicações de aprendizagem de máquina voltadas para problemas de regressão, tenta-se estimar uma função a partir de um conjunto de dados de treinamento. O objetivo é encontrar uma aproximação da função original tal que a diferença entre as duas seja mínima. Um algoritmo de aprendizagem para problemas de regressão tipicamente analisa as instâncias de treinamento e produz uma aproximação da função, a qual poderá ser usada para mapear novos exemplos.

Em muitos problemas de regressão é comum encontrar situações nas quais o conjunto de entrada não está totalmente disponível no início da fase de treinamento. As instâncias dos dados são fornecidas de forma sequencial aos modelos, conforme a sua disponibilidade. Para essa classe de problemas, a tarefa de aproximação de funções consiste em analisar as instâncias de treinamento de forma sequencial, gerando uma nova versão do modelo a cada nova instância ou bloco de dados de entrada. O conceito de aprendizado pode ser aplicado a todo método que permita a incorporação de informação a partir de instâncias de treinamento (DUDA; HART; STORK, 2012). Nesse contexto, os algoritmos de aprendizagem sequencial armazenam apenas a última informação aprendida e realizam a sua atualização somente quando uma nova instância de treinamento é fornecida. Após ter sido usada no aprendizado, a amostra é totalmente descartada, armazenando apenas a informação atualizada. Diversas soluções para aprendizagem sequencial são encontradas na literatura, e entre as mais comuns estão as soluções baseadas em mínimos quadrados (JAIN et al., 2014). Algoritmos de aprendizagem sequencial tornaram-se também comuns para soluções baseadas em redes neurais do tipo *Single Layer Feedforward Network* (SLFN), com unidades do tipo *Radial Basis Function* (RBF) e suas extensões (PLATT, 1991). Os métodos baseados nessa classe de algoritmos buscam reduzir os erros médios de previsão através da combinação de métodos de aprendizagem (LIANG et al., 2006), mecanismos de regularização (HUYNH; WON, 2011; GUO; HAO; LIU, 2014) e penalização (GU et al., 2014).

A abordagem de aprendizagem sequencial é capaz de tratar vários tipos de problemas que envolvam a aproximação de funções. Entre eles, encontram-se os problemas específicos de regressão e previsão de séries temporais. Nos problemas de regressão, procura-se ajustar sequencialmente os pesos da rede para encontrar os coeficientes que minimizam uma função de

custo relacionada com os dados de entrada. No problema de séries temporais, busca-se prever as próximas sequências de observações a partir dos padrões representados pelos dados sequenciais de entrada.

Entre os problemas encontrados nos diversos métodos de aprendizagem sequencial para SLFNs, encontra-se a possibilidade de singularidade e mal condicionamento da matriz que representa a saída da camada escondida (ZHAO; WANG; CHAI, 2013). A consequência direta desse problema é o aumento da variância no cálculo dos pesos de saída da rede, com o correspondente aumento no erro das estimativas geradas (ZHOU; LIU; ZHU, 2014). Além disso, é conhecido que os mecanismos tradicionais de tratamento dos efeitos da singularidade e mal condicionamento aumentam a complexidade computacional do processo de aprendizagem sequencial (GUO; HAO; LIU, 2014; NOBREGA; OLIVEIRA, 2015). Em aplicações que requerem o treinamento sequencial, como por exemplo, em problemas do mercado financeiro onde o fornecimento dos dados ocorre em alta frequência (KEARNS; NEVMYVAKA, 2013), o aumento da complexidade computacional se traduz na elevação do tempo de treinamento, com impacto direto no desempenho dos modelos gerados. Dessa forma, se faz importante a investigação de métodos que tratem os efeitos da singularidade e mal condicionamento em redes do tipo SLFN, sem o consequente aumento no tempo de treinamento.

1.1 Motivação

A maioria dos métodos recentes de aprendizagem sequencial para SLFNs utiliza a solução de mínimos quadrados para estimar os pesos de saída da rede (LIANG et al., 2006). Esses métodos tentam minimizar a norma composta pela diferença entre as previsões do modelo e as observações usadas no treinamento. A abordagem mais usada consiste em ajustar recursivamente os pesos de acordo com a correção da estimativa mais atual. Esse procedimento é realizado sempre que uma nova amostra de treinamento alimenta o algoritmo de forma sequencial. A implementação sequencial da solução de mínimos quadrados é similar ao algoritmo *Recursive Least Squares* (RLS) (ou mínimos quadrados recursivo, em tradução livre) (CANDY, 2005). Comparado com outras soluções similares de mínimos quadrados, o RLS apresenta como vantagem a rápida convergência para um valor estável do erro de previsão. Um problema encontrado nas implementações de SLFNs baseadas em RBF, é a possibilidade que a matriz da camada escondida seja singular ou mal condicionada, como consequência direta da escolha aleatória dos pesos de entrada da rede (ZHAO; WANG; CHAI, 2013). Um dos efeitos colaterais da singularidade e mal condicionamento é a impossibilidade da matriz da camada escondida ser positiva definida. Uma matriz positiva definida possui necessariamente autovalores positivos. Por sua vez, matrizes singulares possuem autovalores negativos ou próximos de zero e o seu determinante também poderá ser igual a zero, o que impede a sua inversão. Portanto, em métodos de aprendizagem sequencial cuja solução seja baseada em RLS, a singularidade e mal condicionamento pode tornar a solução de mínimos quadrados uma tarefa complexa do ponto

de vista computacional, sendo necessária a utilização de técnicas específicas para inversão de matrizes (GOLUB; VAN LOAN, 2012).

De acordo com o nosso conhecimento do estado da arte, os métodos de aprendizagem sequencial para SLFN utilizam mecanismos de regularização e penalização para tratar os efeitos da singularidade e mal condicionamento da matriz da camada escondida (GUO; HAO; LIU, 2014; GU et al., 2014). Porém, essas abordagens impõem a necessidade de otimização de parâmetros adicionais aos modelos. Além disso, as estratégias adotadas para o ajuste dos pesos na presença de singularidade e mal condicionamento implicam no aumento da complexidade computacional durante o treinamento, pois utilizam outros procedimentos iterativos, como por exemplo o método de Newton, para realizar essa tarefa (GU et al., 2014). Dessa forma, a investigação de uma abordagem que permita o tratamento dos efeitos da singularidade e mal condicionamento, sem aumentar a complexidade computacional do treinamento, pode resultar na descrição de métodos de aprendizagem sequencial mais estáveis para os problemas de regressão e previsão de séries temporais.

Um dos efeitos da singularidade e mal condicionamento é o surgimento de dependências lineares entre os elementos da matriz de covariância, usada no RLS para ajustar os pesos da rede. Esse fenômeno, conhecido na estatística como multicolinearidade, aumenta a variância das estimativas dos pesos quando ocorre uma pequena variação nas instâncias de treinamento. Mesmo com a utilização de técnicas conhecidas para inversão de matrizes singulares (PENROSE, 1955; GOLUB; VAN LOAN, 2012), na presença de multicolinearidade os elementos gerados podem ser numericamente imprecisos. A multicolinearidade em si não reduz o poder preditivo do modelo como um todo, mas o resultado da tarefa de aproximação de funções poderá gerar estimativas com aumento do erro de previsão, e em alguns casos, até causar a divergência na curva dos erros.

É conhecido na literatura que a multicolinearidade pode ser tratada de forma eficiente através de técnicas que mapeiam o problema de aprendizado na forma de um modelo linear dinâmico (WATSON, 1983). O filtro de Kalman (KALMAN, 1960) é uma implementação conhecida e eficaz dessa classe de modelos, sendo considerado um estimador ótimo para problemas sequenciais onde o conjunto de parâmetros varia com o tempo. Além disso, a resolução do problema de aprendizagem sequencial através do filtro de Kalman, pode reduzir a complexidade computacional do treinamento de SLFNs, pois é possível eliminar algumas operações de multiplicação e inversão de matrizes através da aplicação de técnicas de discretização matricial (SARKKA, 2007).

Até onde conhecemos, a aplicação do filtro de Kalman combinado com métodos de aprendizagem sequencial para SLFN não é investigada na literatura com o objetivo específico de tratar os efeitos da multicolinearidade. Teoricamente, todos os problemas de aprendizagem sequencial, tradicionalmente resolvidos com soluções de mínimos quadrados, poderiam ser igualmente resolvidos com uma combinação destes com o filtro de Kalman, com a vantagem, em tese, de se obter estimativas mais precisas com baixo custo computacional. Esse é o principal

fator motivacional para a criação de um novo método de aprendizagem sequencial.

1.2 Objetivos

O objetivo desse trabalho é a proposição, implementação e análise de um novo método para aprendizagem sequencial baseado na combinação de redes neurais do tipo *Single Layer Feedforward Network* (SLFN) com o filtro de Kalman, voltado para o tratamento dos efeitos da multicolinearidade. Embora já existam métodos conhecidos para a resolução do problema em questão, esses métodos sofrem com o aumento da complexidade computacional ao tratar a singularidade da matriz da camada escondida. Além disso, existe a necessidade de otimização de parâmetros adicionais para que o tratamento seja eficiente.

Os algoritmos do método proposto nesse trabalho, chamados de *Kalman Learning Machine* (KLM) e *Extended Kalman Learning Machine* (EKLM), possuem como único parâmetro o número de unidades da camada escondida, cujo valor pode ser facilmente otimizado pelo método *simplex* de Nelder-Mead (NELDER; MEAD, 1965). Dessa forma, o método proposto deverá ser capaz de gerar modelos preditivos com desempenho comparável aos métodos tradicionais de aprendizagem sequencial baseados em SLFN, levando em consideração os erros de previsão e o tempo de treinamento desses métodos.

Para avaliar o desempenho do método proposto, utilizamos nesse trabalho várias bases de dados conhecidas da comunidade de aprendizagem de máquina. As bases foram obtidas do repositório UCI (University of California Irvine - Machine Learning Repository) (BACHE; LICHMAN, 2013) ou geradas artificialmente para alguns problemas de previsão de séries temporais. O objetivo da utilização das bases, é a validação do método proposto em relação à sua adequação para a resolução de problemas de regressão e previsão de séries temporais, dentro do escopo de aprendizagem sequencial. O objetivo da validação é a comparação do desempenho do método proposto com métodos do estado da arte relacionados ao problema.

1.3 Estrutura do Documento

Nesse Capítulo, a motivação e os objetivos da tese foram apresentados. Os demais capítulos do documento abordam os seguintes assuntos:

- **Capítulo 2 - Aprendizagem Sequencial.** Esse capítulo oferece uma visão geral do problema de aprendizagem sequencial, mostrando os métodos usados como base para a resolução de problemas dessa classe. São apresentados em detalhes os principais algoritmos sequenciais baseados em SLFNs e a solução recursiva de mínimos quadrados. Uma introdução às principais extensões dos métodos é feita ao longo do capítulo e uma análise crítica das abordagens relacionadas com o tópico dessa tese é apresentada ao final.

- **Capítulo 3 - Filtro de Kalman.** O capítulo introduz as principais características do filtro de Kalman. A derivação de todas as equações do filtro é mostrada formalmente e uma discussão sobre o estado da arte para as estimativas de seus parâmetros é realizada. Também são apresentadas as principais extensões não lineares do filtro.

- **Capítulo 4 - Método Proposto.** Esse capítulo introduz o método proposto neste trabalho. A formulação do problema é apresentada matematicamente e o algoritmo KLM tem os seus passos detalhados de forma textual e em pseudocódigo. Uma versão não linear do algoritmo é apresentada e uma discussão sobre a abordagem para estimativa de parâmetros do método é mostrada em seguida. Ao final, é apresentada uma comparação com os métodos do estado da arte relacionados com o tópico dessa tese.

- **Capítulo 5 - Experimentos.** O capítulo apresenta os resultados dos experimentos realizados. A metodologia adotada é apresentada e os resultados da avaliação estatística dos experimentos são mostrados ao longo do capítulo. Também são apresentados os resultados de um estudo de caso para a previsão de séries temporais do mercado financeiro, voltado para o problema da arbitragem estatística.

- **Capítulo 6 - Conclusão e Trabalhos Futuros.** A conclusão deste trabalho é apresentada e as contribuições geradas são listadas. São apresentadas propostas de trabalhos futuros que podem ser gerados a partir dos métodos propostos nesta tese.

2

Aprendizagem Sequencial

Esse capítulo mostra uma visão geral do problema de aprendizagem sequencial, apresentando os métodos do estado da arte. São apresentados em detalhes os principais algoritmos sequenciais baseados em SLFNs e a solução recursiva de mínimos quadrados. Uma introdução às principais extensões dos métodos é feita ao longo do capítulo e uma análise crítica das abordagens relacionadas com o tópico dessa tese é apresentada ao final.

2.1 Contextualização

Uma rede neural, de forma geral, consiste de várias unidades de processamento interconectadas conhecidas como neurônios. Essas unidades são conectadas de acordo com uma topologia de rede que reproduz o funcionamento de neurônios biológicos (GUPTA; JIN; HOMMA, 2004). As principais características de uma rede neural podem ser divididas em duas partes: (i) características da arquitetura e (ii) propriedades funcionais. A arquitetura de uma rede neural está relacionada com a topologia utilizada na interconexão das unidades, as quais são subdivididas em entrada, saída, unidades da camada escondida e entradas de viés. Essas unidades estão dispostas em uma estrutura de camadas de acordo com a topologia adotada. As propriedades funcionais descrevem o procedimento para o aprendizado através da decomposição dos atributos da informação em elementos fundamentais, tipicamente chamados de pesos, os quais são armazenados nos bancos de memória interconectados (GUPTA; JIN; HOMMA, 2004).

Entre as diversas propriedades funcionais de uma rede neural, destacamos a capacidade de aprendizado, mesmo quando uma representação matemática do modelo em estudo não está explicitamente disponível (KARTALOPOULOS; KARTAKAPOULOS, 1997). Através de um algoritmo de aprendizagem, é possível derivar informação a partir de um conjunto de instâncias dos dados de treinamento, permitindo a construção de uma base de conhecimento descrita em termos de um conjunto de pesos que conectam as unidades da rede. Baseado no conhecimento representado pelos pesos, a rede é capaz de resolver uma gama de tarefas complexas, incluindo problemas de visão computacional (LUO; ZHANG, 2014), reconhecimento de voz (HINTON et al., 2012), previsão de séries temporais (GRIGORIEVSKIY et al., 2014), entre outras.

Os paradigmas de aprendizado para redes neurais podem ser divididos, quanto ao fornecimento dos dados, em duas categorias (JAIN et al., 2014): (i) aprendizagem *batch* ou *off-line* e (ii) aprendizagem sequencial ou *online*. Na primeira categoria, o processo de otimização dos pesos é realizado a partir de todas as instâncias de treinamento disponíveis, as quais são fornecidas de uma única vez à rede. Na segunda categoria, o conhecimento é atualizado de forma sequencial, após a apresentação de cada amostra de treinamento (SAAD, 2009).

Na aprendizagem *batch*, ao final do ciclo de treinamento, normalmente a rede é colocada em operação e nenhum treinamento adicional é necessário, de forma que o conhecimento aprendido mantém-se contante através dos pesos da rede. Esse paradigma é adequado para problemas nos quais o ambiente modelado pela rede é estacionário e as instâncias utilizadas no treinamento são suficientes para representar o conhecimento do problema em questão (JAIN et al., 2014). Entretanto, quando a rede em modo *batch* recebe dados que não foram usados no treinamento, não é possível absorver diretamente essa nova informação e transformá-la em conhecimento. Para que seja possível a incorporação do conhecimento gerado por novas instâncias de dados, é necessário o retreinamento da rede usando todo o conjunto de treinamento previamente utilizado.

Ao contrário da aprendizagem *batch*, a abordagem sequencial permite incorporar dinamicamente as novas informações, mesmo após o fim da fase de treinamento. Um sistema baseado em uma rede neural, com estratégia de treinamento sequencial, poderá atualizar dinamicamente os pesos da rede sempre que o estado do sistema mudar (CHOY; SRINIVASAN; CHEU, 2006). A abordagem sequencial permite formar uma base de conhecimento para problemas dinâmicos e esse tópico tem atraído a atenção para a pesquisa de novos métodos de treinamento para redes neurais. Quando comparados com os algoritmos *batch*, as soluções de aprendizagem sequencial possuem algumas características específicas (HUANG; SARATCHANDRAN; SUNDARARAJAN, 2005):

- As instâncias de treinamento alimentam o algoritmo de forma sequencial, recebendo uma instância por vez ou através de um bloco de instâncias;
- Em um dado instante, apenas uma instância de treinamento é vista pelo algoritmo durante o aprendizado;
- Uma instância de treinamento é descartada após ser vista em um passo do aprendizado;
- O processo de aprendizado não possui o conhecimento prévio das instâncias que alimentarão o algoritmo.

As propriedades listadas acima podem ser aplicadas à uma grande variedade de modelos de redes neurais. Na literatura, encontramos soluções de aprendizagem sequencial para *Multilayer Perceptron* (MLP) (ZHOU; LAI; YEN, 2012), *Radial Basis Function* (RBF) (HO; LEUNG; SUM, 2010), *wavelets* (EL-SOUSY, 2011), redes neurais recorrentes (LIN; CHANG;

LIN, 2013), redes neurais *fuzzy* (ZHANG et al., 2011), *Support Vector Machine* (SVM) (ZHENG et al., 2013) e outros modelos híbridos (WANG; ER; MENG, 2009). Uma revisão detalhada sobre as principais soluções de aprendizagem sequencial para os diversos modelos de redes neurais pode ser vista em JAIN et al. (2014).

Dos modelos citados, o MLP e o RBF são exemplos de redes do tipo *feedforward*, nas quais os sinais são transmitidos através de conexões das unidades de entrada até a saída. Devido à simplicidade de sua topologia, as redes *feedforward* são as mais utilizadas em aplicações práticas, com as suas unidades dispostas em uma ou mais camadas, conforme mostrado nas Figuras 2.1 e 2.2.

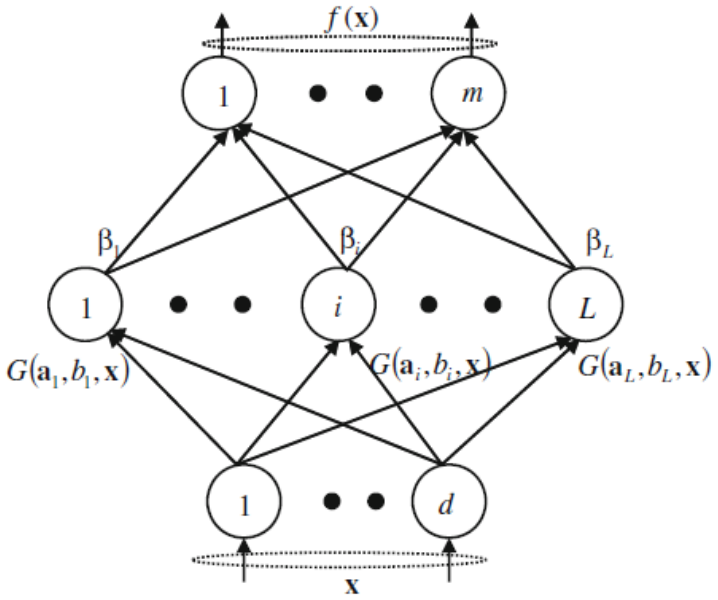


Figura 2.1: Rede *feedforward* com uma camada

Esse capítulo apresenta os principais métodos de aprendizagem sequencial para redes *feedforward* com uma única camada. Uma visão geral dos algoritmos do estado da arte e suas extensões é apresentada ao longo do capítulo. Ao final, é apresentada uma comparação entre os métodos relacionados com o tópico dessa tese.

2.2 Aprendizagem Sequencial para SLFN: Origens

Nas últimas décadas, as redes do tipo *Single Layer Feedforward Network* (SLFN) têm sido estudadas e o seu uso disseminado em aplicações de classificação de padrões e aproximação de funções, entre outras. Em LESHNO et al. (1993), é mostrado que qualquer função contínua poderia ser aproximada por uma SLFN através de uma função de ativação não polinomial. Para problemas de aproximação de funções em um conjunto de treinamento finito, o trabalho de HUANG; BABRI (1998) mostrou que uma SLFN com pelo menos N unidades na camada escondida e com uma função de ativação não linear, pode aprender exatamente N instâncias

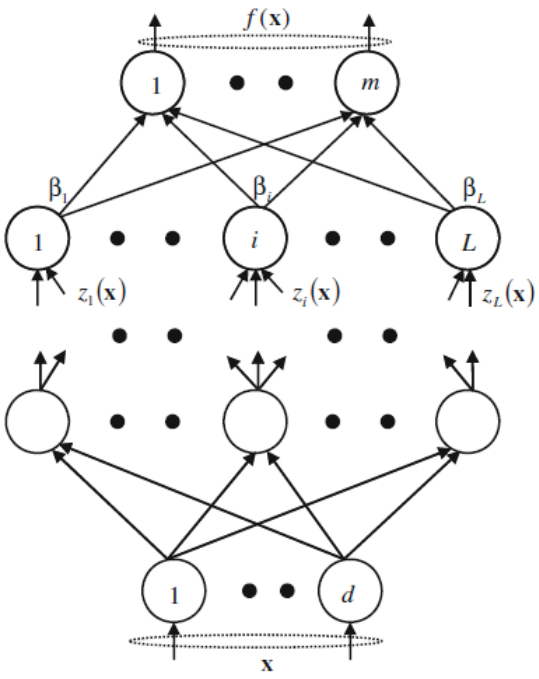


Figura 2.2: Rede *feedforward* com múltiplas camadas

distintas do conjunto de treinamento.

Na abordagem clássica para implementação de redes RBF, as funções de ativação Gaussianas estão entre as mais comuns e o número de unidades da camada escondida é escolhido previamente baseado nas propriedades dos dados de entrada. Os pesos que conectam as unidades da camada escondida e a saída da rede são calculados através de métodos baseados em mínimos quadrados, como por exemplo, *Least Mean Squares* (LMS) (MOODY; DARKEN, 1989; MUSAVI et al., 1992) e *Recursive Least Squares* (RLS) (CHEN; BILLINGS; GRANT, 1992). Essa abordagem gera um número elevado de unidades na camada escondida e está sujeita aos problemas relacionados com a dimensionalidade dos dados de treinamento (BORS; GABBOUJ, 1994). Uma contribuição importante para o tratamento desses problemas foi proposta em PLATT (1991), através do desenvolvimento de um algoritmo que adiciona unidades na camada escondida da rede com base na "inovação" dos dados de entrada. Essa abordagem é mais adequada para problemas de aprendizagem sequencial, pois o número de unidades da camada escondida está relacionada com a complexidade dos dados observados. A rede resultante, chamada de *Resource Allocation Network* (RAN), inicia sem nenhuma unidade na camada escondida e cresce através da alocação de novas unidades baseadas no critério de inovação dos dados de entrada. Os dados são apresentados sequencialmente, e se não existir inovação, os parâmetros da rede são ajustados através do algoritmo LMS para refletir o padrão dos dados observados.

O algoritmo da RAN foi desenvolvido como uma solução para o problema de aprendizagem em redes neurais com tamanho fixo. Em PLATT (1991), esse problema de aprendizagem foi descrito como NP-Completo. A solução para o problema é dada através da alocação de novas unidades à camada escondida, conforme novas instâncias de treinamento alimentam o algoritmo.

Levando em consideração que a curva do erro de previsão converge para um valor estável após o aprendizado de um certo número de instâncias, o nível de alocação de novas unidades tende a se estabilizar, o que permite que o tempo de treinamento seja reduzido para uma escala polinomial. Esse trabalho mostra a RAN como um esquema de memorização com adaptação, na qual a primeira parte é obtida através do armazenamento dos conjuntos de entrada e saída, de forma semelhante às técnicas baseadas em Janelas de Parzen e *K-Nearest Neighbour* (KNN). Nessa abordagem, um conjunto reduzido de observações é armazenado, com o tamanho do bloco de observações crescendo de forma linear, com uma eventual saturação desse tamanho. O algoritmo da RAN encontra o tamanho apropriado para a rede através da interpolação dos dados fornecidos.

Do ponto de vista arquitetural, a RAN é uma rede neural com uma única camada, com a sua saída sendo gerada em função dos padrões de entrada da rede, através de uma combinação linear das unidades da camada escondida. A formulação da saída da RAN é mostrada na equação a seguir:

$$f(\mathbf{x}) = \alpha_0 + \sum_{k=1}^{K} \alpha_k \phi_k(\mathbf{x}), \tag{2.1}$$

em que K é o número de unidades da camada escondida, $\phi_k(\mathbf{x})$ são as saídas das unidades da camada escondida, geradas a partir da entrada $\mathbf{x}$, e α_0 é o viés da rede. Os coeficientes α_k são os pesos que conectam as unidades da camada escondida com as unidades de saída da rede. A função de ativação das unidades da camada escondida é mostrada na equação seguinte:

$$\phi_k(\mathbf{x}) = \exp\left(-\frac{1}{\sigma_k^2}\|\mathbf{x} - u_k\|^2\right), \tag{2.2}$$

em que u_k é o centro da unidade ou a média da função Gaussiana e σ_k é a "largura" da mesma função. Essencialmente, a RAN é uma rede RBF acrescida do termo α_0, em que uma unidade da camada escondida armazena os padrões recebidos pela rede. Os pesos que conectam a camada escondida com a saída da rede, os coeficientes α_k, definem a contribuição de cada unidade escondida para uma resposta em particular da rede.

Uma instância da RAN inicia sem nenhuma unidade na camada escondida. A primeira observação $(\mathbf{x}_0, \mathbf{y}_0)$ é usada para inicializar o coeficiente $\alpha_0 = \mathbf{y}_0$, em que $\mathbf{y}_0$ é a saída esperada. De acordo com fornecimento sequencial das observações, a estrutura da rede cresce através da adição de novas unidades na camada escondida. A decisão sobre o armazenamento de uma observação $(\mathbf{x}_n, \mathbf{y}_n)$ depende de duas condições, descritas através das equações:

$$\|\mathbf{x}_n - u_{nr}\| > \varepsilon_n, \tag{2.3}$$

$$e_n = \mathbf{y}_n - f(\mathbf{x}_n) > e_{\min}, \tag{2.4}$$

em que u_{nr} é o padrão armazenado com proximidade a $\mathbf{x}_n$ no espaço de entrada e ε_n, $e_{\min}$ são limiares definidos empiricamente. A primeira condição diz que a entrada deverá estar suficientemente afastada dos padrões armazenados, com a distância acima do primeiro limiar. A

segunda condição reflete a significância do erro de previsão, de acordo com o segundo limiar. O valor $e_{\min}$ deve ser escolhido de modo a representar a precisão da saída da rede.

Quando uma nova unidade da camada escondida é acrescentada à rede, os pesos associados a ela são atribuídos da seguinte forma:

$$\alpha_{K+1} = e_n, \tag{2.5}$$

$$u_{K+1} = \mathbf{x}_n, \tag{2.6}$$

$$\sigma_{K+1} = \kappa \left\| \mathbf{x}_n - u_{nr} \right\|, \tag{2.7}$$

em que κ é um fator que determina a sobreposição das respostas das unidades da camada escondida. O valor da largura de σ_{K+1} é baseado na heurística do vizinho mais próximo. Quando a observação $(\mathbf{x}_n, \mathbf{y}_n)$ não atende às duas condições, o algoritmo *Least Mean Squares* (LMS) é usado para adaptar os parâmetros $\mathbf{w} = \left[\alpha_0, ..., \alpha_K, u_1^T, ..., u_K^T\right]^T$, de acordo com a seguinte equação:

$$\mathbf{w}_{(n)} = \mathbf{w}_{(n-1)} + \eta e_n a_n, \tag{2.8}$$

em que η é um fator de adaptação e $a_n = \nabla_w f(\mathbf{x}_n)$ é o gradiente da função $f(\mathbf{x}_n)$, definida na Equação 2.1, com respeito ao vetor de parâmetros $\mathbf{w}$, avaliado através de $\mathbf{w}_{(n-1)}$. A RAN inicia com $\varepsilon_n = \varepsilon_{max}$, a maior escala definida pelo tamanho do espaço de entrada, com densidade de probabilidade diferente de zero. A distância ε_n decai exponencialmente, de acordo com a expressão a seguir:

$$\varepsilon_n = \max \left\{ \varepsilon_{\max} \exp\left(\frac{-n}{\gamma_n}\right), \varepsilon_{\min} \right\}, \tag{2.9}$$

em que $0 < \gamma < 1$ é a constante de decaimento. O valor de ε_n decai até atingir o valor de ε_{min}. As principais limitações da RAN estão relacionadas à escolha correta dos parâmetros γ e $\varepsilon_{\min}$ e da convergência lenta em decorrência do uso do LMS.

Como alternativa à abordagem da RAN, em KADIRKAMANATHAN; NIRANJAN (1993) foram introduzidas algumas modificações no algoritmo para reduzir o impacto da escolha dos parâmetros γ e $\varepsilon_{\min}$ no desempenho dos modelos gerados. Nesse trabalho, os autores modificaram o processo de atualização de ε_n, o qual é reduzido gradualmente até atingir um valor mínimo escolhido. O critério de distância baseado em ε_n forma um limite inferior natural para a largura denotada por σ_n, de forma que $\sigma_n > \kappa\varepsilon_n$. O limite inferior de $\varepsilon_{\min}$ para ε_n define a outra faixa para toda função ϕ_k, dado por

$$\sigma_k > \kappa\varepsilon_{min}. \tag{2.10}$$

Essa modificação no algoritmo da RAN garante um limite na suavização da função de ativação como forma de prevenir o *overfitting* do modelo gerado. A principal inovação do trabalho foi a adaptação dos parâmetros da rede quando a alocação de uma nova unidade da

camada escondida não satisfaz os critérios de armazenamento. A RAN adapta os coeficientes α_k e o centro das unidades $\mathbf{u}_k$ quando o algoritmo não encontra nenhuma inovação nos dados sequenciais. Em KADIRKAMANATHAN; NIRANJAN (1993), o LMS é substituído pela versão estendida do filtro de Kalman (*Extended Kalman Filter* (EKF)) (MAYBECK, 1982), melhorando a taxa de convergência da RAN e diminuindo a complexidade da rede.

Dado um vetor de parâmetros $\mathbf{w}$, o algoritmo do EKF obtém a estimativa a posteriori de $\mathbf{w}_{(n)}$ a partir da estimativa a priori de $\mathbf{w}_{(n-1)}$ e da estimativa da covariância do erro $\mathbf{P}_{n-1}$, de acordo com a seguinte expressão:

$$\mathbf{w}_{(n)} = \mathbf{w}_{(n-1)} + e_n \mathbf{k}_n, \tag{2.11}$$

em que $\mathbf{k}_n$ é o vetor formado pela matriz de ganho de Kalman, calculado a partir da equação a seguir:

$$\mathbf{k}_n = \left[\mathbf{R}_n + \mathbf{a}_n^T \mathbf{P}_{n-1} \mathbf{a}_n\right]^{-1} \mathbf{P}_{n-1} \mathbf{a}_n, \tag{2.12}$$

com $\mathbf{a}_n$ sendo o vetor formado pelo gradiente do ruído da medição do filtro e $\mathbf{R}_n$ como a variância do mesmo ruído. A matriz de covariância do erro é atualizada recursivamente através da equação:

$$\mathbf{P}_n = \left[\mathbf{I} - \mathbf{k}_n \mathbf{a}_n^T\right] \mathbf{P}_{n-1}, \tag{2.13}$$

com $\mathbf{I}$ sendo a matriz identidade. Um problema presente em soluções que adotam modelos lineares dinâmicos é rápida convergência do algoritmo. No caso do EKF, isso pode levar à perda de generalização do modelo, medida pelo erro de previsão, visto que impede a adaptação aos dados sequenciais que ainda não foram vistos pela RAN. Para resolver esse problema, os autores adotaram um modelo de passeio aleatório para a atualização da matriz de covariância, a partir da seguinte formulação:

$$\mathbf{P}_n = \left[\mathbf{I} - \mathbf{k}_n \mathbf{a}_n^T\right] \mathbf{P}_{n-1} + q\mathbf{I}, \tag{2.14}$$

em que o parâmetro q é um escalar que determina o nível de aleatoriedade do algoritmo na direção do gradiente. A matriz de covariância do erro $\mathbf{P}_n$ é positiva e simétrica, com dimensões $p \times p$, em que p é o número de parâmetros adaptados. Sempre que uma nova unidade da camada escondida é adicionada à RAN, a dimensão de $\mathbf{P}_n$ cresce, fazendo com que novas linhas e colunas precisem ser inicializadas. Como $\mathbf{P}_n$ é uma estimativa do erro de covariância dos parâmetros, podemos inicializá-la da seguinte forma:

$$\mathbf{P}_n = \begin{pmatrix} \mathbf{P}_{n-1} & 0 \\ 0 & \mathbf{P}_0 \mathbf{I} \end{pmatrix}, \tag{2.15}$$

em que $\mathbf{P}_0$ é uma estimativa da incerteza dos valores atribuídos inicialmente aos parâmetros, o que implica que esse valor é igual à variância das observações $\mathbf{x}_n$ e $\mathbf{y}_n$. A dimensão da matriz identidade $\mathbf{I}$ é igual ao número de novos parâmetros introduzidos após a alocação de uma nova

unidade da camada escondida. As equações do EKF são utilizadas em substituição ao LMS no algoritmo original. Os autores nomearam o algoritmo de *Resource Allocation Network-Extended Kalman Filter* (RAN-EKF), mostrando que as alterações sugeridas aumentam a velocidade e a capacidade de generalização quando aplicadas aos problemas de aprendizagem sequencial.

As melhorias introduzidas no RAN-EKF conseguem produzir uma rede mais compacta, através da substituição do LMS pelo EKF. Entretanto, essas redes não são capazes de remover uma unidade da camada escondida quando esta não contribui para saída da rede. O trabalho apresentado em YINGWEI; SUNDARARAJAN; SARATCHANDRAN (1998) propõe uma solução para o problema, através de uma estratégia de poda para as unidades que não contribuem de forma significativa para o desempenho geral da rede. Para cada observação n, a saída das unidades da camada escondida é normalizada em relação ao valor máximo $|O^n_{\max}|$ obtido de todas as unidades da rede. Os valores normalizados são comparados com um certo limiar δ. Em seguida, as unidades da rede que apresentam valores abaixo de δ em M observações consecutivas são removidas. Os autores nomearam o novo algoritmo de *Minimal Resource Allocation Network* (M-RAN). A estratégia de poda para cada observação $(\mathbf{x}_n, \mathbf{y}_n)$ pode ser resumida nos seguintes passos:

1. Calcule a saída das unidades O^n_k, com $k = 1, ..., K$, através da equação:

$$O^n_k = \alpha_k \exp\left(-\frac{1}{\sigma_k^2}\|\mathbf{x}_n - \mu_k\|^2\right); \tag{2.16}$$

2. Encontre o maior valor absoluto de saída das unidades, $|O^n_{\max}|$;

3. Calcule os valores normalizados r^n_k, com $k = 1, ..., K$, através da equação:

$$r^n_k = \left|\frac{O^n_k}{O^n_{\max}}\right|; \tag{2.17}$$

4. Se $r^n_k < \delta$ para M observações consecutivas, remova a k-ésima unidade e reduza a dimensionalidade através do EKF.

O critério para alocar uma nova unidade à rede, originalmente proposto na RAN, foi ampliado na definição da M-RAN através da introdução de uma condição baseada no erro de previsão, calculado a partir da saída da rede. A condição utiliza uma janela deslizante de tamanho M e avalia o valor do erro e_{rmse}, comparando-o com um limiar $e_{\min}$ previamente estabelecido. A formalização da condição é mostrada na equação a seguir:

$$e_{rmse} = \sqrt{\frac{\sum_{i=n-M-1}^{n} e_i^2}{M}} > e_{\min}, \tag{2.18}$$

em que e_i é o i-ésimo erro de saída da rede, expresso da forma $e_i = \mathbf{y}_i - f(\mathbf{x}_i)$. Essa condição

adicional foi introduzida para garantir a suavização na curva de crescimento das unidades da rede. Após a avaliação dos critérios de alocação de novas unidades, o algoritmo M-RAN realiza as mesmas operações do RAN-EKF para atualização dos parâmetros da rede, através da versão estendida do filtro de Kalman. Ao final do processo, são aplicados os passos descritos anteriormente para remoção de unidades com baixa contribuição à saída da rede.

Uma característica em comum dos algoritmos RAN, RAN-EKF e M-RAN é a utilização de limiares escolhidos a priori para alocação de novas unidades da rede ou para remoção daquelas com baixa contribuição para os pesos de saída. É importante observar que as soluções acima precisam do conhecimento prévio da distribuição dos dados de entrada. O trabalho apresentado em HUANG; SARATCHANDRAN; SUNDARARAJAN (2004) mostra uma solução para esse problema através da definição do *Generalized Growing and Prunning RBF* (GGAP-RBF), no qual o nível de significância de uma unidade da camada escondida é calculado a partir de uma aproximação com a função Gaussiana, reduzindo significativamente o esforço computacional para alocar ou remover uma unidade da rede.

No GGAP-RBF, o processo para adicionar uma unidade à rede utiliza um critério de significância, o qual avalia a precisão da rede antes de uma nova alocação. Se a significância da unidade é maior que a precisão do aprendizado, uma nova unidade é adicionada. Caso contrário, essa unidade é removida. O algoritmo proposto em HUANG; SARATCHANDRAN; SUNDARARAJAN (2004) precisa verificar apenas a unidade mais próxima, baseada na distância euclidiana da instância atual de entrada. Quando os dados de treinamento indicam que não existe a necessidade de crescimento da rede, apenas os parâmetros da unidade mais próxima são ajustados. Dessa forma, a quantidade de computações do processo de aprendizado é reduzida.

Formalmente, o processo de aprendizado sequencial do GGAP-RBF inicia com a escolha aleatória de um conjunto de entradas $(\mathbf{x}_i, \mathbf{y}(\mathbf{x}_i))$, com $i = 1, 2, \ldots$. As instâncias de entrada, $\mathbf{x}_i$, possuem uma distribuição uniforme. Após aprender de forma sequencial n observações, o algoritmo cria uma rede RBF com K unidades. A saída da rede para uma entrada $\mathbf{x}_i$ é dada pela seguinte equação:

$$f_1(\mathbf{x}_i) = \sum_{j=1}^{K} \alpha_j \phi_j(\mathbf{x}_i), \tag{2.19}$$

em que K é o número de unidades da camada escondida e $\phi_j(\mathbf{x}_i)$ são as saídas das unidades da camada escondida, geradas a partir da entrada $\mathbf{x}_i$, conforme mostrado Equação 2.2. Os coeficientes α_j são os pesos que conectam as unidades da camada escondida com as unidades de saída da rede. Se a k-ésima unidade for removida da rede, a saída da rede para as *K-1* unidades é dada por:

$$f_2(\mathbf{x}_i) = \sum_{j=1}^{k-1} \alpha_j \phi_j(\mathbf{x}_i) + \sum_{j=k+1}^{K} \alpha_j \phi_j(\mathbf{x}_i). \tag{2.20}$$

Para uma dada observação $\mathbf{x}_i$, o erro resultante da remoção da k-ésima unidade é a diferença

absoluta entre $f_1(\mathbf{x}_i)$ e $f_2(\mathbf{x}_i)$, conforme a expressão:

$$E(k,i) = |f_1(\mathbf{x}_i) - f_2(\mathbf{x}_i)| = |\alpha_k| \phi_k(\mathbf{x}_i) \ , \tag{2.21}$$

em que *i=1,...,n*. De acordo com a equação acima, a média do erro para as n entradas sequenciais após a remoção da k-ésima unidade será:

$$E_{avg}(k) = \frac{\sum_{i=1}^{n} E(k,i)}{n} = \frac{|\alpha_k|}{n} \sum_{i=1}^{n} \phi_k(\mathbf{x}_i). \tag{2.22}$$

De acordo com a Equação 2.22, a significância de uma unidade da camada escondida é definida como a média das saídas sobre o conjunto de entrada. A significância é calculada como a distância média de duas funções RBF, antes e depois da remoção de uma unidade. Assumindo que as n observações estão uniformemente distribuídas, a significância da unidade k é calculada pela equação a seguir (HUANG; SARATCHANDRAN; SUNDARARAJAN, 2004):

$$E_{sig}(k) = \left| \frac{(1,8\sigma_k)^l \alpha_k}{S(\mathbf{X})} \right|, \tag{2.23}$$

em que l é a dimensão do espaço de entrada e $S(\mathbf{X})$ é o tamanho do domínio $\mathbf{X}$ no qual as instâncias foram escolhidas. O algoritmo introduz um novo critério para a adição de unidades, levando em consideração a equação acima. O novo critério para a chegada de uma nova observação $(\mathbf{x}_n, \mathbf{y}_n)$ é definido como:

$$\frac{(1,8k \|\mathbf{x}_n - \mu_{nr}\|)^l |e_n|}{S(\mathbf{X})} > e_{\min}, \tag{2.24}$$

em que $e_n = \mathbf{y}_n - f(\mathbf{x}_n)$ e μ_{nr} é o centro mais próximo de $\mathbf{x}_n$. Se a significância da unidade k for menor que a precisão esperada $e_{\min}$, a unidade será removida. O restante do algoritmo segue o mesmo processo do M-RAN, com o ajuste dos parâmetros sendo feito pela versão estendida do filtro de Kalman.

2.3 Extreme Learning Machine para Aprendizado Sequencial

Os algoritmos RAN, RAN-EKF, M-RAN e GGAP-RBF são implementações que permitem a aprendizagem sequencial para redes *feedforward* com unidades RBF. As principais limitações dos três primeiros métodos são a grande quantidade de parâmetros que precisam ser ajustados e o elevado esforço computacional requerido no treinamento. A introdução do GGAP-RBF permitiu a simplificação da topologia da rede, com a consequente redução do es-

força computacional. Ainda assim, o GGAP-RBF requer o conhecimento prévio da distribuição dos dados de entrada da rede, com o objetivo de adicionar ou podar as unidades da camada escondida. Além disso, os métodos acima não permitem que a aprendizagem ocorra com o fornecimento de blocos de dados de tamanhos diferentes. Ainda, só é possível armazenar a informação aprendida em unidades do tipo RBF, não sendo possível a utilização de outras funções de ativação.

Na literatura, são encontrados vários métodos de aprendizagem sequencial para redes *feedforward*. A maioria é baseada em abordagens de aprendizado que implementam versões sequenciais dos algoritmos de gradiente descendente (GONZÁLEZ; DORRONSORO, 2008) e *backpropagation* (CAMPOLUCCI et al., 1999; JUNG; KIM, 2008). Especificamente para redes com unidades RBF, encontramos diversas soluções de aprendizagem sequencial que envolvem mecanismos de correção do erro de previsão (YU et al., 2011), adaptação dinâmica dos parâmetros da rede (SURESH; SUNDARARAJAN, 2012) e alocação dinâmica das unidades RBF (LI et al., 2004; LEE; STREET, 2003). De forma genérica, as principais limitações dos métodos citados envolvem os mesmos problemas encontrados nos algoritmos RAN, RAN-EKF, M-RAN e GGAP-RBF. Soluções baseadas em gradiente descendente e *backpropagation* normalmente estão associadas a um elevado tempo de treinamento e um número elevado de parâmetros para ajustar (HUANG et al., 2015).

Uma abordagem que trata as principais limitações dos modelos tradicionais de redes neurais foi proposta em HUANG; ZHU; SIEW (2006) com a introdução da *Extreme Learning Machine* (ELM). Essa método de treinamento tem atraído a atenção da comunidade e diversas extensões para aprendizagem sequencial foram propostas, usando como base a ELM. No restante dessa seção são apresentados os conceitos básicos da versão *batch* do algoritmo e a derivação de sua solução para a aprendizagem sequencial.

2.3.1 Extreme Learning Machine

As redes neurais *feedforward* têm sido largamente estudadas e utilizadas desde a introdução do algoritmo *backpropagation* (RUHMELHART; HINTON; WILIAMS, 1986). Esse algoritmo é essencialmente um método de otimização de parâmetros baseado em um gradiente de primeira ordem. Métodos baseados em gradientes tipicamente possuem convergência lenta e estão sujeitos à problemas relacionados com mínimos locais. A introdução da ELM em HUANG; ZHU; SIEW (2006) trata diretamente essas limitações, de acordo com as suas definições para redes do tipo *Single Layer Feedforward Network* (SLFN). Na ELM, as unidades da camada escondida são inicializadas aleatoriamente e os seus valores permanecem constantes durante o treinamento. Os únicos parâmetros atualizados são os pesos que conectam a camada escondida com a camada de saída da rede. Essa abordagem permite o aprendizado com um baixo custo computacional e com a otimização de um único parâmetro: o número de unidades da camada escondida. Ainda em comparação com as abordagens tradicionais, a ELM não apenas tende

a encontrar o menor erro de treinamento, mas também tenta minimizar a norma dos pesos de saída. De acordo com a teoria das redes neurais (BARTLETT, 1998), para que as redes do tipo *feedforward* consigam o menor erro de treinamento, a norma dos pesos de saída deve ser a menor possível. Uma vantagem do algoritmo de treinamento baseado em ELM é que os parâmetros da camada escondida não precisam ser ajustados, com a solução para os pesos da saída sendo ajustados através do método de mínimos quadrados.

A capacidade de interpolação e aproximação de funções da ELM foi investigada em diversos trabalhos (HUANG; ZHU; SIEW, 2006; HUANG; CHEN; SIEW, 2006; HUANG; CHEN, 2007, 2008). Podemos discutir brevemente esses dois aspectos, olhando inicialmente para a definição de redes SLFN, a partir da equação:

$$f_L(\mathbf{x}) = \sum_{i=1}^{L} \boldsymbol{\beta}_i g_i(\mathbf{x}) = \sum_{i=1}^{L} \boldsymbol{\beta}_i G(\mathbf{a}_i, \mathbf{b}_i, \mathbf{x}), \tag{2.25}$$

em que g_i denota a função de ativação $G(\mathbf{a}_i, \mathbf{b}_i, \mathbf{x})$ para a i-ésima unidade da camada escondida, $\mathbf{x} \in \mathbb{R}^d$ e $\boldsymbol{\beta}_i \in \mathbb{R}^m$. L é o número de unidades da camada escondida, $\mathbf{a}_i$ são os pesos que conectam a camada de entrada com a camada escondida e $\mathbf{b}_i$ são as entradas de viés da rede. Para unidades aditivas com função de ativação g, g_i é definida como

$$g_i = G(\mathbf{a}_i, \mathbf{b}_i, \mathbf{x}) = g(\mathbf{a}_i \cdot \mathbf{x} + \mathbf{b}_i), \mathbf{a}_i \in \mathbb{R}^d, \mathbf{b}_i \in \mathbb{R} \tag{2.26}$$

e para unidades RBF com função de ativação g, g_i é definida como

$$g_i = G(\mathbf{a}_i, \mathbf{b}_i, \mathbf{x}) = g(\mathbf{b}_i \|\mathbf{x} - \mathbf{a}_i\|), \mathbf{a}_i \in \mathbb{R}^d, \mathbf{b}_i \in \mathbb{R}^+. \tag{2.27}$$

Em HUANG; HUANG (1991) e SARTORI; ANTSAKLIS (1991) foi demonstrado que N instâncias distintas podem ser aprendidas por redes SLFN com a alocação de N unidades escondidas com um limiar pré-definido. Em HUANG; BABRI (1998), foi demonstrado que uma SLFN com ao menos N unidades na camada escondida e com qualquer função de ativação não linear limitada e arbitrária com limite infinito, poderia aprender N instâncias distintas com erro zero. Vários trabalhos (HORNIK, 1991; LESHNO et al., 1993; PARK; SANDBERG, 1991; HORNIK; STINCHCOMBE; WHITE, 1989; CYBENKO, 1989; FUNAHASHI, 1989; STINCHCOMBE; WHITE, 1989) provaram na teoria que certas funções de ativação $g(\mathbf{x})$ que satisfazem condições específicas, possuem uma sequência de funções de rede $\{f_L\}$ que conseguem aproximar qualquer função contínua f com um erro de aprendizado $\varepsilon > 0$. Em todas as teorias convencionais de redes neurais, os parâmetros de qualquer f_L são livremente ajustáveis, incluindo os parâmetros da camada escondida $(\mathbf{a}_i, \mathbf{b}_i)$ e os pesos de saída da rede $\boldsymbol{\beta}_i$. De acordo com essas teorias, os parâmetros da camada escondida precisam ser ajustados para que seja possível aproximar f a partir de f_L. Para minimizar o impacto do ajuste dos parâmetros, alguns trabalhos propõem métodos incrementais para as redes SLFN (BARRON, 1993; KWOK;

YEUNG, 1997; MEIR; MAIOROV, 2000), os quais alocam dinamicamente unidades à camada escondida, deixando os parâmetros fixos após o respectivo ajuste. Ao contrário das abordagens tradicionais, todos os parâmetros da camada escondida da ELM não precisam ser ajustados, tornando-os independentes do conjunto de treinamento. Nas implementações típicas da ELM, os parâmetros $(\mathbf{a}_i, \mathbf{b}_i)$ são gerados aleatoriamente.

Para N instâncias distintas e arbitrárias $(\mathbf{x}_i, \mathbf{t}_i) \in \mathbb{R}^d \times \mathbb{R}^m$, uma rede SLFN com L unidades na camada escondida é modelada matematicamente de acordo com a equação:

$$\sum_{i=1}^{L} \boldsymbol{\beta}_i g_i\left(\mathbf{x}_j\right) = \sum_{i=1}^{L} \boldsymbol{\beta}_i G\left(\mathbf{a}_i, \mathbf{b}_i, \mathbf{x}_j\right) = \mathbf{o}_j, j = 1, ..., N. \tag{2.28}$$

Para que a SLFN possa aproximar as N instâncias com erro zero, teríamos $\sum_{j=1}^{L} \left\|\mathbf{o}_j - \mathbf{x}_j\right\| = 0$, ou seja, existe $(\mathbf{a}_i, \mathbf{b}_i)$ e $\boldsymbol{\beta}_i$ tal que:

$$\sum_{i=1}^{L} \boldsymbol{\beta}_i G\left(\mathbf{a}_i, \mathbf{b}_i, \mathbf{x}_j\right) = \mathbf{t}_j, j = 1, ..., N. \tag{2.29}$$

As N equações acima podem ser escritas de forma compacta:

$$\mathbf{H}\boldsymbol{\beta} = \mathbf{T}, \tag{2.30}$$

em que

$$\mathbf{H} = \begin{bmatrix} h\left(\mathbf{x}_1\right) \\ \vdots \\ h\left(\mathbf{x}_N\right) \end{bmatrix} = \begin{pmatrix} G\left(\mathbf{a}_1, \mathbf{b}_1, \mathbf{x}_1\right) & \ldots & G\left(\mathbf{a}_L, \mathbf{b}_L, \mathbf{x}_1\right) \\ \vdots & \cdots & \vdots \\ G\left(\mathbf{a}_1, \mathbf{b}_1, \mathbf{x}_N\right) & \cdots & G\left(\mathbf{a}_L, \mathbf{b}_L, \mathbf{x}_N\right) \end{pmatrix}_{N \times L,} \tag{2.31}$$

$$\boldsymbol{\beta} = \begin{bmatrix} \boldsymbol{\beta}_1^T \\ \vdots \\ \boldsymbol{\beta}_L^T \end{bmatrix}_{L \times m,} \tag{2.32}$$

$$\mathbf{T} = \begin{bmatrix} \mathbf{t}_1^T \\ \vdots \\ \mathbf{t}_N^T \end{bmatrix}_{N \times m.} \tag{2.33}$$

$\mathbf{H}$ é chamada de matriz de saída da camada escondida, na qual a i-ésima coluna de $\mathbf{H}$ é a saída da i-ésima unidade para as entradas $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N$. Na literatura, a matriz $\mathbf{H}$ também é referenciada como sendo a projeção aleatória do conjunto de entrada (HUANG, 2014).

A função $h\left(\mathbf{x}\right) = G\left(\mathbf{a}_1, \mathbf{b}_1, \mathbf{x}\right), ..., G\left(\mathbf{a}_L, \mathbf{b}_L, \mathbf{x}\right)$ é chamada de mapeamento de características da camada escondida. A i-ésima linha de $\mathbf{H}$ é o mapeamento em relação à i-ésima entrada $\mathbf{x}_i : h\left(\mathbf{x}_i\right)$. Em HUANG; ZHU; SIEW (2006), é feita uma análise rigorosa da capacidade de interpolação da ELM, na qual é mostrada formalmente que os parâmetros da camada escondida

podem ser gerados aleatoriamente se a função de ativação g for infinitamente diferenciável em qualquer intervalo. Em HUANG; CHEN; SIEW (2006) é apresentada uma prova para o teorema de aproximação universal da ELM, o qual estabelece que unidades aditivas ou RBF geradas aleatoriamente podem aproximar qualquer função contínua em um subespaço compacto $\mathbf{X} \in \mathbb{R}^d$. Entretanto, como encontrar os parâmetros $(\mathbf{a}_i, \mathbf{b}_i)$ mais adequados ainda é um problema em aberto da literatura da ELM.

Para implementar a ELM, é necessária a execução de três passos:

1. A camada escondida não precisa ser ajustada de forma iterativa (HUANG; ZHU; SIEW, 2006);

2. De acordo com a teoria das redes *feedforward* (BARTLETT, 1998), tanto o erro de treinamento $\|\mathbf{H}\boldsymbol{\beta} - \mathbf{T}\|$ como a norma dos pesos $\|\boldsymbol{\beta}\|$ precisam ser minimizados;

3. O mapeamento de características da camada escondida precisa satisfazer as condições de aproximação universal (HUANG; CHEN, 2008).

Como as unidades da camada escondida são geradas aleatoriamente, o vetor de pesos $\boldsymbol{\beta}_i$ pode ser estimado diretamente através de uma solução de mínimos quadrados ordinários. O treinamento de uma rede SLFN baseada na ELM é equivalente a encontrar a solução de mínimos quadrados $\hat{\boldsymbol{\beta}}$ do sistema linear $\mathbf{H}\boldsymbol{\beta} = \mathbf{T}$, de acordo com a expressão:

$$\left\|\mathbf{H}\hat{\boldsymbol{\beta}} - \mathbf{T}\right\| = \min_{\beta} \|\mathbf{H}\boldsymbol{\beta} - \mathbf{T}\|. \tag{2.34}$$

Se o número L de unidades da camada escondida for igual ao número N de instâncias distintas de treinamento, a matriz $\mathbf{H}$ é quadrada e inversível quando os parâmetros $(\mathbf{a}_i, \mathbf{b}_i)$ são escolhidos aleatoriamente. Como consequência, a SLFN pode aproximar as instâncias de treinamento com erro zero. Entretanto, na maioria dos casos, o número de unidades da camada escondida é menor que o número de instâncias distintas, de forma que $L \ll N$. Nesses casos, $\mathbf{H}$ não é quadrada e pode não existir $\mathbf{a}_i, \mathbf{b}_i, \boldsymbol{\beta}_i$, com $i = 1, ..., L$, tal que $\mathbf{H}\boldsymbol{\beta} = \mathbf{T}$. Assim, a solução para a menor norma através de mínimos quadrados passa a ser o sistema linear:

$$\hat{\boldsymbol{\beta}} = \mathbf{H}^{\dagger}\mathbf{T}, \tag{2.35}$$

em que $\mathbf{H}^{\dagger}$ é a matriz inversa generalizada de Moore-Penrose da matriz $\mathbf{H}$ (BLUM, 2012). Para o cálculo de $\mathbf{H}^{\dagger}$, são encontradas diversas soluções na literatura, incluindo a projeção ortogonal, ortogonalização, métodos iterativos e *Singular Value Decomposition* (SVD), sendo essa última a mais comum encontrada nas implementações da ELM. O algoritmo da ELM pode ser resumido de acordo com os seguintes passos:

Algoritmo 1: Extreme Learning Machine

Entrada:
Conjunto de treinamento $\aleph = \left\{(\mathbf{x}_i, \mathbf{t}_i) \,|\mathbf{x}_i \in \mathbb{R}^d, \mathbf{t}_i \in \mathbb{R}^m, i = 1, ..., N\right\}$,
Função de ativação $G(\mathbf{a}_i, \mathbf{b}_i, x)$,
Número de unidades da camada escondida L.
Saída: Vetor de pesos de saída $\boldsymbol{\beta}$

begin
  Passo 1: Gerar aleatoriamente os parâmetros da camada escondida $(\mathbf{a}_i, \mathbf{b}_i), i = 1, ..., L$.
  Passo 2: Calcular a matriz de saída da camada escondida $\mathbf{H}$ usando a Equação 2.31.
  Passo 3: Calcular o vetor de pesos de saída $\boldsymbol{\beta} = \mathbf{H}^{\dagger}\mathbf{T}$.
end

2.3.2 Mínimos Quadrados Recursivo

Antes de aplicar a solução da ELM para problemas de aprendizagem sequencial, vamos discutir brevemente nessa subseção a solução de mínimos quadrados para esse tipo de algoritmo. De acordo com a teoria da estimação, soluções de mínimos quadrados sequenciais são equivalentes à sua versão recursiva (CANDY, 2005). Nesse contexto, as técnicas de estimativas recursivas evoluíram nas últimas décadas a partir do uso intensivo de métodos computacionais. É importante observar que a solução de mínimos quadrados recursivo é idêntica à sua solução em modo *batch*, pois ambos convergem para o mesmo valor, não sendo possível afirmar que uma solução seja melhor que a outra. No entanto, o número de ciclos computacionais para executar por completo a solução recursiva é menor que a sua versão *batch*. A abordagem recursiva para o problema de mínimos quadrados é parte fundamental da base teórica que define as técnicas de estimativas adaptativas, como por exemplo, o filtro de Kalman (KAILATH, 1981).

A estratégia padrão para estimativas recursivas é baseada no conceito de correção ou atualização da estimativa atual ($\hat{\Theta}_{old}$). O procedimento consiste em corrigir a estimativa atual sempre que as instâncias do conjunto de treinamento alimentam o algoritmo de forma sequencial. As estimativas seguem a seguinte forma recursiva:

$$\hat{\Theta}_{new} = \hat{\Theta}_{old} + KE_{new}, \tag{2.36}$$

em que $E_{new} = Y - \hat{Y}_{old} = Y - C(\Theta_{old})$, para o valor medido Y e a sua previsão $\hat{Y}_{old}$, a qual é baseada na estimativa passada $C(\Theta_{old})$. A equação acima obtém uma nova estimativa a partir da anterior, corrigida por um peso K do erro E_{new}. O termo do erro E_{new} é conhecido também por inovação, ou seja, é a diferença entre o valor medido da amostra e a previsão para esse valor, baseada na estimativa passada. O cálculo da matriz de pesos K depende do critério usado para convergência da estimativa, como por exemplo, o erro médio quadrado. Para desenvolver a forma recursiva da equação acima para uma medida escalar $\mathbf{y}(t)$, vamos inicialmente nos basear

no estimador de mínimos quadrados em sua forma *batch*. Seja

$$\mathbf{y}(t-1) = \begin{bmatrix} \mathbf{y}(1) \\ \vdots \\ \mathbf{y}(t-1) \end{bmatrix}, \Theta(t-1) = \begin{bmatrix} \boldsymbol{\theta}_1 \\ \vdots \\ \boldsymbol{\theta}_{N_\theta} \end{bmatrix}, \mathbf{C}(t-1) = \begin{bmatrix} \mathbf{c}^\mathsf{T}(1) \\ \vdots \\ \mathbf{c}^\mathsf{T}(t-1) \end{bmatrix}$$
$$\Theta \in \mathbb{R}^{N_\Theta \times 1}, \mathbf{c}^\mathsf{T} \in \mathbb{R}^{1 \times N_\Theta}, \mathbf{y} \in \mathbb{R}^{(t-1)\times 1}, \mathbf{C} \in \mathbb{R}^{(t-1)\times N_\Theta},$$

respectivamente o conjunto dos $(t-1)$ valores medidos, a previsão dos valores medidos e as suas estimativas. A estimativa de mínimos quadrados de $\boldsymbol{\theta}$, baseada nos dados passados até $(t-1)$, é dada pela expressão a seguir (CANDY, 2005):

$$\hat{\Theta}_{LS}(t-1) = \mathbf{P}(t-1)\,\mathbf{C}^\mathsf{T}(t-1)\,\mathbf{y}(t-1), \tag{2.37}$$

em que $\mathbf{P}(t-1) = \left[\mathbf{C}^\mathsf{T}(t-1)\,\mathbf{C}(t-1)\right]^{-1}$. Suponha agora que precisamos adicionar um novo escalar $\mathbf{y}(t)$ ao conjunto de valores medidos. De acordo com a equação anterior, temos:

$$\Theta_{LS}(t) = \mathbf{P}(t)\,\mathbf{C}^\mathsf{T}(t)\,\mathbf{y}(t). \tag{2.38}$$

A matriz $\mathbf{C}^\mathsf{T}$ pode ser expandida em termos das instâncias passadas e da atual, na forma:

$$\mathbf{C}^\mathsf{T}(t) = [\mathbf{c}(1)\cdots\mathbf{c}(t-1)\,|\mathbf{c}(t)] = \left[\mathbf{C}^\mathsf{T}(t-1)\,|\mathbf{c}(t)\right]. \tag{2.39}$$

De forma similar, o vetor de valores medidos pode ser expandido na forma:

$$\mathbf{y}(t) = [\mathbf{y}(t-1)\,|\mathbf{y}(t)]. \tag{2.40}$$

Substituindo essas relações pela Equação 2.38 e multiplicando, temos:

$$\mathbf{P}(t) = \left[\mathbf{C}^\mathsf{T}(t-1)\,\mathbf{y}(t-1) + \mathbf{c}(t)\,y(t)\right]. \tag{2.41}$$

Reescrevendo $\mathbf{P}(t)$ em termos de $\mathbf{C}^\mathsf{T}$ através da Equação 2.39, temos:

$$\mathbf{P}(t) = \left[\mathbf{P}^{-1}(t-1) + \mathbf{c}(t)\,\mathbf{c}^\mathsf{T}(t)\right]^{-1}. \tag{2.42}$$

Para prosseguir com a formalização da versão recursiva dos mínimos quadrados, devemos relembrar o lema de inversão de matriz (VAN TREES, 2004), em que $[\mathbf{A} + \mathbf{BCD}]^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{B}\left[\mathbf{DA}^{-1}\mathbf{B} + \mathbf{C}^{-1}\right]^{-1}\mathbf{DA}^{-1}$. Se fizermos $\mathbf{A} = \mathbf{P}^{-1}(t-1), \mathbf{B} = \mathbf{c}(t), \mathbf{C} = 1, \mathbf{D} = \mathbf{c}^\mathsf{T}(t)$, temos:

$$\mathbf{P}(t) = \mathbf{P}(t-1) - \mathbf{P}(t-1)\,\mathbf{c}(t)\left[\mathbf{c}^\mathsf{T}(t)\,\mathbf{P}(t-1)\,\mathbf{c}(t) + 1\right]^{-1}\mathbf{c}^\mathsf{T}(t)\,\mathbf{P}(t-1). \tag{2.43}$$

Vamos definir $\mathbf{K}$ como a matriz de ganho, de acordo com a seguinte expressão:

$$\mathbf{K}(t) = \mathbf{P}(t-1)\,\mathbf{c}(t)\left[\mathbf{c}^{\mathsf{T}}(t)\,\mathbf{P}(t-1)\,\mathbf{c}(t)+1\right]^{-1}. \tag{2.44}$$

Reescrevendo $\mathbf{P}(t)$ pela equação acima, temos:

$$\mathbf{P}(t) = \left[\mathbf{I}-\mathbf{K}(t)\,\mathbf{c}^{\mathsf{T}}(t)\right]\mathbf{P}(t-1). \tag{2.45}$$

Substituindo esse resultado pela Equação 2.41, nós obtemos:

$$\Theta_{LS}(t) = \left[\mathbf{I}-\mathbf{K}(t)\,\mathbf{c}^{\mathsf{T}}(t)\right]\mathbf{P}(t-1)\left[\mathbf{C}^{\mathsf{T}}(t-1)\,\mathbf{y}(t-1)+\mathbf{c}(t)\,\mathbf{y}(t)\right]. \tag{2.46}$$

Dessa forma, usando a Equação 2.37, a estimativa de mínimos quadrados *LS* da Equação 2.38 torna-se:

$$\Theta_{LS}(t) = \left[\mathbf{I}-\mathbf{K}(t)\,\mathbf{c}^{\mathsf{T}}(t)\right]\Theta_{LS}(t-1)+\left[\mathbf{I}-\mathbf{K}(t)\,\mathbf{c}^{\mathsf{T}}(t)\right]\mathbf{P}(t-1)\,\mathbf{c}(t)\,\mathbf{y}(t). \tag{2.47}$$

Concentrando no último termo da expressão e realizando as multiplicações necessárias, podemos reescrevê-lo como

$$\left[\mathbf{I}-\mathbf{K}(t)\,\mathbf{c}^{\mathsf{T}}(t)\right]\mathbf{P}(t-1)\,\mathbf{c}(t)\,\mathbf{y}(t) = \left[\mathbf{P}(t-1)\,\mathbf{c}(t)-\mathbf{K}(t)\,\mathbf{c}^{\mathsf{T}}(t)\,\mathbf{P}(t-1)\,\mathbf{c}(t)\right]\mathbf{y}(t).$$

Substituindo $\mathbf{K}(t)$, temos:

$$\left[\mathbf{P}(t-1)\,\mathbf{c}(t)-\mathbf{P}(t-1)\,\mathbf{c}(t)\left[\mathbf{c}^{\mathsf{T}}(t)\,\mathbf{P}(t-1)\,\mathbf{c}(t)+1\right]^{-1}\mathbf{c}^{\mathsf{T}}(t)\,\mathbf{P}(t-1)\,\mathbf{c}(t)\right]\mathbf{y}(t).$$

Fatorando $\mathbf{P}(t-1)\,\mathbf{c}(t)$ do lado esquerdo, temos:

$$\mathbf{P}(t-1)\,\mathbf{c}(t)\left[\mathbf{I}-\left[\mathbf{c}^{\mathsf{T}}(t)\,\mathbf{P}(t-1)\,\mathbf{c}(t)+1\right]^{-1}\mathbf{c}^{\mathsf{T}}(t)\,\mathbf{P}(t-1)\,\mathbf{c}(t)\right]\mathbf{y}(t).$$

Fatorando a inversão a partir do lado esquerdo da expressão, temos:

$$\mathbf{P}(t-1)\,\mathbf{c}(t)\left[\mathbf{c}^{\mathsf{T}}(t)\,\mathbf{P}(t-1)\,\mathbf{c}(t)+1\right]^{-1}\left\{\left[\mathbf{c}^{\mathsf{T}}(t)\,\mathbf{P}(t-1)\,\mathbf{c}(t)+1\right]-\mathbf{c}^{\mathsf{T}}(t)\,\mathbf{P}(t-1)\,\mathbf{c}(t)\right\}\mathbf{y}(t).$$

Todo o termo acima é simplesmente reduzido a $\mathbf{K}(t)\,\mathbf{y}(t)$. Logo, substituindo essa expressão pela Equação 2.47, nós obtemos:

$$\Theta_{LS}(t) = \left[I-K(t)\,c^{\mathsf{T}}(t)\right]\Theta_{LS}(t-1)+K(t)\,y(t), \tag{2.48}$$

o qual multiplicando, expandindo e combinando os termos, nos dá o estimador de mínimos

quadrados recursivo, de acordo com a equação:

$$\boldsymbol{\Theta}_{LS}(t) = \boldsymbol{\Theta}_{LS}(t-1) + \mathbf{K}(t)\left[\mathbf{y}(t) - \mathbf{c}^{\mathsf{T}}(t)\,\boldsymbol{\Theta}_{LS}(t-1)\right]. \tag{2.49}$$

A base da solução de aprendizagem sequencial da ELM é derivada do estimador de mínimos quadrados da Equação 2.49, conforme veremos na próxima subseção.

2.3.3 Online Sequential Extreme Learning Machine

A solução da *Online Sequential Extreme Learning Machine* (OS-ELM) tem a sua origem na versão *batch* do algoritmo ELM (LIANG et al., 2006). Para as unidades aditivas da camada escondida da OS-ELM, os parâmetros são gerados aleatoriamente e os pesos de saída são determinados de forma analítica através da solução de mínimos quadrados recursivo, apresentado na subseção anterior. Da mesma forma, unidades RBF da OS-ELM possuem os centros e as larguras determinados aleatoriamente, com a mesma solução para a estimativa dos pesos de saída da rede. Ao contrário das soluções clássicas de aprendizagem sequencial, a OS-ELM não precisa ajustar os seus parâmetros durante o processo de treinamento. O único parâmetro previamente ajustável é a quantidade de unidades da camada escondida.

A versão *batch* da ELM, discutida na Seção 2.3.1, precisa que todo o conjunto de treinamento, composto por N instâncias, esteja disponível no início do processo. Entretanto, em aplicações reais os dados podem chegar um-a-um ou de bloco-em-bloco. O algoritmo da ELM foi modificado em LIANG et al. (2006) para torná-lo adequado aos problemas sequenciais. A matriz $\hat{\boldsymbol{\beta}}$, calculada pela Equação 2.35, é uma solução de mínimos quadrados da Equação 2.30. Logo, a matriz inversa generalizada de Moore-Penrose $\mathbf{H}^{\dagger}$ pode ser descrita da seguinte forma:

$$\mathbf{H}^{\dagger} = \left(\mathbf{H}^{T}\mathbf{H}\right)^{-1}\mathbf{H}^{T}. \tag{2.50}$$

Após a substituição da Equação 2.50 na Equação 2.35, a matriz $\hat{\boldsymbol{\beta}}$ pode ser reescrita de acordo com a equação:

$$\hat{\boldsymbol{\beta}} = \left(\mathbf{H}^{T}\mathbf{H}\right)^{-1}\mathbf{H}^{T}\mathbf{T}. \tag{2.51}$$

A OS-ELM resulta da implementação sequencial da solução de mínimos quadrados para a Equação 2.51. Seja um bloco inicial do conjunto de treinamento $\aleph_0 = \{(\mathbf{x}_i, \mathbf{t}_i)\}_{i=1}^{N_0}$, com $N_0 \geq L$, em que L é a quantidade de unidades da camada escondida. De acordo com a Equação 2.34, devemos considerar o problema de minimizar a norma de $\|\mathbf{H}_0\boldsymbol{\beta} - \mathbf{T}_0\|$, em que

$$\mathbf{H}_0 = \begin{pmatrix} G(\mathbf{a}_1, \mathbf{b}_1, \mathbf{x}_1) & \cdots & G(\mathbf{a}_L, \mathbf{b}_L, \mathbf{x}_1) \\ \vdots & \cdots & \vdots \\ G(\mathbf{a}_1, \mathbf{b}_1, \mathbf{x}_{N_0}) & \cdots & G(\mathbf{a}_L, \mathbf{b}_L, \mathbf{x}_{N_0}) \end{pmatrix}_{N_0 \times L} \tag{2.52}$$

$$\mathbf{T}_0 = \begin{bmatrix} \mathbf{t}_1^T \\ \vdots \\ \mathbf{t}_{N_0}^T \end{bmatrix}_{N_0 \times m.} \tag{2.53}$$

A solução para minimizar $\|\mathbf{H}_0\boldsymbol{\beta} - \mathbf{T}_0\|$ é dada por $\boldsymbol{\beta}^{(0)} = \mathbf{K}_0^{-1}\mathbf{H}_0^{\mathbf{T}}T_0$, em que $\mathbf{K}_0 = \mathbf{H}_0^T\mathbf{H}_0$ (HUANG; ZHU; SIEW, 2006). Para um novo bloco de dados $\aleph_1 = \{(\mathbf{x}_i, \mathbf{t}_i)\}_{i=N_0+1}^{N_0+N_1}$, com $\aleph_1$ denotando a quantidade de instâncias no novo bloco, o problema de minimização da norma passa a ser o seguinte:

$$\left\| \begin{bmatrix} \mathbf{H}_0 \\ \mathbf{H}_1 \end{bmatrix} \boldsymbol{\beta} - \begin{bmatrix} \mathbf{T}_0 \\ \mathbf{T}_1 \end{bmatrix} \right\|, \tag{2.54}$$

em que $\mathbf{H}_1$ e $\mathbf{T}_1$ são definidos a seguir:

$$\mathbf{H}_1 = \begin{pmatrix} G(\mathbf{a}_1, \mathbf{b}_1, \mathbf{x}_{N_0+1}) & \cdots & G(\mathbf{a}_L, \mathbf{b}_L, \mathbf{x}_{N_0+1}) \\ \vdots & \cdots & \vdots \\ G(\mathbf{a}_1, \mathbf{b}_1, \mathbf{x}_{N_0+N_1}) & \cdots & G(\mathbf{a}_L, \mathbf{b}_L, \mathbf{x}_{N_0+N_1}) \end{pmatrix}_{N_1 \times L,} \tag{2.55}$$

$$\mathbf{T}_1 = \begin{bmatrix} \mathbf{t}_{N_0+1}^T \\ \vdots \\ \mathbf{t}_{N_0+N_1}^T \end{bmatrix}_{N_1 \times m.} \tag{2.56}$$

Considerando os dois blocos de dados de treinamento, $\aleph_0$ e $\aleph_1$, os pesos de saída $\boldsymbol{\beta}$ são calculados de acordo com a expressão:

$$\boldsymbol{\beta}^{(1)} = \mathbf{K}_1^{-1} \begin{bmatrix} \mathbf{H}_0 \\ \mathbf{H}_1 \end{bmatrix}^T \begin{bmatrix} \mathbf{T}_0 \\ \mathbf{T}_1 \end{bmatrix} \tag{2.57}$$

em que

$$\mathbf{K}_1 = \begin{bmatrix} \mathbf{H}_0 \\ \mathbf{H}_1 \end{bmatrix}^T \begin{bmatrix} \mathbf{H}_0 \\ \mathbf{H}_1 \end{bmatrix}. \tag{2.58}$$

Para o processo de aprendizado sequencial, tivemos que expressar $\boldsymbol{\beta}^{(1)}$ em função de $\boldsymbol{\beta}^{(0)}$, $\mathbf{K}_1$, $\mathbf{H}_1$ e $\mathbf{T}_1$, de forma independente do bloco inicial $\aleph_0$. Dessa forma, $\mathbf{K}_1$ pode ser escrito como

$$\begin{aligned} \mathbf{K}_1 &= \begin{bmatrix} \mathbf{H}_0^T & \mathbf{H}_1^T \end{bmatrix} \begin{bmatrix} \mathbf{H}_0 \\ \mathbf{H}_1 \end{bmatrix} \\ &= \mathbf{K}_0 + \mathbf{H}_1^T\mathbf{H}_1 \end{aligned} \tag{2.59}$$

e

$$\begin{aligned}\begin{bmatrix}\mathbf{H}_0\\ \mathbf{H}_1\end{bmatrix}^T\begin{bmatrix}\mathbf{T}_0\\ \mathbf{T}_1\end{bmatrix} &= \mathbf{H}_0^T\mathbf{T}_0+\mathbf{H}_1^T\mathbf{T}_1\\ &= \mathbf{K}_0\mathbf{K}_0^{-1}\mathbf{H}_0^T\mathbf{T}_0+\mathbf{H}_1^T\mathbf{T}_1\\ &= \mathbf{K}_0\boldsymbol{\beta}^{(0)}+\mathbf{H}_1^T\mathbf{T}_1\\ &= \left(\mathbf{K}_1-\mathbf{H}_1^T\mathbf{H}_1\right)\beta^{(0)}+\mathbf{H}_1^T\mathbf{T}_1\\ &= \mathbf{K}_1\boldsymbol{\beta}^{(0)}-\mathbf{H}_1^T\mathbf{H}_1\beta^{(0)}+\mathbf{H}_1^T\mathbf{T}_1.\end{aligned} \tag{2.60}$$

Combinando as Equações 2.57 e 2.60, $\boldsymbol{\beta}^{(1)}$ pode ser calculado de acordo com a expressão:

$$\begin{aligned}\boldsymbol{\beta}^{(1)} &= \mathbf{K}_1^{-1}\begin{bmatrix}\mathbf{H}_0\\ \mathbf{H}_1\end{bmatrix}^T\begin{bmatrix}\mathbf{T}_0\\ \mathbf{T}_1\end{bmatrix}\\ &= \mathbf{K}_1^{-1}\left(\mathbf{K}_1\beta^{(0)}-\mathbf{H}_1^T\mathbf{H}_1\beta^{(0)}+\mathbf{H}_1^T\mathbf{T}_1\right)\\ &= \boldsymbol{\beta}^{(0)}+\mathbf{K}_1^{-1}\mathbf{H}_1^T\left(\mathbf{T}_1-\mathbf{H}_1\boldsymbol{\beta}^{(0)}\right),\end{aligned} \tag{2.61}$$

em que $\mathbf{K}_1$ é dado por:

$$\mathbf{K}_1 = \mathbf{K}_0+\mathbf{H}_1^T\mathbf{T}_1. \tag{2.62}$$

Generalizando a discussão, sempre que um novo bloco de dados de treinamento alimenta o algoritmo, um procedimento recursivo é utilizado para atualizar a solução de mínimos quadrados, de forma similar ao método mostrado na Subseção 2.3.2. Quando o $(k+1)$-ésimo bloco de dados é recebido pelo algoritmo, de forma que

$$\aleph_{k+1} = \{(\mathbf{x}_i,\mathbf{t}_i)\}_{i=\left(\sum_{j=0}^{k}N_j\right)+1}^{\sum_{j=0}^{k+1}N_j},$$

em que $k\geq 0$, e N_{k+1} denota o número de instâncias do $(k+1)$-ésimo bloco de dados, temos:

$$\begin{aligned}\mathbf{K}_{k+1} &= \mathbf{K}_k+\mathbf{H}_{k+1}^T\mathbf{H}_{k+1}\\ \boldsymbol{\beta}^{(k+1)} &= \boldsymbol{\beta}^{(k)}+K_{k+1}^{-1}H_{k+1}^T\left(T_{k+1}-H_{k+1}\beta^{(k)}\right),\end{aligned} \tag{2.63}$$

em que T_{k+1} e H_{k+1} são definidos a seguir:

$$\mathbf{T}_{k+1} = \begin{bmatrix}\mathbf{t}_{\left(\sum_{j=0}^{k}N_j\right)+1}^T\\ \vdots\\ \mathbf{t}_{\sum_{j=0}^{k+1}N_j}^T\end{bmatrix}_{N_{k+1}\times m,} \tag{2.64}$$

$$\mathbf{H}_{k+1} = \begin{bmatrix} G\left(\mathbf{a}_1, \mathbf{b}_1, \mathbf{x}_{\left(\sum_{j=0}^{k} N_j\right)+1}\right) & \cdots & G\left(\mathbf{a}_L, \mathbf{b}_L, \mathbf{x}_{\left(\sum_{j=0}^{k} N_j\right)+1}\right) \\ \vdots & \cdots & \vdots \\ G\left(\mathbf{a}_1, \mathbf{b}_1, \mathbf{x}_{k+1 \sum_{j=0}^{k} N_j}\right) & \cdots & G\left(\mathbf{a}_L, \mathbf{b}_L, \mathbf{x}_{k+1 \sum_{j=0}^{k} N_j}\right) \end{bmatrix}. \tag{2.65}$$

Observe que usamos $\mathbf{K}_{k+1}^{-1}$ ao invés de $\mathbf{K}_{k+1}$ para calcular $\boldsymbol{\beta}^{(k+1)}$ a partir de $\boldsymbol{\beta}^{(k)}$, de acordo com a Equação 2.63. Para derivar a versão recursiva de atualização de $\boldsymbol{\beta}^{(k+1)}$, vamos reescrever $\mathbf{K}_{k+1}^{-1}$ usando a fórmula de Woodbury (GOLUB; VAN LOAN, 2012):

$$\begin{aligned} \mathbf{K}_{k+1}^{-1} &= \left(\mathbf{K}_k + \mathbf{H}_{k+1}^T \mathbf{H}_{k+1}\right)^{-1} \\ &= \mathbf{K}_k^{-1} - \mathbf{K}_k^{-1} \mathbf{H}_{k+1}^T \left(\mathbf{I} + \mathbf{H}_{k+1} \mathbf{K}_k^{-1} \mathbf{H}_{k+1}^T\right)^{-1} \mathbf{H}_{k+1} \mathbf{K}_k^{-1}. \end{aligned} \tag{2.66}$$

Seja $\mathbf{P}_{k+1} = \mathbf{K}_{k+1}^{-1}$. Podemos então reescrever as equações recursivas para atualizar $\boldsymbol{\beta}^{(k+1)}$ da seguinte forma:

$$\begin{aligned} \mathbf{P}_{k+1} &= \mathbf{P}_k - \mathbf{P}_k \mathbf{H}_{k+1}^T \left(\mathbf{I} + \mathbf{H}_{k+1} \mathbf{P}_k H_{k+1}^T\right)^{-1} \mathbf{H}_{k+1} \mathbf{P}_k \\ \boldsymbol{\beta}^{(k+1)} &= \boldsymbol{\beta}^{(k)} + \mathbf{P}_{k+1} \mathbf{H}_{k+1}^T \left(\mathbf{T}_{k+1} - \mathbf{H}_{k+1} \boldsymbol{\beta}^{(k)}\right). \end{aligned} \tag{2.67}$$

A Equação 2.67 nos dá a fórmula para calcular recursivamente $\boldsymbol{\beta}^{(k+1)}$. Observe que a implementação sequencial da solução de mínimos quadrados da Equação 2.51 é similar ao algoritmo dos mínimos quadrados recursivos, mostrado na Equação 2.49. A matriz de ganhos do RLS pode ser substituída pela expressão $\mathbf{P}_{k+1}\mathbf{H}_{k+1}^T$ e a inovação pode ser atribuída a $\left(\mathbf{T}_{k+1} - \mathbf{H}_{k+1}\boldsymbol{\beta}^{(k)}\right)$. Os mesmos resultados de convergência da solução de mínimos quadrados recursivo podem ser aplicados à saída da Equação 2.67. A sequência completa dos passos da OS-ELM é descrita no Algoritmo 2.

2.4 Trabalhos Relacionados

Nessa seção, vamos destacar os principais trabalhos que estendem a versão original da OS-ELM. As extensões encontradas na literatura se enquadram em cinco categorias distintas:

1. ***Ensemble***. Nesse tipo de extensão, diferentes instâncias da OS-ELM são treinadas através de subconjuntos (disjuntos) das instâncias de entrada, mas compartilhando as mesmas unidades da camada escondida (LAN; SOH; HUANG, 2009);

2. **Otimização**. Nesse tipo, diversos métodos de otimização são empregados para ajustar os pesos de entrada, ou ainda, para otimizar a estrutura da rede (ZHU et al., 2005);

Algoritmo 2: Online Sequential Extreme Learning Machine

Entrada:
Conjunto de treinamento $\aleph = \left\{(\mathbf{x}_i, \mathbf{t}_i) \,|\mathbf{x}_i \in \mathbb{R}^d, \mathbf{t}_i \in \mathbb{R}^m, i = 1, ..., N\right\}$,
Função de ativação $G(\mathbf{a}_i, \mathbf{b}_i, x)$,
Número de unidades da camada escondida L,
Tamanho do bloco inicial N_0.
Saída: Vetor de pesos de saída $\boldsymbol{\beta}$.

begin

Fase de Inicialização

Inicialize o aprendizado utilizando um pequeno bloco do conjunto de treinamento $\aleph_0 = \{(\mathbf{x}_i, \mathbf{t}_i)\}_{i=1}^{N_0}, N_0 \geq L$.

Escolha os parâmetros das unidades da camada escondida: $\mathbf{a}_i$, $\mathbf{b}_i$, $i = 1, ..., L$.

Calcule a matriz inicial de saída da camada escondida:

$$\mathbf{H}_0 = \begin{bmatrix} G(\mathbf{a}_1, \mathbf{b}_1, \mathbf{x}_1) & \cdots & G(\mathbf{a}_L, \mathbf{b}_L, \mathbf{x}_1) \\ \vdots & \ddots & \vdots \\ G(\mathbf{a}_1, \mathbf{b}_1, \mathbf{x}_{N_0}) & \cdots & G(\mathbf{a}_L, \mathbf{b}_L, \mathbf{x}_{N_0}) \end{bmatrix}_{N_0 \times L.}$$

Estime o conjunto inicial dos pesos de saída $\boldsymbol{\beta}^0 = \mathbf{P}_0 \mathbf{H}_0^T \mathbf{T}_0$, em que $\mathbf{P}_0 = \left(\mathbf{H}_0^T \mathbf{H}_0\right)^{-1}$ e $\mathbf{T}_0 = [\mathbf{t}_1, ..., \mathbf{t}_{N_0}]^T$.

Faça $k = 0$.

Fase de Aprendizado Sequencial

Receba o $(k+1)$-ésimo bloco de novas observações $\aleph_{k+1} = \{(\mathbf{x}_i, \mathbf{t}_i)\}_{i=\left(\sum_{j=0}^{k} N_j\right)+1}^{\sum_{j=0}^{k+1} N_j}$, em que N_{k+1} denota o número de observações do $(k+1)$-ésimo bloco.

Calcule a matriz parcial de saída da camada escondida:

$$\mathbf{H}_{k+1} = \begin{bmatrix} G\left(\mathbf{a}_1, \mathbf{b}_1, \mathbf{x}_{\left(\sum_{j=0}^{k} N_j\right)+1}\right) & \cdots & G\left(\mathbf{a}_L, \mathbf{b}_L, \mathbf{x}_{\left(\sum_{j=0}^{k} N_j\right)+1}\right) \\ \vdots & \ddots & \vdots \\ G\left(\mathbf{a}_1, \mathbf{b}_1, \mathbf{x}_{\left(\sum_{j=0}^{k+1} N_j\right)}\right) & \cdots & G\left(\mathbf{a}_L, \mathbf{b}_L, \mathbf{x}_{\left(\sum_{j=0}^{k+1} N_j\right)}\right) \end{bmatrix}_{N_{k+1} \times L.}$$

Atribua $\mathbf{T}_{k+1} = \begin{bmatrix} \mathbf{t}_{\left(\sum_{j=0}^{k} N_j\right)+1, ..., \left(\sum_{j=0}^{k+1} N_j\right)} \end{bmatrix}^T$.

Calcule os pesos parciais de saída $\boldsymbol{\beta}^{(k+1)}$:

$$\mathbf{P}_{k+1} = \mathbf{P}_k - \mathbf{P}_k \mathbf{H}_{k+1}^T \left(\mathbf{I} + \mathbf{H}_{k+1} \mathbf{P}_k \mathbf{H}_{k+1}^T\right)^{-1} \mathbf{H}_{k+1} P_k,$$

$$\boldsymbol{\beta}^{(k+1)} = \boldsymbol{\beta}^{(k)} + \mathbf{P}_{k+1} \mathbf{H}_{k+1}^T \left(\mathbf{T}_{k+1} - \mathbf{H}_{k+1} \boldsymbol{\beta}^{(k)}\right).$$

Faça $k = k + 1$ e repita a Fase de Aprendizado Sequencial até o final do conjunto de treinamento.

end

3. **Incremental**. Nessa categoria de extensão encontramos técnicas para criar novas unidades da camada escondida, de acordo com critérios preestabelecidos (HUANG; CHEN, 2008);

4. **Substituição**. Versões modificadas da OS-ELM substituem as funções de ativação tradicionais (sigmoide e RBF) por outras funções, com o objetivo de melhorar a precisão e taxa de convergência em problemas de aproximação de funções (HAN; HUANG, 2006);

5. **Modificação**. Nessa extensão, o processo de estimativa dos pesos de saída da rede é feito de forma diferente da versão original da OS-ELM. Diversas técnicas são empregadas para esse fim, incluindo regressão *ridge*, regularização, filtragem adaptativa, entre outras.

As principais extensões da OS-ELM que adotam as estratégias de *ensemble*, otimização, incremental e substituição estão descritas na próxima subseção. Em alinhamento aos objetivos da tese, destacamos na seção subsequente os trabalhos relacionados que envolvem alterações no procedimento original de atualização dos pesos de saída.

2.4.1 Extensões Gerais do OS-ELM

Com o objetivo de melhorar o desempenho do algoritmo original, algumas variações relevantes da OS-ELM são encontradas na literatura. Em LAN; SOH; HUANG (2009), é apresentado o *Ensemble of OS-ELM* (EOS-ELM), que consiste em várias redes OS-ELM com o mesmo número de unidades da camada escondida e a mesma função de ativação para cada instância. Cada rede é treinada com a chegada de novas instâncias na fase de aprendizado sequencial. A saída geral da rede é calculada a partir da média das saídas de todas as instâncias da rede. Os autores aplicam o método proposto em conjuntos de dados conhecidos da comunidade e mostram resultados mais estáveis e precisos, comparados com a versão original do algoritmo.

Em LI; LIU; DONG (2010) é mostrada a *Structure-adjustable Online ELM* (SAO-ELM), em que a estrutura é a mesma da OS-ELM original, mas com a possibilidade de ajustes no número de unidades da camada escondida. Para isso, os autores usaram uma técnica na qual uma esfera é modelada ao redor dos dados que alimentam o algoritmo sequencialmente. O centro e o raio da esfera são armazenados e servem como base para a alocação de uma nova unidade RBF à rede. Se as novas instâncias do conjunto de treinamento estiverem fora da área determinada pela esfera, uma nova unidade é alocada e os parâmetros recalculados. Os autores mostram a aplicação do método em problemas clássicos de regressão e classificação, além de um problema prático na área de siderurgia.

No trabalho de SUN; YUAN; WANG (2011) é apresentada uma solução de classificação para ambientes distribuídos baseado na OS-ELM. A proposta de um *ensemble Peer-to-Peer* (P2P) é baseada na seleção de pares da rede através de um processo de cobertura do espaço gerado

pelo conjunto de treinamento. Os autores definiram uma estrutura de duas camadas para reduzir o custo de comunicação e sincronização da rede. O *framework* gerado é capaz de processar os dados sequenciais um-a-um ou de forma paralela. Vários experimentos são reportados, e os resultados obtidos para problemas de classificação mostraram que a solução reduz o tempo de treinamento e mantém a capacidade de generalização da rede, medida em termos do erro de previsão.

Em ER; ZHAI; LI (2012), os autores propõem o algoritmo *Hybrid OS-ELM* (HOS-ELM). Nesse trabalho, é feita a integração entre o algoritmo de aprendizado do M-RAN com a OS-ELM. A estratégia de alocação de novas unidades à camada escondida e o processo de poda são aplicados ao OS-ELM de forma que os parâmetros são ajustados de forma automática, com a subsequente redução no número de unidades da rede. Nessa versão, quando uma nova unidade não é alocada à rede, os parâmetros da OS-ELM são atualizados através do filtro de Kalman estendido (EKF).

Em ZHAO; WANG; PARK (2012), um novo algoritmo de aprendizado é proposto, chamado de *Online Learning Algorithm with Forgetting Mechanism* (FOS-ELM). Nessa abordagem, um mecanismo de "esquecimento" é incorporado ao EOS-ELM, no qual os dados considerados desatualizados após uma certa unidade de tempo são descartados e não participam do treinamento.

Em MIRZA; LIN; TOH (2013) é apresentado um algoritmo sequencial para tratar o problema de desbalanceamento de classes em tarefas de classificação. O algoritmo *Weighted Online Sequential ELM* (WOS-ELM) atualiza os pesos de saída de forma semelhante ao OS-ELM, ajustando-os a partir de um critério baseado na taxa de desbalanceamento das classes. A alteração do algoritmo original ocorre através da aplicação de uma matriz de pesos diagonal, com os valores ajustados de acordo com a precisão de classificação para cada iteração da fase sequencial.

Uma estratégia diferente foi adotada em LIM (2013) com o particionamento recursivo da matriz formada pelo conjunto de treinamento. Nessa abordagem, chamada de *Partitioned OS-ELM* (POS-ELM), o algoritmo sequencial é aplicado a um conjunto de sub-matrizes para gerar os resultados parciais. Ao final da execução do POS-ELM, os resultados são concatenados em um vetor de estimativas, gerando a saída final da rede. O método proposto foi validado em problemas de regressão, nos quais os dados de entrada apresentavam alta dimensionalidade, com os resultados mostrando uma diminuição no tempo de treinamento, em comparação ao OS-ELM.

O trabalho original de RONG et al. (2009) foi estendido em RONG; HUANG; LIANG (2013) com o objetivo de treinar sistemas *fuzzy*, com capacidade de aproximação universal. Os princípios de aprendizado são os mesmos do *Online Sequential Fuzzy ELM* (OSFuzzy-ELM). Nessa abordagem, os parâmetros das funções *fuzzy* são gerados aleatoriamente, com o restante dos parâmetros da rede sendo gerados de forma analítica. As entradas da rede são selecionadas com base em uma medida de correlação. Os experimentos com problemas de regressão para o algoritmo proposto mostram um desempenho similar em comparação com outras abordagens de

aprendizagem *fuzzy*.

Em YANG et al. (2014), o algoritmo utiliza *kernels* com regressão *ridge* para processamento de dados com alta dimensionalidade. Os autores incorporaram ao método um esquema evolucionário para selecionar o melhor conjunto de regressores esparsos. A versão sequencial foi chamada de *Online Sequential Extreme Learning Algorithm* (OS-ELA) e foi validada com problemas de regressão e previsão de séries temporais, sem comparar o método proposto diretamente com a OS-ELM.

Uma abordagem similar foi apresentada em WANG; HAN (2014) para lidar com o problema de predição de séries temporais não-estacionárias, através da aplicação de um esquema baseado em *kernel* para ajustar os pesos de saída da rede de forma adaptativa. O algoritmo *OS-ELM with Kernels* (OS-ELMK) é validado através de problemas clássicos de previsão de séries temporais, alcançando resultados comparáveis às soluções semelhantes que utilizam *kernels*.

O trabalho de ZHOU; LIU; ZHU (2014) define duas extensões da OS-ELM para conjuntos de treinamento que variam com o tempo. A versão *Online Regularized ELM with Forgetting Mechanism* (FORELM) atualiza os pesos de saída da rede de forma recursiva através da fórmula de Sherman-Morrison, combinando os mecanismos descritos nos algoritmos FOS-ELM e ReOS-ELM (HUYNH; WON, 2011), evitando assim problemas de singularidade nas operações de inversão de matrizes através de um mecanismo de regularização. A versão *Online Kernelized ELM with Forgetting Mechanism* (FOKELM) aborda o problema de expansão das matrizes do KB-IELM (GUO; HAO; LIU, 2014) através da remoção das instâncias do conjunto de treinamento usando a fórmula de inversão em bloco de matrizes. As duas versões foram validadas em experimentos com dados que variam com o tempo, apresentando resultados melhores que os algoritmos originais em termos de precisão, estabilidade e complexidade computacional.

2.4.2 Extensões Relacionadas com o Tópico da Tese

Poucos trabalhos encontrados na literatura tratam diretamente os efeitos da multicolinearidade na matriz da camada escondida da OS-ELM. Os trabalhos descritos a seguir utilizam diversos mecanismos de ajuste dos pesos de saída da rede diretamente na fase sequencial do algoritmo. As técnicas utilizadas no tratamento de singularidade e mal-condicionamento incluem penalização, regularização e combinação linear através de funções base.

2.4.2.1 Regularized OS-ELM (ReOS-ELM)

Com o objetivo de evitar o problema de ruído e singularidade na matriz da camada escondida, em HUYNH; WON (2011) é proposto o método *Regularized OS-ELM* (ReOS-ELM), o qual utiliza uma função de otimização multi-objetivo para a minimização do erro empírico e da norma dos pesos de saída da rede. Os autores empregaram a técnica de regularização de Tikhonov para solucionar os problemas citados. A principal diferença do ReOS-ELM em relação

ao algoritmo original é a utilização de uma constante de regularização na fase de inicialização. A solução da Equação 2.51 pode ser reescrita para encontrar $\boldsymbol{\beta}$ que minimize a seguinte expressão:

$$\|\mathbf{H}\boldsymbol{\beta} - \mathbf{T}\|^2 + \lambda\|\boldsymbol{\beta}\|^2, \tag{2.68}$$

em que $\|\bullet\|$ é a norma euclidiana e λ é uma constante positiva. A fase de aprendizagem sequencial do ReOS-ELM segue o mesmo esquema recursivo da OS-ELM, com uma pequena mudança em relação ao cálculo dos pesos de saída. Para problemas nos quais o bloco de instâncias n_k é maior que o número de unidades da camada escondida, a atualização dos pesos segue a Equação 2.67. Caso contrário, os pesos são atualizados de acordo com o conjunto de equações a seguir:

$$\begin{aligned} \mathbf{P}_{k+1} &= \mathbf{P}_k + \mathbf{H}_{k+1}^T\mathbf{H}_{k+1} \\ \boldsymbol{\beta}_{k+1} &= \boldsymbol{\beta}_k + (\mathbf{P}_{k+1})^{-1}\mathbf{H}_{k+1}^T(\mathbf{T}_{k+1} - \mathbf{H}_{k+1}\boldsymbol{\beta}_k). \end{aligned} \tag{2.69}$$

A principal limitação do ReOS-ELM é a necessidade de otimização da constante de regularização λ. Os autores apresentam em HUYNH; WON (2011) quatro diferentes abordagens para o cálculo da constante, sem levar em consideração o custo computacional da otimização do parâmetro. Além disso, a mudança no cálculo dos pesos de acordo com o tamanho do bloco aumenta a complexidade do algoritmo. Levando em consideração que o tamanho do bloco de instâncias na maioria das vezes será menor que o número de unidades da camada escondida, torna-se necessária a inversão da matriz $\mathbf{P}_{k+1}$ que ajusta os pesos de saída. Apesar dos experimentos reportados pelos autores mostrarem que o uso da regularização aumenta a precisão em relação ao OS-ELM em problemas de regressão e classificação, a escolha incorreta da constante poderá afetar o desempenho geral do algoritmo.

2.4.2.2 OS-ELM Time-varying (OS-ELM-TV)

O trabalho de YE; SQUARTINI; PIAZZA (2013) estende o algoritmo *batch* da ELM para problemas de previsão em séries temporais não-estacionárias e realizam a integração com a sua versão sequencial, criando o *OS-ELM Time-varying* (OS-ELM-TV). Nessa abordagem, os pesos de entrada mudam de acordo com a passagem do tempo, sempre que uma nova sequência de instâncias alimentam o algoritmo. A saída da rede é calculada através de uma combinação linear dos pesos, os quais também podem variar de acordo com o tempo. O cálculo dos pesos de entrada é realizado de acordo com a expressão:

$$\mathbf{w}_k = \sum_{n=1}^{B} f_{n_k}\alpha_n, \tag{2.70}$$

em que f_{n_k} é uma função ortogonal de ordem n, calculada no tempo k, α_n é o n-ésimo coeficiente da função, utilizado na construção de $\mathbf{w}_k$, e B é a quantidade total de funções utilizadas na

combinação linear. A inicialização e a fase sequencial do algoritmo é similar ao da OS-ELM, tendo como principal diferença a utilização das funções ortogonais no cálculo da matriz da camada escondida $\mathbf{H}$ de ordem n, de acordo com a expressão:

$$\mathbf{h}_{k_n} = g\left(\sum_{i=0}^{L} x_i \mathbf{w}_{i,k_n}\right), \tag{2.71}$$

em que g é uma função de ativação, $\mathbf{x}_i$ é o i-ésimo atributo da amostra k, $\mathbf{w}_{i,k_n}$ é o i-ésimo peso de entrada de ordem n e L é o número de unidades da camada escondida. O cálculo de $\mathbf{H}$ não pode ser feito diretamente, como no algoritmo original, sendo necessária a aplicação de várias transformações matriciais. Os autores utilizam o produto de Kronecker para estimar a matriz da camada escondida (BREWER, 1978), de acordo com a equação a seguir:

$$\mathbf{H} = \begin{bmatrix} \mathbf{h}_1^T \otimes f_1^T \\ \vdots \\ \mathbf{h}_n^T \otimes f_n^T \end{bmatrix}_{n \times b \cdot k}, \tag{2.72}$$

em que $\otimes$ denota o produto de Kronecker de $\mathbf{h}_i^T$ e f_i^T, com $i = 1, \ldots, n$. A fase sequencial do OS-ELM-TV adota o mesmo procedimento recursivo da OS-ELM, utilizando a Equação 2.67 para atualizar os pesos de saída da rede.

O método é validado através de sete experimentos com problemas de regressão, apresentando um desempenho comparável ao da OS-ELM. Os autores não apresentam testes estatísticos para validar as diferenças de desempenho nos experimentos realizados. Uma limitação do método proposto é necessidade de se gerar previamente a função base e a estimativa dos coeficientes. Essa etapa aumenta a complexidade do algoritmo, o qual tende a aumentar o tempo de treinamento quando o conjunto de entrada possui alta dimensionalidade ou quando a quantidade de instâncias é muito grande. Outro problema aparente é a escolha correta da função ortogonal para o cálculo da matriz $\mathbf{H}$. Os autores testam o algoritmo usando quatro funções base: polinomial de Legendre, polinomial de Chebyshev, senoidal de Fourier e uma função esférica. Por fim, ainda é necessário escolher a quantidade de funções que serão usadas na combinação linear do cálculo de $\mathbf{H}$.

2.4.2.3 Timeliness Online Sequential ELM (TOSELM)

Para problemas nos quais a distribuição dos dados muda de acordo com o tempo, em GU et al. (2014) foi proposto o *Timeliness Online Sequential ELM* (TOSELM). Esse algoritmo utiliza as características de tendência e dispersão dos dados de entrada para ajustar os pesos de saída a cada iteração. A inicialização do algoritmo é a mesma da OS-ELM e a sua fase sequencial atualiza os pesos de forma semelhante ao algoritmo original. A principal diferença em relação à Equação 2.67 é a utilização de um fator de penalização na matriz de ajuste dos pesos. O fator ω

é calculado através da média das diferenças entre a amostra atual e as instâncias adjacentes e as suas respectivas variâncias, de acordo com a seguinte equação:

$$\omega = 1 + 2\exp\left(-\left|\mu_i - \mu_j\right|^{\left|\sigma_i - \sigma_j\right|}\right), \tag{2.73}$$

em que μ_i e μ_j são as médias das i instâncias incrementais e das j instâncias adjacentes a i. De forma semelhante, σ_i e σ_j são as variâncias das i instâncias incrementais e das j instâncias adjacentes a i. Para a atualização dos pesos, a Equação 2.67 passa a ter a seguinte formulação:

$$\boldsymbol{\beta}^{(k+1)} = \boldsymbol{\beta}^{(k)} + \omega \mathbf{P}_{k+1}\mathbf{H}_{k+1}^{T}\left(\mathbf{T}_{k+1} - \mathbf{H}_{k+1}\boldsymbol{\beta}^{(k)}\right). \tag{2.74}$$

Os autores relatam problemas de estabilidade quando uma única correção é realizada durante a atualização dos pesos. Para estabilizar o modelo, os autores propõem a utilização do método de Newton para aproximar o valor de ω. Através de um procedimento iterativo, o fator de correção é escolhido nos j-ésimos pesos adjacentes, de forma que $\left|\boldsymbol{\beta}_{k(j+1)} - \boldsymbol{\beta}_{k(j)}\right| < \varepsilon$, em que ε é um limiar definido previamente.

O desempenho do método proposto foi validado em problemas de regressão, com os resultados mostrando que a utilização de um esquema alternativo de correção dos pesos pode melhorar a precisão e a convergência do algoritmo. Entretanto, o esquema iterativo utilizado para ajustar o fator ω aumenta de forma considerável o tempo de treinamento, com valores mais elevados em relação ao OS-ELM. Além do alto custo computacional, o algoritmo impõe a otimização de dois parâmetros adicionais: o limiar ε e o valor máximo de iterações do método de Newton.

2.4.2.4 Least Squares Incremental ELM (LS-IELM)

Baseado na versão original da OS-ELM, o *Incremental Extreme Learning Machine* (IELM) é proposto em GUO; HAO; LIU (2014), com três versões diferentes que podem ser adaptadas para situações distintas de treinamento. A versão *Minimum Norm Incremental ELM* (MN-IELM) é utilizada quando o número de instâncias do conjunto de treinamento é pequeno, mas a dimensionalidade do conjunto de é alta. Quando a situação inversa ocorre, os autores sugerem o uso do *Least Squares Incremental ELM* (LS-IELM). Os algoritmos citados só podem ser usados quando um mapeamento com as características dos dados for feito previamente. Caso contrário, os autores indicam o uso do *Kernel Based Incremental ELM* (KB-IELM). O algoritmo que possui relação mais direta com o tópico dessa tese é o LS-IELM, devido à sua estratégia de ajuste direto dos pesos de saída da rede. De forma similar ao ReOS-ELM, esse algoritmo utiliza uma constante de regularização para a fase de inicialização. Aqui o objetivo é encontrar $\boldsymbol{\beta}$ que minimize a expressão:

$$\frac{1}{2}\|\boldsymbol{\beta}\|^2 + \lambda\frac{1}{2}\|\xi,\|^2 \tag{2.75}$$

Tabela 2.1: Algoritmos de aprendizagem sequencial - Mecanismos de atualização de pesos e principais parâmetros

Algoritmo	Atualização dos Pesos	Parâmetros	Limitações
OS-ELM	Mínimos quadrados recursivos (RLS)	L, N_0	Multicolinearidade
ReOS-ELM	Regularização e RLS	L, N_0, λ	Atualização dependente da distribuição
OS-ELM-TV	Combinação linear com n funções ortogonais	L, N_0, f, B	Otimização adicional de parâmetros
TOSELM	Fator de correção dos pesos e RLS	L, N_0, ε, j	Custo computacional
LS-IELM	Regularização e RLS	L, N_0, λ	Otimização adicional de parâmetros

sujeito a $\mathbf{H}\boldsymbol{\beta} = \mathbf{T} - \xi$, em que ξ denota o erro de treinamento e λ é a constante de regularização. A fase sequencial segue o mesmo procedimento da OS-ELM, seguindo as regras de atualização de acordo com a Equação 2.67.

O LS-IELM é validado através de problemas de regressão e classificação, com os resultados indicando uma melhoria no desempenho em comparação ao OS-ELM. Entretanto, os autores não apresentam testes estatísticos para validar as comparações entre os algoritmos. A mesma limitação do ReOS-ELM pode ser aplicada ao LS-IELM, em relação à otimização da constante de regularização λ.

2.5 Considerações Finais

Esse capítulo introduziu os conceitos essenciais sobre a aprendizagem sequencial para redes *feedforward*. O processo de treinamento para essa classe de problemas evoluiu a partir do mecanismo de alocação dinâmica das unidades da rede. As extensões aos modelos clássicos de aprendizagem sequencial passam inicialmente pela atualização dos parâmetros da rede, sempre que uma nova unidade não consegue ser incorporada à sua estrutura. As técnicas de poda e a utilização de limiares de significância das unidades tornaram as redes mais compactas e estáveis. O problema do aprendizagem sequencial para SLFNs é tratado a partir da atualização sequencial dos pesos de saída da rede através de soluções baseadas em mínimos quadrados recursivo. A definição da ELM, e sua versão para aprendizagem sequencial, permitiu a sua aplicação em problemas nos quais os dados não estão integralmente disponíveis no início do processo de treinamento.

As principais extensões da OS-ELM podem ser enquadradas em cinco categorias: *Ensemble*, otimização, incremental, substituição e modificação. Os algoritmos da última categoria modificam o mecanismo de atualização dos pesos de saída da rede e foram discutidos na seção anterior. A Tabela 2.1 apresenta um resumo da estratégia adotada para a fase sequencial, a lista de parâmetros de cada algoritmo estudado e suas limitações.

A seguir, o próximo capítulo irá apresentar os conceitos básicos do filtro de Kalman, o qual será considerado na formulação de uma solução para o problema de aprendizagem sequencial.

3

Filtro de Kalman

O problema de aprendizagem sequencial, discutido no Capítulo 2, possui soluções para a estimativa dos pesos de saída da rede através de métodos recursivos de mínimos quadrados. Dentro do contexto, vamos abordar nesse capítulo algumas alternativas para a estimativa dos pesos através do filtro de Kalman (KALMAN, 1960). Essa técnica é baseada na formulação de estado de espaços para modelos lineares dinâmicos, permitindo uma solução recursiva para o problema de filtragem linear. Modelos dinâmicos baseados no filtro de Kalman podem ser utilizados em ambientes estacionários e não estacionários. A recursividade da solução ocorre sempre que uma estimativa do estado é atualizada a partir da estimativa anterior, no momento da chegada de uma nova instância do conjunto de treinamento. Dessa forma, não é necessário armazenar todas as instâncias para o processo de aprendizado, satisfazendo as condições formuladas em HUANG; SARATCHANDRAN; SUNDARARAJAN (2005), na qual apenas uma instância de treinamento é vista pelo algoritmo em um dado instante, sendo descartada em seguida. Além disso, o filtro de Kalman é computacionalmente mais eficiente no processo de filtragem do que o cálculo direto da estimativa dos pesos, baseado no aprendizado do restante das instâncias (HARVEY, 1990).

Nesse capítulo, nós vamos discutir as principais características do filtro de Kalman para pavimentar a sua aplicação no problema de aprendizagem sequencial. Uma discussão mais aprofundada sobre aspectos avançados do filtro pode ser vista em HARVEY (1990) e em KAILATH (1981). Considere um sistema linear dinâmico e discreto no tempo, de acordo com o diagrama de blocos da Figura 3.1. O conceito de estado é fundamental para entender o funcionamento do sistema. O vetor de estados, ou simplesmente estado, denotado por $\mathbf{x}_k$, é definido como o conjunto mínimo de dados, suficiente para descrever de forma única o comportamento dinâmico do sistema, com k representando uma instância discreta do tempo. Em outras palavras, o estado é a quantidade mínima de dados extraída do comportamento passado do sistema, a qual será utilizada para prever o seu comportamento futuro. Normalmente, o estado $\mathbf{x}_k$ é desconhecido, e para estimá-lo, devemos utilizar um conjunto de dados observados, denotados na figura pelo vetor $\mathbf{y}_k$.

Em termos matemáticos, o diagrama de blocos da Figura 3.1 apresenta o seguinte par de

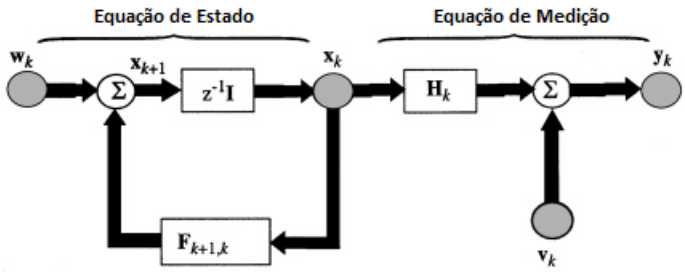


Figura 3.1: Representação de um sistema linear dinâmico e discreto no tempo

equações:

Equação de Estado.

$$\mathbf{x}_{k+1} = \mathbf{A}_{k+1|k}\mathbf{x}_k + \mathbf{w}_k, \tag{3.1}$$

em que $\mathbf{A}_{k+1|k}$ é a matriz de transição do estado $\mathbf{x}_k$ do tempo k para o tempo $k+1$. O ruído do processo $\mathbf{w}_k$ é supostamente aditivo, branco e Gaussiano, com média zero e matriz de covariância $\mathbf{Q}_k$, seguindo a forma:

$$E\left[\mathbf{w}_n\mathbf{w}_k^T\right] = \begin{cases} \mathbf{Q}_k, & n = k \\ 0, & n \neq k. \end{cases} \tag{3.2}$$

Equação de Medição.

$$\mathbf{y}_k = \mathbf{H}_k\mathbf{x}_k + \mathbf{v}_k, \tag{3.3}$$

em que $\mathbf{y}_k$ é a observação no tempo k e $\mathbf{H}_k$ é matriz de medição. O ruído de medição $\mathbf{v}_k$ é supostamente aditivo, branco e Gaussiano, com média zero e matriz de covariância $\mathbf{R}_k$, na forma a seguir:

$$E\left[\mathbf{v}_n\mathbf{v}_k^T\right] = \begin{cases} \mathbf{R}_k, & n = k \\ 0, & n \neq k. \end{cases} \tag{3.4}$$

O ruído de medição $\mathbf{v}_k$ não é correlacionado com o ruído do processo $\mathbf{w}_k$.

O problema do filtro de Kalman, descrito como a solução otimizada e conjunta das equações de estado e medição para um estado desconhecido, pode ser formulado conforme a seguinte definição:

Definição 3.0.1. Problema do Filtro de Kalman. *Usar todo o conjunto de dados observados, consistindo dos vetores $\boldsymbol{y}_1, \boldsymbol{y}_2, ..., \boldsymbol{y}_k$, para encontrar o erro mínimo quadrado para a estimativa do estado $\boldsymbol{x}_i$, com $k \geq 1$.*

O problema acima pode ser classificado em três tipos, dependendo dos valores de i e k:

1. Filtragem, se $i = k$;

2. Predição, se $i > k$; e

3. Suavização, se $1 \leq i < k$.

Nas próximas seções vamos descrever com mais detalhes o processo de estimação ótima, antes de descrever a solução recursiva do filtro de Kalman. Também discutiremos brevemente a estimativa de seus parâmetros pelo método da máxima verossimilhança.

3.1 Estimadores Ótimos

Nessa seção, vamos rever alguns conceitos básicos de estimação ótima. Com o objetivo de simplificar a formalização, vamos descrever aqui o processo de estimação ótima no contexto de variáveis escalares aleatórias. Suponha que seja dada a seguinte observação:

$$\mathbf{y}_k = \mathbf{x}_k + \mathbf{v}_k,$$

em que $\mathbf{x}_k$ é um sinal desconhecido e $\mathbf{v}_k$ é um componente de ruído aditivo. Faça $\hat{\mathbf{x}}_k$ a estimativa a posteriori do sinal $\mathbf{x}_k$, dadas as observações $\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_k$. De forma geral, a estimativa $\hat{\mathbf{x}}_k$ é diferente do sinal desconhecido $\mathbf{x}_k$. Para derivar essa estimativa de maneira ótima, precisamos definir uma função de custo para as estimativas incorretas. A função de custo deverá satisfazer dois requisitos básicos:

1. A função de custo é positiva;

2. A função de custo é definida de forma não-decrescente a partir da estimativa do erro $\tilde{\mathbf{x}}_k$, definida por $\tilde{\mathbf{x}}_k = \mathbf{x}_k - \hat{\mathbf{x}}_k$.

Esses dois requisitos são satisfeitos pela definição do erro médio quadrado, dada por

$$\begin{aligned} J_k &= E\left[(\mathbf{x}_k - \hat{\mathbf{x}}_k)^2\right] \\ &= E\left[\tilde{\mathbf{x}}_k^2\right], \end{aligned}$$

em que E é o operador esperança. A dependência da função de custo J_k no tempo k enfatiza a natureza não-estacionária do processo recursivo de estimativa. Para derivar um valor ótimo para a estimativa $\hat{\mathbf{x}}_k$, vamos utilizar dois teoremas da teoria de processos estocásticos (VAN TREES, 2004):

Teorema 3.1.1. ***Estimador Condicional da Média.*** *Se os processos estocásticos $\{\mathbf{x}_k\}$ e $\{\mathbf{y}_k\}$ são simultaneamente Gaussianos, então o estimador ótimo $\hat{\mathbf{x}}_k$ que minimiza o erro médio quadrado J_k é o estimador da média condicional:*

$$\hat{\mathbf{x}}_k = E\left[\mathbf{x}_k | \mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_k\right].$$

Teorema 3.1.2. ***Princípio da Ortogonalidade.*** *Sejam os processos estocásticos $\{\mathbf{x}_k\}$ e $\{\mathbf{y}_k\}$ com médias zero, tal que $E[\mathbf{x}_k] = E[\mathbf{y}_k] = 0$ para todo k. Então:*

(i) os processos estocásticos $\{\mathbf{x}_k\}$ e $\{\mathbf{y}_k\}$ são simultaneamente Gaussianos; ou

(ii) se o estimador ótimo $\hat{\mathbf{x}}_k$ for restrito a uma função linear das observações e a função de custo for a média do erro quadrado,

(iii) então o estimador ótimo $\hat{\mathbf{x}}_k$, dadas as observações $\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_k$, é a projeção ortogonal de $\mathbf{x}_k$ no espaço gerado por essas observações.

A partir da definição desses dois teoremas, cujas provas podem ser vistas em VAN TREES (2004), vamos mostrar a definição formal do filtro de Kalman na próxima seção.

3.2 Filtro de Kalman Linear

Suponha que a medição de um sistema linear dinâmico, descrito pelas Equações 3.1 e 3.3, tenha sido feita no tempo k. O principal requisito desse modelo é usar a informação contida em uma nova medição $\mathbf{y}_k$ para atualizar a estimativa do estado desconhecido $\mathbf{x}_k$. Seja $\hat{\mathbf{x}}_k^-$ a estimativa a priori do estado, o qual já está disponível no tempo k. Com o objetivo de obter o estimador linear, vamos expressar a estimativa a posteriori $\hat{\mathbf{x}}_k$ como uma combinação linear da estimativa a priori e da nova medição, de acordo com a equação:

$$\hat{\mathbf{x}}_k = \mathbf{G}_k^{(1)} \hat{\mathbf{x}}_k^- + \mathbf{G}_k \mathbf{y}_k, \tag{3.5}$$

em que as matrizes com os fatores $\mathbf{G}_k^{(1)}$ e $\mathbf{G}_k$ precisam ser determinadas. Para encontrá-las, vamos usar o princípio da ortogonalidade, mostrado no Teorema 3.1.2. O vetor de erro de estado é definido por:

$$\tilde{\mathbf{x}}_k = \mathbf{x}_k - \hat{\mathbf{x}}_k. \tag{3.6}$$

Aplicando o princípio da ortogonalidade para o problema, podemos escrever:

$$E\left[\tilde{\mathbf{x}}_k \mathbf{y}_i^T\right] = 0, \quad i = 1, 2, ..., k-1. \tag{3.7}$$

Usando as Equações 3.3, 3.5 e 3.6 em 3.7, temos:

$$E\left[\left(\mathbf{x}_k - \mathbf{G}_k^{(1)} \hat{\mathbf{x}}_k^- - \mathbf{G}_k \mathbf{H}_k \mathbf{x}_k - \mathbf{G}_k \mathbf{w}_k\right) \mathbf{y}_i^T\right] = 0, \quad i = 1, 2, ..., k-1. \tag{3.8}$$

Sabemos que o ruído do processo $\mathbf{w}_k$ e o ruído da medição $\mathbf{v}_k$ não são correlacionados. Logo,

$$E\left[\mathbf{w}_k \mathbf{y}_i^T\right] = 0.$$

Usando essa relação e rearranjando os termos, podemos reescrever a Equação 3.8 da seguinte forma:

$$E\left[\left(\mathbf{I} - \mathbf{G}_k \mathbf{H}_k - \mathbf{G}_k^{(1)}\right) \mathbf{x}_k \mathbf{y}_i^T + \mathbf{G}_k^{(1)} \left(\mathbf{x}_k - \hat{\mathbf{x}}_k^-\right) \mathbf{y}_i^T\right] = 0, \tag{3.9}$$

em que $\mathbf{I}$ é a matriz identidade. Levando em consideração o princípio da ortogonalidade, sabemos que

$$E\left[\left(\mathbf{x}_k - \hat{\mathbf{x}}_k^-\right)\mathbf{y}_i^T\right] = 0.$$

Portanto, podemos simplificar a Equação 3.9 da seguinte forma:

$$\left(\mathbf{I} - \mathbf{G}_k\mathbf{H}_k - \mathbf{G}_k^{(1)}\right) E\left[\mathbf{x}_k\mathbf{y}_i^T\right] = 0. \tag{3.10}$$

Para valores arbitrários do estado $\mathbf{x}_k$ e da observação $\mathbf{y}_i$, a Equação 3.10 só será satisfeita se os fatores de escala $\mathbf{G}_k^{(1)}$ e $\mathbf{G}_k$ forem relacionados da seguinte forma:

$$\mathbf{I} - \mathbf{G}_k\mathbf{H}_k - \mathbf{G}_k^{(1)} = 0$$

ou, se definirmos de forma equivalente $\mathbf{G}_k^{(1)}$ em função de $\mathbf{G}_k$, conforme a expressão:

$$\mathbf{G}_k^{(1)} = \mathbf{I} - \mathbf{G}_k\mathbf{H}_k. \tag{3.11}$$

Substituindo a Equação 3.11 pela Equação 3.5, podemos expressar a estimativa a posteriori do estado no tempo k como:

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{G}_k\left(\mathbf{y}_k - \mathbf{H}_k\hat{\mathbf{x}}_k^-\right). \tag{3.12}$$

$\mathbf{G}_k$ é chamada de matriz de ganho de Kalman. Para derivar explicitamente uma fórmula para essa matriz, usaremos novamente o princípio da ortogonalidade. Sabemos que

$$E\left[\left(\mathbf{x}_k - \hat{\mathbf{x}}_k^-\right)\mathbf{y}_k^T\right] = 0. \tag{3.13}$$

Isso implica em

$$E\left[\left(\mathbf{x}_k - \hat{\mathbf{x}}_k^-\right)\hat{\mathbf{y}}_k^T\right] = 0, \tag{3.14}$$

em que $\hat{\mathbf{y}}_k^T$ é uma estimativa de de $\mathbf{y}_k$ a partir das medições anteriores $\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_{k-1}$. Vamos definir a inovação do processo como:

$$\tilde{\mathbf{y}}_k = \mathbf{y}_k - \hat{\mathbf{y}}_k. \tag{3.15}$$

A inovação do processo representa a medida da nova informação contida em $\mathbf{y}_k$ e pode ser expressa da seguinte forma:

$$\begin{aligned}\tilde{\mathbf{y}}_k &= \mathbf{y}_k - \mathbf{H}_k\hat{\mathbf{x}}_k^- \\ &= \mathbf{H}_k\mathbf{x}_k + \mathbf{v}_k - \mathbf{H}_k\hat{\mathbf{x}}_k^- \\ &= \mathbf{H}_k\tilde{\mathbf{x}}_k^- + \mathbf{v}_k.\end{aligned} \tag{3.16}$$

Dessa forma, subtraindo a Equação 3.14 da Equação 3.13 e usando a definição da Equação 3.15, podemos escrever:

$$E\left[\left(\mathbf{x}_k - \hat{\mathbf{x}}_k\right)\tilde{\mathbf{y}}_k^T\right] = 0. \tag{3.17}$$

Usando as Equações 3.3 e 3.12, podemos expressar o vetor de erro de estado $\mathbf{x}_k - \hat{\mathbf{x}}_k$ como:

$$\begin{aligned} \mathbf{x}_k - \hat{\mathbf{x}}_k &= \tilde{\mathbf{x}}_k^- - \mathbf{G}_k \left(\mathbf{H}_k \tilde{\mathbf{x}}_k^- + \mathbf{v}_k \right) \\ &= \left(\mathbf{I} - \mathbf{G}_k \mathbf{H}_k \right) \tilde{\mathbf{x}}_k^- - \mathbf{G}_k \mathbf{v}_k. \end{aligned} \tag{3.18}$$

Consequentemente, substituindo as Equações 3.16 e 3.18 em 3.17, temos:

$$E \left[\left\{ \left(\mathbf{I} - \mathbf{G}_k \mathbf{H}_k \right) \tilde{\mathbf{x}}_k^- - \mathbf{G}_k \mathbf{v}_k \right\} \left(\mathbf{H}_k \tilde{\mathbf{x}}_k^- + \mathbf{v}_k \right) \right]. \tag{3.19}$$

Uma vez que o ruído de medição $\mathbf{v}_k$ é independente do estado $\mathbf{x}_k$ e, por consequência, também é independente do estado $\tilde{\mathbf{x}}_k^-$, a esperança da Equação 3.19 se resume a:

$$\left(\mathbf{I} - \mathbf{G}_k \mathbf{H}_k \right) E \left[\tilde{\mathbf{x}}_k \tilde{\mathbf{x}}_k^{T-} \right] \mathbf{H}_k^T - \mathbf{G}_k E \left[\mathbf{v}_k \mathbf{v}_k^T \right] = 0. \tag{3.20}$$

Vamos definir a matriz de covariância a priori, como sendo:

$$\begin{aligned} \mathbf{P}_k^- &= E \left[\left(\mathbf{x}_k - \hat{\mathbf{x}}_k^- \right) \left(\mathbf{x}_k - \hat{\mathbf{x}}_k^- \right)^T \right] \\ &= E \left[\tilde{\mathbf{x}}_k^- \cdot \tilde{\mathbf{x}}_k^{T-} \right]. \end{aligned} \tag{3.21}$$

Dessa forma, utilizando as Equações 3.4 e 3.21, podemos reescrever a Equação 3.20 da seguinte forma:

$$\left(\mathbf{I} - \mathbf{G}_k \mathbf{H}_k \right) \mathbf{P}_k^- \mathbf{H}_k^T - \mathbf{G}_k \mathbf{R}_k = 0.$$

Resolvendo a equação anterior para $\mathbf{G}_k$, encontramos a fórmula para a matriz de ganhos de Kalman:

$$\mathbf{G}_k = \mathbf{P}_k^- \mathbf{H}_k^T \left[\mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^T + \mathbf{R}_k \right]^{-1}. \tag{3.22}$$

Note que a equação anterior é definida em termos da matriz de covariância a priori, $\mathbf{P}_k^-$.

Para completar o processo de estimativa recursiva, vamos levar em consideração a propagação do erro de covariância, a qual descreve o efeito do tempo nas matrizes de covariância dos erros de estimativa. Essa propagação envolve dois estágios de cálculo:

1. A matriz de covariância a priori $\mathbf{P}_k^-$ é definida pela Equação 3.21. A partir de $\mathbf{P}_k^-$, calcule a matriz de covariância a posteriori $\mathbf{P}_k$ no tempo k, cuja definição é dada por:

$$\begin{aligned} \mathbf{P}_k &= E \left[\tilde{\mathbf{x}}_k \tilde{\mathbf{x}}_k^T \right] \\ &= E \left[\left(\mathbf{x}_k - \hat{\mathbf{x}}_k \right) \left(\mathbf{x}_k - \hat{\mathbf{x}}_k \right)^T \right]. \end{aligned} \tag{3.23}$$

2. A partir da "antiga" matriz de covariância a posteriori $\mathbf{P}_{k-1}$, calcule a matriz "atualizada" de covariância a priori $\mathbf{P}_k^-$.

Para realizar os cálculos do estágio 1, vamos substituir a Equação 3.18 pela Equação 3.23. Observe que o ruído do processo $\mathbf{v}_k$ é independente do erro de estimativa a priori $\tilde{\mathbf{x}}_k^-$. Dessa

forma, calculamos $\mathbf{P}_k$ de acordo com a equação:

$$\begin{aligned}\mathbf{P}_k &= (\mathbf{I}-\mathbf{G}_k\mathbf{H}_k)\,E\left[\tilde{\mathbf{x}}_k^-\tilde{\mathbf{x}}_k^{T-}\right](\mathbf{I}-\mathbf{G}_k\mathbf{H}_k)^T+\mathbf{G}_kE\left[\mathbf{v}_n\mathbf{v}_k^T\right]\mathbf{G}_k^T\\ &= (\mathbf{I}-\mathbf{G}_k\mathbf{H}_k)\,\mathbf{P}_k^-(\mathbf{I}-\mathbf{G}_k\mathbf{H}_k)^T+\mathbf{G}_k\mathbf{R}_k\mathbf{G}_k^T.\end{aligned} \tag{3.24}$$

Expandindo os termos da Equação 3.24 e usando a Equação 3.22, podemos reformular a dependência da matriz de covariância a posteriori $\mathbf{P}_k$ em termos da matriz de covariância a priori $\mathbf{P}_k^-$ de forma simplificada:

$$\begin{aligned}\mathbf{P}_k &= (\mathbf{I}-\mathbf{G}_k\mathbf{H}_k)\,\mathbf{P}_k^- - (\mathbf{I}-\mathbf{G}_k\mathbf{H}_k)\,\mathbf{P}_k^-\mathbf{H}_k^T\mathbf{G}_k^T+\mathbf{G}_k\mathbf{R}_k\mathbf{G}_k^T\\ &= (\mathbf{I}-\mathbf{G}_k\mathbf{H}_k)\,\mathbf{P}_k^- - \mathbf{G}_k\mathbf{R}_k\mathbf{G}_k^T+\mathbf{G}_k\mathbf{R}_k\mathbf{G}_k^T\\ &= (\mathbf{I}-\mathbf{G}_k\mathbf{H}_k)\,\mathbf{P}_k^-.\end{aligned} \tag{3.25}$$

Para o segundo estágio da propagação do erro de covariância, vamos lembrar que a estimativa a priori do estado é definida em termos da "velha" estimativa a posteriori, na forma a seguir:

$$\hat{x}_k^- = \mathbf{A}_{k|k-1}\hat{\mathbf{x}}_{k-1}. \tag{3.26}$$

Podemos agora utilizar as Equações 3.1 e 3.26 para expressar a estimativa a priori do erro de outra forma:

$$\begin{aligned}\tilde{\mathbf{x}}_k^- &= \mathbf{x}_k-\hat{\mathbf{x}}_k^-\\ &= \left(\mathbf{A}_{k|k-1}\mathbf{x}_{k-1}+\mathbf{w}_{k-1}\right)-\left(\mathbf{A}_{k|k-1}\hat{\mathbf{x}}_{k-1}\right)\\ &= \mathbf{A}_{k|k-1}\left(\mathbf{x}_{k-1}-\hat{x}_{k-1}\right)+\mathbf{w}_{k-1}\\ &= \mathbf{A}_{k|k-1}\tilde{\mathbf{x}}_{k-1}+\mathbf{w}_{k-1}.\end{aligned} \tag{3.27}$$

Portanto, usando as Equações 3.27 e 3.21 e levando em consideração que o ruído do processo $\mathbf{w}_k$ é independente de $\tilde{\mathbf{x}}_{k-1}$, temos:

$$\begin{aligned}\mathbf{P}_k^- &= \mathbf{A}_{k|k-1}E\left[\tilde{\mathbf{x}}_{k-1}\tilde{\mathbf{x}}_{k-1}^T\right]\mathbf{A}_{k|k-1}^T+E\left[\mathbf{w}_{k-1}\mathbf{w}_{k-1}^T\right]\\ &= \mathbf{A}_{k|k-1}\mathbf{P}_{k-1}\mathbf{A}_{k|k-1}^T+\mathbf{Q}_{k-1}.\end{aligned} \tag{3.28}$$

A equação anterior define a dependência da matriz de covariância a priori $\mathbf{P}_k^-$ em função da "antiga" matriz de covariância a posteriori $\mathbf{P}_{k-1}$.

Com as Equações 3.26, 3.28, 3.22, 3.12 e 3.25 podemos agora resumir a estimativa recursiva do estado, conforme mostrado no Algoritmo 3. Esse algoritmo também inclui a inicialização do procedimento recursivo. Na ausência de qualquer observação no tempo $k=0$, podemos escolher a estimativa do estado inicial simplesmente como:

$$\hat{\mathbf{x}}_0 = E\left[\mathbf{x}_0\right]. \tag{3.29}$$

O valor inicial da matriz de covariância a posteriori é definido como:

$$\mathbf{P}_0 = E\left[\left(\mathbf{x}_0 - E\left[\mathbf{x}_0\right]\right)\left(\mathbf{x}_0 - E\left[\mathbf{x}_0\right]\right)^T\right]. \tag{3.30}$$

As escolhas para as condições iniciais são sugeridas de forma empírica em KALMAN (1960), com a vantagem de ambas serem estimativas não enviesadas de $\mathbf{x}_k$.

Algoritmo 3: Filtro de Kalman

Entrada: Modelo de espaço de estado $\mathbf{x}_{k+1} = \mathbf{A}_{k+1|k}\mathbf{x}_k + \mathbf{w}_k$ e $\mathbf{y}_k = \mathbf{H}_k\mathbf{x}_k + \mathbf{v}_k$, em que $\mathbf{w}_k$ e $\mathbf{v}_k$ são ruídos Gaussianos independentes, com média zero das matrizes de covariância $\mathbf{Q}_k$ e $\mathbf{R}_k$, respectivamente.

Saída: $\hat{\mathbf{x}}$, $\mathbf{P}$ e $\mathbf{G}$, em que $\hat{\mathbf{x}}$ é o vetor estimado de estados, $\mathbf{P}$ é a covariância do processo e $\mathbf{G}$ é a matriz de ganhos de Kalman.

begin

  Inicialização: Para $k = 0$, faça:

  $\hat{\mathbf{x}}_0 = E\left[\mathbf{x}_0\right]$,

  $\mathbf{P}_0 = E\left[\left(\mathbf{x}_0 - E\left[\mathbf{x}_0\right]\right)\left(\mathbf{x}_0 - E\left[\mathbf{x}_0\right]\right)^T\right]$.

  Computação: Para $k = 1, 2, ...$, calcule:

  `// Propagação da estimativa do estado`

  $\hat{\mathbf{x}}_k^- = \mathbf{A}_{k|k-1}\hat{\mathbf{x}}_{k-1}^-$

  `// Propagação do erro de covariância`

  $\mathbf{P}_k^- = \mathbf{A}_{k|k-1}\mathbf{P}_{k-1}\mathbf{A}_{k|k-1}^T + \mathbf{Q}_{k-1}$

  `// Matriz de ganho de Kalman`

  $\mathbf{G}_k = \mathbf{P}_k^-\mathbf{H}_k^T\left[\mathbf{H}_k\mathbf{P}_k^-\mathbf{H}_k^T + \mathbf{R}_k\right]^{-1}$

  `// Atualização da estimativa do estado`

  $\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{G}_k\left(\mathbf{y}_k - \mathbf{H}_k\hat{\mathbf{x}}_k^-\right)$

  `// Atualização erro de covariância`

  $\mathbf{P}_k = \left(\mathbf{I} - \mathbf{G}_k\mathbf{H}_k\right)\mathbf{P}_k^-$

end

3.3 Estimativa de Parâmetros do Filtro de Kalman

A modelagem de sistemas dinâmicos tem sido uma importante ferramenta em diversas áreas do conhecimento, com aplicações em campos tão distintos como engenharia e ciências sociais (CANDY, 2005). Em aplicações práticas, é essencial conhecer as suas características mais importantes. Primeiro, essas ferramentas são estocásticas, pois as observações de suas saídas podem ser descritas como uma função de suas entradas acrescidas de um termo de ruído. Segundo, elas podem ser caracterizadas por um estado interno em uma dimensão finita, o qual não é diretamente observável, mas que pode resumir a qualquer tempo todas as informações sobre o comportamento passado do modelo, com o objetivo de prever a sua evolução no futuro.

Do ponto de vista da modelagem, o comportamento estocástico é essencial para a utilização de modelos com um conjunto pequeno de parâmetros fixos. Através da modelagem do

estado interno, é possível "desacoplar" a sua dinâmica interna do processo de observação. Por exemplo, para modelar a sequência de imagens de vídeo de um balão flutuando ao vento, seria computacionalmente muito custoso tentar prever diretamente todos os vetores de intensidade de *pixels* a partir dos vários vetores com as características passadas das imagens. É muito mais eficiente tentar inferir o verdadeiro estado do balão, no qual o estado é formado pelas características que definem a sua posição, velocidade e orientação. Dessa forma, o processo que rege a dinâmica do balão é desacoplado do seu processo de observação, o qual mapeia o estado atual em um vetor de *pixels*.

É possível escrever equações que mapeiam diretamente esses sistemas dinâmicos, com base no conhecimento prévio da estrutura do problema e suas possíveis fontes de ruído. Nesses casos, queremos inferir o estado escondido do sistema a partir de uma sequência de observações das entradas e saídas do próprio sistema. Entretanto, em muitos casos, o número exato de parâmetros ou a estrutura interna do sistema em si são desconhecidos. Para esses cenários, a dinâmica do sistema deverá ser identificada a partir das sequências de suas observações.

O algoritmo do filtro de Kalman, visto na Seção 3.2, foi desenvolvido originalmente para aplicações da engenharia, nas quais os modelos físicos que descrevem a dinâmica dos sistemas são previamente conhecidos. Se conhecemos todos os parâmetros do filtro, a sua utilização resume-se à aplicação dos passos mostrados no Algoritmo 3 para inferir a sequência dos estados escondidos. A incerteza do processo é representada pela distribuição a posteriori das variáveis que descrevem o estado escondido, dada uma certa sequência de observações. O filtro de Kalman permite uma solução ótima para esse problema quando as funções que descrevem essas distribuições assumem um formato linear.

Baseado nas Equações 3.1 e 3.3, as quais definem o modelo de estado de espaço, assumimos que as matrizes $\mathbf{H}_k$, $\mathbf{A}_k$, $\mathbf{R}_k$ e $\mathbf{Q}_k$ são previamente conhecidas. Isso permitiu a derivação das equações do filtro de Kalman e o estudo do seu comportamento recursivo. Entretanto, em aplicações de aprendizagem sequencial com modelos lineares dinâmicos, é raro encontrar essas matrizes com valores previamente conhecidos. Nesse cenário, vamos assumir que as matrizes do modelo em questão dependem de um vetor de parâmetros desconhecidos, denotados aqui por ψ. Esses parâmetros normalmente são constantes no tempo, mas existem modelos que permitem atualizações de forma temporal. De toda forma, a dinâmica de ψ deverá ser capaz de preservar a natureza linear e Gaussiana dos modelos em questão.

A abordagem clássica para estimar ψ é através do cálculo da máxima verossimilhança (CANDY, 2005), a qual assume que os parâmetros são determinísticos e desconhecidos ao mesmo tempo. Esse método produz a "melhor" estimativa do valor que maximiza a probabilidade das medições, dado que o valor do parâmetro é provavelmente verdadeiro. No problema de estimativa, os dados da medição são fornecidos em conjunto com a estrutura da função de densidade de probabilidade, mas com parâmetros desconhecidos e que precisam ser determinados a partir da sequência de medições. Portanto, a estimativa da máxima verossimilhança pode ser considerada heuristicamente como o valor dos parâmetros que melhor "explica" os dados analisados a partir

de sua estimativa mais provável.

Formalmente, suponha que temos n vetores aleatórios, $\mathbf{y}_1, ..., \mathbf{y}_n$, cuja distribuição depende de um parâmetro desconhecido ψ. Vamos denotar a densidade conjunta das observações para um valor específico do parâmetro como $p(\mathbf{y}_1, ..., \mathbf{y}_n; \psi)$. A função de verossimilhança é definida como sendo a probabilidade da densidade dos dados observados em função de ψ, até um certo fator constante. Se denotarmos a verossimilhança por L, podemos escrever a função $L(\psi) = C \cdot p(\mathbf{y}_1, ..., \mathbf{y}_n; \psi)$, em que C é uma constante. Para um modelo linear dinâmico, podemos descrever a densidade conjunta das observações da seguinte forma:

$$p(\mathbf{y}_1, ..., \mathbf{y}_n; \psi) = \prod_{k=1}^{n} p(\mathbf{y}_k | \mathbf{y}_{1:k-1}; \psi), \tag{3.31}$$

em que $p(\mathbf{y}_k | \mathbf{y}_{1:k-1}; \psi)$ é a densidade condicional de $\mathbf{y}_k$, dadas as observações até o tempo $k-1$, assumindo que ψ é o valor do parâmetro desconhecido. Se assumirmos que os termos da Equação 3.31 são densidades Gaussianas com média f_k e variância $\mathbf{Q}_k$, podemos escrever a função logarítmica da verossimilhança como:

$$\ell(\psi) = -\frac{1}{2} \sum_{k=1}^{n} \log |\mathbf{Q}_k| - \frac{1}{2} \sum_{k=1}^{n} (\mathbf{y}_k - f_k)^T \mathbf{Q}_k^{-1} (\mathbf{y}_k - f_k), \tag{3.32}$$

em que f_k e $\mathbf{Q}_k$ dependem implicitamente de ψ. A Equação 3.32 pode ser numericamente maximizada para se obter a estimativa da máxima verossimilhança de ψ, de acordo com a expressão:

$$\hat{\psi} = \arg\max_{\psi} \ell(\psi). \tag{3.33}$$

O método de estimativa da máxima verossimilhança possui uma série de propriedades desejáveis, cujas provas podem ser vistas em detalhes em PORAT (1994). As estimativas da máxima verossimilhança são consistentes, assintoticamente eficientes, assintoticamente Gaussianas, invariantes e estatisticamente suficientes. Entretanto, alguns problemas podem ser observados no processo de otimização númerica de $\ell(\psi)$. Em SHUMWAY; STOFFER (2010) são mostradas situações nas quais a estimativa da máxima verossimilhança pode apresentar vários máximos locais. Isso implica que ao iniciar o processo de otimização a partir de diferentes pontos, podemos encontrar diferentes valores para a máxima verossimilhança. Uma verossimilhança "plana" é outro problema associado ao método, no qual podemos encontrar valores praticamente iguais para a estimativa, mesmo iniciando o processo com valores diferentes.

Uma alternativa ao método de estimativa da máxima verossimilhança é proposto em RAJAMANI; RAWLINGS (2009) com a definição da autocovariância de mínimos quadrados (ALS), a qual estima as matrizes de covariância dos ruídos a partir das observações do modelo. O trabalho de MONTANA; TRIANTAFYLLOPOULOS; TSAGARIS (2009), chamado de *Flexible Least Squares* (FLS), propõe a estimativa das matrizes através da definição de uma função de custo dependente de um limiar pré-estabelecido. Nessa abordagem, os autores mostraram a

equivalência da estimativa do FLS com o método da máxima verossimilhança, apontando como vantagem o baixo custo computacional da primeira. Outra alternativa para o cálculo das matrizes é apresentada em XI; PENG; CHEN (2014). Nesse trabalho, as matrizes de covariância dos processos de estado e medição são atualizadas de forma recursiva, no mesmo esquema iterativo do filtro de Kalman estendido (EKF). A justificativa para essa abordagem deriva do fato que o ruído do processo poderá variar com o tempo, exigindo um ajuste nesses parâmetros. Cada matriz é mapeada como um estado do modelo não linear em questão e através da maximização de uma função de densidade seus valores são atualizados a cada ciclo do EKF. Os autores do método mostram que a atualização sequencial dos parâmetros permite a redução da linearização do erro de previsão, aumentando a precisão dos modelos que utilizam o algoritmo. Entretanto, uma análise preliminar da complexidade dessa abordagem indica um aumento no custo computacional para o processo de aprendizagem sequencial, pois o número de inversões e multiplicações de matrizes cresce em relação ao algoritmo original do filtro de Kalman e de suas extensões não lineares.

Uma solução de baixo custo computacional para a estimativa das covariâncias dos ruídos é apresentada em SARKKA (2007). Nesse trabalho, as equações de estado são mapeadas em um modelo linear invariante com o tempo, de acordo com a seguinte equação:

$$\frac{dx(t)}{dt} = \mathbf{F}x(t) + \mathbf{L}w(t), \tag{3.34}$$

em que $\mathbf{F}$ e $\mathbf{L}$ são matrizes constantes que descrevem o comportamento do modelo e $\mathbf{w}(t)$ é um ruído branco com densidade $\mathbf{Q}_c$. As condições iniciais do sistema acima são dadas por $\mathbf{x}(0) \sim N(\mathbf{m}(0), \mathbf{P}(0))$, em que $\mathbf{m}(0)$ e $\mathbf{P}(0)$ são parâmetros escolhidos a partir do conhecimento prévio do sistema linear em questão. O cálculo da estimativa dos ruídos passa pela discretização do modelo da Equação 3.34 na forma das Equações 3.1 e 3.3. A solução para discretizar as matrizes $\mathbf{A}_k$ e $\mathbf{Q}_k$ é apresentada em detalhes em SARKKA (2007) e seguem a forma a seguir:

$$\mathbf{A}_k = \exp(\mathbf{F}\Delta t_k), \tag{3.35}$$

$$\mathbf{Q}_k = \int_0^{\Delta t_k} \exp(\mathbf{F}(\Delta t_k - \tau))\mathbf{L}\mathbf{Q}_c\mathbf{L}^T \exp(F(\Delta t_k - \tau))^T d\tau, \tag{3.36}$$

em que $\Delta t_k = t_{k+1} - t_k$ é a dimensão da escala de discretização. Em alguns casos, a matriz de covariância $\mathbf{Q}_k$ não pode ser calculada analiticamente através da Equação 3.36. Para essas situações, a matriz poderá ser calculada através da integração por decomposição de frações parciais, de acordo com a expressão:

$$\begin{pmatrix} \mathbf{C}_k \\ \mathbf{D}_k \end{pmatrix} = \exp\left\{ \begin{pmatrix} \mathbf{F} & \mathbf{L}\mathbf{Q}_c\mathbf{L}^T \\ 0 & -\mathbf{F}^T \end{pmatrix} \Delta t_k \right\} \begin{pmatrix} 0 \\ \mathbf{I} \end{pmatrix}. \tag{3.37}$$

De forma direta, a matriz $\mathbf{Q}_k$ pode ser calculada pela equação:

$$\mathbf{Q}_k = \mathbf{C}_k {\mathbf{D}_k}^{-1}. \tag{3.38}$$

Do ponto de vista de complexidade computacional, essa abordagem é mais eficiente que as soluções baseadas nos métodos de máxima verossimilhança e de autocovariância de mínimos quadrados, pois não é necessário adotar um esquema iterativo para estimar as matrizes $\mathbf{A}_k$ e $\mathbf{Q}_k$.

3.4 Filtro de Kalman e o Problema da Multicolinearidade

O problema da multicolinearidade ocorre nos modelos de regressão sempre que existe uma forte dependência linear entre as variáveis independentes do modelo. A multicolinearidade pode ser gerada por uma perturbação nos dados de treinamento, podendo ocasionar um aumento da variância nas inferências estatísticas produzidas pelo modelo, conforme discutido na Seção 1.1. Em comparação aos métodos tradicionais de mínimos quadrados ordinários (OLS) ou recursivos (RLS), a aplicação do filtro de Kalman em problemas de aprendizagem sequencial é uma forma eficiente de tratar os efeitos da multicolinearidade (WATSON, 1983). Esse tratamento poderá ocorrer de duas formas: (i) agregando mais informação no início do processo de filtragem, e (ii) reduzindo ou eliminando a necessidade de inversões de matrizes (JAZWINSKI, 2007).

A primeira estratégia está relacionada à discussão da Seção anterior. Através da estimativa das matrizes de covariância do filtro, podemos incorporar informação sobre a distribuição dos parâmetros e a incerteza associada a eles. A atualização do erro de covariância do filtro de Kalman é dependente desses parâmetros, e em tese, essa estratégia poderia ser utilizada para reduzir a variância da estimativa dos estados.

A segunda estratégia independe da estimativa correta das matrizes de covariância do processo e medição. Em problemas de mínimos quadrados, quando dois ou mais elementos de uma matriz $\mathbf{X}$ são perfeitamente colineares, a matriz $\mathbf{X}^T\mathbf{X}$ é singular e a sua inversa $\left(\mathbf{X}^T\mathbf{X}\right)^{-1}$ não existe, tornando impossível o cálculo direto das estimativas das variáveis do modelo. Quando a multicolinearidade não é perfeita, a matriz inversa poderá ser calculada através de um procedimento numérico, ficando sujeita à precisão da máquina onde o cálculo foi realizado. Uma vantagem da aplicação do filtro de Kalman em substituição aos métodos tradicionais de mínimos quadrados é a quantidade reduzida de inversões matriciais de seu algoritmo. De acordo com o Algoritmo 3, apenas na atualização da matriz de ganhos é realizada a inversão do termo representado pela covariância da medição. É fácil observar que a matriz $\mathbf{H}_k\mathbf{P}_k^-\mathbf{H}_k^T$ não é singular e a sua inversa pode ser facilmente calculada. A única possibilidade de $\mathbf{H}_k\mathbf{P}_k^-\mathbf{H}_k^T$ ser singular é se $\mathbf{P}_k^-$ for a covariância de uma combinação linear "perfeita" das variáveis dependentes, o que reduziria o termo à matriz identidade. Ainda assim, seria possível utilizar o filtro na presença de multicolinearidade, pois a atualização da matriz de ganhos dependeria apenas da covariância da medição $\mathbf{R}_k$. Esse comportamento do filtro garante que a sua aplicação é possível, mesmo

quando existe colinearidade entre os elementos de $\mathbf{H}$.

3.5 Filtro de Kalman Não Linear

Em muitas aplicações práticas, encontramos sistemas dinâmicos que não são lineares por natureza. Para essa classe de sistemas, tanto a dinâmica quanto o processo de medição podem ser não lineares. Como consequência, a estimativa de estados através do filtro de Kalman não se beneficia das vantagens descritas nas seções anteriores.

O *Extended Kalman Filter* (EKF) amplia o escopo da versão original do algoritmo para problemas de filtragem ótima em sistemas não-lineares (JAZWINSKI, 2007). Essa extensão utiliza uma aproximação Gaussiana da distribuição conjunta dos estados e das medições usando uma transformação linear baseada na série de Taylor, de acordo com as seguintes condições:

$$\begin{aligned} \mathbf{x} &\sim N(\mathbf{m}, \mathbf{P}), \\ y &= \mathbf{g}(\mathbf{x}), \end{aligned} \tag{3.39}$$

em que $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{y} \in \mathbb{R}^m$, $\mathbf{g} : \mathbb{R}^n \mapsto \mathbb{R}^m$ é uma função não linear, $\mathbf{m}$ é a média estimada do estado e $\mathbf{P}$ é a covariância do estado. A aproximação linear das variáveis $\mathbf{x}$ e $\mathbf{y}$ é dada pela equação:

$$\begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix} \sim N\left(\begin{pmatrix} \mathbf{m} \\ \mu_L \end{pmatrix}, \begin{pmatrix} \mathbf{P} & \mathbf{C}_L \\ \mathbf{C}_L^T & \mathbf{S}_L \end{pmatrix} \right), \tag{3.40}$$

em que

$$\begin{aligned} \mu_L &= \mathbf{g}(\mathbf{m}) \\ \mathbf{S}_L &= \mathbf{G}_\mathbf{x}(\mathbf{m}) \mathbf{P} \mathbf{G}_\mathbf{x}^T(\mathbf{m}) \\ \mathbf{C}_L &= \mathbf{P} \mathbf{G}_\mathbf{x}^T(\mathbf{m}) \end{aligned} \tag{3.41}$$

e $\mathbf{G}_\mathbf{x}(\mathbf{m})$ é a matriz Jacobiana de $\mathbf{g}$ com os elementos

$$[\mathbf{G}_\mathbf{x}(\mathbf{m})]_{j,j'} = \left. \frac{\partial \mathbf{g}_j(\mathbf{x})}{\partial \mathbf{x}_{j'}} \right|_{\mathbf{x}=\mathbf{m}}. \tag{3.42}$$

A partir das definições acima, a descrição do EKF pode ser feita de acordo com as seguintes equações do modelo de estado de espaço:

$$\mathbf{x}_k = \mathbf{f}(\mathbf{x}_{k-1}, k-1) + \mathbf{q}_{k-1}, \tag{3.43}$$

$$\mathbf{y}_k = \mathbf{h}(\mathbf{x}_k, k) + \mathbf{r}_k, \tag{3.44}$$

em que $\mathbf{x}_k \in \mathbb{R}^n$ é o estado, $\mathbf{y}_k \in \mathbb{R}^m$ é a medição, $\mathbf{q}_{k-1} \sim N(0, \mathbf{Q}_{k-1})$ é o ruído do processo de estado, $\mathbf{r}_{k-1} \sim N(0, \mathbf{R}_{k-1})$ é o ruído da medição, $\mathbf{f}$ é a função não linear do modelo de estado e $\mathbf{h}$ é a função não linear do modelo de medição. O filtro de Kalman estendido aproxima a

distribuição do estado $\mathbf{x}_k$ para as observações $\mathbf{y}_{1:k}$ através da seguinte Gaussiana:

$$p(\mathbf{x}_k|\mathbf{y}_{1:k}) \approx N(\mathbf{x}_k|\mathbf{m}_k, \mathbf{P}_k). \tag{3.45}$$

De forma semelhante à versão linear do filtro de Kalman, a extensão não linear é dividida em duas fases de previsão e correção das estimativas, de acordo com as equações:

Previsão.

$$\begin{aligned} \mathbf{m}_k^- &= \mathbf{f}(\mathbf{m}_{k-1}, k-1) \\ \mathbf{P}_k^- &= \mathbf{F}_x(\mathbf{m}_{k-1}, k-1)\,\mathbf{P}_{k-1}\mathbf{F}_x^T(\mathbf{m}_{k-1}, k-1) + \mathbf{Q}_{k-1} \end{aligned} \tag{3.46}$$

Correção.

$$\begin{aligned} \mathbf{v}_k &= \mathbf{y}_k - \mathbf{h}\left(\mathbf{m}_k^-, k\right) \\ \mathbf{S}_k &= \mathbf{H}_x\left(\mathbf{m}_k^-, k\right)\mathbf{P}_k^-\mathbf{H}_x^T\left(\mathbf{m}_k^-, k\right) + \mathbf{R}_k \\ \mathbf{G}_k &= \mathbf{P}_k^-\mathbf{H}_x^T\left(\mathbf{m}_k^-, k\right)\mathbf{S}_k^{-1} \\ \mathbf{m}_k &= \mathbf{m}_k^- + \mathbf{G}_k\mathbf{v}_k \\ \mathbf{P}_k &= \mathbf{P}_k^- - \mathbf{G}_k\mathbf{S}_k\mathbf{G}_k^T \end{aligned} \tag{3.47}$$

em que as matrizes $\mathbf{F}_x(\mathbf{m}_{k-1}, k-1)$ e $\mathbf{H}_x\left(\mathbf{m}_k^-, k\right)$ são as Jacobianas de $\mathbf{f}$ e $\mathbf{h}$, com os elementos

$$\left[\mathbf{F}_x(\mathbf{m}, k-1)\right]_{j,j'} = \left.\frac{\partial \mathbf{f}_j(\mathbf{x}, k-1)}{\partial \mathbf{x}_{j'}}\right|_{\mathbf{x}=\mathbf{m}} \tag{3.48}$$

$$\left[\mathbf{H}_x(\mathbf{m}, k)\right]_{j,j'} = \left.\frac{\partial \mathbf{h}_j(\mathbf{x}, k)}{\partial \mathbf{x}_{j'}}\right|_{\mathbf{x}=\mathbf{m}}. \tag{3.49}$$

Observe que as únicas diferenças do EKF em relação à sua versão linear são as matrizes Jacobianas $\mathbf{F}_x(\mathbf{m}_{k-1}, k-1)$ e $\mathbf{H}_x\left(\mathbf{m}_k^-, k\right)$ em substituição às matrizes $\mathbf{A}_k$ e $\mathbf{H}_k$.

Apesar de ser a solução mais utilizada para problemas não lineares, o EKF possui algumas limitações, conforme o relato de JULIER; UHLMANN (2004), entre as quais destacamos a possibilidade de divergência do erro de aprendizado e a obrigatoriedade da existência das matrizes Jacobianas e Hessianas do filtro para realizar a transformação de Taylor.

3.6 Trabalhos Relacionados

Nessa seção, vamos discutir brevemente algumas extensões do filtro de Kalman que podem ser aplicadas para a estimativa de estados em sistemas dinâmicos não lineares, através de aproximações Gaussianas da distribuição conjunta dos estados e suas medições.

Uma alternativa à transformação de Taylor para gerar uma aproximação Gaussiana é descrita nos trabalhos de JULIER; UHLMANN; DURRANT-WHYTE (1995) e JULIER; UHLMANN (2004), com a aplicação da transformada de incerteza (tradução livre de *unscented transform*) para esse fim. Nessa abordagem, é escolhido um número fixo de pontos que capturam

os momentos (média e covariância) da distribuição original dos estados. Em seguida, esses pontos são propagados através de uma função não linear. A vantagem da transformada de incerteza sobre a aproximação de Taylor é que a primeira captura melhor os momentos de alta ordem gerados pela transformação não linear. Além disso, as matrizes Jacobianas e Hessianas não são necessárias no processo, tornando a estimativa mais simples do ponto de vista computacional. A extensão é chamada de *Unscented Kalman Filter* (UKF) e utiliza a transformada de incerteza para obter a aproximação Gaussiana em soluções que envolvam filtragem para problemas não lineares.

É possível modificar o procedimento do UKF através da formulação de uma variável de estado expandida, na qual o estado e o ruído são concatenados de forma conjunta, de forma que o efeito dos ruídos do processo e da medição possam ser utilizados para capturar as informações dos momentos de ordem ímpar. A definição do *Augmented Unscented Kalman Filter* (AUKF) estabelece que os pontos gerados na fase de predição também sejam usados na fase de atualização do filtro, garantindo dessa forma que os efeitos dos termos representados pelos ruídos sejam propagados corretamente (WU et al., 2006). Entretanto, se forem gerados novos pontos na fase de atualização, a abordagem expandida dará os mesmos resultados que a sua versão não-expandida, assumindo que os ruídos são aditivos. Se os ruídos não forem aditivos, a versão expandida deverá produzir estimativas mais precisas que a sua versão não expandida, mesmo se novos pontos forem criados durante a fase de atualização.

Com o objetivo de unificar as variantes do filtro de Kalman, alguns trabalhos tentam uma formulação comum para o problema da não linearidade (MAYBECK, 1982; ITO; XIONG, 2000). Em problemas de filtragem Gaussiana, assume-se que as distribuições apresentam comportamento de normalidade, o que permite que as equações lineares do filtro de Kalman possam ser adaptadas para um modelo não linear de estado de espaço. O modelo do *Gauss–Hermite Kalman Filter* (GHKF) (WU et al., 2006) utiliza uma regra de quadratura definida através de uma soma ponderada para gerar a aproximação Gaussiana. Os pesos são calculados de forma analítica através da utilização de um *kernel* Gaussiano. A solução para o cálculo dos pesos melhora a precisão das estimativas de estado, mas sofre com o problema da dimensionalidade, tornando difícil a sua aplicação em problemas reais.

Uma solução para o problema da dimensionalidade do GHKF é a utilização de métodos de cubatura para o cálculo dos pesos que irão gerar a aproximação Gaussiana. A proposta do *Cubature Kalman Filter* (CKF) (ARASARATNAM; HAYKIN, 2009) restringe o domínio dos pesos a uma região do espaço definida por uma integral esférica, reduzindo o esforço computacional do algoritmo.

Em vários cenários de utilização do filtro de Kalman, é razoável assumir que a dinâmica do sistema muda com a passagem do tempo. Esse fenômeno pode ser modelado como um problema de transição das matrizes de probabilidade de uma cadeia de Markov de primeira ordem (BLOM; BAR-SHALOM, 1988). Essa estrutura é responsável por caracterizar as transições de estado, sendo comumente referenciadas como sistemas de comutação Markovianos. Nesse tipo

de sistema, a filtragem dos estados precisa ser feita para cada sequência possível do modelo, ou seja, para n modelos são necessários n^k filtros para processar a k-ésima medição. Com base nesse conceito, em YAAKOV; THIAGALINGAM; RONG (2002) é proposto o *Interacting Multiple Model Kalman Filter* (IMM-KF) para sistemas de comutação Markovianos. Essa versão do filtro consiste de três passos básicos: interação, filtragem e combinação. A cada mudança do modelo no tempo são produzidas combinações das estimativas dos estados, as quais são geradas por versões anteriores de todas as sequências de filtros. Em seguida, o filtro de Kalman original é executado para cada modelo gerado, com a saída final sendo fornecida por uma combinação linear ponderada dos estados estimados por cada filtro. Os pesos são escolhidos de acordo com a probabilidade de cada modelo, calculada durante a fase de filtragem.

3.7 Considerações Finais

Esse capítulo mostrou os conceitos básicos do processo de aprendizagem baseado no filtro de Kalman. A partir da definição de um modelo de estado de espaço, usamos todo o conjunto de dados observados para encontrar o erro mínimo quadrado da estimativa de um determinado estado. Apresentamos a derivação das equações do filtro de Kalman e o procedimento recursivo para a sua aplicação em problemas de filtragem de modelos lineares dinâmicos. Uma atenção especial foi dada ao problema da estimativa dos parâmetros desconhecidos do filtro, através do método da máxima verossimilhança e outras alternativas com baixa complexidade computacional. As principais extensões do filtro de Kalman abordam o problema da não linearidade dos processos de medição, com soluções voltadas para a aproximação Gaussiana das distribuições desses processos.

O próximo capítulo irá apresentar o método proposto na tese em maiores detalhes.

4

Método Proposto

Este Capítulo apresenta os algoritmos do método proposto nesta tese. As versões do método tratam de problemas de aprendizagem sequencial através de um modelo de espaço de estados, de acordo com as condições de linearidade do sistema em estudo. Após a formulação do problema, os algoritmos são formalmente descritos, com uma discussão sobre a melhor estratégia para estimativa de seus parâmetros. Em seguida, é feita uma análise da complexidade computacional das duas soluções. Por fim, uma discussão das limitações do método proposto é apresentada e suas características são comparadas com os trabalhos relacionados.

4.1 Justificativa

Apesar dos avanços recentes na teoria de aprendizagem sequencial com a OS-ELM, alguns pontos ainda encontram-se em aberto, relacionados principalmente com a definição da versão *batch* da ELM. Em LIU et al. (2015) e HUANG et al. (2015), são relatados problemas decorrentes da distribuição dos parâmetros da camada escondida. É mostrado que o caráter aleatório dos pesos afeta a capacidade de generalização dos modelos baseados na ELM, medida pelo erro de previsão. Outro ponto de atenção importante foi estudado em ZHAO; WANG; PARK (2012) e LI; NIU (2013), no qual é mostrado que o poder de generalização da ELM pode ser fortemente influenciado pela singularidade da matriz de saída da camada escondida. Levando em consideração que a estimativa dos pesos de saída da OS-ELM é feita, via de regra, através da solução de mínimos quadrados da Equação 2.67, podemos nos deparar com o problema da multicolinearidade na matriz que representa esses pesos. Na teoria da análise de regressão, quando duas ou mais variáveis independentes estão fortemente correlacionadas, dizemos que ocorreu o fenômeno estatístico da multicolinearidade. A principal consequência desse fenômeno é o aumento no desvio padrão dos coeficientes do modelo de regressão. Em ZHAO; WANG; CHAI (2013) é mostrado que a multicolinearidade na matriz de saída da camada escondida da ELM gera distorções no cálculo dos pesos, ocasionando a instabilidade do modelo através do aumento da variância de suas estimativas.

Entre os algoritmos que estendem a OS-ELM, encontramos poucos trabalhos que tratam

diretamente do problema da multicolinearidade na matriz dos pesos de saída da camada escondida. Dos trabalhos relacionados, mostrados na Seção 2.4.2, apenas o ReOS-ELM e o LS-IELM apresentam propostas para tratar diretamente a multicolinearidade através de mecanismos de regularização. Entretanto, as soluções mostradas nesses trabalhos apresentam como desvantagem a inclusão de novos parâmetros nos métodos e a consequente necessidade de otimização. Especificamente para o ReOS-ELM, a abordagem adotada para atualizar os pesos de saída aumenta o tempo de treinamento na execução do algoritmo. Os demais trabalhos relacionados, OS-ELM-TV e TOSELM, tratam os efeitos da multicolinearidade através da adoção de parâmetros que variam com a passagem do tempo. Nessas abordagens, além da introdução de parâmetros adicionais, a estratégia adotada para o ajuste dos pesos aumenta a complexidade computacional dos algoritmos, pois os mecanismos adotados requerem novos processos iterativos para ajustar as estimativas dos pesos.

Uma solução eficiente para o tratamento dos efeitos da multicolinearidade em problemas de aprendizagem sequencial é a utilização de modelos de espaço de estados baseados no filtro de Kalman (KALMAN, 1960). Em WATSON (1983) é apresentada uma prova formal de que essa classe de modelos é capaz de tratar do problema de multicolinearidade com desempenho superior à solução de mínimos quadrados recursivo (RLS). Na Seção 3.4, uma discussão foi apresentada sobre o tratamento dos efeitos da multicolinearidade pelo filtro de Kalman. Uma vantagem adicional dessa abordagem é a redução da complexidade computacional no processo de aprendizagem sequencial, em comparação ao RLS. Além disso, não é necessária a adoção de novos parâmetros, como constantes de regularização, ou de mecanismos de ajustes adicionais para as estimativas dos pesos da rede. Dessa forma, a abordagem baseada no filtro de Kalman mostra-se adequada para tratar o problema de multicolinearidade da OS-ELM, reduzindo a variância das estimativas e diminuindo o tempo de treinamento para problemas de aprendizagem sequencial.

Por fim, é importante destacar que os algoritmos propostos no presente trabalho podem ser aplicados a problemas genéricos de aprendizagem sequencial para regressão e classificação. Entretanto, o foco da pesquisa foi voltado para problemas de regressão e previsão de séries temporais, ficando fora do escopo dessa tese a utilização dos algoritmos para outros problemas de aprendizagem.

4.2 Formulação do Problema

O método proposto nesta tese tem como premissa a aprendizagem sequencial através de um modelo de espaço de estados. De forma simplificada, o problema consiste em definir um modelo baseado no filtro de Kalman para calcular a estimativa dos pesos de forma sequencial.

Sejam $\mathbf{H}$, $\boldsymbol{\beta}$ e $\mathbf{T}$ a matriz da camada escondida, a matriz dos pesos de saída da ELM e o vetor com as observações, de acordo com as Equações 2.31, 2.32 e 2.33, respectivamente. De acordo com a Equação 2.34, queremos encontrar $\hat{\boldsymbol{\beta}}$ de forma que o valor da norma $\left\|\mathbf{H}\hat{\boldsymbol{\beta}} - \mathbf{T}\right\|$

seja mínimo. Precisamos descrever o problema da norma mínima da ELM em termos de um modelo de espaço de estados. Vamos usar as equações de estado e de medição (Equações 3.1 e 3.3, respectivamente), fazendo com que $\mathbf{T}_k$ seja o conjunto de dados "observáveis" no tempo k, a matriz de saída da camada escondida $\mathbf{H}_k$ seja a matriz de medição no tempo k, $\boldsymbol{\beta}_k$ seja o estado atual no tempo k e $\boldsymbol{\beta}_{k-1}$ o seu estado anterior. Podemos descrever então a ELM em termos de um modelo de espaço de estados, conforme o conjunto de equações a seguir:

$$\boldsymbol{\beta}_k = \mathbf{A}\boldsymbol{\beta}_{k-1} + \mathbf{w}_k, \tag{4.1}$$

$$\mathbf{T}_k = \mathbf{H}_k\boldsymbol{\beta}_k + \mathbf{v}_k. \tag{4.2}$$

$\mathbf{A}$ é a matriz de transição de estados, $\mathbf{w}_k$ e $\mathbf{v}_k$ são ruídos Gaussianos independentes com média zero das matrizes de covariância $\mathbf{Q}_k$ e $\mathbf{R}_k$, respectivamente. Definimos a Equação 4.1 como sendo a Equação de Estado da ELM e a Equação 4.2 como a Equação de Medição da ELM.

Para validar a proposição das equações acima, vamos inicialmente usar os resultados da Equação 2.34, na qual o menor erro de treinamento denotado por $\boldsymbol{\varepsilon}$ é obtido pela solução de $\left\|\mathbf{H}\hat{\boldsymbol{\beta}} - \mathbf{T}\right\|$. Dessa forma, podemos escrever diretamente a seguinte expressão:

$$\mathbf{T} = \mathbf{H}\hat{\boldsymbol{\beta}} + \boldsymbol{\varepsilon}. \tag{4.3}$$

Sabendo que o erro de estimativa é ortogonal a qualquer função de $\mathbf{T}$, podemos deduzir pelo princípio da ortogonalidade, mostrado no Teorema 3.1.2, que o erro da estimativa é ortogonal a todos os dados históricos de $\mathbf{T}$. Assim, para a estimativa de $\mathbf{T}$ no tempo k, denotada por $\hat{\mathbf{T}}_k$, temos

$$E\left\{\hat{\mathbf{T}}_k|\mathbf{T}_k\right\} = 0. \tag{4.4}$$

Para a condição anterior, o estimador de variância mínima assume exatamente a forma da Equação 4.2, com o termo $\mathbf{v}_k$ sendo descrito como um ruído branco, com média zero e variância $\mathbf{R}_k$ (CANDY, 2005).

Uma solução sequencial para a Equação 4.1 é dada pela estimativa de $\boldsymbol{\beta}_k$ através do procedimento recursivo do filtro de Kalman, conforme mostrado no Algoritmo 3. Os ruídos $\mathbf{w}_k$ e $\mathbf{v}_k$ são inicialmente desconhecidos, pois devemos estimar a priori as matrizes de covâriancias $\mathbf{Q}_k$ e $\mathbf{R}_k$ através dos métodos descritos na Seção 3.3.

4.3 O Algoritmo Kalman Learning Machine (KLM)

O método proposto nesta tese, chamado de *Kalman Learning Machine* (KLM), tem como ideia principal a atualização dos pesos de saída da rede através de um modelo de espaço de estados baseado no filtro de Kalman. De forma similar à OS-ELM, o algoritmo proposto poderá ser inicializado com um pequeno bloco de instâncias de treinamento. No entanto, esse passo é

opcional, pois existe a possibilidade de não contarmos com um bloco de instâncias no início da aprendizagem sequencial. Após a inicialização do algoritmo, é feita a estimativa da matriz de transição $\mathbf{A}$ e da matriz de covariância $\mathbf{Q}_k$ usando as Equações 3.35 e 3.38. O cálculo da matriz $\mathbf{R}_k$ pode ser feito através da covariância das medições, quando o algoritmo é inicializado com um bloco de instâncias. Em seguida, para cada instância de treinamento que alimenta o KLM de forma sequencial, os pesos de saída são atualizados pelo filtro de Kalman em duas etapas: (i) previsão do estado $\boldsymbol{\beta}_k$ e da covariância do erro de previsão, e (ii) correção da previsão e da matriz de covariância dos estados. Todo o processo de treinamento do KLM, incluindo a fase de inicialização, pode ser visto no Algoritmo 4.

Levando em consideração que a fase de inicialização do KLM é semelhante ao da OS-ELM, vamos detalhar os passos da fase sequencial, na qual o filtro de Kalman é aplicado para atualizar os pesos de saída da rede. A fase de aprendizagem sequencial do KLM pode ser resumida através da seguinte sequência de passos:

1. Na fase de inicialização, após estimar a matriz inicial de pesos $\boldsymbol{\beta}_k$ no tempo k, calcule a matriz de transição $\mathbf{A}_k$ e a matriz de covariância $\mathbf{Q}_k$ através das Equações 3.35 e 3.38.

2. Se a matriz $\boldsymbol{\beta}_k$ tiver sido estimada através de um pequeno bloco de instâncias de treinamento, a estimativa de $\mathbf{R}_k$ poderá ser feita pelo cálculo direto da covariância das medições e a matriz de covariância do estado $\mathbf{P}_k$ será igual a $\left(\mathbf{H}_0^T \mathbf{H}_0\right)^{-1}$. Caso contrário, inicialize $\mathbf{R}_k$ e $\mathbf{P}_k$ como a matriz identidade.

3. Após a apresentação da instância de treinamento $(k+1)$, na seguinte forma

$$n_{k+1} = \{\mathbf{x}_i, \mathbf{y}_i\}, i = \left(\sum_{l=0}^{k} N_l\right) + 1, \ldots, \left(\sum_{l=0}^{k+1} N_l\right),$$

   calcule $\mathbf{T}_{t+1}$ e $\mathbf{H}_{k+1}$ usando as Equações 2.64 e 2.65, respectivamente.

4. Atualize a estimativa da medição do modelo, usando a matriz da camada escondida $\mathbf{H}_{k+1}$ e o valor anterior do estado $\boldsymbol{\beta}_k$, na forma da expressão:

$$\hat{\mathbf{T}}_{k+1} = \mathbf{H}_{k+1} \boldsymbol{\beta}_k.$$

5. Faça a propagação da estimativa do estado $\hat{\boldsymbol{\beta}}_{k+1}$ usando o estado anterior $\boldsymbol{\beta}_k$ e a matriz de transição, na seguinte forma:

$$\hat{\boldsymbol{\beta}}_{k+1} = \mathbf{A} \boldsymbol{\beta}_k.$$

   Observe que se definirmos a matriz $\mathbf{A}$ como sendo a identidade, a Equação de Estado será reduzida a um passeio aleatório com ruído $\mathbf{w}_k$.

6. Faça a propagação do erro da covariância na estimativa do estado através da expressão

$$\hat{\mathbf{P}}_{k+1} = \mathbf{A}\mathbf{P}_k\mathbf{A}^T + \mathbf{Q}_k.$$

7. Calcule a matriz de ganho de Kalman $\mathbf{G}_{k+1}$, de acordo com a expressão:

$$\mathbf{G}_{k+1} = \hat{\mathbf{P}}_{k+1}\mathbf{H}_{k+1}^T\left(\mathbf{H}_{k+1}\hat{\mathbf{P}}_{k+1}\mathbf{H}_{k+1}^T + \mathbf{R}_k\right)^{-1}.$$

   O termo $\left(\mathbf{H}_{k+1}\hat{\mathbf{P}}_{k+1}\mathbf{H}_{k+1}^T + \mathbf{R}_k\right)^{-1}$ é a previsão da covariância de medição.

8. Faça a correção da estimativa do estado $\boldsymbol{\beta}_{k+1}$ usando a matriz de ganho de Kalman $\mathbf{G}_{k+1}$ e a inovação do processo de medição, através da expressão:

$$\boldsymbol{\beta}_{k+1} = \beta_k + \mathbf{G}_{k+1}\left(\mathbf{T}_{k+1} - \hat{\mathbf{T}}_{t+1}\right).$$

9. Atualize a matriz de covariância do estado usando a matriz de ganho de Kalman $\mathbf{G}_{k+1}$, através da expressão:

$$\mathbf{P}_{k+1} = \hat{\mathbf{P}}_{k+1} - \mathbf{G}_{k+1}\left(\mathbf{H}_{k+1}\hat{\mathbf{P}}_{k+1}\mathbf{H}_{k+1}^T + \mathbf{R}_k\right)\mathbf{G}_{k+1}^T.$$

A sequência de passos acima é executada de forma recursiva até que todas as instâncias de treinamento sejam fornecidas de forma sequencial. Ao final do ciclo iterativo, o estado atual $\boldsymbol{\beta}_N$ determina a saída final do algoritmo, em que N é tamanho do conjunto de treinamento. Observe que as matrizes $\mathbf{A}$, $\mathbf{Q}_k$ e $\mathbf{R}_k$ não são atualizadas durante o aprendizado sequencial, seguindo os mesmos princípios da ELM em relação à atualização de parâmetros. O algoritmo do KLM atualiza apenas os pesos de saída da rede.

4.4 Versão Não Linear: Extended KLM (EKLM)

De acordo com a discussão da Seção 3.5, não podemos garantir *a priori* a linearidade dos modelos baseados no KLM. Para permitir a sua utilização em sistemas nos quais a dinâmica do estado ou do processo de medição sejam não lineares, podemos estender o KLM usando a versão não linear do filtro de Kalman. O algoritmo *Extended Kalman Learning Machine* (EKLM) substitui a matriz de transição $\mathbf{A}$ e a matriz da camada escondida $\mathbf{H}_k$ pelas matrizes Jacobianas $\mathbf{F}_{\mathbf{x}}(\boldsymbol{\beta}_{k-1}, k-1)$ e $\mathbf{H}_{\mathbf{x}}(\boldsymbol{\beta}_k, k)$, em que $\mathbf{x} \sim N\left(\boldsymbol{\mu}_\beta, \mathbf{Q}_k\right)$, de acordo com a Equação 3.42.

De forma intuitiva, podemos observar que a complexidade computacional do EKLM é maior que a sua versão linear, devido à necessidade de se calcular as derivadas parciais dos elementos das matrizes Jacobianas. No Capítulo 5 mostraremos a comparação de desempenho entre as versões linear e não linear do KLM, além de compará-los com os trabalhos relacionados da Seção 2.4.2. Observe que a fase de inicialização do EKLM é a mesma do KLM, com exceção do cálculo da matriz $\mathbf{A}$, que será igual à identidade. A sequência completa de passos do EKLM é apresentada no Algoritmo 5.

Algoritmo 4: Kalman Learning Machine (KLM)

Data: $\{\mathbf{x}_i, \mathbf{t}_i\}, \mathbf{x}_i \in \mathbb{R}^d,\ t_i \in \mathbb{R}^q,\ i = 1, \ldots, N$

Result: $\hat{\beta}$

begin /* Fase de Inicialização */

$N_0 \leftarrow \{x_i, t_i\}, x_i \in \mathbb{R}^d, t_i \in \mathbb{R}^q, i = 1, \ldots, N_0, N_0 \in N$;

$L \leftarrow nhid, N_0 \geq L$;

$\mathbf{a}_j \in \mathbb{R}^d$ and $\mathbf{b}_j \in \mathbb{R}^1, j = 1, \ldots, L$;

$$\mathbf{H}_0 \leftarrow \begin{bmatrix} G(\mathbf{a}_1, \mathbf{b}_1, \mathbf{x}_1) & \cdots & G(\mathbf{a}_L, \mathbf{b}_L, \mathbf{x}_1) \\ \vdots & \ddots & \vdots \\ G(\mathbf{a}_1, \mathbf{b}_1, \mathbf{x}_{N_0}) & \cdots & G(\mathbf{a}_L, \mathbf{b}_L, \mathbf{x}_{N_0}) \end{bmatrix}_{N_0 \times L};$$

$$\mathbf{T}_0 \leftarrow \begin{bmatrix} \mathbf{t}_1^T \\ \vdots \\ \mathbf{t}_{N_0}^T \end{bmatrix}_{N_0 \times m};$$

$k \leftarrow N_0$;

$\mathbf{P}_k \leftarrow \left(\mathbf{H}_0^T \mathbf{H}_0\right)^{-1}$;

$\boldsymbol{\beta}_k \leftarrow \mathbf{P}_k \mathbf{H}_0^T \mathbf{T}_0$;

$\mathbf{R}_k \leftarrow \mathbf{I}$;

$\mathbf{A} \leftarrow \exp(\mathbf{F}\Delta t_k)$; /* Equação 3.35 */

$\mathbf{Q}_k \leftarrow \mathbf{C}_k {\mathbf{D}_k}^{-1}$; /* Equação 3.38 */

end

repeat /* Fase de Aprendizado Sequencial */

increment(k);

$$N_{k+1} \leftarrow \{\mathbf{x}_i, \mathbf{t}_i\}, i = \left(\sum_{l=0}^{k} N_l\right) + 1, \ldots, \left(\sum_{l=0}^{k+1} N_l\right);$$

$$\mathbf{H}_{k+1} \leftarrow \begin{bmatrix} G\left(\mathbf{a}_1, \mathbf{b}_1, \mathbf{x}_{\left(\sum_{j=0}^{k} N_j\right)+1}\right) & \cdots & G\left(\mathbf{a}_L, \mathbf{b}_L, \mathbf{x}_{\left(\sum_{j=0}^{k} N_j\right)+1}\right) \\ \vdots & \cdots & \vdots \\ G\left(\mathbf{a}_1, \mathbf{b}_1, \mathbf{x}_{\sum_{j=0}^{k+1} N_j}\right) & \cdots & G\left(\mathbf{a}_L, \mathbf{b}_L, \mathbf{x}_{\sum_{j=0}^{k+1} N_j}\right) \end{bmatrix};$$

$$\mathbf{T}_{k+1} \leftarrow \begin{bmatrix} \mathbf{t}_{\left(\sum_{j=0}^{k} N_j\right)+1, \ldots, \left(\sum_{j=0}^{k+1} N_j\right)} \end{bmatrix}^T;$$

$\hat{\mathbf{T}}_{k+1} \leftarrow \mathbf{H}_{k+1} \beta_k$;

$\hat{\mathbf{P}}_{k+1} \leftarrow \mathbf{A}\mathbf{P}_k\mathbf{A}^T + \mathbf{Q}_k$;

$\mathbf{G}_{k+1} \leftarrow \hat{\mathbf{P}}_{k+1}\mathbf{H}_{k+1}^T\left(\mathbf{H}_{k+1}\hat{\mathbf{P}}_{k+1}\mathbf{H}_{k+1}^T + \mathbf{R}_k\right)^{-1}$;

$\boldsymbol{\beta}_{k+1} \leftarrow \boldsymbol{\beta}_k + \mathbf{G}_{k+1}\left(\mathbf{T}_{k+1} - \hat{\mathbf{T}}_{k+1}\right)$;

$\mathbf{P}_{k+1} \leftarrow \hat{\mathbf{P}}_{k+1} - \mathbf{G}_{k+1}\left(\mathbf{H}_{k+1}\hat{\mathbf{P}}_{k+1}\mathbf{H}_{k+1}^T + \mathbf{R}_k\right)\mathbf{G}_{k+1}^T$;

until $k = N$;

$\hat{\boldsymbol{\beta}} \leftarrow \beta_N$

Algoritmo 5: Extended Kalman Learning Machine (EKLM)

Data: $\{\mathbf{x}_i, \mathbf{t}_i\}, \mathbf{x}_i \in \mathbb{R}^d, \mathbf{t}_i \in \mathbb{R}^q, i = 1, \ldots, N$

Result: $\hat{\beta}$

begin /* Fase de Inicialização */

$N_0 \leftarrow \{\mathbf{x}_i, \mathbf{t}_i\}, \mathbf{x}_i \in \mathbb{R}^d, \mathbf{t}_i \in \mathbb{R}^q, i = 1, \ldots, N_0, N_0 \in N$;

$L \leftarrow nhid, N_0 \geq L$;

$\mathbf{a}_j \in \mathbb{R}^d$ and $\mathbf{b}_j \in \mathbb{R}^1, j = 1, \ldots, L$;

$$\mathbf{H}_0 \leftarrow \begin{bmatrix} G(\mathbf{a}_1, \mathbf{b}_1, \mathbf{x}_1) & \cdots & G(\mathbf{a}_L, \mathbf{b}_L, \mathbf{x}_1) \\ \vdots & \ddots & \vdots \\ G(\mathbf{a}_1, \mathbf{b}_1, \mathbf{x}_{N_0}) & \cdots & G(\mathbf{a}_L, \mathbf{b}_L, \mathbf{x}_{N_0}) \end{bmatrix}_{N_0 \times L};$$

$$\mathbf{T}_0 \leftarrow \begin{bmatrix} \mathbf{t}_1^T \\ \vdots \\ \mathbf{t}_{N_0}^T \end{bmatrix}_{N_0 \times m};$$

$k \leftarrow N_0$;

$\mathbf{P}_k \leftarrow \left(\mathbf{H}_0^T \mathbf{H}_0\right)^{-1}$;

$\boldsymbol{\beta}_k \leftarrow \mathbf{P}_k \mathbf{H}_0^T \mathbf{T}_0$;

$\mathbf{R}_k \leftarrow \mathbf{I}$;

$\mathbf{A} \leftarrow \mathbf{I}$

$\mathbf{Q}_k \leftarrow \mathbf{C}_k {\mathbf{D}_k}^{-1}$; /* Equação 3.38 */

end

repeat /* Fase de Aprendizado Sequencial */

increment(k);

$N_{k+1} \leftarrow \{\mathbf{x}_i, \mathbf{t}_i\}, i = \left(\sum_{l=0}^{k} N_l\right) + 1, \ldots, \left(\sum_{l=0}^{k+1} N_l\right)$;

$\mathbf{H}_{k+1} \leftarrow \mathbf{H}_{\mathbf{x}}\left(\boldsymbol{\beta}_{k+1}, k+1\right)$;

$$\mathbf{T}_{k+1} \leftarrow \left[\mathbf{t}_{\left(\sum_{j=0}^{k} N_j\right)+1, \ldots, \left(\sum_{j=0}^{k+1} N_j\right)}\right]^T;$$

$\hat{\mathbf{T}}_{k+1} \leftarrow \mathbf{H}_{k+1} \beta_k$;

$\hat{\mathbf{P}}_{k+1} \leftarrow \mathbf{F}_x\left(\boldsymbol{\beta}_k, k\right) \mathbf{P}_k \mathbf{F}_x\left(\boldsymbol{\beta}_k, k\right)^T + \mathbf{Q}_k$;

$\mathbf{G}_{k+1} \leftarrow \hat{\mathbf{P}}_{k+1} \mathbf{H}_{k+1}^T \left(\mathbf{H}_{k+1} \hat{\mathbf{P}}_{k+1} \mathbf{H}_{k+1}^T + \mathbf{R}_k\right)^{-1}$;

$\boldsymbol{\beta}_{k+1} \leftarrow \boldsymbol{\beta}_k + \mathbf{G}_{k+1}\left(\mathbf{T}_{k+1} - \hat{\mathbf{T}}_{k+1}\right)$;

$\mathbf{P}_{k+1} = \hat{\mathbf{P}}_{k+1} - \mathbf{G}_{k+1}\left(\mathbf{H}_{k+1} \hat{\mathbf{P}}_{k+1} \mathbf{H}_{k+1}^T + \mathbf{R}_k\right) \mathbf{G}_{k+1}^T$;

until $k = N$;

$\hat{\boldsymbol{\beta}} \leftarrow \beta_N$

4.5 Estimativa de Parâmetros do Método

A formulação dos algoritmos KLM e EKLM permite que um conjunto com N instâncias distintas possa ser aprendido de forma sequencial, com o tratamento dos efeitos da multicolinearidade da matriz $\mathbf{H}$ sendo feito através da filtragem dos ruídos. Para utilizar os algoritmos, precisamos definir a priori apenas um único parâmetro, o número de unidades da camada escondida, para em seguida realizar o treinamento de forma sequencial. Para a inicialização dos algoritmos propostos, deve-se observar também o tamanho do conjunto inicial de treinamento. Esse passo é opcional, pois o processo de treinamento pode ser iniciado a partir da primeira instância dos dados. Entretanto, a utilização do método proposto depende da inicialização correta da matriz de transição $\mathbf{A}$ e das matrizes de covariância $\mathbf{Q}_k$ e $\mathbf{R}_k$. Em CANDY (2005), é mostrada que a escolha da estratégia de inicialização dos parâmetros do filtro de Kalman pode ser considerada uma "arte", mas existem procedimentos numéricos que dispensam a otimização desses parâmetros. Em casos nos quais o treinamento começa a partir da primeira instância, devemos escolher os parâmetros iniciais $\boldsymbol{\beta}_0$ e $\mathbf{P}_0$ para permitir o esquema recursivo dos algoritmos. No entanto, conforme mostrado em JAZWINSKI (2007), os valores desses parâmetros são rapidamente "esquecidos" conforme as instâncias de treinamento alimentam os algoritmos de forma sequencial. Esse fato é importante, pois o valor de $\mathbf{P}_0$ raramente é conhecido no início do treinamento. De fato, as escolhas de $\boldsymbol{\beta}_0$ e $\mathbf{P}_0$ não influenciam as estimativas dos estados, pois os valores de $\boldsymbol{\beta}$ são rapidamente ajustados pela matriz de ganho de Kalman. O valor de $\mathbf{A}$ pode ser calculado diretamente pela Equação 3.35, para problemas nos quais é possível utilizar um pequeno conjunto de instâncias na inicialização dos algoritmos. Para problemas nos quais esse cenário não é possível, podemos atribuir $\mathbf{A} = \mathbf{I}$, fazendo com que o cálculo de $\boldsymbol{\beta}_k$ torne-se um simples passeio aleatório em que o termo do ruído possui covariância $\mathbf{Q}_k$.

Conforme visto nos Algoritmos 4 e 5, a matriz de ganhos de Kalman $\mathbf{G}_k$ é o componente responsável por ajustar o valor de $\boldsymbol{\beta}$ a cada iteração. De acordo com o cálculo dessa matriz, a variação do ganho depende de $\hat{\mathbf{P}}_{k+1}$, $\mathbf{Q}_k$ e $\mathbf{R}_k$. De fato, observamos que no cálculo de $\hat{\mathbf{P}}_{k+1}$ a incerteza da previsão é representada pela covariância do processo $\mathbf{Q}_k$. Podemos ver, de forma teórica, que se $k \to \infty$, então $\hat{\mathbf{P}}_{k+1} \to \mathbf{Q}_k$, o que implica que $\mathbf{Q}_k$ é um limite teórico para o erro de covariância. Para valores elevados de $\mathbf{Q}_k$, teremos valores elevados de $\hat{\mathbf{P}}_{k+1}$, o que indica maior incerteza do modelo. Para reduzir a incerteza, o valor de $\mathbf{Q}_k$ deverá ser pequeno, mas suficiente para ajustar o erro da covariância na estimativa do estado. De forma similar, se $k \to \infty$, então $\left(\mathbf{H}_{k+1}\hat{\mathbf{P}}_{k+1}\mathbf{H}_{k+1}^T + \mathbf{R}_k\right)^{-1} \to \mathbf{R}_k$. A previsão da covariância de medição influencia no cálculo da matriz de ganhos, de forma que valores menores de $\mathbf{R}_k$ aumentam a incerteza no processo de medição, enquanto valores elevados diminuem a incerteza. De forma heurística, podemos mostrar que a matriz de ganho é influenciada pela razão das covariâncias, de acordo com a seguinte expressão:

$$\mathbf{G}_k \propto \frac{\mathbf{Q}_k}{\mathbf{R}_k}. \tag{4.5}$$

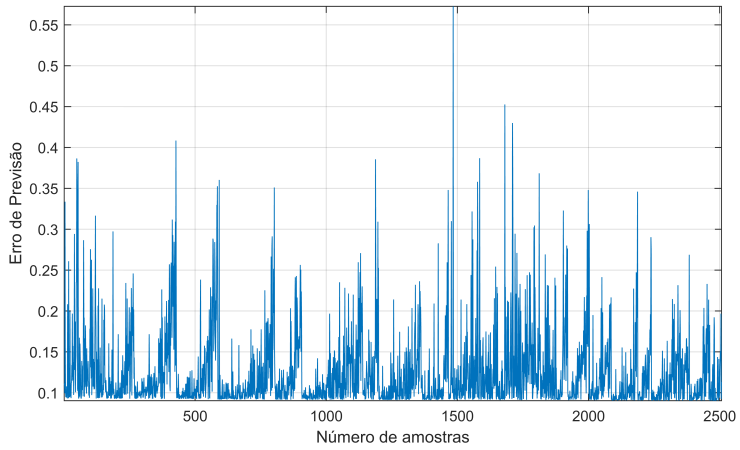
Podemos mostrar visualmente a dependência de $\mathbf{G}_k$ através das Figuras 4.1 e 4.2. Para simular os efeitos das escolhas dos parâmetros, usamos o problema da base de dados Abalone, obtida do repositório da *University of California Irvine* (UCI) (BACHE; LICHMAN, 2013). De forma empírica, escolhemos três valores para $\mathbf{Q}_k$ (1; 0,0001 e 10^{-16}) e $\mathbf{R}_k$ (10; 0,01 e 10^{-4}). Podemos observar na Figura 4.1 que para o maior valor de $\mathbf{Q}_k$ as estimativas apresentam um nível maior de ruído. Para o menor valor, as estimativas apresentam um comportamento de suavização. Visualmente, é possível comprovar para o exemplo em questão que o valor de $\mathbf{Q}_k$ é diretamente proporcional à covariância da previsão do estado, o que influencia no cálculo da matriz de ganho, e por consequência, no ajuste do estado $\boldsymbol{\beta}_{k+1}$. Um efeito similar pode ser visto na Figura 4.2 para os valores de $\mathbf{R}_k$. O valor do parâmetro é inversamente proporcional à matriz de ganhos de Kalman, pois o nível de ruído das estimativas aumenta quando $\mathbf{R}_k$ diminui. Apesar de não afetar diretamente $\hat{\mathbf{P}}_{k+1}$, esse parâmetro têm forte influência na matriz de ganhos através da covariância das inovações, de forma que o aumento de seus valores suaviza a curva do erro. De fato, mesmo no cenário no qual o ruído aumenta em função de valores pequenos de $\mathbf{R}_k$, a curva do erro de estimativa converge para um valor estável após o aprendizado de um certo número de instâncias.

Conforme discutido na Seção 3.3, encontramos na literatura as três principais abordagens para o cálculo de $\mathbf{Q}_k$: (i) estimativa por máxima verossimilhança, (ii) autocovariância de mínimos quadrados (ALS), e (iii) discretização matricial. A abordagem baseada na máxima verossimilhança possui um custo computacional mais elevado que as soluções baseadas em autocovariância de mínimos quadrados e discretização matricial, e por esse motivo, a escolha da estratégia de cálculo concentrou-se nos dois últimos métodos. O método ALS utiliza o produto de Kronecker para o cálculo da covariância, gerando matrizes com um grande número de dimensões, mesmo para problemas simples de aprendizagem sequencial. O tamanho interno das matrizes da ALS pode crescer rapidamente, o que pode ser impeditivo para aplicações que demandam aprendizagem em tempo real.

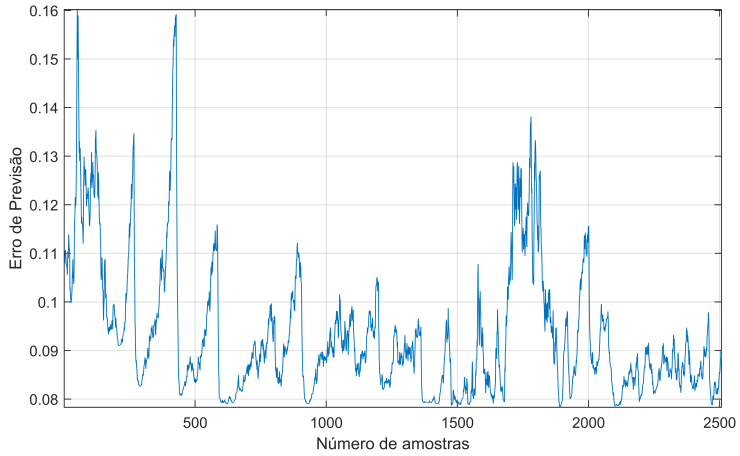
Em aplicações práticas do filtro de Kalman, o cálculo da matriz $\mathbf{R}_k$ poderá ser feito de forma simples e direta a partir da covariância das medições, calculadas em um pequeno conjunto de treinamento, ou ser inicializada com a matriz identidade. O cálculo direto de $\mathbf{Q}_k$ normalmente é mais complexo, pois não conhecemos a priori o processo que estamos estimando. O método proposto nessa tese utiliza a discretização matricial para calcular de forma direta o valor da covariância $\mathbf{Q}_k$, usando a Equação 3.38 a partir de um pequeno conjunto com instâncias de treinamento. Sob as condições em que as matrizes $\mathbf{Q}_k$ e $\mathbf{R}_k$ são constantes, tanto $\hat{\mathbf{P}}_{k+1}$ quanto $\mathbf{G}_k$ irão se estabilizar rapidamente.

4.6 Análise da Complexidade Computacional

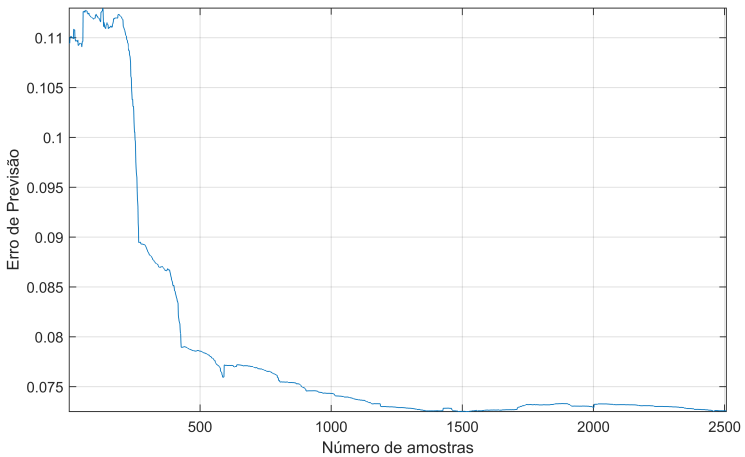
A complexidade assintótica do algoritmo KLM é analisada nessa seção em duas partes: (i) fase de inicialização e (ii) fase de aprendizado. Vamos assumir como premissas que a computação matricial de elementos individuais tem complexidade $O(1)$ e a multiplicação de



(a) Erro de Previsão para $Q_k = 1$

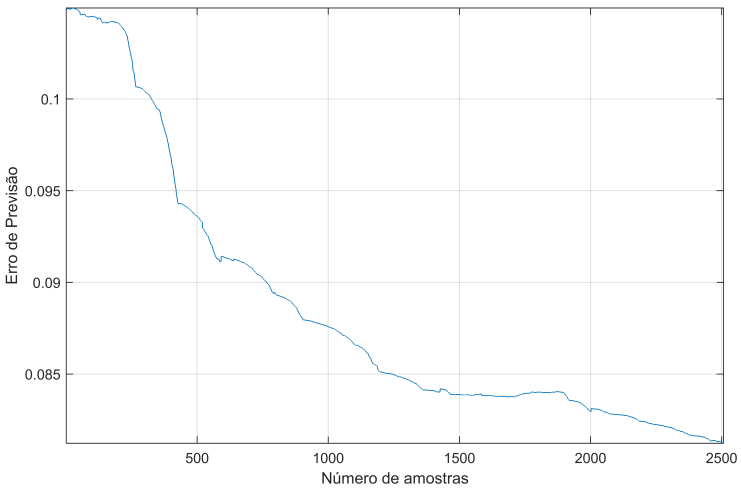


(b) Erro de Previsão para $Q_k = 0,0001$

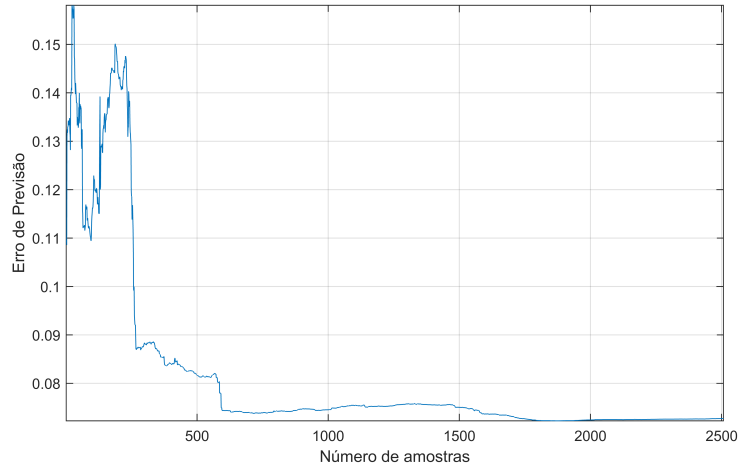


(c) Erro de Previsão para $Q_k = 10^{-16}$

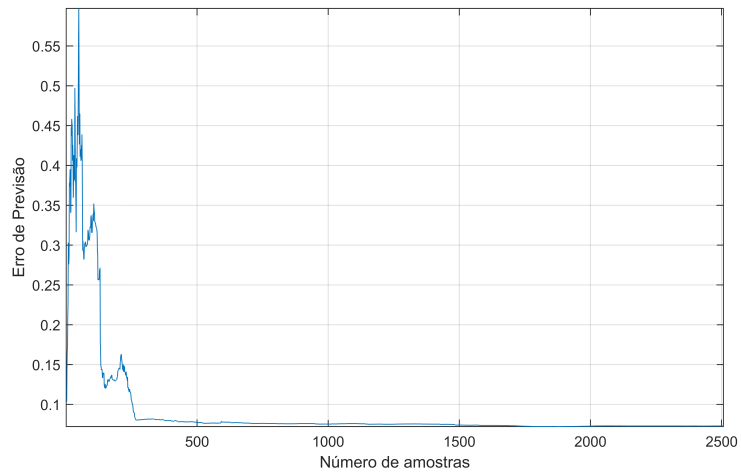
Figura 4.1: Ajuste na variância do processo



(a) Erro de Previsão para $R_k = 10$



(b) Erro de Previsão para $R_k = 0,01$



(c) Erro de Previsão para $R_k = 10^{-4}$

Figura 4.2: Ajuste na variância da medição

uma matriz $m \times n$ por outra matriz $n \times p$ tem complexidade $O(mnp)$. Assumimos também que a inversão de uma matriz $n \times n$ tem complexidade $O\left(n^3\right)$. Pela formulação do problema, temos $\mathbf{P} \in \mathbb{R}^{L\times L}$, $\mathbf{A} \in \mathbb{R}^{L\times L}$, $\mathbf{H} \in \mathbb{R}^{N\times L}$, $\mathbf{T} \in \mathbb{R}^{N\times 1}$, $\boldsymbol{\beta} \in \mathbb{R}^{L\times 1}$, $\mathbf{G} \in \mathbb{R}^{L\times N}$, $\mathbf{C} \in \mathbb{R}^{L\times L}$ e $\mathbf{D} \in \mathbb{R}^{L\times L}$, em que L é o número de unidades da camada escondida e N é o tamanho do bloco de instâncias de treinamento.

4.6.1 Fase de Inicialização

No Passo 8 do algoritmo, a complexidade computacional de duas multiplicações e uma inversão de matriz é $O\left(L^2N\right)$ e $O\left(L^3\right)$, respectivamente. Para calcular $\boldsymbol{\beta}_k$ no Passo 9, temos complexidade $O\left(L^2N + LN\right)$. O cálculo da matriz de transição $\mathbf{A}$ no passo 12 tem complexidade $O\left(L^3\right)$.

4.6.2 Fase de Aprendizagem Sequencial

Na segunda parte, a fase de aprendizagem sequencial inicia quando a instância $(k+1)$ alimenta o algoritmo. No passo 19, o cálculo de $\hat{\mathbf{T}}_{k+1}$ tem complexidade $O(LN)$. A previsão da covariância do estado $\hat{\mathbf{P}}_{k+1}$ no Passo 20 tem complexidade $O\left(L^3\right)$. O cálculo da matriz de ganhos $\mathbf{G}_{k+1}$ no Passo 21 possui um custo computacional de $O\left(L^3 + L^2N + LN^2\right)$. A atualização do estado $\boldsymbol{\beta}_{k+1}$ no Passo 22 tem complexidade $O(LN)$. Por fim, a atualização da matriz de covariância $\mathbf{P}_{k+1}$ tem complexidade $O\left(L^2N + LN^2\right)$.

A complexidade total do algoritmo KLM, incluindo a fase de inicialização, é definida como $O\left(L^3 + L^2N + LN^2 + LN\right)$. Para efeito de comparação, a complexidade assintótica do KLM é exatamente a mesma da OS-ELM original e das extensões ReOS-ELM e LS-IELM. Intuitivamente, podemos perceber também que a complexidade computacional do EKLM é maior que a do KLM. A explicação desse fato decorre da necessidade de se calcular a cada iteração as matrizes Jacobianas $\mathbf{F}_x$ e $\mathbf{H}_x$, o que implica em calcular as derivadas parciais de primeira ordem para cada elemento dessas matrizes. Levando em consideração que a complexidade computacional para o cálculo de uma derivada é função de um número T de operações aritméticas, as quais dependem da função não linear em análise, podemos concluir que o desempenho do EKLM em termos do custo computacional é sempre inferior ao do KLM. De fato, como mostraremos no Capítulo 5, o EKLM apresenta o maior tempo de treinamento entre todos os algoritmos estudados.

4.7 Limitações do Método

A aplicação do método proposto para problemas de aprendizagem sequencial apresenta algumas limitações derivadas dos algoritmos utilizados em sua formulação. Separamos as limitações do método em duas partes: (i) limitações derivadas da ELM e (ii) limitações relacionadas com a aplicação do filtro de Kalman no cálculo dos pesos de saída. Vale ressaltar que as restrições

relacionadas à ELM se aplicam a todos os métodos descritos na Subseção 2.4.2 e que estão relacionados ao objeto dessa tese.

4.7.1 Limitações Relacionadas com a ELM

1. **Comportamento Estocástico.** A aleatoriedade da ELM gera um problema adicional de incerteza no uso do KLM e EKLM. A incerteza ocorre durante a geração aleatória dos pesos da camada escondida. Como consequência, é difícil julgar se uma simples execução dos algoritmos consegue obter o menor erro de previsão para um dado problema. Na fase de treinamento, é fundamental executar os algoritmos de forma repetida e observar a variância do erro de previsão para avaliar a sua capacidade de generalização.

2. **Degradação na Generalização.** O método proposto depende da correta escolha da função de ativação e do número de unidades da camada escondida. Recentemente, foi mostrado em LIU et al. (2015) que existe um fenômeno de degradação na generalização da ELM quando alguns tipos de funções de ativação baseados em *kernels* Gaussianos são utilizados em problemas genéricos de aprendizagem. A degradação é medida em termos do aumento da variância do erro de previsão. No artigo citado acima, é mostrado que a capacidade de generalização da ELM depende do número de unidades da camada escondida e o seu valor deverá ser otimizado durante o treinamento.

3. **Maldição da Dimensionalidade.** A fase de inicialização do método proposto poderá sofrer com o problema de dimensionalidade para bases de dados com um número muito grande de instâncias de treinamento e atributos. Nesse caso, recomenda-se começar o aprendizado sequencial a partir da primeira instância, sem aplicar a fase de inicialização. Outra alternativa para uso do método é a redução do número de atributos através de abordagens de seleção de características, como por exemplo as técnicas baseadas em análise dos componentes principais ou projeções aleatórias (MICHE; SCHRAUWEN; LENDASSE, 2010).

4.7.2 Limitações Relacionadas com o Filtro de Kalman

1. **Premissa de Linearidade**. A formulação do problema na Seção 4.2 tem como premissa a linearidade do sistema de estados baseado na ELM. Para sistemas lineares dinâmicos, o filtro de Kalman pode ser considerado como um estimador ótimo (HARVEY, 1990). Mesmo em problemas nos quais o sistema apenas se aproxima de uma solução linear, o filtro é capaz de produzir bons resultados. Entretanto, para alguns problemas nos quais o sistema não se comporta de forma linear é necessária a

avaliação das duas versões do método proposto, KLM e EKLM, pois a versão não linear poderá obter resultados melhores, conforme discutiremos no Capítulo 5.

2. **Premissa de Normalidade dos Ruídos.** Na formulação do problema, as matrizes de covariância $\mathbf{Q}_k$ e $\mathbf{R}_k$ são calculadas diretamente a partir de uma pequena amostra do conjunto de treinamento, mantendo-se constantes durante a fase de aprendizagem sequencial. É assumida a normalidade dos ruídos na previsão dos estados e das medições. Entretanto, é comum o caso no qual o erro de medição não se mantém constante no tempo, com a consequente mudança em sua distribuição. De forma semelhante, o ruído do processo de estados também poderá se alterar de acordo com a dinâmica do sistema. Nesses casos, pode ser necessário utilizar outra abordagem de aprendizado sequencial (por exemplo, o OS-ELM-TV) que atualize dinamicamente as covariâncias com a passagem do tempo, implicando no aumento do tempo de treinamento.

3. **Transformação Linear.** Para os casos nos quais o EKLM é utilizado, a transformação linear produz bons resultados apenas quando o erro de propagação pode ser aproximado por uma função linear, sendo essa uma restrição da versão estendida do filtro de Kalman. Em casos nos quais a aproximação não é possível, a convergência do erro na estimativa dos estados não é garantida pelo algoritmo.

4.8 Comparação com os Trabalhos Relacionados

Nessa Seção, vamos discutir as características do método proposto, comparando-o com os trabalhos relacionados da Subseção 2.4.2. Um resumo das características do método e dos trabalhos relacionados pode ser visto na Tabela 4.1.

1. **Tratamento dos Efeitos da Multicolinearidade.** O método proposto consegue tratar o efeito da multicolinearidade na matriz $\mathbf{H}$ através da filtragem do ruído de medição e da estimativa dos estados. Essa abordagem é mais eficiente para reduzir a variância nas estimativas do que o método de mínimos quadrados recursivo, adotado pela OS-ELM e suas extensões. Uma vantagem do método proposto é não depender diretamente da escolha de uma constante de regularização para tratar o problema da multicolinearidade, como aplicado nos algoritmos ReOS-ELM e LS-IELM.

2. **Dependência da Distribuição do Conjunto de Treinamento.** Apesar da utilização de uma pequena amostra do conjunto de treinamento na fase de inicialização, o método proposto não depende da distribuição dos dados de treinamento para decidir a melhor estratégia de ajuste dos pesos na fase sequencial, como é aplicado ao caso do ReOS-ELM. Ainda assim, o método proposto nessa tese permite que a fase sequencial possa ser executada sem a totalidade da fase de inicialização, logo a partir da primeira

instância de treinamento. Além disso, não é necessário conhecer a distribuição dos dados de treinamento na fase de aprendizagem sequencial para ajustar os pesos de saída, como é feito no caso do TOSELM.

3. **Otimização Adicional de Parâmetros.** O método proposto utiliza apenas um parâmetro para o processo de aprendizagem sequencial: o número de unidades da camada escondida L. O tamanho da amostra inicial do conjunto de treinamento N_0, usado na fase de inicialização, também poderá ser definido no início do treinamento. Apenas o parâmetro L precisa ser otimizado durante o treinamento, sendo opcional a otimização de N_0. De fato, para encontrar diretamente o valor de N_0 podemos aplicar a restrição descrita em LIANG et al. (2006), com $N_0 \geq L$. Ainda assim, se não for aplicada a fase de inicialização, com o treinamento iniciando a partir da primeira instância, não é necessário estabelecer um valor a priori para N_0.

4. **Custo Computacional.** A complexidade computacional do KLM é a mesma da OS-ELM, ReOS-ELM e LS-IELM, mas teoricamente é menor que o OS-ELM-TV e TOSELM. Esses dois últimos métodos utilizam um segundo procedimento iterativo dentro do ciclo de atualização dos pesos, aumentando a complexidade computacional. É importante observar que o custo computacional do KLM pode ser reduzido em relação aos métodos relacionados através da definição das matrizes de covariância $\mathbf{Q}_k$ e $\mathbf{R}_k$ como simples escalares. O cálculo da discretização matricial pode ser feito com escalares que definem o comportamento do sistema linear em estudo, reduzindo a quantidade de multiplicações e inversões matriciais no calculo da matriz de ganhos $\mathbf{G}_{k+1}$ e no ajuste da covariância do processo $\mathbf{P}_{k+1}$.

Tabela 4.1: Algoritmos Sequenciais - Características do KLM e EKLM

Algoritmo	Atualização dos Pesos	Parâmetros	Limitações
OS-ELM	Mínimos quadrados recursivos	L, N_0	Multicolinearidade
ReOS-ELM	Regularização	L, N_0, λ	Atualização dependente da distribuição
OS-ELM-TV	Combinação linear com n funções ortogonais	L, N_0, f, B	Otimização adicional de parâmetros
TOSELM	Fator de correção dos pesos	L, N_0, ε, j	Custo computacional
LS-IELM	Regularização	L, N_0, λ	Otimização adicional de parâmetros
KLM	Filtragem do ruído	L, N_0 (opcional)	Linearidade e normalidade
EKLM	Filtragem do ruído	L, N_0 (opcional)	Transformação linear

4.9 Considerações Finais

Neste capítulo foram introduzidas as duas versões do método proposto. Através do mapeamento da dinâmica da ELM em um modelo de espaço de estados, a versão KLM do método proposto permite o aprendizado sequencial para problemas genéricos de regressão com complexidade computacional similar aos métodos relacionados. Para problemas nos quais a

dinâmica do sistema seja não linear, a extensão EKLM permite a aprendizagem sequencial através de uma transformação linear. Foram discutidas as estratégias de estimativas de parâmetros para o método proposto, com foco em uma solução de baixo custo computacional.

5

Experimentos

Nesse Capítulo são apresentados os resultados experimentais da aplicação dos algoritmos do método proposto, mostrados no Capítulo anterior, em diferentes bases de dados obtidas a partir do repositório UCI (BACHE; LICHMAN, 2013) e geradas artificialmente. Os trabalhos relacionados, os quais descrevem os métodos utilizados na comparação com o método proposto, foram descritos na Seção 2.4.2. Os resultados obtidos a partir dos experimentos são apresentados e analisados com base em métodos estatísticos.

Foram utilizadas quinze bases de dados conhecidas da comunidade de aprendizagem de máquina. Um estudo de caso também é apresentado nesse Capítulo, voltado para o problema de previsão de séries temporais do mercado financeiro. Nesse estudo, os resultados da aplicação do método proposto e dos trabalhos relacionados são analisados e validados estatisticamente.

5.1 Metodologia Adotada

Os algoritmos propostos nessa tese e os métodos relacionados foram avaliados em bases de dados obtidas do repositório UCI (BACHE; LICHMAN, 2013) ou geradas artificialmente. Para a escolha das bases foram utilizados como critérios a sua aplicação em problemas de regressão e de previsão para séries temporais. O número de atributos das bases variam de 2 até 281. No total, foram selecionadas ou geradas quinze bases de dados. Após a seleção, todos os atributos foram normalizados no intervalo $[0,1]$, com exceção da base SinC, na qual os atributos foram normalizados no intervalo $[-1,1]$ com o objetivo de manter o mesmo padrão de normalização mostrado em HUANG; ZHU; SIEW (2006). Na inicialização de cada método avaliado, incluindo os trabalhos relacionados, os pesos de entrada foram gerados de forma aleatória e normalizados no intervalo $[-1,1]$, conforme recomendado em HUANG; CHEN; SIEW (2006). Para os experimentos reportados nessa tese, foram escolhidas as funções de ativação para unidades do tipo RBF e sigmoide. Para os problemas de regressão, foi utilizado o método *holdout* repetido, o qual reserva uma porção do conjunto original para fins de treinamento, validação e teste (WITTEN; FRANK, 2005), com os elementos das partições escolhidos aleatoriamente a cada iteração. Foram geradas 50 variações distintas de cada base de dados, para cada função de

ativação, com as instâncias dispostas de forma aleatória. Para os problemas de séries temporais, foi utilizado o método de *holdout* simples (WITTEN; FRANK, 2005), o qual mantém a ordem das observações com o objetivo de preservar a informação gerada pelas correlações entre as observações. As séries artificiais, todas voltadas para problemas de previsão de séries temporais, foram geradas usando a biblioteca *Chaotic Time Series*[1] do Matlab, com 5000 instâncias cada. Foram definidos 20 atributos para cada série, gerados a partir das observações anteriores do atributo alvo e de suas médias móveis. Para cada variação das bases de dados normalizadas, foram criadas partições para os conjuntos de treinamento, validação e teste, na proporção de 50%, 25% e 25%, respectivamente. Para as séries temporais, foram criadas três partições sem variação na mesma proporção usada nos problemas de regressão, com a manutenção da ordem temporal das observações.

A melhor configuração de parâmetros para cada base de dados foi encontrada a partir do conjunto de validação. O número L de unidades da camada escondida foi otimizado seguindo a abordagem descrita na Subseção 5.1.2 e todos os algoritmos avaliados utilizam o mesmo valor para cada base. Para o valor de N_0, foi fixado o limite de 10% do conjunto de treinamento, ou o valor que tornasse $N_0 \geq L$. Os algoritmos TOSELM, LS-IELM e ReOS-ELM possuem parâmetros específicos (ε, υ e λ), os quais foram otimizados para cada método usando o procedimento descrito na Subseção 5.1.2.

Para cada uma das 50 variações de cada base de dados de regressão, os algoritmos do método proposto e os dos trabalhos relacionados foram executados, com os parâmetros obtidos a partir do procedimento de otimização. Para os problemas de séries temporais, os algoritmos foram executados 50 vezes nas mesmas partições. Os resultados de cada execução foram armazenados para a posterior comparação entre os métodos. Foram medidos o erro de previsão no conjunto completo de treinamento (incluindo o de validação) e no conjunto de teste. Foi medido também o tempo total de treinamento para cada execução. No total foram realizadas $50 \times 2 \times 15 = 1500$ execuções em uma máquina com o sistema operacional Windows 8.1, processador Intel Core i5, com 3,10 GHz e 8 GB de memória. Os algoritmos foram implementados usando a versão R2014b do Matlab e os códigos-fonte do KLM[2] e EKLM[3] estão disponíveis para *download*.

Ao final das execuções, as medidas de desempenho foram comparadas estatisticamente usando o teste não paramétrico de Friedman (FRIEDMAN, 1940), com nível de confiança $\alpha = 0,05$, segundo a abordagem proposta por DEMŠAR (2006). A escolha do teste levou em consideração a capacidade de se comparar simultaneamente todos os métodos para cada base de dados e o estabelecimento de um *ranking* de desempenho. Após o cálculo do *ranking*, um segundo teste *post-hoc* foi realizado para comparar par a par as diferenças de desempenho entre todos os métodos. Todos os algoritmos foram comparados uns com outros através da aplicação do teste de Tukey (TUKEY, 1949) com nível de confiança $\alpha = 0,05$. A descrição detalhada do

[1] `http://www.mathworks.com/matlabcentral/fileexchange/1597-chaotic-systems-toolbox`

[2] `https://dl.dropboxusercontent.com/u/1634663/matlab/klm/KLM.m`

[3] `https://dl.dropboxusercontent.com/u/1634663/matlab/eklm/EKLM.m`

teste de Friedman e dos testes *post-hoc* é apresentada no Apêndice B.

5.1.1 Bases de Dados

A Tabela 5.1 mostra o número de instâncias de treinamento, validação e teste, além da quantidade de atributos. Abaixo, apresentamos uma descrição resumida de cada base utilizada nos experimentos.

Tabela 5.1: Bases de dados e número de atributos

Base de Dados	Tipo	Tam. Total	Treinamento	Validação	Teste	Atributos
Abalone	Regressão	4177	2089	1044	1044	8
Auto MPG	Regressão	392	196	98	98	8
Bike	Regressão	17379	8989	4345	4345	16
Blog	Regressão	60021	30011	15005	15005	281
Communities	Regressão	2215	1107	554	554	128
Cycle	Regressão	9568	4784	2392	2392	4
Energy	Regressão	768	384	192	192	8
Housing	Regressão	506	252	127	127	14
Istanbul	Regressão	536	268	134	134	8
Logistic	Série Temporal	5000	2500	1250	1250	20
Lorentz	Série Temporal	5000	2500	1250	1250	22
Mackey-Glass	Série Temporal	5000	2500	1250	1250	20
Rossler	Série Temporal	5000	2500	1250	1250	22
Sinc	Série Temporal	10000	5000	2500	2500	2
Wine	Regressão	4898	2448	1225	1225	12

Abalone - Essa base de dados é utilizada para prever a idade de um molusco (haliote) a partir da quantidade de anéis encontrados em seu corpo. Os atributos que descrevem a idade do molusco são o tamanho, diâmetro, altura, peso total, peso das vísceras, peso da concha e peso sem a concha. Essa base foi utilizada originalmente no trabalho de UYSAL; GÜVENIR (2004) e recentemente em NOBREGA; OLIVEIRA (2015). A base está disponível para *download* no repositório UCI (BACHE; LICHMAN, 2013) [4].

Auto MPG - Os dados dessa base são utilizados para a previsão do consumo de combustível de um veículo, medido em milhas por galão. Essa é uma versão modificada da base encontrada no repositório Statlib [5], contendo os atributos: número de cilindros, deslocamento, potência do motor, peso, aceleração, ano do modelo e origem. A base foi utilizada inicialmente em QUINLAN (1993) e recentemente em NOBREGA; OLIVEIRA (2015). A base está disponível para *download* no repositório UCI (BACHE; LICHMAN, 2013) [6].

[4] https://archive.ics.uci.edu/ml/datasets/Abalone
[5] http://lib.stat.cmu.edu/datasets/
[6] https://archive.ics.uci.edu/ml/datasets/Auto+MPG

Bike - Esse conjunto de dados é utilizado na previsão da quantidade de aluguéis de bicicletas durante o período de uma hora. Os principais atributos da base incluem informações da temperatura, estação do ano, hora atual, velocidade do vento, umidade, dia da semana, entre outros. A base foi utilizada no trabalho de FANAEE-T; GAMA (2014) e está disponível para *download* no repositório UCI (BACHE; LICHMAN, 2013) [7].

Blog - Os dados têm a sua origem na postagem de mensagens em vários blogs. O objetivo é prever o número de postagem de comentários para as próximas 24 horas. A base foi extraída da vários blogs nos anos de 2010 e 2011 e possui um número grande de atributos, incluindo o número de comentários de posts anteriores, os intervalos de tempo entre as postagens, os tamanhos das postagens originais, quantidade de palavras, dia da semana da postagem, entre outros. A base foi utilizada inicialmente em BUZA (2014) e está disponível para *download* no repositório UCI (BACHE; LICHMAN, 2013) [8].

Communities - Essa base de dados é utilizada para a previsão do número de crimes por habitante a partir de atributos extraídos de comunidades em várias cidades americanas. Os atributos incluem o perfil racial das comunidades, taxas de desemprego, nível de escolaridade, tamanho das famílias, tamanho das residências, valor médio das residências, densidade populacional, entre outros. Uma versão modificada dessa base foi utilizada no trabalho de REDMOND; BAVEJA (2002). A base original está disponível para *download* no repositório UCI (BACHE; LICHMAN, 2013) [9].

Cycle - Os dados dessa base foram coletados a partir dos sensores espalhados em várias turbinas de uma fábrica durante os anos de 2006 a 2011, sendo utilizada para a previsão da geração de energia pelas turbinas durante o período de uma hora. Os seus atributos são a temperatura e pressão do ambiente, umidade relativa e o vácuo de exaustão. A base foi utilizada recentemente em TÜFEKCI (2014) e está disponível para *download* no repositório UCI (BACHE; LICHMAN, 2013) [10].

Energy - Essa base de dados é utilizada na análise do consumo de energia para edifícios em construção. Os dados foram coletados de doze diferentes formatos de plantas arquitetônicas e levam em consideração atributos como a área da superfície dos prédios, área do teto, orientação, altura dos edifícios, área coberta por vidros, entre outros atributos. A base foi utilizada inicialmente em TSANAS; XIFARA (2012) e está disponível para *download* no repositório UCI (BACHE; LICHMAN, 2013) [11].

[7] https://archive.ics.uci.edu/ml/datasets/Bike+Sharing+Dataset
[8] https://archive.ics.uci.edu/ml/datasets/BlogFeedback
[9] https://archive.ics.uci.edu/ml/datasets/https://archive.ics.uci.edu/ml/datasets/Communities+and+Crime
[10] https://archive.ics.uci.edu/ml/datasets/Combined+Cycle+Power+Plant
[11] https://archive.ics.uci.edu/ml/datasets/Energy+efficiency

Housing - Essa base é utilizada na previsão dos preços de residências da cidade de Boston. Seus principais atributos são a taxa de crimes por habitante da vizinhança, número de quartos, distância do centro da cidade, distância do acesso às rodovias, entre outros. A base é apresentada em detalhes no livro BELSLEY; KUH; WELSCH (2005) e está disponível para *download* no repositório UCI (BACHE; LICHMAN, 2013) [12].

Istanbul - Essa base é utilizada na previsão do retorno financeiro do índice da bolsa de valores de Istanbul. Seus principais atributos são os índices de outras bolsas globais, como as da Alemanha, Reino Unido, Japão, Brasil e outros índices econômicos. A base foi utilizada em AKBILGIC; BOZDOGAN; BALABAN (2014) e está disponível para *download* no repositório UCI (BACHE; LICHMAN, 2013) [13].

Logistic - A série temporal do mapa logístico é um mapeamento polinomial de grau dois com comportamento caótico, derivado de uma simples equação não linear. A definição matemática do mapa logístico pode ser vista na equação a seguir:

$$x_{n+1} = r x_n (1 - x_n) \tag{5.1}$$

em que x_n é um número entre zero e um, representando a população no tempo n e r representa a taxa de crescimento da população. Maiores detalhes sobre a definição e utilização do mapa logístico podem ser vistos em MAY et al. (1976). Para os experimentos realizados nessa tese, os parâmetros escolhidos foram $r = 4$ e $x_0 = 0,1$, com 5000 instâncias de treinamento. Foram definidos 20 atributos, gerados a partir das observações anteriores do atributo alvo x e de suas médias móveis. A Figura 5.1 mostra um exemplo da série com um número reduzido de instâncias.

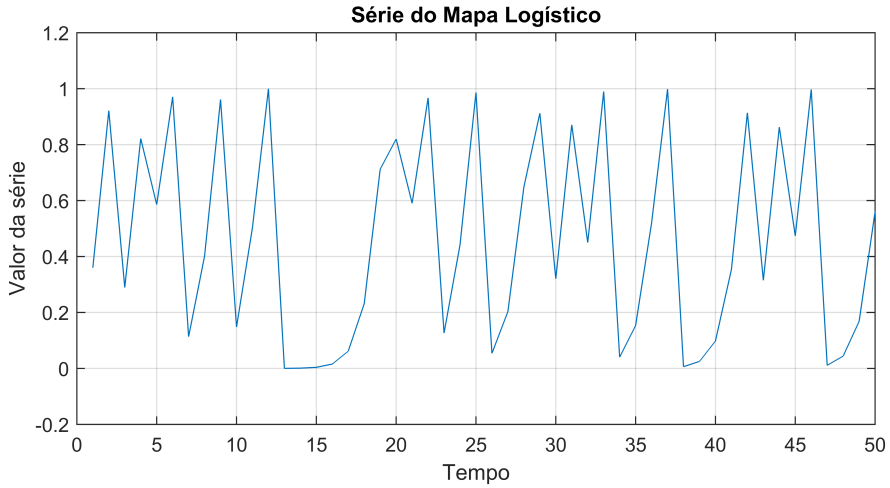


Figura 5.1: Série Temporal do Mapa Logístico ($r = 4$ e $x_0 = 0,1$).

[12] https://archive.ics.uci.edu/ml/datasets/Housing
[13] https://archive.ics.uci.edu/ml/datasets/ISTANBUL+STOCK+EXCHANGE

Lorenz - A série temporal de Lorenz é formada por um sistema de equações diferenciais, apresentando comportamento caótico para um conjunto específico de parâmetros. Em 1963, Edward Lorenz desenvolveu um modelo matemático composto por três equações diferenciais ordinárias, conhecidas como equações de Lorenz (LORENZ, 1963):

$$\begin{aligned} \frac{dx}{dt} &= \sigma\,(y-x)\,, \\ \frac{dy}{dt} &= x\,(\rho-z)-y, \\ \frac{dz}{dt} &= xy-\beta z, \end{aligned} \tag{5.2}$$

em que x, y e z representam o estado do sistema no tempo t, σ, ρ e β são valores positivos que definem os parâmetros do sistema. Para os experimentos realizados nessa tese, os parâmetros escolhidos foram $\sigma = 16$, $\rho = 45,92$ e $\beta = 4$, com 5000 instâncias de treinamento. Foram definidos 20 atributos, gerados a partir das observações anteriores do atributo alvo x e de suas médias móveis. Os valores de y e z foram incluídos como atributos no conjunto de treinamento. A Figura 5.2 mostra um exemplo da série com um número reduzido de instâncias.

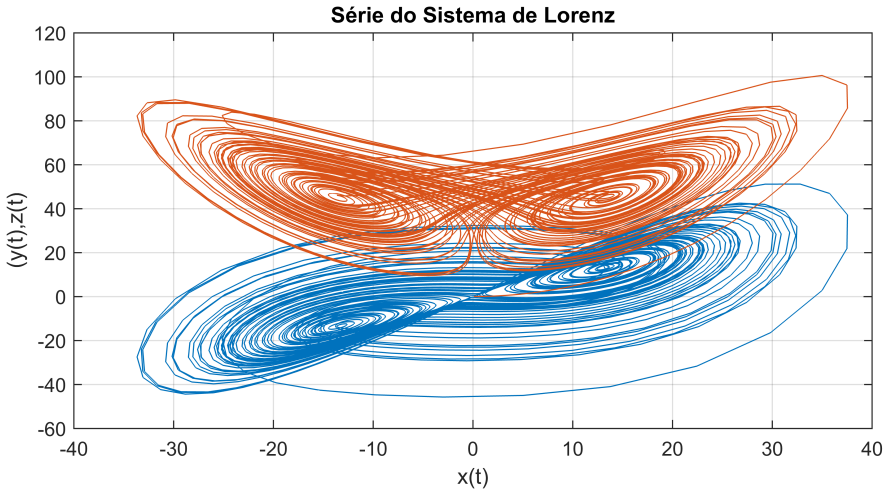
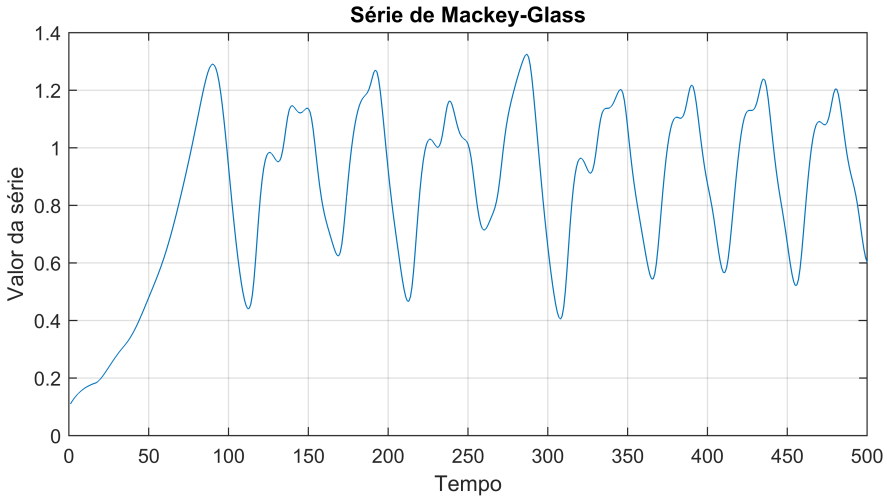


Figura 5.2: Série temporal do sistema de Lorenz ($\sigma = 16$, $\rho = 45,92$ e $\beta = 4$).

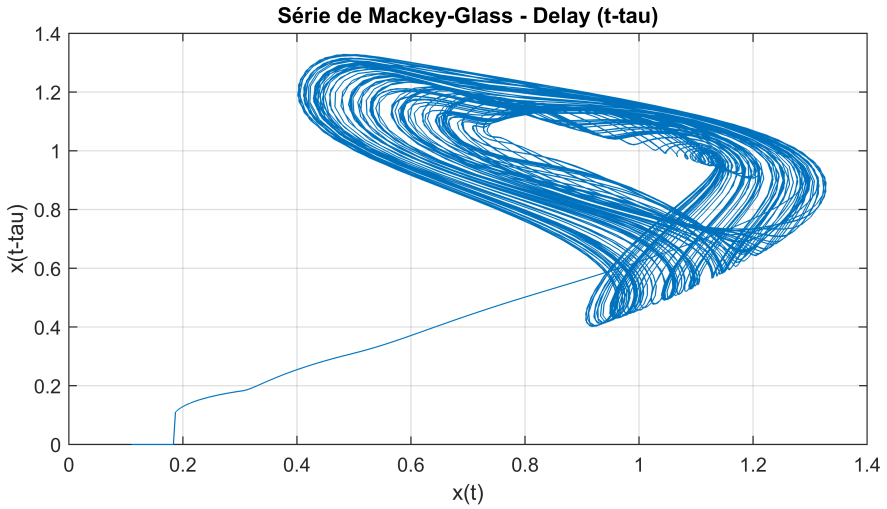
Mackey-Glass - A série temporal de Mackey-Glass é definida por uma equação diferencial, de acordo com a expressão (MACKEY; GLASS et al., 1977):

$$\frac{dx}{dt} = \beta \frac{x_\tau}{1+x_\tau^n} - \gamma x \tag{5.3}$$

em que β, γ, τ e n são números reais positivos e x_τ representa o valor de x no tempo $(t-\tau)$. Para os experimentos realizados nessa tese, os parâmetros escolhidos foram $\beta = 0,2$, $n = 10$, $\gamma = 0,1$, $\tau = 17$ e $x_0 = 0,1$, com 5000 instâncias de treinamento. Foram definidos 20 atributos, gerados a partir das observações anteriores do atributo alvo x e de suas médias móveis. A Figura 5.3 mostra um exemplo da série com um número reduzido de instâncias, apresentando a série sem atraso e a dinâmica caótica da série com atraso de $(t-\tau)$.



(a) Série de Mackey-Glass sem atraso



(b) Dinâmica da série de Mackey-Glass com atraso $(t-\tau)$

Figura 5.3: Série temporal de Mackey-Glass ($\beta = 0,2$, $n = 10$, $\gamma = 0,1$, $\tau = 17$ e $x_0 = 0,1$).

Rossler - A série temporal de Rossler é composta por três equações diferenciais ordinárias que definem um sistema dinâmico contínuo, com comportamento caótico. A série foi estudada inicialmente por Otto Rössler em 1976, tendo as suas equações definidas de acordo com as seguintes expressões (RÖSSLER, 1976):

$$\begin{aligned} \frac{dx}{dt} &= -y - z, \\ \frac{dy}{dt} &= x + ay, \\ \frac{dz}{dt} &= b + z\,(x - c)\,, \end{aligned} \tag{5.4}$$

em que x, y e z representam o estado do sistema no tempo t, a, b e c são os parâmetros do sistema. Para os experimentos realizados nessa tese, os parâmetros escolhidos foram $a = 0,2$, $b = 0,4$ e $c = 5,7$, com 5000 instâncias de treinamento. Foram definidos 20 atributos, gerados a partir das observações anteriores do atributo alvo x e de suas médias móveis. Os valores de y e z foram

incluídos como atributos no conjunto de treinamento. A Figura 5.4 mostra um exemplo da série com um número reduzido de instâncias.

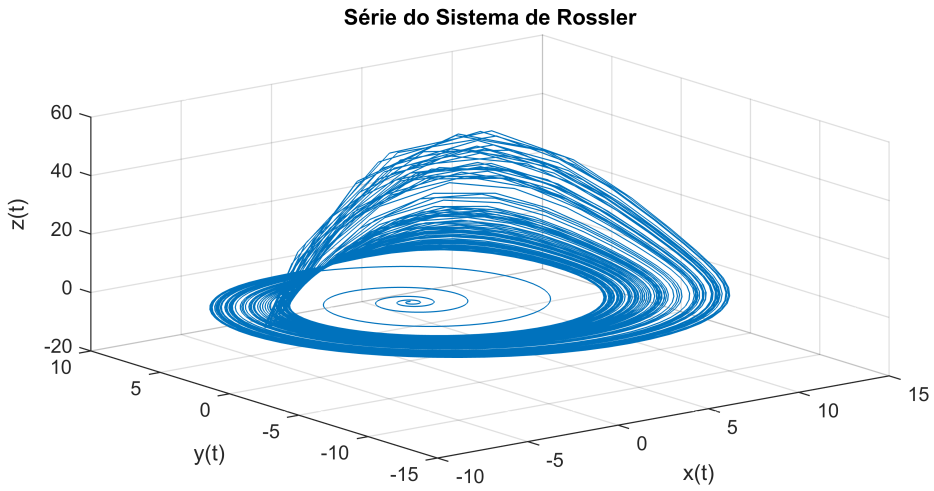


Figura 5.4: Série temporal do sistema de Rossler ($a = 0,2$, $b = 0,4$ e $c = 5,7$).

SinC - A base de dados SinC é utilizada como exemplo na aproximação de funções no artigo original da ELM (HUANG; ZHU; SIEW, 2006) e a sua geração ocorre de acordo com a seguinte equação:

$$y(x) = \begin{cases} \frac{\sin(x)}{x}, & x \neq 0, \\ 1, & x = 0. \end{cases} \tag{5.5}$$

Para os experimentos reportados nessa tese, foram geradas 10000 instâncias de y, com x_i uniformemente distribuído no intervalo $[-10; 10]$. Foi acrescentado à base um ruído uniformemente distribuído no intervalo $[-0,2; 0,2]$ para todas as instâncias do conjunto de treinamento gerado, enquanto o conjunto de testes ficou livre do ruído. A base foi utilizada em HUANG; ZHU; SIEW (2006) nos experimentos da versão *batch* da ELM. A Figura 5.5 mostra um exemplo da série com ruído no conjunto de treinamento.

Wine Quality - Essa base de dados está relacionada com a avaliação da qualidade do vinho português conhecido como "Vinho Verde". Seus principais atributos incluem os níveis de acidez, açúcar residual, níveis de dióxido de enxofre, densidade, pH, teor alcoólico, entre outros. A base foi utilizada em CORTEZ et al. (2009) e está disponível para *download* no repositório UCI (BACHE; LICHMAN, 2013) [14].

5.1.2 Otimização de Parâmetros

Para uma comparação justa entre os métodos avaliados, foram executados procedimentos de otimização dos parâmetros para cada algoritmo. O número N_0 de instâncias para inicializar

[14] https://archive.ics.uci.edu/ml/datasets/Wine+Quality

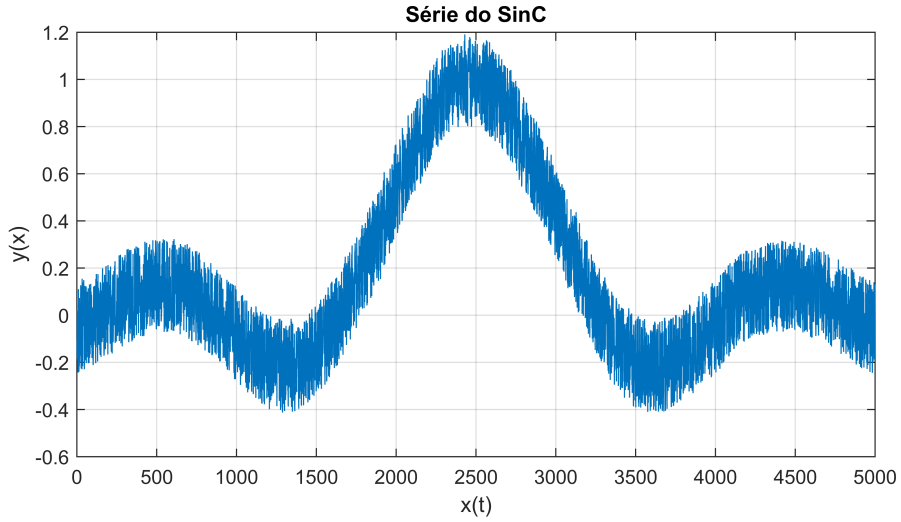


Figura 5.5: Série temporal do SinC com ruído

os algoritmos foi fixado em 10% do tamanho total do conjunto de treinamento, ou o valor que tornasse $N_0 \geq L$. Para os métodos KLM, EKLM e OS-ELM, o único parâmetro otimizado foi o número L de unidades da camada escondida. Para os algoritmos ReOS-ELM, OS-ELM-TV, TOSELM e LS-IELM, além do número de unidades, foram otimizados os parâmetros específicos de cada método:

- λ para ReOS-ELM e LS-IELM;
- ε e j para o TOSELM;
- Função base f e B para o OS-ELM-TV.

A otimização de L utilizou o método *simplex* de Nelder–Mead (NELDER; MEAD, 1965) para encontrar o menor erro de previsão medido no conjunto de validação em cada base de dados. Foi utilizada a função *fminsearch*[15] do Matlab, com a posterior conversão de L para inteiro (a função *fminsearch* só retorna valores reais). O método Nelder–Mead ou *downhill simplex* é utilizado para encontrar o valor mínimo de uma função objetivo em um espaço multidimensional. Pode ser aplicado em problemas de otimização nos quais as derivadas da função objetivo não são previamente conhecidas. No entanto, Nelder–Mead é um método de busca heurística que pode convergir para pontos não estacionários em problemas que podem ser resolvidos por métodos mais eficientes, como por exemplo, a técnica baseada em *Particle Swarm Optimization* (PSO) (KENNEDY, 2010). O método consegue a aproximação de um mínimo local para um problema com n variáveis, quando a função objetivo é unimodal e varia de forma suave na vizinhança dos pontos analisados. Um descrição detalhada do método de Nelder–Mead é apresentada no Apêndice A.

Para os experimentos realizados, a função objetivo foi definida como a média do erro de previsão, medida no conjunto de validação, para trinta execuções de cada algoritmo em uma base

[15] `http://www.mathworks.com/help/matlab/ref/fminsearch.html`

de dados. Na inicialização do algoritmo Nelder–Mead, precisamos definir um valor inicial para o parâmetro a ser otimizado. Para L, calculamos o valor inicial L_0 a partir da fórmula apresentada em LIN et al. (2014), a qual estabelece uma relação entre o número de unidades e o tamanho do conjunto de treinamento da ELM:

$$L_0 = N^{d/(2r+d)}, \tag{5.6}$$

em que N é o tamanho do conjunto de treinamento, d é o número de atributos de uma instância de treinamento e r é um número inteiro positivo com $r \geq \frac{d}{2}$. O critério de parada do algoritmo foi estabelecido a partir do limiar ν, que representa uma pequena diferença relativa entre os valores da função objetivo. O critério de parada sugerido em NELDER; MEAD (1965) é dado pelas expressões a seguir:

$$\begin{aligned} &\tfrac{1}{n+1}\sum_{i=1}^{n+1}\left(f_i - \bar{f}\right)^2 < \nu \\ &\bar{f} = \tfrac{1}{n+1}\sum_{i=1}^{n+1} f_i \end{aligned} \tag{5.7}$$

em que n é o número de iterações do algoritmo e f_i é o i-ésimo valor computado da função objetivo. O valor de $\nu = 10^{-4}$ foi escolhido de forma empírica para o critério de parada.

De forma similar, a otimização do parâmetro λ para o ReOS-ELM e LS-IELM foi realizada usando o método de Nelder–Mead, com os valores iniciais de λ obtidos a partir dos artigos que descrevem os algoritmos. Para o ReOS-ELM foi escolhido $\lambda_0 = 250$ e para o LS-IELM o valor foi $\lambda_0 = 2^{15}$. Os parâmetros ε e j para o TOSELM foram otimizados de forma semelhante, com $\varepsilon_0 = 0,001$ e $j_0 = 100$. Por fim, foram testadas combinações dos parâmetros f e B para o OS-ELM-TV, em que $f \in \{\text{Fourier}, \text{Prolate}, \text{Chebyshev}, \text{Legendre}\}$ e $1 \leq B \leq 4$, sem a necessidade de rodar o algoritmo de otimização.

A Figura 5.6 apresenta um exemplo do comportamento do método proposto KLM em relação à variação do parâmetro L na função objetivo. Usando as bases de dados Abalone, Auto MPG, Energy e Mackey-Glass como exemplos, podemos observar a relação entre o número de unidades da camada escondida e o erro de previsão. Podemos verificar visualmente que existe uma faixa de estabilidade do erro para um conjunto de valores de L. Para a base Abalone, a faixa varia de 20 a 80 unidades. Para a base Auto MPG, L varia de 24 a 43 unidades. Para a base Mackey-Glass, os valores variam de 165 até 400 unidades. Nos exemplos mostrados, apenas a base Energy apresentou um mínimo local bem definido. O número de unidades da camada escondida determina as dimensões da matriz $\mathbf{H}$, e como consequência, o esforço computacional do KLM é fortemente influenciado pela escolha correta do parâmetro L. Em problemas nos quais encontramos faixas de estabilidade para o erro de previsão em função do número de unidades da camada escondida, devemos escolher o menor valor de L que minimize o erro com o objetivo de reduzir o esforço computacional nos cálculos que envolvem a matriz $\mathbf{H}$.

Os experimentos preliminares mostraram que a escolha do parâmetro através do processo de otimização também influencia na convergência do erro de aprendizado. A Figura 5.7 mostra para as mesmas bases da figura anterior que o erro de previsão converge rapidamente para um

valor estável. No exemplo com a base Abalone, a convergência ocorre com aproximadamente 25% do número de instâncias fornecidas sequencialmente. Paras as bases Auto MPG, Energy e Mackey-Glass esses valores são da ordem de 55%, 38% e 40%, respectivamente. A evolução do erro de treinamento não é uniforme para todos as bases, conforme mostrado na Figura 5.7. Na base Abalone, o erro de convergência se estabiliza com aproximadamente 68% do valor inicial. Para as bases Auto MPG, Energy e Mackey-Glass, a proporção é de 15%, 1,25% e 92%, respectivamente. Pode-se observar que a capacidade de aprendizagem baseada na aplicação do KLM não garante um comportamento uniforme na evolução do erro de treinamento, o que implica que as características específicas de cada base determinam a evolução do erro em função do número de instâncias.

5.1.3 Métricas para Análise de Desempenho

Com o objetivo de comparar o desempenho dos métodos abordados nesse trabalho, utilizamos as métricas de desempenho para problemas de regressão baseadas na raiz do erro médio quadrado (*Root Mean Square Error* - RMSE) e no coeficiente U de Theil.

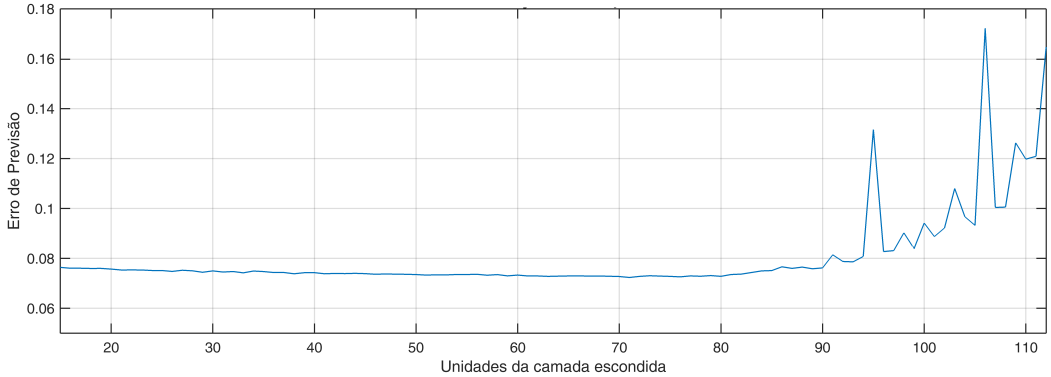
O RMSE é uma métrica frequentemente usada para medir as diferenças dos valores previstos por um dado modelo e os valores reais observados, representando o desvio padrão amostral das diferenças entre esses valores. As diferenças individuais são chamadas de residuais quando os cálculos são realizados a partir das amostras usadas nas estimativas e são chamados de erros de previsão quando são computados em um conjunto de dados não utilizados no treinamento ("*out-of-sample*"). O RMSE consegue agregar as diferentes magnitudes dos erros de previsão, calculados a partir de várias instâncias de um modelo preditivo, em uma única medida preditiva. O RMSE é uma boa medida de precisão, mas apenas para comparar erros de previsão em diferentes modelos que possuam a mesma escala nos dados. A definição do RMSE pode ser vista na equação a seguir:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \bar{y}_i)^2}, \tag{5.8}$$

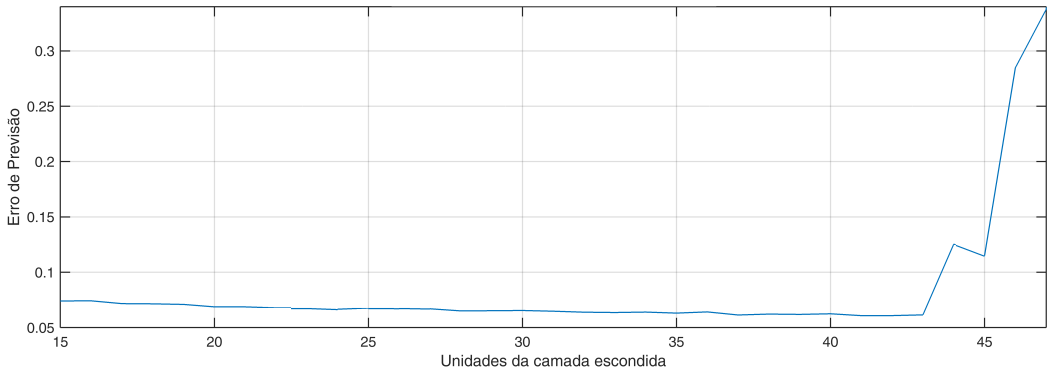
em que N é o número de amostras y_i é o i-ésimo valor observado e $\bar{y}_i$ é o i-ésimo valor previsto.

Na estatística, o coeficiente de incerteza, também chamado de coeficiente de entropia ou coeficiente U de Theil, é uma medida relativa de precisão usada na comparação dos resultados de um modelo preditivo (THEIL et al., 1971). O coeficiente se baseia no conceito de entropia da informação e compara as previsões do modelo através do cálculo de um coeficiente que representa o nível de proximidade entre as séries das previsões e dos valores observados. Para os experimentos reportados nessa tese, utilizaremos a seguinte fórmula:

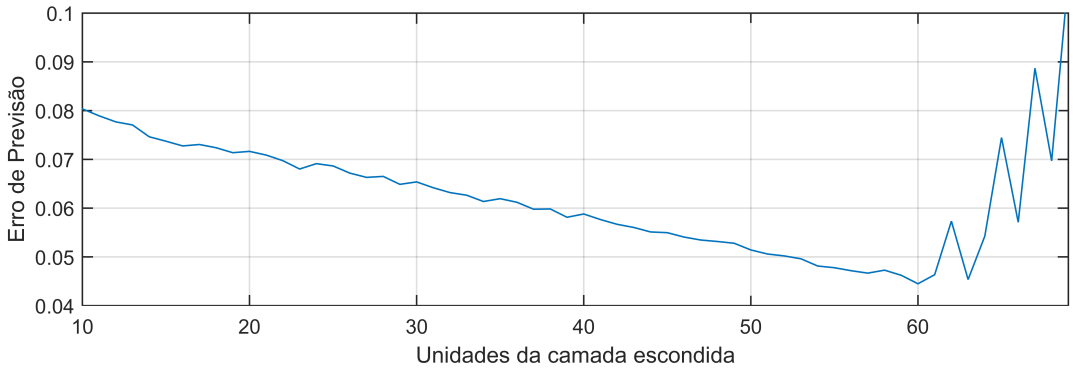
$$U = \frac{RMSE}{\sqrt{\sum_{i=1}^{N} y_i^2} + \sqrt{\sum_{i=1}^{N} \bar{y}_i^2}} \tag{5.9}$$



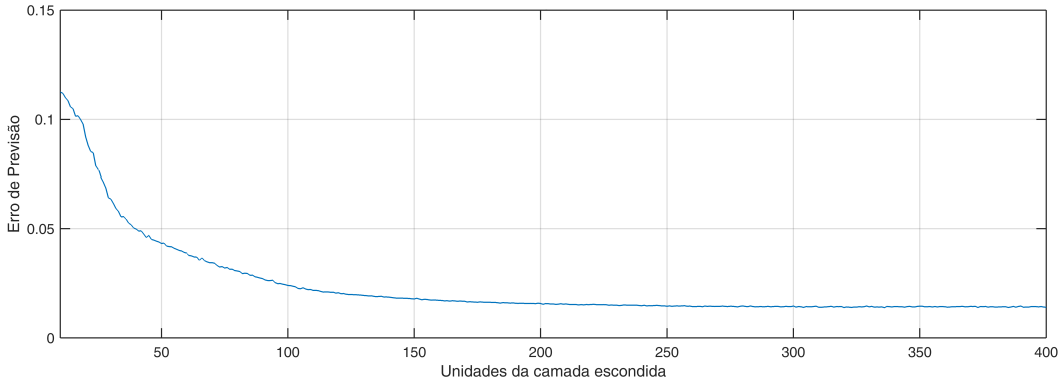
(a) Evolução do erro por unidade - Abalone



(b) Evolução do erro por unidade - Auto MPG

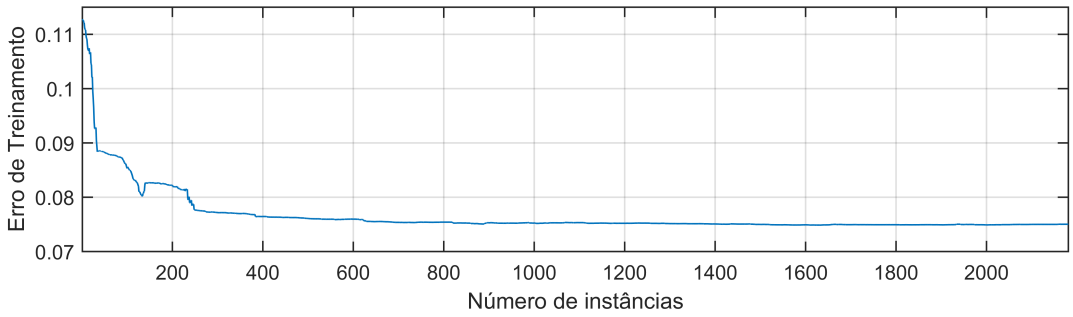


(c) Evolução do erro por unidade - Energy

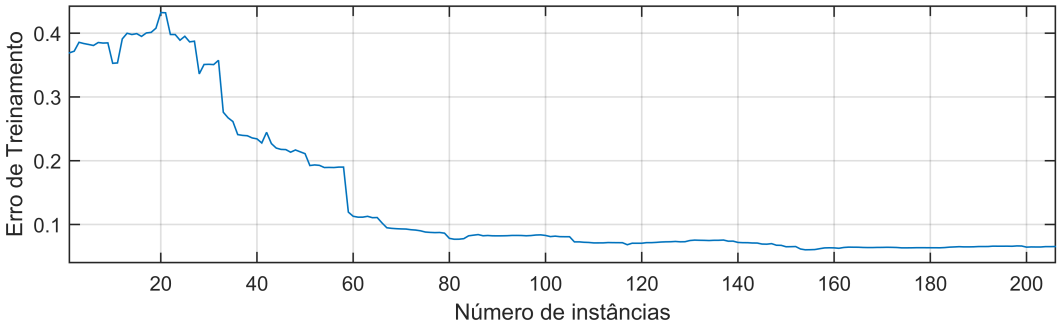


(d) Evolução do erro por unidade - Mackey-Glass

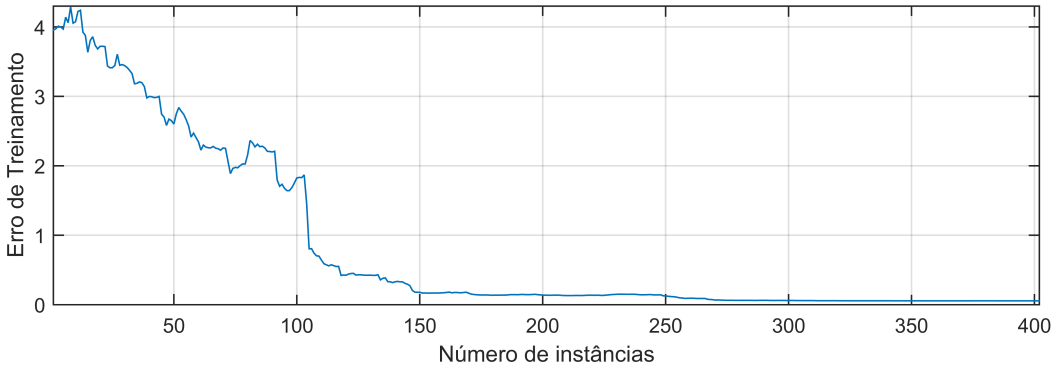
Figura 5.6: Evolução do erro por unidade para o KLM



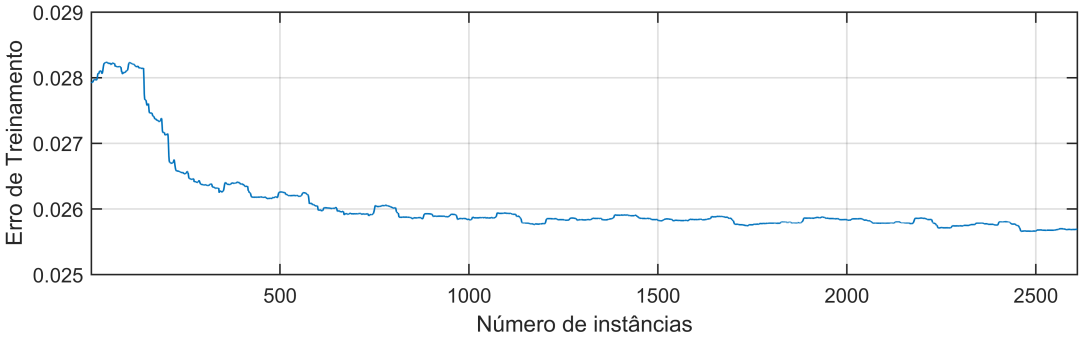
(a) Evolução do erro por número de instâncias - Abalone (L=24)



(b) Evolução do erro por número de instâncias - Auto MPG (L=25)



(c) Evolução do erro por número de instâncias - Energy (L=60)



(d) Evolução do erro por número de instâncias - Mackey-Glass (L=180)

Figura 5.7: Evolução do erro de treinamento para o KLM

O valor do coeficiente U de Theil varia entre zero e um, com valores próximos a zero indicando alta precisão no resultado das previsões.

A justificativa para o uso de duas métricas diferentes baseia-se no fato que o RMSE é dependente da escala dos dados usados no treinamento e teste do modelo, enquanto o coeficiente U de Theil é independente de escala.

5.2 Avaliação Estatística dos Resultados

Com o objetivo de mostrar a aplicação do método proposto, analisamos visualmente os valores previstos para os problemas de séries temporais e comparamos com os valores observados dessas bases de dados. A Figura 5.8 mostra os dois valores, obtidos a partir do conjunto de teste, quase sobrepostos no gráfico. O conjunto de treinamento foi gerado de forma artificial, com a adição de um termo de ruído, para dificultar a tarefa de aprendizagem. Visualmente, a previsão da série mostra que o método proposto conseguiu eliminar o ruído do treinamento. A Figura 5.9 mostra o ajuste da curva na série não estacionária de Lorenz, com um comportamento mais complexo que a base do SinC. Mais uma vez, a previsão dos valores da série no conjunto de teste mostra que a filtragem do ruído pelo KLM permitiu um ajuste quase idêntico à série original. De forma similar, o mesmo nível de ajuste pode ser observado para as séries caóticas de Mackey-Glass, Rossler e Logistic, de acordo com as Figuras 5.10, 5.11 e 5.12, respectivamente. Nesses gráficos pode ser observado que é difícil distinguir os valores observados dos previstos, devido ao processo de previsão ter se ajustado quase que perfeitamente aos valores da série original. A correlação entre os valores previstos e observados das séries pode ser vista nos gráficos de dispersão da Figura 5.13. Com exceção da série de Lorenz, as demais bases apresentam um relacionamento aproximadamente linear para a maioria dos pontos observados, levando em consideração a variância das previsões. As diferenças de linearidade observadas podem ser explicadas em função da natureza de cada série estudada. As séries de Lorenz e Mackey-Glass apresentam níveis de ruído mais elevados em relação às demais séries, o que dificulta a filtragem na fase de aprendizagem sequencial do KLM.

As distribuições dos dados utilizados no processo de previsão apresentam diversos formatos para as bases utilizadas nos experimentos, conforme visualizados nos histogramas da Figura 5.14. Apenas as bases do Abalone, Blog e Istanbul apresentam certa simetria na distribuição dos dados. Algumas bases apresentam forte assimetria à direita (Bike, Communities), enquanto as demais bases mostram comportamento bimodal ou multimodal em suas distribuições.

A Tabela 5.2 contém os valores das raízes do erro médio quadrado (RMSE) para todos os experimentos envolvendo as bases de dados descritas na Seção 5.1.1. Cada célula contém a média dos 50 valores de RMSE obtidos para cada base e respectivo método, seguida por seu desvio padrão em parênteses. Os melhores valores de RMSE, sem considerar a significância dos testes *post hoc*, estão destacados em negrito. Nos experimentos realizados, utilizamos o método de *holdout* repetido (WITTEN; FRANK, 2005), o qual permite avaliar a variação do

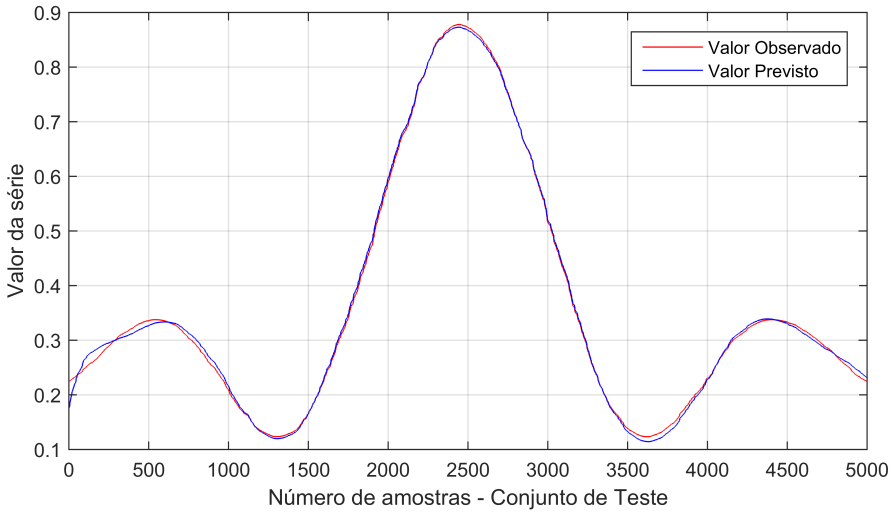


Figura 5.8: SinC - Comparação entre os valores observados e previstos com o KLM

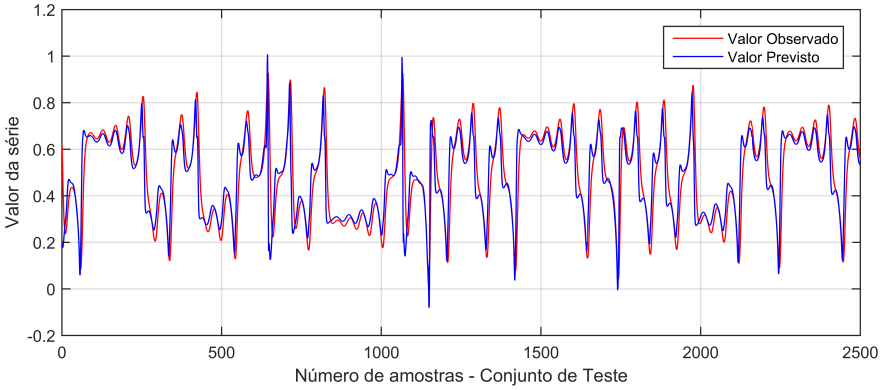


Figura 5.9: Lorenz - Comparação entre os valores observados e previstos com o KLM

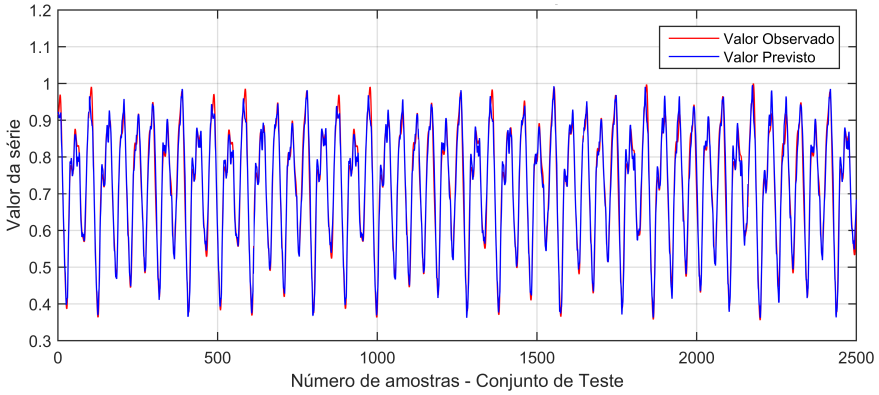


Figura 5.10: Mackey-Glass - Comparação entre os valores observados e previstos com o KLM

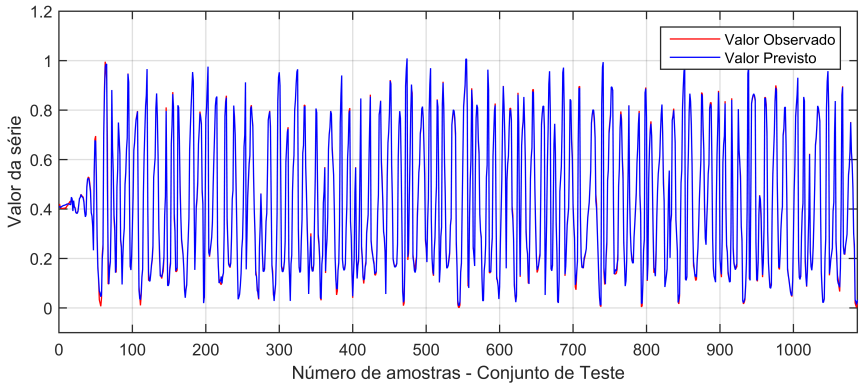


Figura 5.11: Rossler - Comparação entre os valores observados e previstos com o KLM

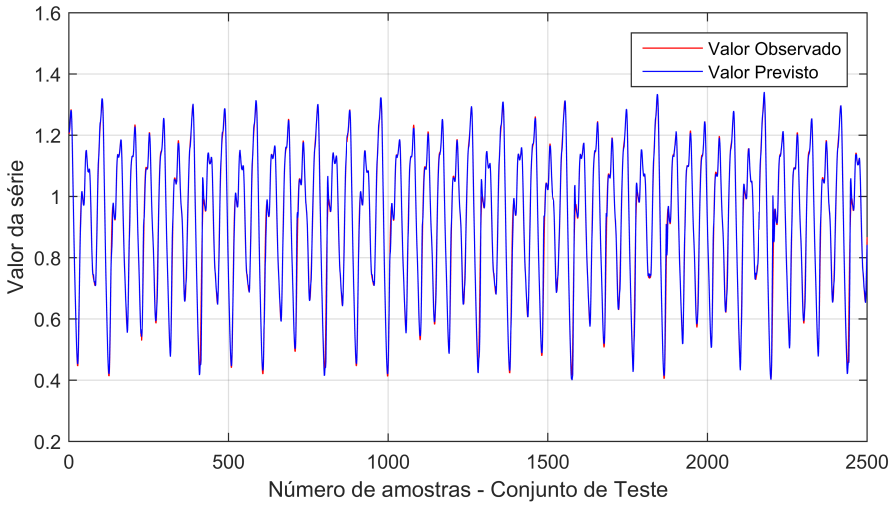
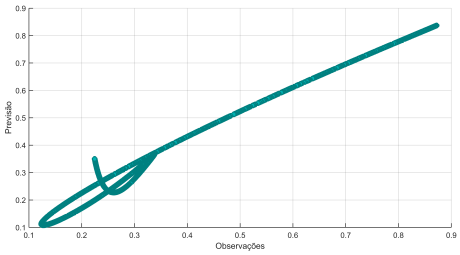
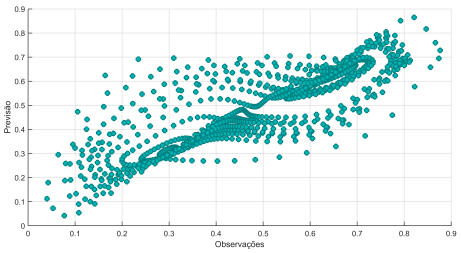


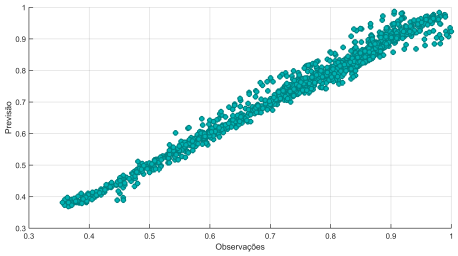
Figura 5.12: Logistic - Comparação entre os valores observados e previstos com o KLM



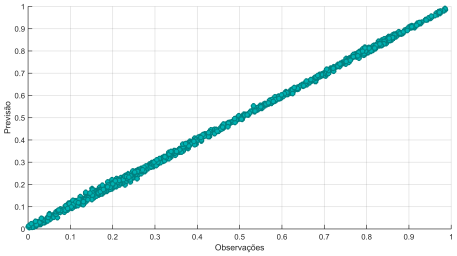
(a) SinC



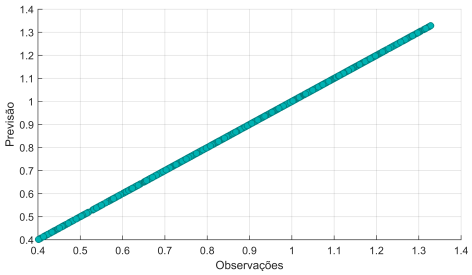
(b) Lorenz



(c) Mackey-Glass

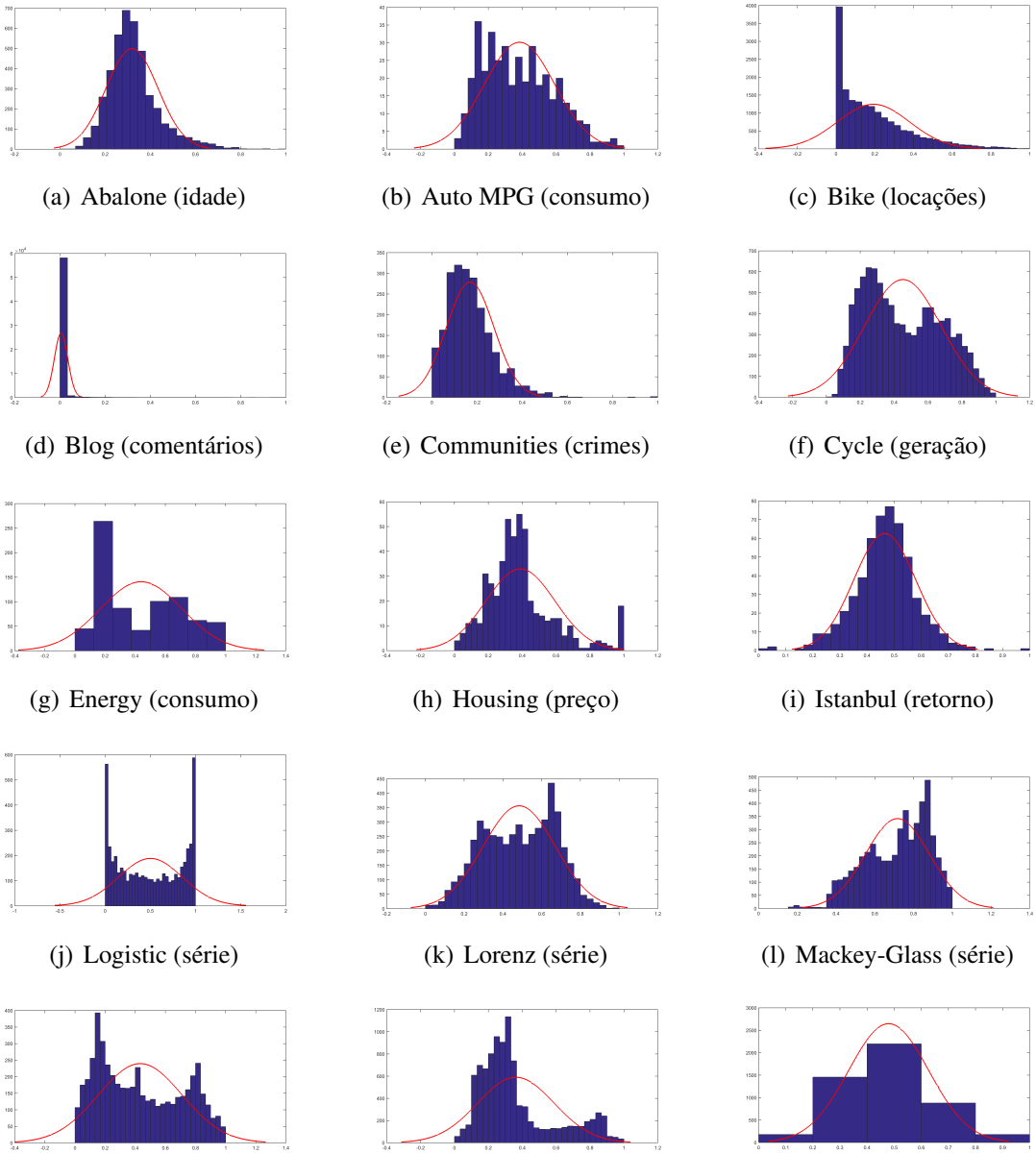


(d) Rossler



(e) Logistic

Figura 5.13: Valores Previstos e Observados



(a) Abalone (idade) (b) Auto MPG (consumo) (c) Bike (locações)

(d) Blog (comentários) (e) Communities (crimes) (f) Cycle (geração)

(g) Energy (consumo) (h) Housing (preço) (i) Istanbul (retorno)

(j) Logistic (série) (k) Lorenz (série) (l) Mackey-Glass (série)

(m) Rossler (série) (n) SinC (série) (o) Wine (qualidade)

Figura 5.14: Histogramas do atributo alvo para cada base de dados

erro de estimativa após múltiplas execuções do método. Observando o desvio padrão dos erros de previsão no conjunto de teste para cada base estudada, podemos inferir que não ocorreu *overfitting* durante o treinamento.

De acordo com a Tabela 5.2, o KLM conseguiu os melhores valores em 27% dos experimentos, incluindo a maior base de dados em número de observações. O método EKLM alcançou o menor erro médio de previsão em 23% dos experimentos, com destaque para os valores obtidos nos problemas de previsão de séries temporais não lineares e não estacionárias. No total, os métodos propostos nessa tese conseguiriam os melhores resultados em metade dos experimentos. Observa-se ainda que em nenhum experimento o KLM e o EKLM ficaram entre os piores desempenhos de forma isolada, considerando apenas o valor medido do erro. O OS-ELM-TV obteve os menores erros de previsão em 20% dos experimentos, enquanto os métodos LS-IELM e ReOS-ELM alcançaram resultados de 13% e 10%, respectivamente. Nota-se que os métodos com abordagens baseadas em regularização possuem valores semelhantes, denotando uma equivalência entre eles, mesmo sem levar em consideração a significância dos testes. O algoritmo original OS-ELM alcançou os menores erros em 7% dos experimentos, enquanto o TOSELM apresentou os maiores erros de previsão em praticamente todos os experimentos.

Tabela 5.2: Erros médios de previsão para todos os experimentos

Base	EKLM	KLM	OS-ELM	TOSELM	ReOS-ELM	OS-ELM-TV	LS-IELM
Abalone(RBF)	**0,0767** (0,0030)	0,0787 (0,0055)	0,0787 (0,0078)	0,0794 (0,0072)	0,0778 (0,0042)	0,0780 (0,0061)	0,0788 (0,0060)
Abalone(Sig)	**0,0777** (0,0045)	0,0795 (0,0083)	0,0803 (0,0106)	0,0799 (0,0095)	0,0779 (0,0049)	0,0787 (0,0081)	0,0799 (0,0074)
Auto MPG(RBF)	0,0768 (0,0080)	**0,0744** (0,0082)	0,0779 (0,0071)	0,0805 (0,0092)	0,0746 (0,0079)	0,0775 (0,0079)	0,0744 (0,0081)
Auto MPG(Sig)	0,0780 (0,0076)	**0,0768** (0,0088)	0,0778 (0,0085)	0,0816 (0,0078)	0,0792 (0,0071)	0,0770 (0,0089)	0,0787 (0,0078)
Bike(RBF)	0,0035 (0,0010)	**0,0033** (0,0009)	0,0034 (0,0010)	0,0035 (0,0010)	0,0037 (0,0009)	0,0179 (0,0014)	0,0036 (0,0011)
Bike(Sig)	0,0022 (0,0002)	**0,0021** (0,0001)	0,0022 (0,0002)	0,0022 (0,0001)	0,0021 (0,0001)	0,0022 (0,0002)	0,0021 (0,0002)
Blog(RBF)	0,0341 (0,0033)	0,0326 (0,0029)	0,0332 (0,0040)	0,0332 (0,0039)	0,0290 (0,0015)	0,0288 (0,0018)	**0,0285** (0,0019)
Blog(Sig)	0,0272 (0,0017)	0,0273 (0,0015)	0,0277 (0,0015)	0,0274 (0,0016)	0,0272 (0,0012)	**0,0271** (0,0016)	0,0275 (0,0019)
Communities(RBF)	0,0778 (0,0080)	0,0776 (0,0104)	0,0743 (0,0058)	0,0771 (0,0083)	0,0696 (0,0070)	0,0888 (0,0083)	**0,0670** (0,0058)
Communities(Sig)	0,0500 (0,0037)	0,0508 (0,0058)	0,0501 (0,0036)	0,0549 (0,0054)	**0,0496** (0,0036)	0,0524 (0,0089)	0,0498 (0,0037)
Cycle(RBF)	0,0546 (0,0017)	0,0544 (0,0014)	0,0552 (0,0020)	0,0569 (0,0103)	0,0546 (0,0018)	**0,0543** (0,0013)	0,0555 (0,0038)
Cycle(Sig)	0,0554 (0,0020)	**0,0546** (0,0015)	0,0578 (0,0062)	0,0559 (0,0031)	0,0567 (0,0070)	0,0553 (0,0013)	0,0571 (0,0097)
Energy(RBF)	0,0729 (0,0056)	0,0602 (0,0039)	**0,0599** (0,0045)	0,0733 (0,0164)	0,0603 (0,0045)	0,0693 (0,0041)	0,0604 (0,0045)
Energy(Sig)	0,0559 (0,0047)	0,0586 (0,0062)	0,0564 (0,0055)	0,0752 (0,0191)	0,0554 (0,0046)	**0,0549** (0,0051)	0,0559 (0,0043)
Housing(RBF)	0,1027 (0,0146)	0,1015 (0,0140)	0,0994 (0,0155)	0,1145 (0,0172)	0,0983 (0,0148)	0,1052 (0,0165)	**0,0977** (0,0124)
Housing(Sig)	0,0963 (0,0168)	**0,0914** (0,0111)	0,0914 (0,0109)	0,1161 (0,0187)	0,0939 (0,0129)	0,0951 (0,0166)	0,0958 (0,0128)
Istanbul(RBF)	0,0817 (0,0069)	0,0805 (0,0054)	0,0802 (0,0054)	0,0850 (0,0096)	**0,0793** (0,0084)	0,0805 (0,0066)	0,0827 (0,0068)
Istanbul(Sig)	0,0787 (0,0051)	0,0791 (0,0061)	0,0786 (0,0054)	0,0791 (0,0065)	**0,0781** (0,0069)	0,0791 (0,0060)	0,0797 (0,0063)
Logistic(RBF)	9,00e-4 (0,0003)	9,44e-4 (0,0003)	1,067e-3 (0,0003)	1,20e-3 (0,0005)	1,32e-3 (0,0004)	**2,00e-4** (0,0001)	1,07e-4 (0,0002)
Logistic(Sig)	**1,63e-4** (9,00e-6)	1,64e-4 (9,00e-6)	1,80e-4 (2,6e-5)	1,96e-4 (5,3e-5)	1,97e-4 (4,40e-5)	1,65e-4 (1,10e-5)	1,64e-4 (1,80e-5)
Lorentz(RBF)	0,0843 (0,0046)	0,0835 (0,0060)	0,0858 (0,0068)	0,0892 (0,0080)	0,0858 (0,0101)	**0,0826** (0,0052)	0,0856 (0,0043)
Lorentz(Sig)	**0,0831** (0,0049)	0,0833 (0,0059)	0,0856 (0,0048)	0,0888 (0,0049)	0,0851 (0,0039)	0,0837 (0,0098)	0,0862 (0,0039)
Mackey-Glass(RBF)	**0,0230** (0,0026)	0,0272 (0,0036)	0,0275 (0,0025)	0,0291 (0,0062)	0,0362 (0,0064)	0,0263 (0,0030)	0,0389 (0,0079)
Mackey-Glass(Sig)	**0,0231** (0,0030)	0,0291 (0,0020)	0,0331 (0,0066)	0,0305 (0,0034)	0,0464 (0,0072)	0,0297 (0,0031)	0,0451 (0,0025)
Rossler(RBF)	0,0122 (0,0029)	0,0116 (0,0023)	0,0128 (0,0030)	0,0121 (0,0027)	0,0130 (0,0025)	**0,0106** (0,0023)	0,0139 (0,0032)
Rossler(Sig)	0,0140 (0,0034)	**0,0124** (0,0035)	0,0134 (0,0027)	0,0141 (0,0040)	0,0169 (0,0037)	0,0138 (0,0036)	0,0142 (0,0035)
SinC(RBF)	0,0088 (0,0011)	0,0086 (0,0005)	**0,0085** (0,0015)	0,0093 (0,0015)	0,0331 (0,0044)	0,0093 (0,0008)	0,0320 (0,0031)
SinC(Sig)	**0,0120** (0,0020)	0,0123 (0,0024)	0,0137 (0,0026)	0,0133 (0,0030)	0,1753 (0,0507)	0,0123 (0,0021)	0,1593 (0,0332)
Wine(RBF)	0,1260 (0,0032)	0,1263 (0,0035)	0,1258 (0,0028)	0,1266 (0,0028)	0,1266 (0,0030)	0,1265 (0,0026)	**0,1249** (0,0031)
Wine(Sig)	0,1279 (0,0040)	**0,1264** (0,0031)	0,1268 (0,0036)	0,1279 (0,0038)	0,1277 (0,0035)	0,1274 (0,0029)	0,1269 (0,0030)

Os valores apresentados na Tabela 5.2 não podem ser analisados apenas por seus valores médios, sendo necessária uma avaliação estatística para confirmar as diferenças de desempenho quando comparados par a par. De acordo com a metodologia descrita na Seção 5.1, foi aplicado o teste não paramétrico de Friedman para estabelecer um *ranking* de desempenho dos métodos abordados. A hipótese nula do teste é que não existem diferenças entre as médias da medida de desempenho dos grupos, formados pelos algoritmos. Todos os métodos tiveram a posição

relativa de seu desempenho medida em cada base, com o menor valor atribuído ao menor erro de previsão. Se dois ou mais algoritmos apresentam o mesmo desempenho para uma base de dados, o valor médio do *ranking* é atribuído a esse grupo. O teste apresentou o valor de 37,49 para a estatística χ^2 e *p-value* de 1,4101e-06, o que rejeita a hipótese nula que o desempenho dos métodos é igual. A Tabela 5.3 apresenta o *ranking* do teste de Friedman para as médias da medida de desempenho, com nível de confiança $\alpha = 0,05$ para um valor crítico de 2,850, de acordo com a distribuição de Nemenyi (NEMENYI, 1962).

Tabela 5.3: Ranking do teste de Friedman para as médias de desempenho ($\chi^2 = 37,49$, p-value=1,4101e-06 e valor crítico=2,850)

Posição	Método	Valor do Ranking
1	KLM	2,72
2	EKLM	3,38
3	OS-ELM-TV	3,43
4	OS-ELM	4,03
5	ReOS-ELM	4,18
6	LS-IELM	4,45
7	TOSELM	5,80

Após a rejeição da hipótese nula, um teste *post hoc* foi executado para avaliar cada algoritmo em relação aos demais, em uma comparação par a par. Foi adotado o teste de Tukey (TUKEY, 1949) para avaliar as diferenças entre os sete métodos com nível de confiança $\alpha = 0,05$. A Tabela 5.4 apresenta os resultados das comparações par a par. A primeira coluna mostra o par comparado. A segunda coluna apresenta a estatística do teste, baseada na diferença das médias do *ranking* de Friedman. A terceira coluna mostra o intervalo de confiança de 95% para a diferença das médias. A última coluna apresenta o *p-value* do teste, com os valores em negrito representando os casos nos quais a hipótese nula de igualdade de desempenho foi rejeitada. A Figura 5.15 apresenta visualmente os resultados dos testes de Friedman e Tukey, com as posições do *ranking* e os intervalos de confiança. Observe que em casos nos quais existem diferenças estatisticamente significativas, os valores dos *rankings* nos intervalos de confiança não se sobrepõem.

A análise da Figura 5.15 mostra que as diferenças dos erros de previsão do KLM são estatisticamente significativas quando comparadas com os métodos OSELM, TOSELM, ReOS-ELM e LS-IELM. No entanto, não é possível afirmar que o KLM apresenta um desempenho superior nos experimentos realizados em relação ao EKLM e o OS-ELM-TV.

A análise dos resultados a partir do RMSE mostra que o método proposto possui desempenho superior à maioria dos trabalhos relacionados com o tópico dessa tese. Entretanto,

Tabela 5.4: Comparação par a par do RMSE - Teste de Tukey com intervalo de confiança de 95%

Comparação	Estatística	Intervalo	p-value
EKLM - KLM	0,67	[-0,43;1,76]	0,23
EKLM - OS-ELM	-0,65	[-1,74;0,44]	0,24
EKLM - TOSELM	-2,42	[-3,51;-1,32]	$< \mathbf{10^{-2}}$
EKLM - ReOS-ELM	-0,80	[-1,89;0,29]	0,15
EKLM - OS-ELM-TV	-0,05	[-1,14;1,04]	0,93
EKLM - LS-IELM	-1,01	[-2,15;0,02]	0,05
KLM - OS-ELM	-1,32	[-2,41;-0,22]	**0,02**
KLM - TOSELM	-3,08	[-4,17;-1,99]	$< \mathbf{10^{-2}}$
KLM - ReOS-ELM	-1,47	[-2,55;-0,37]	**0,01**
KLM - OS-ELM-TV	-0,72	[-1,81;0,37]	0,20
KLM - LS-IELM	-1,73	[-2,82;-0,64]	$< \mathbf{10^{-2}}$
OS-ELM - TOSELM	-1,77	[-2,86;-0,67]	**0,00**
OS-ELM - ReOS-ELM	-0,15	[-1,24;0,94]	0,79
OS-ELM - OS-ELM-TV	0,60	[-0,49;1,69]	0,28
OS-ELM - LS-IELM	-0,42	[-1,51;0,67]	0,45
TOSELM - ReOS-ELM	1,62	[0,52;2,71]	$< \mathbf{10^{-2}}$
TOSELM - OS-ELM-TV	2,37	[1,27;3,46]	$< \mathbf{10^{-2}}$
TOSELM - LS-IELM	1,35	[0,26;2,44]	**0,01**
ReOS-ELM - OS-ELM-TV	0,75	[-0,34;1,84]	0,18
ReOS-ELM - LS-IELM	-0,27	[-1,36;0,82]	0,63
OS-ELM-TV - LS-IELM	-1,02	[-2,11;0,07]	0,07

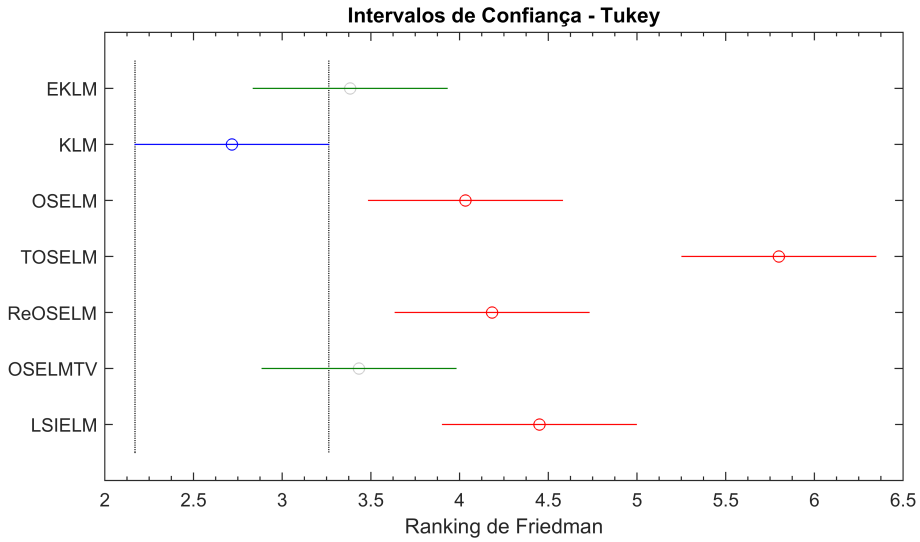


Figura 5.15: Intervalos de confiança de 95% com a distribuição de Tukey para o erro de previsão.

conforme discutido na Seção 5.1.3, o RMSE é dependente das escalas dos valores de cada base, mesmo com a normalização dos atributos de seus atributos. Para avaliar o desempenho dos métodos de forma independente da escala, calculamos o coeficiente U de Theil para as mesmas estimativas da Tabela 5.2. De acordo com a Tabela 5.5, os resultados foram semelhantes à avaliação do erro de previsão, com o KLM e EKLM apresentando valores menores para o coeficiente em metade dos experimentos. De forma similar, os algoritmos do método proposto só apresentaram o pior desempenho entre seus pares em apenas dois dos trinta experimentos realizados. O OS-ELM-TV obteve o menor valor do coeficiente em nove dos trinta experimentos, enquanto o ReOS-ELM conseguiu os menores valores em seis experimentos. O desempenho do OS-ELM e LS-IELM foi similar, com dois valores entre os menores para os trinta experimentos. Novamente, o TOSELM apresentou os maiores valores para a métrica. Na Figura 5.16 podemos observar que as medianas dos coeficientes de Theil ficaram abaixo de 0,1 para todos os métodos, indicando que os valores estimados estão próximos das observações de cada base. Os *outliers* da figura estão relacionados com a base Blog, na qual o atributo alvo apresenta uma grande quantidade de valores iguais a zero.

Tabela 5.5: Coeficiente U de Theil para todos os experimentos

Base	EKLM	KLM	OS-ELM	TOSELM	ReOS-ELM	OS-ELM-TV	LS-IELM
Abalone(RBF)	0,1145	**0,1141**	0,1149	0,1162	0,1151	0,1146	0,1172
Abalone(Sig)	0,1159	**0,1154**	**0,1154**	0,1189	0,1161	**0,1154**	0,1171
Auto MPG(RBF)	0,0876	0,0864	0,0847	0,0891	**0,0831**	0,0875	0,0879
Auto MPG(Sig)	0,0865	**0,0844**	0,0863	0,0898	0,0872	0,0866	0,0870
Bike(RBF)	0,0070	0,0064	0,0067	0,0064	0,0069	0,0333	**0,0063**
Bike(Sig)	0,0040	0,0040	0,0040	0,0041	0,0040	0,0040	0,0040
Blog(RBF)	0,6969	0,6962	**0,6946**	0,7025	0,7665	0,7523	0,7675
Blog(Sig)	0,6319	0,6263	0,6304	0,6368	**0,6262**	0,6294	0,6281
Communities(RBF)	0,1965	0,1890	0,1967	0,1996	0,1732	0,1918	**0,1700**
Communities(Sig)	0,1232	0,1217	0,1217	0,1346	0,1221	**0,1211**	0,1223
Cycle(RBF)	**0,0539**	0,0540	0,0572	0,0549	0,0547	**0,0539**	0,0548
Cycle(Sig)	**0,0545**	0,0546	0,0566	0,0565	0,0565	0,0547	0,0553
Energy(RBF)	0,0586	0,0507	0,0502	0,0622	**0,0481**	0,0620	0,0505
Energy(Sig)	0,0548	0,0573	0,0435	0,0567	**0,0434**	0,0587	0,0438
Housing(RBF)	**0,1029**	0,1032	0,1037	0,1611	0,1060	0,1119	0,1041
Housing(Sig)	0,0972	**0,0928**	0,0989	0,1867	0,1015	0,1003	0,0965
Istanbul(RBF)	0,0854	**0,0825**	0,0853	0,0850	0,0840	**0,0825**	0,0827
Istanbul(Sig)	0,0830	0,0819	0,0826	0,0849	**0,0816**	0,0831	0,0827
Logistic(RBF)	0,3168	0,3165	0,3170	0,3166	0,3165	**0,3160**	0,3169
Logistic(Sig)	0,3170	0,3182	0,3168	0,3168	0,3173	**0,3167**	0,3169
Lorentz(RBF)	0,0806	0,0795	0,1002	0,0916	0,0815	**0,0788**	0,0808
Lorentz(Sig)	**0,0783**	0,0791	0,0838	0,0836	0,0799	0,0784	0,0826
Mackey-Glass(RBF)	0,0159	0,0156	0,1289	0,0170	0,0608	**0,0146**	0,2131
Mackey-Glass(Sig)	0,0159	0,0157	0,0171	0,0171	0,0956	**0,0156**	0,0311
Rossler(RBF)	0,0116	0,0114	0,0408	0,0134	**0,0108**	0,0105	0,2178
Rossler(Sig)	**0,0124**	0,0129	0,0140	0,0135	0,0709	0,0126	0,0935
SinC(RBF)	**0,0110**	**0,0110**	0,0118	0,0119	0,0469	0,0120	0,0420
SinC(Sig)	0,0156	**0,0153**	0,0199	0,0164	0,2333	0,0154	0,2869
Wine(RBF)	**0,1264**	0,1277	0,1276	0,1287	0,1274	0,1286	0,1268
Wine(Sig)	0,1288	**0,1278**	0,1288	0,1304	0,1279	0,1279	0,1288

De forma similar ao tratamento dado à métrica do erro de previsão, foi executado o teste não paramétrico de Friedman para gerar o *ranking* com os melhores desempenhos do coeficiente

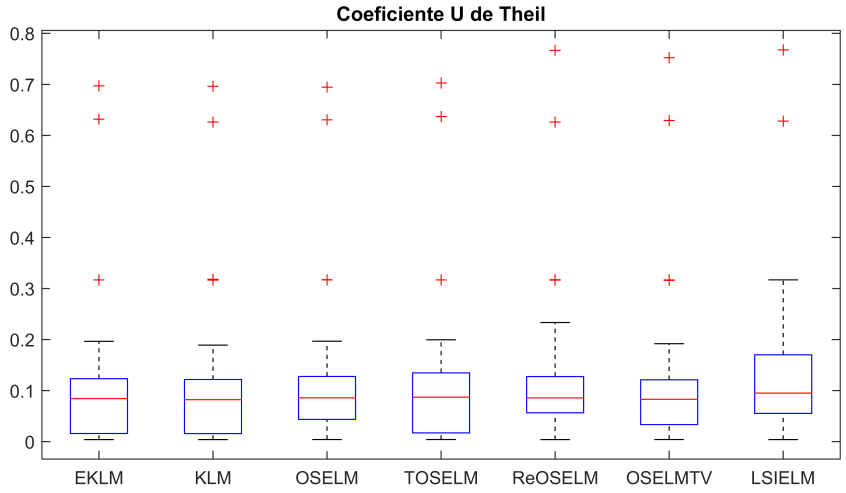


Figura 5.16: Coeficiente U de Theil para os métodos avaliados (box-plot).

U de Theil. O teste apresentou o valor de 36,91 para a estatística χ^2 e *p-value* de 1,8299e-06, rejeitando a hipótese nula de desempenhos semelhantes entre os modelos. A Tabela 5.6 apresenta o *ranking* de Friedman para o coeficiente U de Theil com nível de confiança $\alpha = 0,05$. Em comparação aos resultados da Tabela 5.4, podemos observar que o KLM apresentou a melhor média das relativas. Diferente da análise do erro de previsão, o OS-ELM-TV subiu uma posição, trocando de lugar com o EKLM. Como veremos adiante, as diferenças de precisão entre o EKLM e o OS-ELM-TV não são estatisticamente significativas.

Tabela 5.6: Ranking do teste de Friedman para o coeficiente U de Theil ($\chi^2 = 36,91$, p-value=1,8299e-06 e valor crítico=2,850)

Posição	Método	Valor do Ranking
1	KLM	2,50
2	OS-ELM-TV	3,36
3	EKLM	3,60
4	ReOS-ELM	4,10
5	OS-ELM	4,33
6	LS-IELM	4,50
7	TOSELM	5,60

As comparações par a par do coeficiente são apresentadas na Tabela 5.7 através da execução do teste de Tukey, com os valores em negrito na última coluna representando os casos nos quais a hipótese nula de igualdade de desempenho foi rejeitada. A Figura 5.17 apresenta os testes de forma consolidada. Podemos observar que as diferenças entre o KLM e os métodos EKLM, OS-ELM, TOSELM, ReOS-ELM e LS-IELM são estatisticamente significativas. Não foi possível afirmar que o KLM apresenta maior precisão que o OS-ELM-TV, devido à não rejeição da hipótese nula na comparação entre os dois métodos.

As avaliações estatísticas das duas métricas de desempenho mostraram cenários semelhantes em relação à comparação dos métodos. Tanto pelo critério do erro de previsão quanto pela precisão das estimativas, os métodos propostos nessa tese apresentaram desempenho superior à maioria dos algoritmos dos trabalhos relacionados. Os resultados dos testes confirmam que a aplicação do filtro de Kalman na fase de aprendizagem sequencial consegue reduzir o ruído presente nos dados de treinamento e trata de forma eficiente os efeitos da multicolinearidade durante o processo de aprendizado.

Tabela 5.7: Comparação par a par do coeficiente U de Theil - Teste de Tukey com intervalo de confiança de 95%

Comparação	Estatística	Intervalo	p-value
EKLM - KLM	1,10	[0,01;2,19]	**0,04**
EKLM - OS-ELM	-0,73	[-1,83;0,36]	0,19
EKLM - TOSELM	-2,00	[-3,09;-0,91]	$< \mathbf{10^{-2}}$
EKLM - ReOS-ELM	-0,50	[-1,59;0,59]	0,37
EKLM - OS-ELM-TV	0,23	[-0,86;1,33]	0,67
EKLM - LS-IELM	-0,90	[-1,99;0,19]	0,11
KLM - OS-ELM	-1,83	[-2,93;-0,74]	$< \mathbf{10^{-2}}$
KLM - TOSELM	-3,10	[-4,19;-2,01]	$< \mathbf{10^{-2}}$
KLM - ReOS-ELM	-1,60	[-2,69;-0,51]	$< \mathbf{10^{-2}}$
KLM - OS-ELM-TV	-0,87	[-1,95;0,23]	0,12
KLM - LS-IELM	-2,00	[-3,09;-0,91]	$< \mathbf{10^{-2}}$
OS-ELM - TOSELM	-1,27	[-2,36;-0,17]	0,02
OS-ELM - ReOS-ELM	0,23	[-0,86;1,32]	0,67
OS-ELM - OS-ELM-TV	0,97	[-0,13;2,06]	0,08
OS-ELM - LS-IELM	-0,17	[-1,26;0,93]	0,76
TOSELM - ReOS-ELM	1,50	[0,41;2,59]	$< \mathbf{10^{-2}}$
TOSELM - OS-ELM-TV	2,23	[1,14;3,33]	$< \mathbf{10^{-2}}$
TOSELM - LS-IELM	1,10	[0,01;2,19]	**0,04**
ReOS-ELM - OS-ELM-TV	0,73	[-0,36;1,83]	0,19
ReOS-ELM - LS-IELM	-0,40	[-1,49;0,69]	0,47
OS-ELM-TV - LS-IELM	-1,13	[-2,23;-0,04]	**0,04**

5.3 Análise do Custo Computacional

O último item avaliado nos experimentos foi o custo computacional, medido através do tempo de treinamento. Conforme discutido na Seção 4.6, o tempo de treinamento dos algoritmos baseados na OS-ELM é fortemente influenciado pelo número de unidades L da camada escondida. Mesmo com a otimização desse parâmetro, não podemos fazer uma comparação justa do tempo de treinamento se os valores forem diferentes para cada método comparado. Para avaliar esse

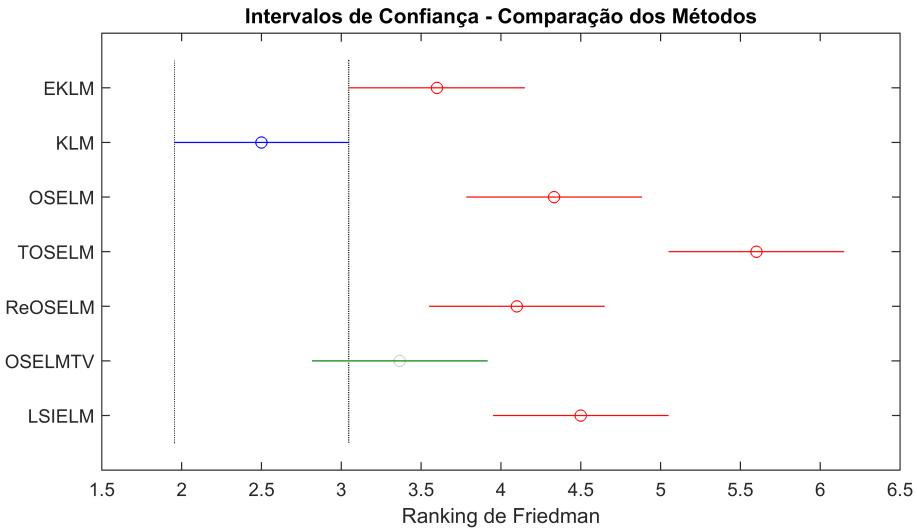


Figura 5.17: Intervalos de confiança de 95% com a distribuição de Tukey para o coeficiente U de Theil.

item, usamos os mesmos valores de L para todos os experimentos, mantendo a regra de inicializar o aprendizado com 10% do conjunto de treinamento. Os valores de L para cada base foram obtidos a partir da média calculada no processo de otimização. A Tabela 5.8 mostra os valores de L e de N_0 para as bases estudadas.

Tabela 5.8: Parâmetros usados na medição do tempo de treinamento

Base	**L**	**N_0**
Abalone	20	417
Auto MPG	27	39
Bike	198	1737
Blog	102	1000
Communities	200	221
Cycle	80	956
Energy	71	76
Housing	58	50
Istanbul	20	53
Logistic	155	500
Lorentz	110	500
Mackey-Glass	113	500
Rossler	110	500
SinC	12	500
Wine	62	489

Na Tabela 5.9 são apresentados em cada célula o tempo médio de treinamento para cinquenta execuções de cada experimento, seguido pelo desvio padrão em parênteses. Os menores valores do tempo de treinamento, sem considerar a significância dos testes *post hoc*, estão

destacados em negrito. Para todos os casos, o KLM conseguiu os menores tempos de treinamento quando o experimento utilizou a função de ativação sigmoide. Quando a função utilizada foi a RBF, o OS-ELM-TV obteve os melhores resultados na maioria dos experimentos. Levando em consideração que a escala do tempo é a mesma para todos os modelos avaliados, podemos observar na Figura 5.18 que as medianas do KLM, OS-ELM e OS-ELM-TV apresentaram valores próximos. Os maiores tempos de treinamento foram observados nos experimentos nos quais o EKLM foi aplicado. Esse fato pode ser explicado em função da complexidade computacional no cálculo das matrizes Jacobianas do algoritmo, o qual requer uma grande quantidade de multiplicações e inversões adicionais da matriz **H**.

Tabela 5.9: Tempo médio de treinamento (em segundos) para todos os experimentos

Base	EKLM	KLM	OS-ELM	TOSELM	ReOS-ELM	OS-ELM-TV	LS-IELM
Abalone(RBF)	1,1575 (0,0936)	0,4312 (0,0214)	0,4790 (0,0247)	0,8696 (0,0253)	0,4837 (0,0615)	**0,4193** (0,0236)	0,4812 (0,0291)
Abalone(Sig)	0,7165 (0,0690)	**0,1284** (0,0115)	0,1656 (0,0133)	0,5671 (0,0356)	0,1640 (0,0119)	0,3737 (0,0160)	0,1590 (0,0203)
Auto MPG(RBF)	0,1721 (0,0168)	**0,0437** (0,0130)	0,0775 (0,0130)	0,1103 (0,0123)	0,0731 (0,0119)	0,0746 (0,0115)	0,0753 (0,0177)
Auto MPG(Sig)	0,1175 (0,0131)	**0,0168** (0,0099)	0,0287 (0,0106)	0,0796 (0,0085)	0,0303 (0,0127)	0,0646 (0,0094)	0,0290 (0,0130)
Bike(RBF)	144,9284 (2,6196)	28,6487 (0,4350)	45,5503 (5,7547)	47,4493 (0,7844)	45,6340 (3,0443)	**13,2375** (0,1848)	41,0825 (4,4317)
Bike(Sig)	118,8556 (1,7598)	**6,8925** (0,1092)	18,7203 (1,6799)	28,1634 (2,8547)	22,0453 (2,3715)	12,8575 (0,2275)	17,2690 (0,7366)
Blog(RBF)	95,0012 (1,1790)	67,5246 (0,9343)	76,0053 (1,5244)	135,9693 (1,4791)	83,4590 (0,7222)	**26,9834** (0,3554)	77,1253 (0,5883)
Blog(Sig)	34,6115 (0,6596)	**10,5184** (0,2562)	17,2215 (0,2820)	74,1428 (0,6027)	22,2037 (0,2573)	25,2537 (0,3222)	16,8978 (0,3454)
Communities(RBF)	18,9700 (0,7573)	4,0846 (0,1209)	5,0959 (0,1821)	5,8437 (0,2223)	5,6531 (0,5295)	**1,4625** (0,0481)	5,1040 (0,4332)
Communities(Sig)	15,1556 (0,4541)	**0,7153** (0,0456)	1,8471 (0,1246)	2,4912 (0,1115)	2,1637 (0,1103)	1,3615 (0,0603)	1,9134 (0,1410)
Cycle(RBF)	12,0490 (0,3191)	5,9031 (0,2170)	6,5487 (0,1835)	10,0412 (0,2005)	7,5065 (0,3081)	**3,1321** (0,0860)	6,7634 (0,4386)
Cycle(Sig)	7,1437 (0,1995)	**1,0593** (0,0314)	1,6071 (0,0852)	4,9509 (0,1039)	2,3546 (0,0879)	2,9381 (0,0971)	1,6100 (0,0553)
Energy(RBF)	0,9600 (0,0138)	0,1646 (0,0152)	0,3762 (0,0304)	0,5496 (0,0410)	0,4356 (0,0346)	**0,1537** (0,0221)	0,3962 (0,0419)
Energy(Sig)	0,5846 (0,0187)	**0,0665** (0,0093)	0,0943 (0,0206)	0,2515 (0,0299)	0,1387 (0,0233)	0,1496 (0,0214)	0,0893 (0,0161)
Housing(RBF)	0,4937 (0,0277)	0,1203 (0,0138)	0,2037 (0,0236)	0,3175 (0,0367)	0,2050 (0,0253)	**0,0921** (0,0152)	0,2246 (0,0386)
Housing(Sig)	0,2806 (0,0147)	**0,0225** (0,0126)	0,0537 (0,0167)	0,1556 (0,0202)	0,0543 (0,0176)	0,0837 (0,0125)	0,0568 (0,0121)
Istanbul(RBF)	0,1843 (0,0189)	0,0546 (0,0127)	0,0843 (0,0099)	0,1287 (0,0116)	0,0800 (0,0136)	0,0790 (0,0079)	**0,0518** (0,0106)
Istanbul(Sig)	0,1287 (0,0111)	**0,0206** (0,0102)	0,0406 (0,0154)	0,0956 (0,0074)	0,0306 (0,0133)	0,0756 (0,0073)	0,0218 (0,0104)
Logistic(RBF)	1,3637 (0,1009)	0,4850 (0,0302)	0,4903 (0,0269)	1,0553 (0,0386)	0,5096 (0,0551)	**0,4775** (0,0273)	0,4921 (0,0319)
Logistic(Sig)	0,8587 (0,0726)	**0,1731** (0,0137)	0,1856 (0,0237)	0,7378 (0,0350)	0,1837 (0,0107)	0,4328 (0,0158)	0,1884 (0,0146)
Lorentz(RBF)	3,8065 (0,0877)	0,8762 (0,0292)	2,1512 (0,0722)	3,5490 (0,1518)	2,1659 (0,1094)	**0,5471** (0,0281)	2,1750 (0,0975)
Lorentz(Sig)	2,0946 (0,0960)	**0,2065** (0,0135)	0,4975 (0,0442)	1,7562 (0,0511)	0,5240 (0,0260)	0,5378 (0,0478)	0,5231 (0,0708)
Mackey-Glass(RBF)	11,4571 (0,3467)	3,7884 (0,1066)	4,1034 (0,2092)	5,6143 (0,3971)	4,5690 (0,2030)	**1,9353** (0,1052)	4,1212 (0,1243)
Mackey-Glass(Sig)	7,5846 (0,2749)	**0,6165** (0,0322)	1,0153 (0,0532)	2,2331 (0,0694)	1,4278 (0,0646)	1,6553 (0,0582)	1,0421 (0,0438)
Rossler(RBF)	7,3453 (0,2245)	3,7762 (0,0824)	4,4409 (0,0846)	5,4875 (0,1301)	4,9081 (0,3355)	**1,8146** (0,0611)	4,5325 (0,3138)
Rossler(Sig)	4,3975 (0,2101)	**0,6475** (0,0346)	1,1768 (0,0556)	2,3281 (0,0579)	1,5509 (0,0932)	1,6925 (0,0592)	1,1803 (0,0935)
SinC(RBF)	20,7065 (0,0482)	1,0728 (0,0165)	1,1137 (0,0892)	2,9300 (0,0944)	1,1896 (0,1394)	**0,9409** (0,0657)	1,1862 (0,0947)
SinC(Sig)	14,6781 (0,0275)	**0,3837** (0,0081)	0,4165 (0,0112)	2,2384 (0,2052)	0,4306 (0,0416)	0,8390 (0,0582)	0,3878 (0,0663)
Wine(RBF)	4,5296 (0,1185)	1,0828 (0,0455)	2,5734 (0,1934)	3,7078 (0,1663)	3,0662 (0,2521)	**0,7556** (0,0531)	2,7143 (0,4432)
Wine(Sig)	2,4637 (0,0658)	**0,2281** (0,0172)	0,5962 (0,0343)	1,7253 (0,0533)	0,9181 (0,0518)	0,7181 (0,0444)	0,5950 (0,0299)

Para avaliar estatisticamente as diferenças de desempenho em relação ao tempo de treinamento, foi aplicado o teste de Friedman para as médias exibidas na Tabela 5.9. Os valores do *ranking* são apresentados na Tabela 5.10 com nível de confiança $\alpha = 0,05$. O teste apresentou o valor de 140,61 para a estatística χ^2 e *p-value* de 7,4356e-28, o que rejeita a hipótese nula que as médias dos tempos de treinamento sejam iguais para todos os métodos. O resultado do *ranking* mostra que o KLM obteve as melhores posições relativas nos experimentos, com metade do valor encontrado para o OS-ELM-TV. Essa diferença pode ser explicada pelos valores da Tabela 5.9, na qual o KLM sempre foi o melhor nos experimentos baseados na função sigmoide, ficando com o segundo menor tempo de treinamento em todos os experimentos quando a função RBF foi utilizada.

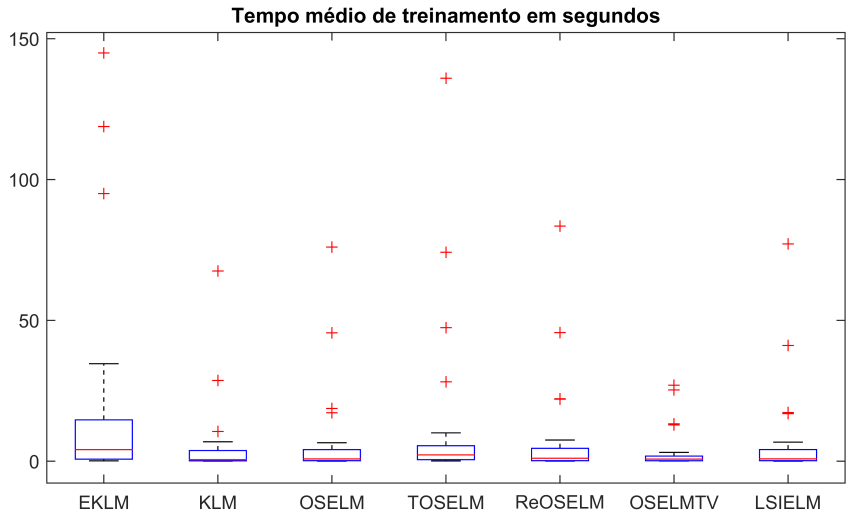


Figura 5.18: Tempo de treinamento para os métodos avaliados (*box-plot*).

Tabela 5.10: Ranking do teste de Friedman para os tempos médios de treinamento ($\chi^2 = 140,61$, p-value=7,4356e-28 e valor crítico=2,850)

Posição	Método	Valor do Ranking
1	KLM	1,47
2	OS-ELM-TV	2,90
3	OS-ELM	3,07
4	LS-IELM	3,33
5	ReOS-ELM	4,23
6	TOSELM	6,07
7	EKLM	6,93

A comparação par a par das diferenças entre os tempos de treinamento para todos os métodos pode ser vista na Tabela 5.11. Os valores em negrito na última coluna representam os casos nos quais a hipótese nula de igualdade de desempenho foi rejeitada. Na Figura 5.19 podemos observar que as diferenças entre o KLM e os demais métodos estudados são estatisticamente significativas.

A avaliação do tempo de treinamento para as bases de dados desse estudo mostra que apesar da complexidade assintótica do KLM ser a mesma da OS-ELM original e das extensões ReOS-ELM e LS-IELM, podemos definir as matrizes de covariância do algoritmo como simples escalares, reduzindo o tempo total de treinamento, conforme discutido no Item 4 da Seção 4.8.

Tabela 5.11: Comparação par a par do tempo de treinamento - Teste de Tukey com intervalo de confiança de 95%

Comparação	Estatística	Intervalo	p-value
EKLM - KLM	5,47	[4,37;6,56]	$< \mathbf{10^{-2}}$
EKLM - OS-ELM	3,87	[2,77;4,96]	$< \mathbf{10^{-2}}$
EKLM - TOSELM	0,87	[-0,23;1,96]	0,12
EKLM - ReOS-ELM	2,70	[1,61;3,79]	$< \mathbf{10^{-2}}$
EKLM - OS-ELM-TV	4,03	[2,94;5,13]	$< \mathbf{10^{-2}}$
EKLM - LS-IELM	3,60	[2,51;4,69]	$< \mathbf{10^{-2}}$
KLM - OS-ELM	-1,60	[-2,69;-0,51]	$< \mathbf{10^{-2}}$
KLM - TOSELM	-4,60	[-5,69;-3,51]	$< \mathbf{10^{-2}}$
KLM - ReOS-ELM	-2,77	[-3,86;-1,67]	$< \mathbf{10^{-2}}$
KLM - OS-ELM-TV	-1,43	[-2,53;-0,34]	**0,01**
KLM - LS-IELM	-1,87	[-2,96;-0,77]	$< \mathbf{10^{-2}}$
OS-ELM - TOSELM	-3,00	[-4,09;-1,91]	$< \mathbf{10^{-2}}$
OS-ELM - ReOS-ELM	-1,17	[-2,26;-0,07]	**0,04**
OS-ELM - OS-ELM-TV	0,17	[-0,93;1,26]	0,76
OS-ELM - LS-IELM	-0,27	[-1,34;0,83]	0,63
TOSELM - ReOS-ELM	1,83	[0,74;2,93]	$< \mathbf{10^{-2}}$
TOSELM - OS-ELM-TV	3,17	[2,07;4,26]	$< \mathbf{10^{-2}}$
TOSELM - LS-IELM	2,73	[1,64;3,83]	$< \mathbf{10^{-2}}$
ReOS-ELM - OS-ELM-TV	1,33	[0,24;2,43]	**0,02**
ReOS-ELM - LS-IELM	0,90	[-0,19;1,99]	0,11
OS-ELM-TV - LS-IELM	-0,43	[-1,53;0,66]	0,44

5.4 Estudo de Caso: Previsão de Séries Temporais Cointegradas para o Problema da Arbitragem Estatística

Após a aplicação e validação dos algoritmos do método proposto em problemas conhecidos da comunidade de aprendizagem de máquina, foram realizados experimentos voltados

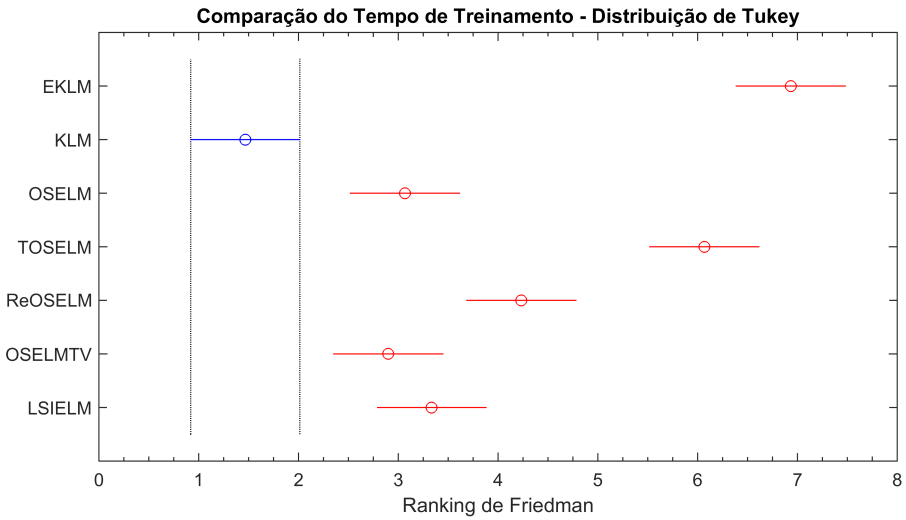


Figura 5.19: Intervalos de confiança de 95% com a distribuição de Tukey para o tempo médio de treinamento.

para um problema real de previsão de séries temporais para o mercado financeiro, conhecido como arbitragem estatística. No mundo das finanças, o termo arbitragem estatística engloba um conjunto de estratégias de negociação nos quais o principal objetivo é obter lucro a partir das discrepâncias de preços observadas em um grupo de ativos. Uma das técnicas mais utilizadas na arbitragem estatística é a negociação de pares de ativos financeiros (VIDYAMURTHY, 2004). Essa estratégia identifica pares de ativos nos quais as séries dos preços possuem forte correlação positiva ao longo do tempo. A Figura 5.20 mostra um exemplo de um par de ativos no qual o movimento dos preços está fortemente correlacionado. Nesse exemplo, as séries das ações ordinárias e preferenciais da empresa Vale S.A. apresentam um movimento similar na frequência de um minuto para o período de Fevereiro de 2009 a Março de 2013, com o coeficiente de correlação linear de Pearson igual a 0,97. A estratégia de negociação consiste em explorar os desvios que ocorrem na diferença relativa entre os dois ativos para pequenos intervalos de tempo. Na área de Econometria, o movimento dos preços dos ativos financeiros é modelado normalmente usando uma distribuição Log-normal, com comportamento baseado em um passeio aleatório (*random walk*). As séries temporais com esse tipo de comportamento são tipicamente não estacionárias e o seu processo de previsão é complexo. No trabalho publicado em ENGLE; GRANGER (1987), ganhador do prêmio Nobel de Economia de 2003, foi mostrado que mesmo para séries não estacionárias, é possível encontrar, em alguns casos, uma combinação linear dessas séries que formam uma nova série estacionária. Engle e Granger usaram o termo cointegração para definir as séries cuja combinação linear seja reversível à média com variabilidade finita. Se em pequeno período de tempo ocorrer um desvio na média da série estacionária, uma das séries, ou ambas, corrigem os seus preços para restaurar o equilíbrio definido pela média.

O modelo de séries temporais cointegradas permite fazer previsões para a diferença relativa entre os pares de ativos. O termo *spread* é usado para definir essa diferença. Na Figura 5.21 podemos observar o *spread* formado pela combinação linear das séries mostradas na Figura

5.20. Se uma dada observação da série do *spread* for maior ou menor que a sua previsão, podemos comprar ou vender simultaneamente os ativos originais e obter lucro quando a série reverter à média. Um histórico e uma discussão mais aprofundada das estratégias de negociação baseadas em arbitragem estatística pode ser vista em VIDYAMURTHY (2004).

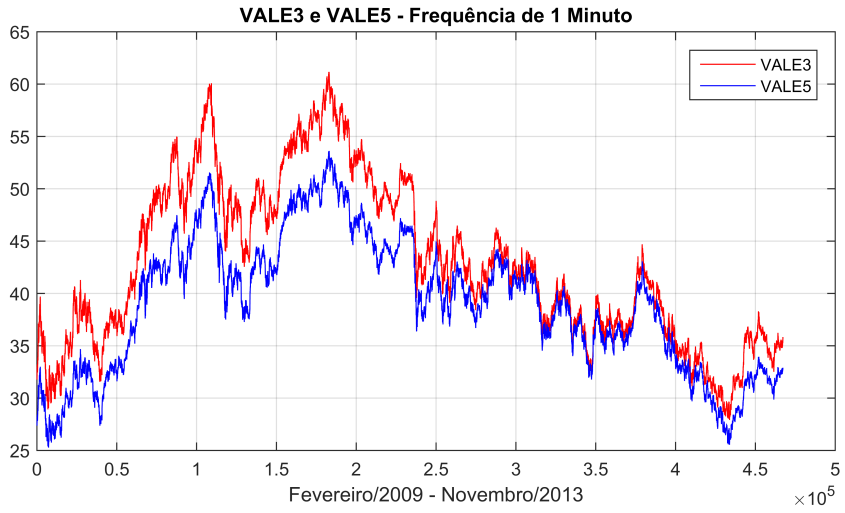


Figura 5.20: Séries dos preços de VALE3 e VALE5 ($\rho = 0,97$).

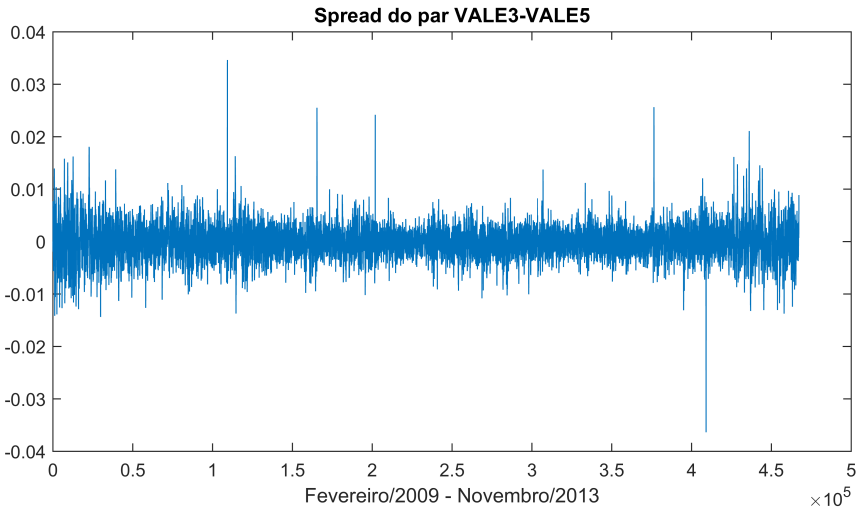


Figura 5.21: Série do *spread* (logaritmo) entre VALE3 e VALE5.

Na literatura, são encontrados diversos trabalhos que descrevem séries temporais financeiras através de um modelo de espaço de estados (SERMPINIS et al., 2012; NOBREGA; OLIVEIRA, 2013, 2014). De forma específica, o filtro de Kalman é utilizado para construir modelos preditivos para séries reversíveis à média, nas quais os valores observados apresentam um componente de ruído. São encontrados na área de Econometria diversos trabalhos que aplicam o filtro de Kalman para o problema de arbitragem estatística (AVELLANEDA; LEE, 2010), incluindo trabalhos voltados para o mercado brasileiro (CALDEIRA; MOURA, 2013).

Nesse estudo de caso, pretende-se estimar o valor do *spread* para a próxima observação

da série (previsão *one-steap-ahead*). A abordagem de aprendizagem sequencial, discutida no Capítulo 2, é adequada para o problema de arbitragem estatística, devido à natureza temporal das séries dos *spreads*. As regras de negociação baseadas em arbitragem estatística levam em consideração o valor da estimativa e a volatilidade, medida pelo desvio padrão do *spread*, para operar no mercado real comprando ou vendo simultaneamente os ativos do par. O estudo de caso apresentado nessa tese apresenta a avaliação do desempenho do método proposto em relação ao erro de previsão, medido pelo RMSE, além de discutir o desempenho relativo ao tempo de treinamento. Por uma questão de escopo da tese, não são discutidos aspectos relacionados com os retornos dos investimentos quando o modelo em estudo é aplicado no mundo real.

5.4.1 Bases de Dados

As bases de dados utilizadas nessa pesquisa foram geradas pelo processo de cointegração definido em ENGLE; GRANGER (1987), a partir de vinte séries de ativos negociados na BMF&Bovespa[16]. Antes de mostrar a definição formal de cointegração, precisamos rever o conceito de integração. Uma série temporal univariada $\mathbf{y}_t$ é integrada se o seu comportamento estacionário for definido por diferenciação. O número de diferenciações para que a série se torne estacionária define a sua ordem de integração. Séries com ordem d são denotadas por $I(d)$.

Formalmente, seja $\mathbf{Y}_t = (\mathbf{y}_{1t}, \ldots, \mathbf{y}_{nt})^T$ um vetor $(n \times 1)$ que denota um conjunto de séries temporais $I(1)$. $\mathbf{Y}_t$ é cointegrada se existe $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_n)^T$, em que $\boldsymbol{\beta}$ é um vetor $(n \times 1)$ e $\beta_1, \ldots, \beta_n$ são números reais diferentes de zero, tal que:

$$\boldsymbol{\beta}^T \mathbf{Y}_t = (\beta_1 \mathbf{y}_{1t} + \ldots + \beta_n \mathbf{y}_{nt}) \sim I(0). \tag{5.10}$$

Na teoria definida por Engle e Granger, $\boldsymbol{\beta}$ é chamado de vetor de cointegração. Podemos observar que a cointegração de duas ou mais séries não estacionárias ocorre quando é possível encontrar uma combinação linear de forma que a série resultante seja estacionária, ou $I(0)$. Nos experimentos do estudo de caso, as séries dos *spreads* entre dois ativos da BMF&Bovespa foram gerados por cointegração, de acordo com a seguinte relação linear entre as séries:

$$\log(\mathbf{x}_t) - \boldsymbol{\beta} \log(\mathbf{y}_t) = \mu + \varepsilon_t = \mathbf{S}, \tag{5.11}$$

em que $\mathbf{x}_t$ e $\mathbf{y}_t$ são as séries não estacionárias dos ativos, $\boldsymbol{\beta}$ é o vetor de cointegração entre $\mathbf{x}_t$ e $\mathbf{y}_t$, μ é a média que define o equilíbrio da relação entre $\mathbf{x}_t$ e $\mathbf{y}_t$ e ε_t é uma série temporal com média zero, representando o fator de pertubação do equilíbrio. A série cointegrada $\mathbf{S}$ forma o *spread* entre dois ativos financeiros. Para os experimentos do estudo de caso, o valor de $\boldsymbol{\beta}$ foi obtido por regressão linear. A Tabela 5.12 apresenta a lista de *spreads* gerados por cointegração.

Após a geração dos *spreads*, foram definidos os atributos de cada base. Dada uma observação da série no tempo t, os dez primeiros atributos são formados por atrasos de um a dez

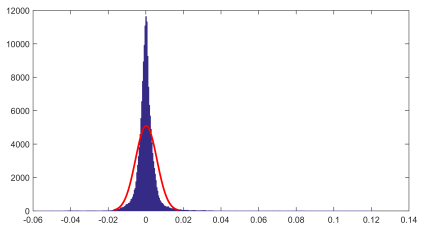
[16] www.bmfbovespa.com.br

Tabela 5.12: Spread de Pares de Ativos Gerados por Cointegração

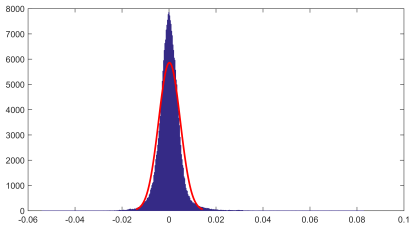
Série	Descrição
AMBV3-AMBV4	Companhia de Bebidas Das Americas ON/PN
BBDC3-BBDC4	Banco Bradesco ON/PN
BOVA11-PIBB11	Bovespa IShares / PIBB IShares
BRAP3-BRAP4	Bradesco Holding ON/PN
CMIG3-CMIG4	Companhia Energetica MG ON/PN
CPLE3-CPLE6	Companhia Paulista de Energia ON/PNA
GOAU4-GGBR4	Gerdau Metalurgica ON/PN
LAME3-LAME4	Lojas Americanas ON/PN
PETR3-PETR4	Petrobras ON/PN
VALE3-VALE5	Vale ON/PN

minutos da observação original, nos tempos $(t-1,...,t-10)$. De forma similar, os dez próximos atributos são as volatilidades, medidas pelo desvio padrão da série com atrasos de um a dez minutos. Os dez atributos seguintes são os atrasos das médias de todo o *spread*. Os últimos dez atributos são formados pelos atrasos das estimativas do tempo de reversão à média, fornecidos de acordo com a fórmula de decaimento exponencial da meia-vida do *spread* apresentada em BIBBONA; PANFILO; TAVELLA (2008). A Tabela 5.13 apresenta o resumo dos atributos iniciais de cada base. Na Figura 5.22 podemos observar as distribuições dos *spreads* para cada par de ativos cointegrados, com comportamento simétrico para todas as bases. Em comparação com as bases mostradas na Figura 5.14, podemos observar que existem diferenças em relação à simetria na distribuição dos *spreads*. Essas diferenças são explicadas pela natureza estacionária das séries geradas através de cointegração, onde a média é aproximadamente constante e próxima de zero.

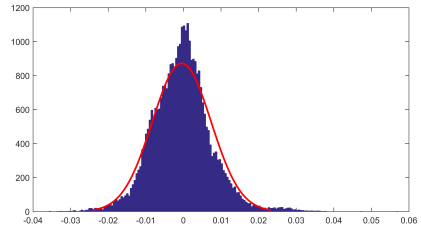
A simples geração dos *spreads* pelo processo descrito acima não garante estatisticamente que a série resultante seja cointegrada. Precisamos aplicar uma série de testes estatísticos para garantir que as séries de cada ativo seja não estacionária e que o *spread* resultante seja estacionário e com, pelo menos, um vetor de cointegração. Para os testes de estacionaridade aplicamos o método de Dickey–Fuller aumentado (SAID; DICKEY, 1984) com nível de confiança $\alpha = 0,05$. Uma breve descrição do teste pode ser vista no Apêndice B. Para os testes de cointegração, usamos a abordagem proposta por Johansen em JOHANSEN (1995) com o mesmo nível de confiança. Uma descrição mais detalhada desse método pode ser vista no Apêndice B. A Tabela 5.14 mostra um resumo dos testes de estacionaridade e cointegração. A segunda e terceira coluna mostram os *p-values* dos testes de estacionaridade para cada ativo individual. A quarta coluna apresenta o resultado do teste de estacionaridade para o *spread* gerado. As duas últimas colunas mostram o *p-value* do teste de Johansen e a sua respectiva estatística. A hipótese nula do teste de estacionaridade é que existe a presença de uma raiz unitária para a amostra da série temporal. A hipótese nula do teste de Johansen é que não existe nenhum vetor de cointegração para o *spread* das séries. Os valores em negrito denotam os casos nos quais a hipótese nula foi rejeitada.



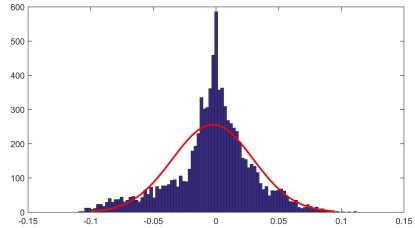
(a) AMBV3-AMBV4



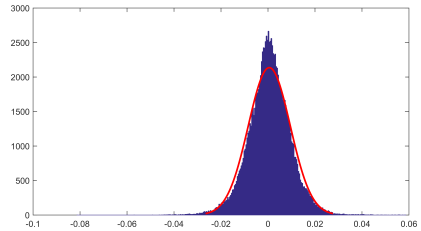
(b) BBDC3-BBDC4



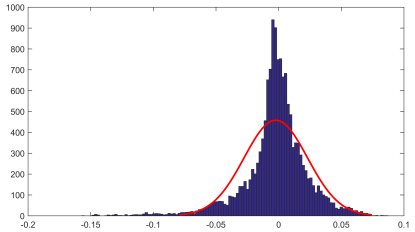
(c) BOVA11-PIBB11



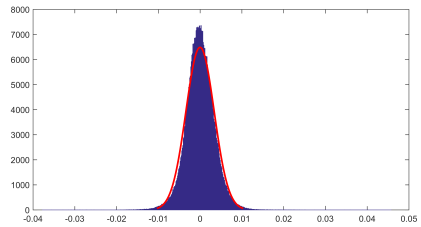
(d) BRAP3-BRAP4



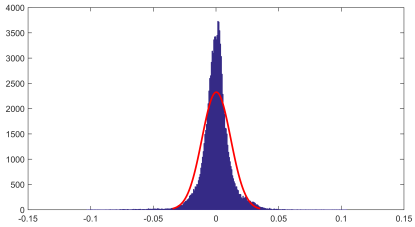
(e) CMIG3-CMIG4



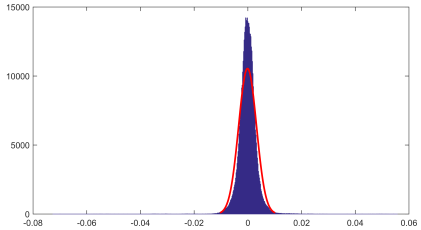
(f) CPLE3-CPLE6



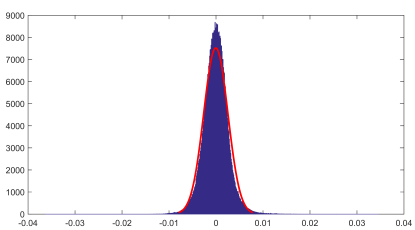
(g) GOAU4-GGBR4



(h) LAME3-LAME4



(i) PETR3-PETR4



(j) VALE3-VALE5

Figura 5.22: Histogramas dos *spreads* dos pares em escala logarítmica.

Tabela 5.13: Atributos das bases de dados

Número	Atributo	Lag
1-10	*Spread* do par	1-10
11-20	Volatilidade do *spread*	1-10
21-30	Média do *spread*	1-10
31-40	Estimativa da velocidade de reversão à média	1-10

Tabela 5.14: Testes de estacionaridade e cointegração,

Par	p-value Série 1	p-value Série 2	p-value Spread	p-value Cointegração	Estatística ($r = 0$)
AMBV3-AMBV4	0,2517	0,2264	**0,0010**	**0,0010**	43,9352
BBDC3-BBDC4	0,7653	0,6988	**0,0010**	**0,0010**	30,9862
BOVA11-PIBB11	0,6961	0,7653	**0,0010**	**0,0010**	56,6469
BRAP3-BRAP4	0,4581	0,5559	**0,0010**	**0,0010**	93,8376
CMIG3-CMIG4	0,4192	0,3486	**0,0010**	**0,0010**	36,0515
CPLE3-CPLE6	0,4780	0,6622	**0,0010**	**0,0010**	57,9678
GOAU4-GGBR4	0,5070	0,5426	**0,0010**	**0,0010**	340,9578
LAME3-LAME4	0,5951	0,7203	**0,0010**	**0,0010**	210,2902
PETR3-PETR4	0,3626	0,4464	**0,0010**	**0,0010**	63,7427
VALE3-VALE5	0,5500	0,5944	**0,0010**	**0,0010**	49,1316

5.4.2 Seleção de Atributos e Otimização dos Parâmetros

Como etapa prévia da execução dos experimentos, foi realizado um procedimento para reduzir a dimensionalidade das bases de dados através de um procedimento de seleção de atributos. O método utilizado avalia um subconjunto de atributos considerando a capacidade preditiva de cada atributo isolado em relação ao grau de redundância entre eles. Para escolher os subconjuntos de atributos que estão correlacionados com o valor medido do spread, foram calculadas as matrizes de correlação para cada subconjunto. Para explorar o espaço dos subconjuntos de atributos foi utilizado o algoritmo *Particle Swarm Optimization* (PSO) (KENNEDY, 2010), com o objetivo de encontrar os melhores subconjuntos. Uma breve descrição do PSO pode ser encontrada no Apêndice C. Os parâmetros usados na otimização do espaço de atributos foram definidos empiricamente: componente social, peso da inércia e os pesos individuais. O número de partículas foi definido em 20, com probabilidade de mutação igual a 0,01. O número máximo de iterações foi limitado a 200. O processo de seleção de atributos foi realizado no conjunto de treinamento, usando o conjunto de validação para selecionar os melhores resultados.

De forma semelhante ao procedimento mostrado na Subseção 5.1.2, os parâmetros de cada método foram otimizados usando a abordagem *simplex* de Nelder–Mead para encontrar o melhor valor de L que minimiza o erro de previsão medido no conjunto de validação. Uma breve descrição do método de Nelder–Mead pode ser vista no Apêndice A. Os parâmetros específicos de cada método foram otimizados utilizando o mesmo procedimento. A Tabela 5.15 mostra um resumo dos tamanhos dos conjuntos de treinamento, validação e teste, com partições 50%, 25% e

25%, respectivamente. A ordem temporal das observações foi preservada nos três conjuntos, com o objetivo de garantir a correlação entre as observações das séries. A última coluna apresenta o número final de atributos para cada base de dados, após a execução do procedimento de seleção de atributos.

Tabela 5.15: Divisão das bases de dados.

Série	Tamanho Total	Treinamento	Validação	Teste	Atributos
AMBV3-AMBV4	155126	77563	38781	38781	5
BBDC3-BBDC4	233127	116563	58281	58281	10
BOVA11-PIBB11	32688	16344	8172	8172	9
BRAP3-BRAP4	8357	4178	2089	2089	7
CMIG3-CMIG4	110365	55182	27591	27591	4
CPLE3-CPLE6	14036	7018	3509	3509	7
GOAU4-GGBR4	410099	205049	102524	102524	30
LAME3-LAME4	99964	49982	24991	24991	4
PETR3-PETR4	443201	221600	110800	110800	6
VALE3-VALE5	442375	221187	110593	110593	6

5.4.3 Avaliação Estatística dos Resultados

A aplicação do método proposto para o problema da arbitragem estatística pode ser visualmente observada através do exemplo mostrado na Figura 5.23, na qual a previsão do *spread* para o par VALE3-VALE5 é mostrada junto com os valores observados. De forma similar às séries temporais da Subseção 5.1.1, as previsões se ajustam quase que perfeitamente à curva dos valores observados. A Tabela 5.16 mostra os valores medidos do RMSE para todas as bases e o seu respectivo método, seguido pelo desvio padrão em parênteses. Os valores em negrito denotam os menores erros de previsão medidos para cada base de dados, sem levar em consideração a significância dos testes estatísticos sobre a diferença das médias. Observa-se que o KLM e o EKLM juntos alcançaram os menores erros de previsão em mais da metade dos experimentos, com o OS-ELM-TV conseguindo os menores valores em 30% dos experimentos.

Para validar estatisticamente os valores medidos para cada grupo de métodos, aplicamos o teste não paramétrico de Friedman para estabelecer um *ranking* de desempenho. A Tabela 5.17 mostra o *ranking* gerado para os erros de previsão com nível de confiança $\alpha = 0,05$. O valor da estatística χ^2 foi de 32,79, com *p-value* de 1,1492e-05, o que rejeita a hipótese nula de igualdade de desempenho entre os métodos. Em seguida, foram executados os testes *post hoc* para as diferenças de desempenho par a par. A Tabela 5.18 mostra os resultados das comparações para os métodos abordados. Os valores em negrito denotam os casos nos quais a hipótese nula de igualdade de desempenho foi rejeitada. A Figura 5.24 mostra uma visão consolidada dos

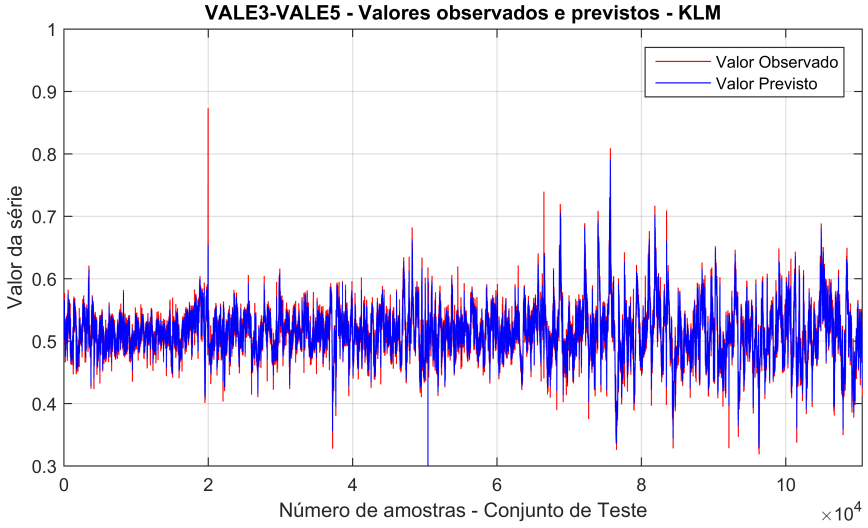


Figura 5.23: VALE3-VALE5 - Comparação entre os valores observados e previstos com o KLM.

Tabela 5.16: Erro de previsão para as bases de dados geradas por cointegração

Base	EKLM	KLM	OS-ELM	TOSELM	ReOS-ELM	OS-ELM-TV	LS-IELM
AMBV3-AMBV4 (RBF)	**0,00997** (0,00033)	**0,00997** (0,00033)	0,00999 (0,00032)	0,01001 (0,00039)	0,00998 (0,00033)	0,00998 (0,00042)	0,01005 (0,00035)
AMBV3-AMBV4 (Sig)	0,01007 (0,00047)	0,01007 (0,00047)	0,01006 (0,00046)	0,01037 (0,00131)	0,01005 (0,00042)	**0,01003** (0,00044)	0,01004 (0,00035)
BBDC3-BBDC4 (RBF)	0,01117 (0,00020)	0,01115 (0,00022)	**0,01114** (0,00018)	0,01118 (0,00021)	0,01116 (0,00018)	0,01116 (0,00020)	0,01116 (0,00021)
BBDC3-BBDC4 (Sig)	0,01116 (0,00017)	0,01113 (0,00020)	**0,01106** (0,00016)	0,01116 (0,00016)	0,01115 (0,00015)	0,01112 (0,00026)	0,01109 (0,00017)
BOVA11-PIBB11 (RBF)	0,02851 (0,00054)	0,02841 (0,00064)	0,02842 (0,00060)	0,02862 (0,00056)	0,02852 (0,00054)	**0,02827** (0,00066)	0,02849 (0,00064)
BOVA11-PIBB11 (Sig)	0,02840 (0,00059)	0,02821 (0,00057)	0,02844 (0,00068)	0,02844 (0,00059)	0,02839 (0,00058)	**0,02815** (0,00066)	0,02844 (0,00069)
BRAP3-BRAP4 (RBF)	0,04631 (0,00280)	**0,04584** (0,00148)	0,04611 (0,00204)	0,04667 (0,00399)	0,04630 (0,00276)	0,04764 (0,00648)	0,04630 (0,00134)
BRAP3-BRAP4 (Sig)	0,05056 (0,01119)	0,04637 (0,00417)	0,04665 (0,00580)	0,04823 (0,00752)	0,04868 (0,00766)	0,04669 (0,00463)	**0,04595** (0,00386)
CMIG3-CMIG4 (RBF)	0,01784 (0,00032)	**0,01781** (0,00034)	0,09473 (0,40831)	0,01789 (0,00040)	0,01782 (0,00028)	0,01791 (0,00033)	0,01795 (0,00055)
CMIG3-CMIG4 (Sig)	**0,01777** (0,00025)	**0,01777** (0,00025)	0,01939 (0,00323)	0,01784 (0,00028)	0,01835 (0,00265)	0,01791 (0,00032)	0,01789 (0,00021)
CPLE3-CPLE6 (RBF)	0,03288 (0,00371)	**0,03206** (0,00135)	0,03228 (0,00149)	0,03428 (0,00640)	0,03319 (0,00421)	0,03302 (0,00316)	0,03211 (0,00095)
CPLE3-CPLE6 (Sig)	0,03241 (0,00259)	**0,03188** (0,00135)	0,03341 (0,00473)	0,03385 (0,00557)	0,03214 (0,00191)	0,03346 (0,00534)	0,03242 (0,00338)
GOAU4-GGBR4 (RBF)	**0,01419** (0,00014)	0,01420 (0,00015)	0,01424 (0,00017)	0,01421 (0,00015)	0,01420 (0,00015)	**0,01419** (0,00014)	0,01420 (0,00015)
GOAU4-GGBR4 (Sig)	0,01405 (0,00011)	0,01407 (0,00013)	0,01405 (0,00011)	0,01407 (0,00013)	0,01407 (0,00013)	**0,01404** (0,00012)	0,01405 (0,00011)
LAME3-LAME4 (RBF)	0,01698 (0,00035)	0,01691 (0,00032)	0,01699 (0,00027)	0,01773 (0,00158)	0,01692 (0,00033)	**0,01688** (0,00029)	0,01694 (0,00037)
LAME3-LAME4 (Sig)	0,01788 (0,00258)	**0,01736** (0,00075)	0,01964 (0,00405)	0,01919 (0,00355)	0,01742 (0,00161)	0,01758 (0,00186)	0,01759 (0,00138)
PETR3-PETR4 (RBF)	**0,00689** (0,00011)	0,00690 (0,00012)	0,00690 (0,00011)	0,00693 (0,00012)	0,00692 (0,00012)	0,00695 (0,00015)	0,00691 (0,00011)
PETR3-PETR4 (Sig)	0,00692 (0,00011)	0,00689 (0,00009)	0,00695 (0,00022)	0,00691 (0,00018)	**0,00688** (0,00012)	0,00689 (0,00013)	0,00692 (0,00015)
VALE3-VALE5 (RBF)	0,01063 (0,00018)	**0,01059** (0,00012)	0,01062 (0,00014)	0,01063 (0,00013)	**0,01059** (0,00011)	0,01060 (0,00011)	0,01060 (0,00018)
VALE3-VALE5 (Sig)	**0,01056** (0,00013)	0,01060 (0,00013)	0,01061 (0,00015)	0,01064 (0,00015)	0,01060 (0,00014)	0,01059 (0,00013)	0,01059 (0,00011)

resultados, na qual o KLM apresenta diferenças de desempenho estatisticamente significantes em relação a quase todos os métodos, com exceção do OS-ELM-TV, no qual não foi possível confirmar a superioridade do método proposto nessa tese.

Tabela 5.17: Ranking do teste de Friedman para os erros de previsão - Estudo de Caso ($\chi^2 = 32,79$, p-value=1,1492e-05 e valor crítico=2,850)

Posição	Método	Valor do Ranking
1	KLM	2,33
2	OS-ELM-TV	3,50
3	LS-IELM	3,75
4	ReOS-ELM	3,78
5	EKLM	4,10
6	OS-ELM	4,50
7	TOSELM	6,05

Tabela 5.18: Comparação par a par do erro de previsão para o estudo de caso - Teste de Tukey com intervalo de confiança de 95%

Comparação	Estatística	Intervalo	p-value
EKLM - KLM	1,77	[0,44;3,11]	**0,01**
EKLM - OS-ELM	-0,40	[-1,74;0,94]	0,56
EKLM - TOSELM	-1,95	[-3,29;-0,61]	$< \mathbf{10^{-2}}$
EKLM - ReOS-ELM	0,32	[-1,01;1,66]	0,63
EKLM - OS-ELM-TV	0,60	[-0,74;1,94]	0,38
EKLM - LS-IELM	0,35	[-0,99;1,69]	0,61
KLM - OS-ELM	-2,17	[-3,51;-0,84]	$< \mathbf{10^{-2}}$
KLM - TOSELM	-3,72	[-5,06;-2,39]	$< \mathbf{10^{-2}}$
KLM - ReOS-ELM	-1,45	[-2,79;-0,11]	**0,03**
KLM - OS-ELM-TV	-1,17	[-2,51;0,16]	0,09
KLM - LS-IELM	-1,42	[-2,76;-0,09]	**0,04**
OS-ELM - TOSELM	-1,55	[-2,89;-0,21]	**0,02**
OS-ELM - ReOS-ELM	0,73	[-0,61;2,06]	0,29
OS-ELM - OS-ELM-TV	1,00	[-0,34;2,34]	0,14
OS-ELM - LS-IELM	0,75	[-0,59;2,09]	0,27
TOSELM - ReOS-ELM	2,27	[0,94;3,61]	$< \mathbf{10^{-2}}$
TOSELM - OS-ELM-TV	2,55	[1,21;3,89]	$< \mathbf{10^{-2}}$
TOSELM - LS-IELM	2,30	[0,96;3,64]	$< \mathbf{10^{-2}}$
ReOS-ELM - OS-ELM-TV	0,27	[-1,06;1,61]	0,69
ReOS-ELM - LS-IELM	0,02	[-1,31;1,36]	0,97
OS-ELM-TV - LS-IELM	-0,25	[-1,59;1,09]	0,71

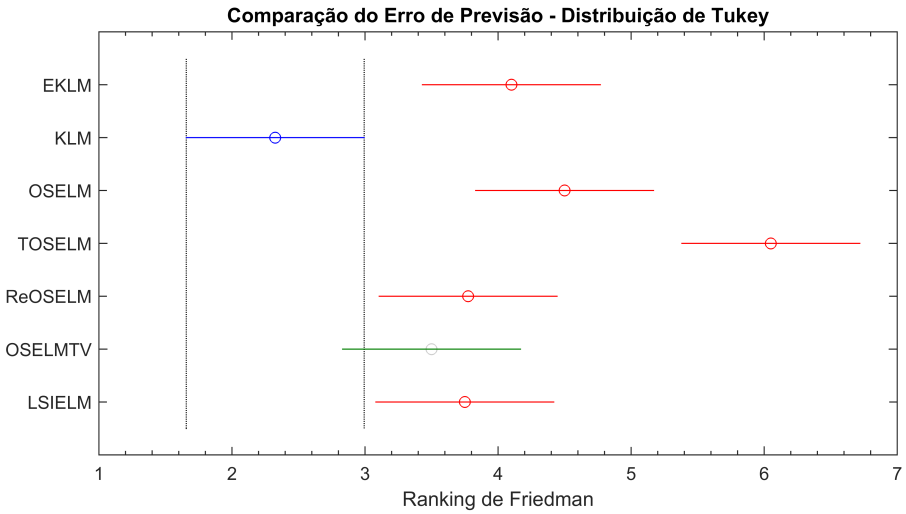


Figura 5.24: Intervalos de confiança de 95% com a distribuição de Tukey para o erro de previsão - Estudo de caso.

Em relação ao tempo médio de treinamento, o KLM obteve os melhores resultados em 80% dos experimentos, conforme mostrado na Tabela 5.19. Sem levar em consideração a significância dos testes, podemos afirmar que o desempenho do KLM em relação aos métodos relacionados não degrada quando aumenta o tamanho das bases usadas na fase sequencial do treinamento. A Figura 5.25 mostra que os *outliers* da previsão do erro do KLM estão próximos numericamente dos valores obtidos da mediana.

Tabela 5.19: Tempo médio de treinamento para as bases de dados geradas por cointegração

Base	EKLM	KLM	OS-ELM	TOSELM	ReOS-ELM	OS-ELM-TV	LS-IELM
AMBV3-AMBV4 (RBF)	23,20 (1,35)	**12,93** (0,12)	13,14 (0,13)	120,06 (1,73)	17,68 (2,51)	25,33 (0,43)	29,26 (1,17)
AMBV3-AMBV4 (Sig)	11,67 (0,18)	4,74 (0,17)	**4,18** (0,03)	110,51 (1,13)	4,77 (0,11)	23,95 (0,27)	9,16 (0,48)
BBDC3-BBDC4 (RBF)	34,67 (0,29)	20,31 (0,20)	**20,11** (0,07)	262,14 (4,74)	24,40 (0,34)	55,60 (2,77)	44,62 (2,00)
BBDC3-BBDC4 (Sig)	17,67 (0,24)	**5,58** (0,05)	6,30 (0,04)	245,98 (2,35)	7,21 (0,13)	52,81 (1,10)	13,83 (0,66)
BOVA11-PIBB11 (RBF)	17,65 (0,30)	4,60 (0,11)	5,84 (0,13)	15,84 (0,43)	13,63 (0,27)	**3,58** (0,05)	14,64 (0,25)
BOVA11-PIBB11 (Sig)	7,19 (0,17)	**0,90** (0,02)	2,06 (0,13)	12,36 (0,35)	3,32 (0,15)	3,27 (0,09)	5,03 (0,15)
BRAP3-BRAP4 (RBF)	1,25 (0,05)	**0,70** (0,01)	0,72 (0,02)	2,10 (0,15)	0,87 (0,05)	0,80 (0,03)	1,59 (0,13)
BRAP3-BRAP4 (Sig)	0,64 (0,03)	**0,20** (0,01)	0,23 (0,01)	1,57 (0,13)	0,26 (0,01)	0,73 (0,04)	0,56 (0,07)
CMIG3-CMIG4 (RBF)	14,30 (0,21)	**7,81** (0,03)	7,93 (0,04)	70,35 (0,89)	9,61 (0,16)	13,57 (0,54)	18,16 (0,77)
CMIG3-CMIG4 (Sig)	7,91 (0,16)	**2,52** (0,02)	2,71 (0,02)	64,96 (0,99)	3,13 (0,07)	12,49 (0,50)	5,89 (0,39)
CPLE3-CPLE6 (RBF)	2,08 (0,07)	**1,16** (0,02)	1,20 (0,02)	4,32 (0,23)	1,46 (0,04)	1,37 (0,09)	2,69 (0,21)
CPLE3-CPLE6 (Sig)	1,06 (0,04)	**0,34** (0,01)	0,38 (0,02)	3,50 (0,21)	0,43 (0,02)	1,22 (0,07)	0,86 (0,12)
GOAU4-GGBR4 (RBF)	271,99 (3,35)	**66,90** (0,16)	90,28 (0,94)	1284,58 (6,01)	242,63 (1,37)	359,48 (20,95)	227,14 (1,23)
GOAU4-GGBR4 (Sig)	107,08 (1,78)	**13,15** (0,05)	30,82 (0,79)	1227,71 (4,62)	77,00 (0,63)	355,71 (34,65)	77,55 (0,75)
LAME3-LAME4 (RBF)	13,61 (0,55)	**6,94** (0,03)	7,19 (0,03)	60,62 (0,96)	8,71 (0,16)	11,67 (0,50)	16,43 (0,47)
LAME3-LAME4 (Sig)	7,36 (0,33)	**2,33** (0,02)	2,45 (0,02)	55,59 (0,81)	2,85 (0,10)	10,48 (0,20)	5,37 (0,39)
PETR3-PETR4 (RBF)	57,11 (0,49)	**31,23** (0,37)	32,04 (0,41)	1476,70 (34,54)	38,73 (0,32)	134,39 (3,29)	71,66 (1,10)
PETR3-PETR4 (Sig)	31,55 (0,27)	**10,18** (0,18)	10,90 (0,04)	1435,20 (11,02)	12,62 (0,20)	136,47 (12,25)	24,23 (0,92)
VALE3-VALE5 (RBF)	72,09 (0,40)	**41,09** (0,63)	43,41 (0,61)	1474,61 (22,91)	52,62 (0,45)	177,81 (10,35)	92,94 (3,37)
VALE3-VALE5 (Sig)	35,84 (2,68)	**10,89** (0,23)	12,68 (0,04)	1428,67 (5,80)	14,20 (0,17)	171,78 (5,57)	27,97 (0,92)

O *ranking* gerado pela aplicação do teste de Friedman para os valores do tempo médio de treinamento pode ser visto na Tabela 5.20. A estatística χ^2 medida no teste foi de 103,22, com p-value igual a 5,3337e-20 para o nível de confiança $\alpha = 0,05$, o que rejeita a hipótese nula de igualdade de desempenho entre os métodos. Os resultados dos testes *post hoc* para o tempo de treinamento podem ser vistos na Tabela 5.21, com os valores em negrito denotando os casos

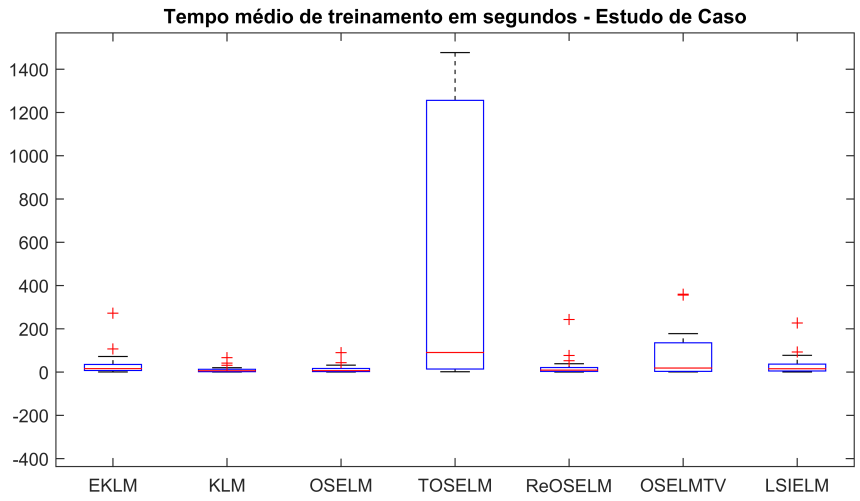


Figura 5.25: Tempo de treinamento para os métodos avaliados no estudo de caso (*box-plot*).

nos quais a hipótese nula de igualdade de desempenho foi rejeitada. A Figura 5.26 apresenta uma visão consolidada dos testes. Observa-se que o KLM apresenta diferenças estatisticamente significativas para o tempo médio de treinamento em comparação a quase todos os métodos abordados, com exceção da OS-ELM.

Tabela 5.20: Ranking do teste de Friedman para os tempos médios de treinamento - Estudo de Caso ($\chi^2 = 103,22$, p-value=5,3337e-20 e valor crítico=2,850)

Posição	Método	Valor do Ranking
1	KLM	1,15
2	OS-ELM	1,95
3	ReOS-ELM	3,25
4	LS-IELM	4,70
5	EKLM	4,95
6	OS-ELM-TV	5,05
7	TOSELM	6,95

5.5 Considerações Finais

Esse capítulo apresentou os resultados experimentais da aplicação dos algoritmos do método proposto em quinze bases de dados com tamanhos e características diferentes. As bases voltadas para problemas de previsão de séries temporais foram geradas artificialmente e as

Tabela 5.21: Comparação par a par do tempo médio de treinamento para o estudo de caso - Teste de Tukey com intervalo de confiança de 95%

Comparação	Estatística	Intervalo	p-value
EKLM - KLM	3,80	[2,46;5,14]	$< \mathbf{10^{-2}}$
EKLM - OS-ELM	3,00	[1,66;4,34]	$< \mathbf{10^{-2}}$
EKLM - TOSELM	-2,00	[-3,34;-0,66]	$< \mathbf{10^{-2}}$
EKLM - ReOS-ELM	1,70	[0,36;3,04]	**0,01**
EKLM - OS-ELM-TV	-0,10	[-1,44;1,24]	0,88
EKLM - LS-IELM	0,25	[-1,09;1,59]	0,71
KLM - OS-ELM	-0,80	[-2,14;0,54]	0,24
KLM - TOSELM	-5,80	[-7,14;-4,46]	$< \mathbf{10^{-2}}$
KLM - ReOS-ELM	-2,10	[-3,44;-0,76]	$< \mathbf{10^{-2}}$
KLM - OS-ELM-TV	-3,90	[-5,24;-2,56]	$< \mathbf{10^{-2}}$
KLM - LS-IELM	-3,55	[-4,89;-2,21]	$< \mathbf{10^{-2}}$
OS-ELM - TOSELM	-5,00	[-6,34;-3,66]	$< \mathbf{10^{-2}}$
OS-ELM - ReOS-ELM	-1,30	[-2,64;0,04]	0,06
OS-ELM - OS-ELM-TV	-3,10	[-4,44;-1,76]	$< \mathbf{10^{-2}}$
OS-ELM - LS-IELM	-2,75	[-4,09;-1,41]	$< \mathbf{10^{-2}}$
TOSELM - ReOS-ELM	3,70	[2,36;5,04]	$< \mathbf{10^{-2}}$
TOSELM - OS-ELM-TV	1,90	[0,56;3,24]	**0,01**
TOSELM - LS-IELM	2,25	[0,91;3,59]	$< \mathbf{10^{-2}}$
ReOS-ELM - OS-ELM-TV	-1,80	[-3,14;-0,46]	**0,01**
ReOS-ELM - LS-IELM	-1,45	[-2,79;-0,11]	**0,03**
OS-ELM-TV - LS-IELM	0,35	[-0,99;1,69]	0,61

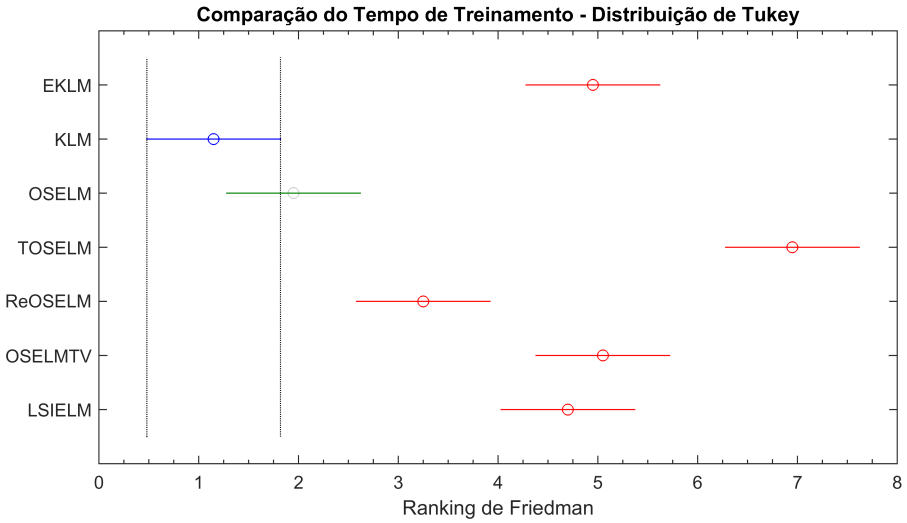


Figura 5.26: Intervalos de confiança de 95% com a distribuição de Tukey para o tempo de treinamento - Estudo de caso.

demais foram obtidas em repositórios públicos. A análise visual dos valores previstos mostra que o método proposto consegue eliminar o ruído presente dos dados de treinamento e se ajustar de forma eficiente à curva dos dados observados.

Os resultados dos testes estatísticos mostraram que tanto pelo critério do erro de previsão quanto pela precisão das estimativas, o método proposto consegue um desempenho superior à maioria dos trabalhos relacionados. De forma semelhante, a análise estatística do tempo médio de treinamento mostra que o KLM consegue reduzir o custo computacional da fase de aprendizagem sequencial, com desempenho superior em todas as bases de dados do primeiro grupo de experimentos. A versão não linear do método proposto sempre obteve resultados semelhantes à versão linear, quando comparados os erros de previsão, não sendo encontradas diferenças estatisticamente significativas nesse aspecto. Entretanto, a complexidade computacional do cálculo das matrizes Jacobianas do EKLM aumenta de forma considerável o tempo médio de treinamento, quando comparado com os métodos relacionados.

Um estudo de caso voltado para a previsão de séries temporais foi realizado, com resultados semelhantes aos reportados para o primeiro grupo de experimentos. O problema de previsão para a arbitragem estatística foi abordado e a aplicação do método proposto nessa tese apresentou desempenho superior aos trabalhos relacionados nos aspectos relativos aos erros de previsão e tempos de treinamento.

6

Conclusão e Trabalhos Futuros

Essa tese introduziu duas versões de um novo método para o aprendizado sequencial em problemas de regressão e previsão de séries temporais: *Kalman Learning Machine* (KLM) e *Extended Kalman Learning Machine* (EKLM). A primeira versão é capaz de atualizar sequencialmente os pesos de saída da rede através de um modelo de espaço de estados baseado no filtro de Kalman. A segunda versão pode ser aplicada em problemas nos quais a dinâmica do sistema seja não linear, com a atualização dos pesos de saída sendo realizada através da versão estendida do filtro de Kalman. Baseado na versão sequencial da ELM, essa tese propôs o mapeamento desses algoritmos em equações de estado e de medição. Foram apresentadas alternativas de baixo custo computacional para o cálculo analítico dos parâmetros do filtro de Kalman.

O método proposto preenche algumas lacunas encontradas nos trabalhos relacionados, incluindo o tratamento dos efeitos da multicolinearidade na ELM através da filtragem do ruído, a independência da distribuição do conjunto de treinamento, a otimização de um único parâmetro para os algoritmos propostos e a redução do custo computacional na atualização dos pesos de saída da rede.

Para validar o método proposto, foram executadas duas séries de experimentos utilizando bases de dados geradas artificialmente e outras obtidas em repositórios públicos (BACHE; LICHMAN, 2013). A primeira série de experimentos consistiu em aplicar o método proposto em quinze bases distintas, alternando a função de ativação utilizada na rede. No total foram realizados trinta experimentos com cinquenta repetições para o método proposto. A mesma carga de experimentos foi realizada para outros cinco métodos com características semelhantes e os resultados foram medidos e analisados. A avaliação dos experimentos consistiu em comparar o desempenho de todos os métodos em relação ao erro médio de previsão gerado, à precisão das estimativas e ao tempo médio de treinamento. Considerando o erro de previsão, o método proposto apresentou desempenho superior a quatro dos cinco métodos relacionados. Em relação à precisão das estimativas, a versão linear do método obteve melhores resultados que quatro dos cinco métodos comparados. Do ponto de vista do tempo médio de treinamento, o KLM obteve melhor desempenho que todos os demais métodos. Os resultados da aplicação do método

proposto e dos métodos relacionados foram comparados estatisticamente, com os resultados dos testes indicando diferenças significativas de desempenho entre eles.

Um estudo de caso foi realizado, aplicando o método proposto ao problema da arbitragem estatística. Esse estudo gerou dez bases de dados de séries temporais através do processo de cointegração. As séries foram obtidas através da coleta de preços de ativos financeiros negociados na bolsa de valores. Os resultados dos experimentos mostraram que a versão linear do método proposto alcançou os menores erros de previsão, com desempenho estatisticamente superior que quatro dos cinco métodos analisados. O mesmo nível de desempenho foi observado em relação ao tempo de treinamento, no qual o KLM obteve os menores valores quando comparado com os trabalhos relacionados.

As principais contribuições desse trabalho podem ser resumidas na lista a seguir:

- O mapeamento do algoritmo ELM em um modelo de espaço de estados e a definição das suas equações de estado e medição;
- Um novo método de aprendizagem baseado na versão sequencial da ELM com atualização dos pesos através da versão linear do filtro de Kalman (algoritmo KLM);
- Uma extensão do método para problemas nos quais a natureza do sistema é não linear (algoritmo EKLM);
- Uma proposta alternativa para o cálculo dos parâmetros do filtro através de discretização matricial;
- Um estudo comparativo entre os principais métodos relacionados com o tópico da tese.

6.1 Trabalhos Publicados

Publicações em revistas:

- Nobrega, J. P., Oliveira, A. L. I., Kalman Filter-based Method for Online Sequential Extreme Learning Machine for Regression Problems. Engineering Applications of Artificial Intelligence, [S.l.], v.44, p.101–110, 2015.

Publicações em conferências:

- Nobrega, J. P., Oliveira, A. L. I., Improving the Statistical Arbitrage Strategy in Intraday Trading by Combining Extreme Learning Machine and Support Vector Regression with Linear Regression Models. In: IEEE 25th International Conference on Tools with Artificial Intelligence (ICTAI), p.182–188, 2013.

- Nobrega, J. P., Oliveira, A. L. I., A combination forecasting model using machine learning and Kalman filter for statistical arbitrage. In: IEEE International Conference on Systems, Man and Cybernetics (SMC), p.1294–1299, 2014.

6.2 Proposta de Trabalhos Futuros

Essa seção propõe alguns temas para trabalhos futuros usando os algoritmos KLM e EKLM. O espaço de pesquisa do tópico inclui os seguintes temas:

- **Abordagem construtiva.** Um ponto em comum dos algoritmos de aprendizagem sequencial é a necessidade de otimização do número de unidades da camada escondida. Conforme discutido no Capítulo 5, a escolha incorreta desse valor induz à degradação no desempenho do método proposto. Um ponto a ser investigado em uma nova pesquisa é a adoção de uma abordagem construtiva para a arquitetura da rede. Usando um critério de significância, como por exemplo, o adotado no algoritmo GGAP-RBF (HUANG; SARATCHANDRAN; SUNDARARAJAN, 2004), pode-se incluir um mecanismo dinâmico de alocação e poda de unidades à camada escondida. Essa extensão, em tese, poderia resolver o problema da escolha *a priori* do número de unidades da camada escondida.

- **Atualização dinâmica de parâmetros.** Uma característica comum aos modelos baseados no filtro de Kalman é a estimativa das matrizes de covariância do estado e da medição antes do início da fase sequencial. Esses parâmetros do filtro permanecem constantes ao longo do processo de aprendizado. Uma extensão do método proposto poderá estudar mecanismos de atualização dinâmica das covariâncias do estado e da medição, usando a distribuição dos dados fornecidos até uma certa unidade de tempo. Outra alternativa para correção dinâmica das matrizes é a utilização de filtros adaptativos com ajuste de parâmetros através de processos de otimização (FARHANG-BOROUJENY, 2013). O objetivo é investigar o impacto da correção dinâmica das matrizes no erro de previsão. Uma abordagem similar foi proposta em XI; PENG; CHEN (2014), com os resultados reportados indicando desempenho superior a outras técnicas de aprendizagem sequencial.

- **Mecanismos de esquecimento.** Na execução dos experimentos dessa tese, os dados foram fornecidos um a um, de forma sequencial. No entanto, uma característica comum aos métodos baseados na OS-ELM é a capacidade de aprender sequencialmente através de blocos de dados de tamanhos distintos. Conforme mostrado em ZHAO; WANG; PARK (2012), em problemas nos quais os padrões dos dados mudam com o tempo, um subconjunto de um certo bloco de dados poderá representar um padrão “desatualizado”. Uma possível extensão ao método proposto nessa tese

poderia estudar a incorporação de mecanismos de esquecimento para dados desatualizados, descantando-os do treinamento ou atribuindo pesos mais baixos que os dados mais "recentes". Uma hipótese a ser testada é se a adoção desses mecanismos melhora o desempenho do método proposto em relação ao erro de previsão e tempo de treinamento.

- ***Ensemble* do KLM.** Os resultados dos experimentos reportados em NOBREGA; OLIVEIRA (2013, 2014) mostraram que a combinação de diferentes modelos preditivos baseados no filtro de Kalman reduz o erro de previsão. Uma nova pesquisa sobre o tema poderia incluir a definição de um modelo de *ensemble* para o KLM e EKLM, com a combinação das previsões sendo realizadas por um modelo de regressão baseado no filtro de Kalman.

- **KLM/EKLM *auto-encoder*.** Recentemente, foram propostas extensões do algoritmo original da ELM para problemas de aprendizagem profunda (*deep learning*) (YU et al., 2015). Um tema a ser explorado como extensão dessa tese é a possível capacidade do KLM e EKLM em aprender representações profundas dos dados de entrada, através da definição de uma arquitetura multicamada para os algoritmos propostos e o empilhamento de *autoencoders* baseados no KLM e EKLM.

- **Problemas de classificação.** Os experimentos realizados com o método proposto ficaram restritos ao escopo de problemas de regressão e de previsão de séries temporais. Entretanto, é perfeitamente possível estender o KLM e EKLM para aplicação em problemas genéricos de classificação. As versões dos algoritmos implementadas nessa tese já suportam tarefas de classificação para duas ou mais classes.

Referências

AKBILGIC, O.; BOZDOGAN, H.; BALABAN, M. A novel Hybrid RBF Neural Networks model as a forecaster. **Statistics and Computing**, [S.l.], v.24, n.3, p.365–375, 2014.

ARASARATNAM, I.; HAYKIN, S. Cubature kalman filters. **Automatic Control, IEEE Transactions on**, [S.l.], v.54, n.6, p.1254–1269, 2009.

AVELLANEDA, M.; LEE, J.-H. Statistical arbitrage in the US equities market. **Quantitative Finance**, [S.l.], v.10, n.7, p.761–782, 2010.

BACHE, K.; LICHMAN, M. UCI machine learning repository. **URL http://archive. ics. uci. edu/ml**, [S.l.], v.901, 2013.

BARRON, A. R. Universal approximation bounds for superpositions of a sigmoidal function. **Information Theory, IEEE Transactions on**, [S.l.], v.39, n.3, p.930–945, 1993.

BARTLETT, P. L. The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network. **Information Theory, IEEE Transactions on**, [S.l.], v.44, n.2, p.525–536, 1998.

BELSLEY, D. A.; KUH, E.; WELSCH, R. E. **Regression diagnostics**: identifying influential data and sources of collinearity. [S.l.]: John Wiley & Sons, 2005. v.571.

BIBBONA, E.; PANFILO, G.; TAVELLA, P. The Ornstein–Uhlenbeck process as a model of a low pass filtered white noise. **Metrologia**, [S.l.], v.45, n.6, p.S117, 2008.

BLOM, H. A.; BAR-SHALOM, Y. The interacting multiple model algorithm for systems with Markovian switching coefficients. **Automatic Control, IEEE Transactions on**, [S.l.], v.33, n.8, p.780–783, 1988.

BLUM, K. **Density matrix theory and applications**. [S.l.]: Springer, 2012. v.64.

BORS, A. G.; GABBOUJ, M. Minimal topology for a radial basis functions neural network for pattern classification. **Digital Signal Processing**, [S.l.], v.4, n.3, p.173–188, 1994.

BREWER, J. W. Kronecker products and matrix calculus in system theory. **Circuits and Systems, IEEE Transactions on**, [S.l.], v.25, n.9, p.772–781, 1978.

BUZA, K. Feedback prediction for blogs. In: **Data Analysis, Machine Learning and Knowledge Discovery**. [S.l.]: Springer, 2014. p.145–152.

CALDEIRA, J.; MOURA, G. V. Selection of a portfolio of pairs based on cointegration: a statistical arbitrage strategy. **Social Science Research Network**, [S.l.], 2013.

CAMPOLUCCI, P. et al. On-line learning algorithms for locally recurrent neural networks. **Neural Networks, IEEE Transactions on**, [S.l.], v.10, n.2, p.253–271, 1999.

CANDY, J. V. **Model-based signal processing**. [S.l.]: John Wiley & Sons, 2005. v.36.

CHEN, S.; BILLINGS, S.; GRANT, P. Recursive hybrid algorithm for non-linear system identification using radial basis function networks. **International Journal of Control**, [S.l.], v.55, n.5, p.1051–1070, 1992.

CHOY, M. C.; SRINIVASAN, D.; CHEU, R. L. Neural networks for continuous online learning and control. **Neural Networks, IEEE Transactions on**, [S.l.], v.17, n.6, p.1511–1531, 2006.

CORTEZ, P. et al. Modeling wine preferences by data mining from physicochemical properties. **Decision Support Systems**, [S.l.], v.47, n.4, p.547–553, 2009.

CYBENKO, G. Approximation by superpositions of a sigmoidal function. **Mathematics of control, signals and systems**, [S.l.], v.2, n.4, p.303–314, 1989.

DEMŠAR, J. Statistical comparisons of classifiers over multiple data sets. **The Journal of Machine Learning Research**, [S.l.], v.7, p.1–30, 2006.

DUDA, R. O.; HART, P. E.; STORK, D. G. **Pattern classification**. [S.l.]: John Wiley & Sons, 2012.

EL-SOUSY, F. F. Robust wavelet-neural-network sliding-mode control system for permanent magnet synchronous motor drive. **IET electric power applications**, [S.l.], v.5, n.1, p.113–132, 2011.

ENGLE, R. F.; GRANGER, C. W. Co-integration and error correction: representation, estimation, and testing. **Econometrica: Journal of the Econometric Society**, [S.l.], p.251–276, 1987.

ER, M.; ZHAI, L.; LI, X. A Hybrid Online Sequential Extreme Learning Machine with Simplified Hidden Network. **International Journal of Computer Science**, [S.l.], v.39, n.1, 2012.

FANAEE-T, H.; GAMA, J. Event labeling combining ensemble detectors and background knowledge. **Progress in Artificial Intelligence**, [S.l.], v.2, n.2-3, p.113–127, 2014.

FARHANG-BOROUJENY, B. **Adaptive filters**: theory and applications. [S.l.]: John Wiley & Sons, 2013.

FRIEDMAN, M. A comparison of alternative tests of significance for the problem of m rankings. **The Annals of Mathematical Statistics**, [S.l.], v.11, n.1, p.86–92, 1940.

FULLER, W. A. **Introduction to statistical time series**. [S.l.]: John Wiley & Sons, 2009. v.428.

FUNAHASHI, K.-I. On the approximate realization of continuous mappings by neural networks. **Neural networks**, [S.l.], v.2, n.3, p.183–192, 1989.

GOLUB, G. H.; VAN LOAN, C. F. **Matrix computations**. [S.l.]: JHU Press, 2012. v.3.

GONZÁLEZ, A.; DORRONSORO, J. R. Natural conjugate gradient training of multilayer perceptrons. **Neurocomputing**, [S.l.], v.71, n.13, p.2499–2506, 2008.

GRIGORIEVSKIY, A. et al. Long-term time series prediction using OP-ELM. **Neural Networks**, [S.l.], v.51, p.50–56, 2014.

GU, Y. et al. TOSELM: timeliness online sequential extreme learning machine. **Neurocomputing**, [S.l.], v.128, p.119–127, 2014.

GUO, L.; HAO, J.-h.; LIU, M. An incremental extreme learning machine for online sequential learning problems. **Neurocomputing**, [S.l.], v.128, p.50–58, 2014.

GUPTA, M.; JIN, L.; HOMMA, N. **Static and dynamic neural networks**: from fundamentals to advanced theory. [S.l.]: John Wiley & Sons, 2004.

HAN, F.; HUANG, D.-S. Improved extreme learning machine for function approximation by encoding a priori information. **Neurocomputing**, [S.l.], v.69, n.16, p.2369–2373, 2006.

HARVEY, A. C. **Forecasting, structural time series models and the Kalman filter**. [S.l.]: Cambridge University Press, 1990.

HINTON, G. et al. Deep neural networks for acoustic modeling in speech recognition: the shared views of four research groups. **Signal Processing Magazine, IEEE**, [S.l.], v.29, n.6, p.82–97, 2012.

HO, K. I.; LEUNG, C.-S.; SUM, J. Convergence and objective functions of some fault/noise-injection-based online learning algorithms for RBF networks. **Neural Networks, IEEE Transactions on**, [S.l.], v.21, n.6, p.938–947, 2010.

HORNIK, K. Approximation capabilities of multilayer feedforward networks. **Neural networks**, [S.l.], v.4, n.2, p.251–257, 1991.

HORNIK, K.; STINCHCOMBE, M.; WHITE, H. Multilayer feedforward networks are universal approximators. **Neural networks**, [S.l.], v.2, n.5, p.359–366, 1989.

HUANG, G.-B. An insight into extreme learning machines: random neurons, random features and kernels. **Cognitive Computation**, [S.l.], v.6, n.3, p.376–390, 2014.

HUANG, G.-B.; BABRI, H. A. Upper bounds on the number of hidden neurons in feedforward networks with arbitrary bounded nonlinear activation functions. **Neural Networks, IEEE Transactions on**, [S.l.], v.9, n.1, p.224–229, 1998.

HUANG, G.-B.; CHEN, L. Convex incremental extreme learning machine. **Neurocomputing**, [S.l.], v.70, n.16, p.3056–3062, 2007.

HUANG, G.-B.; CHEN, L. Enhanced random search based incremental extreme learning machine. **Neurocomputing**, [S.l.], v.71, n.16, p.3460–3468, 2008.

HUANG, G.-B.; CHEN, L.; SIEW, C.-K. Universal approximation using incremental constructive feedforward networks with random hidden nodes. **Neural Networks, IEEE Transactions on**, [S.l.], v.17, n.4, p.879–892, 2006.

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

HUANG, G.-B.; SARATCHANDRAN, P.; SUNDARARAJAN, N. A generalized growing and pruning RBF (GGAP-RBF) neural network for function approximation. **Neural Networks, IEEE Transactions on**, [S.l.], v.16, n.1, p.57–67, 2005.

HUANG, G.-B.; ZHU, Q.-Y.; SIEW, C.-K. Extreme learning machine: theory and applications. **Neurocomputing**, [S.l.], v.70, n.1, p.489–501, 2006.

HUANG, G. et al. Trends in extreme learning machines: a review. **Neural Networks**, [S.l.], v.61, p.32–48, 2015.

HUANG, S.-C.; HUANG, Y.-F. Bounds on the number of hidden neurons in multilayer perceptrons. **Neural Networks, IEEE Transactions on**, [S.l.], v.2, n.1, p.47–55, 1991.

HUYNH, H. T.; WON, Y. Regularized online sequential learning algorithm for single-hidden layer feedforward neural networks. **Pattern Recognition Letters**, [S.l.], v.32, n.14, p.1930–1935, 2011.

ITO, K.; XIONG, K. Gaussian filters for nonlinear filtering problems. **Automatic Control, IEEE Transactions on**, [S.l.], v.45, n.5, p.910–927, 2000.

JAIN, L. C. et al. A review of online learning in supervised neural networks. **Neural Computing and Applications**, [S.l.], v.25, n.3-4, p.491–509, 2014.

JAZWINSKI, A. H. **Stochastic processes and filtering theory**. [S.l.]: Courier Corporation, 2007.

JOHANSEN, S. Likelihood-based inference in cointegrated vector autoregressive models. **OUP Catalogue**, [S.l.], 1995.

JULIER, S. J.; UHLMANN, J. K.; DURRANT-WHYTE, H. F. A new approach for filtering nonlinear systems. In: AMERICAN CONTROL CONFERENCE. **Anais...** [S.l.: s.n.], 1995. v.3, p.1628–1632.

JULIER, S.; UHLMANN, J. Unscented filtering and nonlinear estimation. **Proceedings of the IEEE**, [S.l.], v.92, n.3, p.401–422, 2004.

JUNG, S.; KIM, S. S. Control experiment of a wheel-driven mobile inverted pendulum using neural network. **Control Systems Technology, IEEE Transactions on**, [S.l.], v.16, n.2, p.297–303, 2008.

KADIRKAMANATHAN, V.; NIRANJAN, M. A function estimation approach to sequential learning with neural networks. **Neural Computation**, [S.l.], v.5, n.6, p.954–975, 1993.

KAILATH, T. **Lectures on Kalman and Wiener Filtering Theory**. [S.l.]: New York: Springer-Verlag, 1981.

KALMAN, R. E. A new approach to linear filtering and prediction problems. **Journal of Fluids Engineering**, [S.l.], v.82, n.1, p.35–45, 1960.

KARTALOPOULOS, S. V.; KARTAKAPOULOS, S. V. **Understanding neural networks and fuzzy logic**: basic concepts and applications. [S.l.]: Wiley-IEEE Press, 1997.

KEARNS, M.; NEVMYVAKA, Y. Machine learning for market microstructure and high frequency trading. **High-Frequency Trading–New Realities for Traders, Markets and Regulators**, [S.l.], p.91–124, 2013.

KENNEDY, J. Particle swarm optimization. In: **Encyclopedia of Machine Learning**. [S.l.]: Springer, 2010. p.760–766.

KWOK, T.-Y.; YEUNG, D.-Y. Objective functions for training new hidden units in constructive neural networks. **Neural Networks, IEEE Transactions on**, [S.l.], v.8, n.5, p.1131–1148, 1997.

LAN, Y.; SOH, Y. C.; HUANG, G.-B. Ensemble of online sequential extreme learning machine. **Neurocomputing**, [S.l.], v.72, n.13, p.3391–3395, 2009.

LEE, K.-M.; STREET, W. N. An adaptive resource-allocating network for automated detection, segmentation, and classification of breast cancer nuclei topic area: image processing and recognition. **Neural Networks, IEEE Transactions on**, [S.l.], v.14, n.3, p.680–687, 2003.

LESHNO, M. et al. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. **Neural networks**, [S.l.], v.6, n.6, p.861–867, 1993.

LI, G.; LIU, M.; DONG, M. A new online learning algorithm for structure-adjustable extreme learning machine. **Computers & Mathematics with Applications**, [S.l.], v.60, n.3, p.377–389, 2010.

LI, G.; NIU, P. An enhanced extreme learning machine based on ridge regression for regression. **Neural Computing and Applications**, [S.l.], v.22, n.3-4, p.803–810, 2013.

LI, Y. et al. Robust neuro controller design for aircraft auto-landing. **Aerospace and Electronic Systems, IEEE Transactions on**, [S.l.], v.40, n.1, p.158–167, 2004.

LIANG, N.-Y. et al. A fast and accurate online sequential learning algorithm for feedforward networks. **Neural Networks, IEEE Transactions on**, [S.l.], v.17, n.6, p.1411–1423, 2006.

LIM, J. Partitioned online sequential extreme learning machine for large ordered system modeling. **Neurocomputing**, [S.l.], v.102, p.59–64, 2013.

LIN, S. et al. Almost optimal estimates for approximation and learning by radial basis function networks. **Machine learning**, [S.l.], v.95, n.2, p.147–164, 2014.

LIN, Y.-Y.; CHANG, J.-Y.; LIN, C.-T. Identification and prediction of dynamic systems using an interactively recurrent self-evolving fuzzy neural network. **Neural Networks and Learning Systems, IEEE Transactions on**, [S.l.], v.24, n.2, p.310–321, 2013.

LIU, X. et al. Is Extreme Learning Machine Feasible? A Theoretical Assessment (Part I). **Neural Networks and Learning Systems, IEEE Transactions on**, [S.l.], v.26, n.1, p.7–20, Jan 2015.

LORENZ, E. N. Deterministic nonperiodic flow. **Journal of the atmospheric sciences**, [S.l.], v.20, n.2, p.130–141, 1963.

LUO, M.; ZHANG, K. A hybrid approach combining extreme learning machine and sparse representation for image classification. **Engineering Applications of Artificial Intelligence**, [S.l.], v.27, p.228–235, 2014.

MACKEY, M. C.; GLASS, L. et al. Oscillation and chaos in physiological control systems. **Science**, [S.l.], v.197, n.4300, p.287–289, 1977.

MAY, R. M. et al. Simple mathematical models with very complicated dynamics. **Nature**, [S.l.], v.261, n.5560, p.459–467, 1976.

MAYBECK, P. S. **Stochastic models, estimation, and control**. [S.l.]: Academic Press, 1982. v.3.

MEIR, R.; MAIOROV, V. E. On the optimality of neural-network approximation using incremental algorithms. **Neural Networks, IEEE Transactions on**, [S.l.], v.11, n.2, p.323–337, 2000.

MICHE, Y.; SCHRAUWEN, B.; LENDASSE, A. Machine Learning Techniques based on Random Projections. In: ESANN. **Anais...** [S.l.: s.n.], 2010.

MIRZA, B.; LIN, Z.; TOH, K.-A. Weighted online sequential extreme learning machine for class imbalance learning. **Neural Processing Letters**, [S.l.], v.38, n.3, p.465–486, 2013.

MONTANA, G.; TRIANTAFYLLOPOULOS, K.; TSAGARIS, T. Flexible least squares for temporal data mining and statistical arbitrage. **Expert Systems with Applications**, [S.l.], v.36, n.2, p.2819–2830, 2009.

MOODY, J.; DARKEN, C. J. Fast learning in networks of locally-tuned processing units. **Neural computation**, [S.l.], v.1, n.2, p.281–294, 1989.

MUSAVI, M. T. et al. On the training of radial basis function classifiers. **Neural networks**, [S.l.], v.5, n.4, p.595–603, 1992.

NELDER, J. A.; MEAD, R. A simplex method for function minimization. **The computer journal**, [S.l.], v.7, n.4, p.308–313, 1965.

NEMENYI, P. Distribution-free multiple comparisons. In: BIOMETRICS. **Anais...** [S.l.: s.n.], 1962. v.18, n.2, p.263.

NOBREGA, J. P.; OLIVEIRA, A. L. A combination forecasting model using machine learning and Kalman filter for statistical arbitrage. In: SYSTEMS, MAN AND CYBERNETICS (SMC), 2014 IEEE INTERNATIONAL CONFERENCE ON. **Anais...** [S.l.: s.n.], 2014. p.1294–1299.

NOBREGA, J. P.; OLIVEIRA, A. L. I. Kalman Filter-based Method for Online Sequential Extreme Learning Machine for Regression Problems. **Engineering Applications of Artificial Intelligence**, [S.l.], v.44, p.101–110, 2015.

NOBREGA, J. P.; OLIVEIRA, A. L. I. D. Improving the Statistical Arbitrage Strategy in Intraday Trading by Combining Extreme Learning Machine and Support Vector Regression with Linear Regression Models. In: TOOLS WITH ARTIFICIAL INTELLIGENCE (ICTAI), 2013 IEEE 25TH INTERNATIONAL CONFERENCE ON. **Anais...** [S.l.: s.n.], 2013. p.182–188.

PARK, J.; SANDBERG, I. W. Universal approximation using radial-basis-function networks. **Neural computation**, [S.l.], v.3, n.2, p.246–257, 1991.

PENROSE, R. A generalized inverse for matrices. In: MATHEMATICAL PROCEEDINGS OF THE CAMBRIDGE PHILOSOPHICAL SOCIETY. **Anais...** [S.l.: s.n.], 1955. v.51, n.03, p.406–413.

PLATT, J. A resource-allocating network for function interpolation. **Neural computation**, [S.l.], v.3, n.2, p.213–225, 1991.

PORAT, B. **Digital processing of random signals**: theory and methods. [S.l.]: Prentice-Hall, Inc., 1994.

QUINLAN, J. R. Combining instance-based and model-based learning. In: TENTH INTERNATIONAL CONFERENCE ON MACHINE LEARNING. **Proceedings...** [S.l.: s.n.], 1993. p.236–243.

RAJAMANI, M. R.; RAWLINGS, J. B. Estimation of the disturbance structure from data using semidefinite programming and optimal weighting. **Automatica**, [S.l.], v.45, n.1, p.142–148, 2009.

REDMOND, M.; BAVEJA, A. A data-driven software tool for enabling cooperative information sharing among police departments. **European Journal of Operational Research**, [S.l.], v.141, n.3, p.660–678, 2002.

RONG, H.-J. et al. Online sequential fuzzy extreme learning machine for function approximation and classification problems. **Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on**, [S.l.], v.39, n.4, p.1067–1072, 2009.

RONG, H.-J.; HUANG, G.-B.; LIANG, Y.-Q. FUZZY EXTREME LEARNING MACHINE FOR A CLASS OF FUZZY INFERENCE SYSTEMS. **International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems**, [S.l.], v.21, n.supp02, p.51–61, 2013.

RÖSSLER, O. E. An equation for continuous chaos. **Physics Letters A**, [S.l.], v.57, n.5, p.397–398, 1976.

RUHMELHART, D.; HINTON, G.; WILIAMS, R. Learning representations by back-propagation errors. **Nature**, [S.l.], v.323, p.533–536, 1986.

SAAD, D. **On-line learning in neural networks**. [S.l.]: Cambridge University Press, 2009. v.17.

SAID, S. E.; DICKEY, D. A. Testing for unit roots in autoregressive-moving average models of unknown order. **Biometrika**, [S.l.], v.71, n.3, p.599–607, 1984.

SARKKA, S. On unscented Kalman filtering for state estimation of continuous-time nonlinear systems. **Automatic Control, IEEE Transactions on**, [S.l.], v.52, n.9, p.1631–1641, 2007.

SARTORI, M. A.; ANTSAKLIS, P. J. A simple method to derive bounds on the size and to train multilayer neural networks. **Neural Networks, IEEE Transactions on**, [S.l.], v.2, n.4, p.467–471, 1991.

SERMPINIS, G. et al. Forecasting and trading the EUR/USD exchange rate with stochastic Neural Network combination and time-varying leverage. **Decision Support Systems**, [S.l.], v.54, n.1, p.316–329, 2012.

SHUMWAY, R. H.; STOFFER, D. S. **Time series analysis and its applications**: with r examples. [S.l.]: Springer, 2010.

STINCHCOMBE, M.; WHITE, H. Universal approximation using feedforward networks with non-sigmoid hidden layer activation functions. In: NEURAL NETWORKS, 1989. IJCNN., INTERNATIONAL JOINT CONFERENCE ON. **Anais...** [S.l.: s.n.], 1989. p.613–617.

SUN, Y.; YUAN, Y.; WANG, G. An OS-ELM based distributed ensemble classification framework in P2P networks. **Neurocomputing**, [S.l.], p.2438–2443, 2011.

SURESH, S.; SUNDARARAJAN, N. An on-line learning neural controller for helicopters performing highly nonlinear maneuvers. **Applied Soft Computing**, [S.l.], v.12, n.1, p.360–371, 2012.

THEIL, H. et al. **Applied economic forecasting**. [S.l.]: North-Holland Publishing Company Amsterdam, 1971.

TSANAS, A.; XIFARA, A. Accurate quantitative estimation of energy performance of residential buildings using statistical machine learning tools. **Energy and Buildings**, [S.l.], v.49, p.560–567, 2012.

TÜFEKCI, P. Prediction of full load electrical power output of a base load operated combined cycle power plant using machine learning methods. **International Journal of Electrical Power & Energy Systems**, [S.l.], v.60, p.126–140, 2014.

TUKEY, J. W. Comparing individual means in the analysis of variance. **Biometrics**, [S.l.], p.99–114, 1949.

UYSAL, İ.; GÜVENIR, H. A. Instance-based regression by partitioning feature projections. **Applied Intelligence**, [S.l.], v.21, n.1, p.57–79, 2004.

VAN TREES, H. L. **Detection, estimation, and modulation theory**. [S.l.]: John Wiley & Sons, 2004.

VIDYAMURTHY, G. **Pairs Trading**: quantitative methods and analysis. [S.l.]: John Wiley & Sons, 2004. v.217.

WANG, N.; ER, M. J.; MENG, X. A fast and accurate online self-organizing scheme for parsimonious fuzzy neural networks. **Neurocomputing**, [S.l.], v.72, n.16, p.3818–3829, 2009.

WANG, X.; HAN, M. Online sequential extreme learning machine with kernels for nonstationary time series prediction. **Neurocomputing**, [S.l.], v.145, p.90–97, 2014.

WATSON, P. Kalman filtering as an alternative to Ordinary Least Squares—Some theoretical considerations and empirical results. **Empirical Economics**, [S.l.], v.8, n.2, p.71–85, 1983.

WITTEN, I. H.; FRANK, E. **Data Mining**: practical machine learning tools and techniques. [S.l.]: Morgan Kaufmann, 2005.

WU, Y. et al. A numerical-integration perspective on Gaussian filters. **Signal Processing, IEEE Transactions on**, [S.l.], v.54, n.8, p.2910–2921, 2006.

XI, Y.; PENG, H.; CHEN, X. A sequential learning algorithm based on adaptive particle filtering for RBF networks. **Neural Computing and Applications**, [S.l.], v.25, n.3-4, p.807–814, 2014.

YAAKOV, B.-S.; THIAGALINGAM, K.; RONG, L. **Estimation with applications to tracking and navigation**. [S.l.]: John Wiley & Sons, Inc.: New York, NY, USA, 2002.

YANG, S. et al. Sparse Ridgelet Kernel Regressor and its online sequential extreme learning. **Neurocomputing**, [S.l.], v.134, p.173–180, 2014.

YE, Y.; SQUARTINI, S.; PIAZZA, F. Online sequential extreme learning machine in nonstationary environments. **Neurocomputing**, [S.l.], v.116, p.94–101, 2013.

YINGWEI, L.; SUNDARARAJAN, N.; SARATCHANDRAN, P. Performance evaluation of a sequential minimal radial basis function (RBF) neural network learning algorithm. **Neural Networks, IEEE Transactions on**, [S.l.], v.9, n.2, p.308–318, 1998.

YU, H. et al. Advantages of radial basis function networks for dynamic system design. **Industrial Electronics, IEEE Transactions on**, [S.l.], v.58, n.12, p.5438–5450, 2011.

YU, W. et al. Learning deep representations via extreme learning machines. **Neurocomputing**, [S.l.], v.149, p.308–315, 2015.

ZHANG, H. et al. Data-core-based fuzzy min–max neural network for pattern classification. **Neural Networks, IEEE Transactions on**, [S.l.], v.22, n.12, p.2339–2352, 2011.

ZHAO, J.; WANG, Z.; PARK, D. S. Online sequential extreme learning machine with forgetting mechanism. **Neurocomputing**, [S.l.], v.87, p.79–89, 2012.

ZHAO, L.; WANG, D.; CHAI, T. Estimation of effluent quality using PLS-based extreme learning machines. **Neural Computing and Applications**, [S.l.], v.22, n.3-4, p.509–519, 2013.

ZHENG, J. et al. An online incremental learning support vector machine for large-scale data. **Neural Computing and Applications**, [S.l.], v.22, n.5, p.1023–1035, 2013.

ZHOU, S.; LAI, K. K.; YEN, J. A dynamic meta-learning rate-based model for gold market forecasting. **Expert Systems with Applications**, [S.l.], v.39, n.6, p.6168–6173, 2012.

ZHOU, X.; LIU, Z.; ZHU, C. Online Regularized and Kernelized Extreme Learning Machines with Forgetting Mechanism. **Mathematical Problems in Engineering**, [S.l.], v.2014, 2014.

ZHU, Q.-Y. et al. Evolutionary extreme learning machine. **Pattern Recognition**, [S.l.], v.38, n.10, p.1759–1763, 2005.

Apêndices

A

Método de Nelder-Mead

O método *simplex* de Nelder-Mead é utilizado para encontrar o mínimo local de uma função com uma ou mais variáveis. Sua descoberta é atribuída a J. A. Nelder e R. Mead em 1965. Para duas variáveis, um *simplex* é um triângulo, e o método resume-se a uma busca de padrões que compara os valores da função nos três vértices do triângulo. O pior vértice, em que $f(x,y)$ possui o maior valor, é rejeitado e substituído por um novo vértice. Um novo triângulo é formado e a busca continua. O processo gera uma sequência de triângulos, os quais podem ter diferentes formatos e os valores dos vértices tornam-se cada vez menores. O tamanho dos triângulos é reduzido e as coordenadas do ponto no qual o valor da função atinge o minímo são encontradas. A denominação *simplex* para o algoritmo vem da descrição de um triângulo genérico com n dimensões. O método é efetivo para encontrar o mínimo de uma função com n variáveis e é computacionalmente compacto. O problema consiste em minimizar a função $f(x)$, em que $x \in \mathbb{R}^n$. Os pontos atuais de teste são denotados por $x_1, \ldots, x_{n+1}$. Uma descrição dos passos do algoritmo é apresentada a seguir.

1. **Ordenação.**
   os valores da função nos pontos de teste, de acordo com os valores dos vértices: $f(x_1) \leq f(x_2) \leq \ldots \leq f(x_{n+1})$.

2. **Centroide.**
   Calcule o valor de x_0, o centroide de todos os pontos, exceto x_{n+1}.

3. **Reflexão.**
   Compute o ponto de reflexão $x_r = x_0 + \alpha(x_0 - x_{n+1})$, em que α é o coeficiente de reflexão (o valor padrão é $\alpha = 1$). Se o valor do ponto de reflexão estiver dentro do intervalo entre o segundo pior ponto e o melhor ponto, ou seja, $f(x_1) \leq f(x_r) \leq f(x_n)$, então obtenha um novo *simplex* através da troca do pior ponto x_{n+1} pelo ponto de reflexão x_r e volte para o passo 1.

4. **Expansão.**
   Se o ponto de reflexão é o melhor ponto encontrado, de forma que $f(x_r) < f(x_1)$,

então compute o ponto de expansão $x_e = x_0 + \gamma(x_0 - x_{n+1})$, em que γ é o coeficiente de expansão (o valor padrão é $\gamma = 2$). Se o ponto expandido é melhor que o ponto de reflexão, de forma que $f(x_e) < f(x_r)$, então obtenha um novo *simplex* através da troca do pior ponto x_{n+1} pelo ponto expandido x_e e volte para o passo 1. Senão, obtenha um novo *simplex* através da troca do pior ponto x_{n+1} pelo ponto de reflexão x_r e volte para o passo 1. Se o ponto de reflexão não for o melhor ponto encontrado no início desse passo, vá para o passo seguinte.

5. **Contração**.
   Nesse passo, é certo que $f(x_r) \geq f(x_n)$. Compute o ponto de contração $x_c = x_0 + \rho(x_0 - x_{n+1})$, em que ρ é o coeficiente de contração (valor padrão $\rho = -0,5$). Se o ponto de contração é melhor que o pior ponto, de forma que $f(x_c) < f(x_{n+1})$, então obtenha um novo *simplex* através da troca do pior ponto x_{n+1} pelo ponto de contração x_c e volte para o passo 1. Senão, vá para o passo seguinte.

6. **Redução.**
   Para todos os pontos x_i, com exceção do melhor ponto x_1, faça a troca por $x_i = x_1 + \sigma(x_1 - x_1)$ para todo $i \in \{2, \ldots n+1\}$, em que σ é o coeficiente de redução (valor padrão $\sigma = 0,5$). Volte para o passo 1.

O critério de parada do algoritmo pode ser estabelecido a partir de um número máximo de iterações ou de acordo com um limiar ν, que representa uma pequena diferença relativa entre os valores dos vértices do *simplex*. O critério de parada sugerido em NELDER; MEAD (1965) é dado pelas seguintes expressões:

$$\begin{aligned} &\tfrac{1}{n+1}\sum_{i=1}^{n+1}\left(f_i - \bar{f}\right)^2 < \nu \\ &\bar{f} = \tfrac{1}{n+1}\sum_{i=1}^{n+1} f_i \end{aligned} \tag{A.1}$$

em que n é o número de iterações do algoritmo e f_i é o i-ésimo valor computado da função objetivo. Para os experimentos realizados nessa tese, o valor de $\nu = 10^{-4}$ foi escolhido de forma empírica para o critério de parada.

B

Testes Estatísticos

B.1 Teste de Friedman

O teste de Friedman (FRIEDMAN, 1940) é um método não paramétrico equivalente ao teste ANOVA para medidas repetidas. Ele fornece um *ranking* dos algoritmos para cada base de dados, com o melhor algoritmo obtendo a posição 1, o segundo melhor a posição 2, e assim sucessivamente. No caso de empate entre dois ou mais algoritmos, a média das posições empatadas é atribuída como a posição de cada algoritmo. Uma descrição do procedimento para o teste é apresentada a seguir.

Seja $\{x_{ij}\}_{n\times k}$ uma matriz com n linhas representando as bases de dados e k colunas representando os métodos avaliados. Dado um certo critério de comparação (por exemplo, o menor erro de previsão), calcule as posições de cada método para cada base de dados. Em seguida, crie uma nova matriz $\{r_{ij}\}_{n\times k}$, em que o elemento r_{ij} é a posição de x_{ij} para a base de dados i. Para chegar na formulação da estatística do teste, precisamos da seguinte sequência de cálculos:

$$\bar{r}_j = \frac{1}{n}\sum_{i=1}^{n} r_{ij}, \tag{B.1}$$

$$\bar{r} = \frac{1}{nk}\sum_{i=1}^{n}\sum_{j=1}^{k} r_{ij}, \tag{B.2}$$

$$SS_t = n\sum_{j=1}^{k}\left(\bar{r}_j - \bar{r}\right)^2, \tag{B.3}$$

$$SS_e = \frac{1}{n\,(k-1)}\sum_{i=1}^{n}\sum_{j=1}^{k}\left(r_{ij} - \bar{r}\right)^2. \tag{B.4}$$

$$Q = \frac{SS_t}{SS_e}. \tag{B.5}$$

A estatística do teste é dada pelo cálculo de Q. Para valores elevados de n ou k, a distribuição de probabilidade de Q se aproxima da Qui-quadrado com $(k-1)$ graus de liberdade, com o *p-value* fornecido por $\mathbf{P}\left(\chi^2_{k-1} \geq Q\right)$. Em DEMŠAR (2006), são sugeridos os valores

de $n > 10$ e $k > 5$ para a distribuição citada. Para um número menor de algoritmos e bases de dados, os valores críticos são computados diretamente, sendo preparados especificamente para o teste. Se o *p-value* for significativo, testes *post hoc* para múltiplas comparações são necessários para confirmar se os métodos são significativamente diferentes uns dos outros, levando em consideração as diferenças do *ranking* entre os métodos.

B.2 Teste *post hoc* de Tukey

Na estatística, os testes *post hoc* têm o seu nome derivado do latim ("depois do fato") e são utilizados para múltiplas comparações quando já existe um teste prévio indicando diferenças significativas. Os erros de inferência, incluindo os relacionados com intervalos de confiança ou dos testes que rejeitam incorretamente as hipóteses nulas, podem ocorrer com maior frequência quando são realizadas múltiplas comparações. Diversas técnicas estatísticas estão disponíveis para prevenir essa classe de erros, permitindo que diferentes níveis de significância possam ser diretamente comparados. Essas técnicas necessitam de um limiar alto para a significância das comparações individuais, de forma a compensar o número de inferências realizado.

O procedimento padrão para múltiplas comparações par a par foi proposto por Tukey (TUKEY, 1949). Esse teste basicamente obtém o máximo entre os valores absolutos de todas as estatísticas par a par, na forma da expressão:

$$t_{ij} = \frac{\bar{r}_i - \bar{r}_j}{\sigma\sqrt{\frac{2}{n}}}, \tag{B.6}$$

em que $\bar{r}_i$ é a estimativa de mínimos quadrados para a média do desempenho de r_i, σ é o desvio padrão e n é a quantidade de métodos em comparação. Cada estatística t_{ij} é univariada e distribuída de acordo com a distribuição *t de Student*. O vetor formado pelas estatísticas do teste segue uma distribuição *t de Student* multivariada, e o intervalo de confiança do teste é dado por

$$\left[\bar{r}_i - \bar{r}_j - u_{1-\alpha}\sigma\sqrt{\frac{2}{n}}; \bar{r}_i - \bar{r}_j + u_{1-\alpha}\sigma\sqrt{\frac{2}{n}}\right], \tag{B.7}$$

em que $u_{1-\alpha}$ denota o $1-\alpha$-quartil da distribuição *t de Student* multivariada.

A aplicação do teste *post hoc* de Tukey é considerada a melhor opção para múltiplas comparações quando os tamanhos das amostras não são iguais ou quando os intervalos de confiança são necessários. Quando essas premissas são violadas o poder do teste de Tukey é ligeiramente reduzido, mas ainda adequado para as comparações par a par, o que motivou a sua escolha para os experimentos dessa tese.

B.3 Teste de Cointegração de Johansen

O teste de Johansen, proposto por Søren Johansen em JOHANSEN (1995), é um procedimento para testar a cointegração de duas ou mais séries temporais não estacionárias, ou $I(1)$. Existem duas versões do teste: o do máximo autovalor e o do traço. As inferências de cada teste são ligeiramente diferentes, mas as suas interpretações permitem avaliar a existência de um vetor de cointegração com um certo nível de confiança. Para aplicar o teste, devemos antes descrever um vetor de autoregressão (*Vector Autoregression* (VAR)) de ordem p, de acordo com a expressão:

$$\mathbf{y}_t = \mu + A_1\mathbf{y}_{t-1} + \ldots + A_p\mathbf{y}_{t-p} + \boldsymbol{\varepsilon}_t, \tag{B.8}$$

em que $\mathbf{y}_t$ é um vetor $n \times 1$ de variáveis $I(1)$, A_p são os coeficientes, μ é a média e ε_t é um vetor $n \times 1$ de inovações. O VAR pode ser reescrito da seguinte forma:

$$\Delta\mathbf{y}_t = \mu + \Pi\mathbf{y}_{t-1} + \sum_{i=1}^{p-1} \Gamma_i \Delta\mathbf{y}_{t-1} + \boldsymbol{\varepsilon}_t, \tag{B.9}$$

em que

$$\Pi = \sum_{i=1}^{p} A_i - I, \tag{B.10}$$

$$\Gamma_i = -\sum_{j=i+1}^{p} A_j. \tag{B.11}$$

Se a matriz de coeficientes Π tem posto reduzido $r < n$, então existem nr matrizes $\boldsymbol{\alpha}$ e $\boldsymbol{\beta}$, ambas com posto r, tal que $\Pi = \boldsymbol{\alpha}\boldsymbol{\beta}^T$ e $\boldsymbol{\beta}^T\mathbf{y}_t$ é $\mathbf{I}(0)$, em que r é o número de vetores de cointegração, os elementos de $\boldsymbol{\alpha}$ são os fatores de ajuste do modelo de correção de erros do vetor e as colunas de $\boldsymbol{\beta}$ são os r vetores de cointegração. Se queremos testar a existência de n vetores de cointegração, a hipótese nula para o teste do traço é $r \leq n$. Para o teste do máximo autovalor, a hipótese nula é $r = n$. Para o cálculo das estatísticas dos testes, Johansen propôs duas formulações, baseadas no teste da razão de verossimilhança, de acordo com as seguintes equações:

$$J_{trace} = -T \sum_{i=r+1}^{n} \log\left(1 - \hat{\lambda}_i\right), \tag{B.12}$$

$$J_{\max} = -T \log\left(1 - \hat{\lambda}_{r+1}\right), \tag{B.13}$$

em que T é o tamanho do conjunto de amostras e $\hat{\lambda}_i$ é a i-ésima maior correlação canônica. O teste do traço avalia a hipótese nula de r vetores de cointegração contra a hipótese alternativa de n vetores de cointegração. O teste do máximo autovalor avalia a hipótese nula de r vetores de cointegração contra a hipótese alternativa de $r+1$ vetores de cointegração. Para os experimentos do estudo de caso dessa tese, foram utilizados os testes do máximo autovalor.

B.4 Teste de Estacionaridade de Dickey-Fuller Aumentado

O teste de Dickey-Fuller aumentado ou Teste ADF (do acrônimo em inglês *Augmented Dickey-Fuller*) é um teste originalmente criado para verificar se um modelo autorregressivo possui ou não raiz unitária (SAID; DICKEY, 1984). O teste também é aplicado para verificar a evidência de estacionaridade em séries temporais. Considere o modelo a seguir:

$$\mathbf{Y}_t = \alpha \mathbf{Y}_{t-1} + \mathbf{X}_t, t = 1, 2, ...$$

em que $\{\mathbf{X}_t\}$ é um processo estacionário. O processo $\{\mathbf{Y}_t\}$ é não estacionário se o coeficiente $\alpha = 1$, mas é estacionário se $|\alpha| < 1$. Suponha que $\{\mathbf{X}_t\}$ seja um processo $AR(k)$, de acordo com a seguinte expressão:

$$\mathbf{X}_t = \phi_1 \mathbf{X}_{t-1} + \ldots + \phi_k \mathbf{X}_{t-k} + e_t. \tag{B.14}$$

Se assumirmos a condição da hipótese nula, com $\alpha = 1$, temos:

$$\mathbf{X}_t = \mathbf{Y}_t - \mathbf{Y}_{t-1}. \tag{B.15}$$

Fazendo $a = \alpha - 1$, temos:

$$\begin{aligned} \mathbf{Y}_t - \mathbf{Y}_{t-1} &= (\alpha - 1)\mathbf{Y}_{t-1} + \mathbf{X}_t \\ &= \alpha \mathbf{Y}_{t-1} + \phi_1 \mathbf{X}_{t-1} + \ldots + \phi_k \mathbf{X}_{t-k} + e_t \\ &= \alpha \mathbf{Y}_{t-1} + \phi_1 (\mathbf{Y}_{t-1} - \mathbf{Y}_{t-2}) + \ldots + \phi_k (\mathbf{Y}_{t-k} - \mathbf{Y}_{t-k-1}) + e_t, \end{aligned} \tag{B.16}$$

em que $a = 0$ sob a hipótese que $\mathbf{Y}_t$ é a diferença não estacionária. Por outro lado, se $\{\mathbf{Y}_t\}$ é estacionário, tal que $-1 < \alpha < 1$, então pode ser verificado que $\mathbf{Y}_t$ continua satisfazendo uma expressão similar à Equação B.16, mas com coeficientes diferentes. De fato, $\{\mathbf{Y}_t\}$ é um processo $AR(k+1)$, cuja equação é dada por

$$\Phi(x)(1 - \alpha x) = 0, \tag{B.17}$$

em que $\Phi(x) = 1 - \phi_1 x - \ldots - \phi_k x^k$. Dessa forma, a hipótese nula corresponde ao caso no qual o modelo AR tem raiz unitária e a hipótese alternativa estabelece que o modelo em questão não possui raiz unitária.

Pela análise anterior, a hipótese nula que $\alpha = 1$ (ou, de forma similar, $a = 0$) pode ser facilmente testada através da regressão da primeira diferença da série com atraso 1 sobre a série com os últimos k atrasos. Podemos então verificar se o coeficiente $a = 0$, com a hipótese nula sendo que a diferença do processo é não estacionária. Ou seja, o processo é não estacionário, mas torna-se estacionário após a primeira diferenciação. A hipótese alternativa é que $a < 0$ e $\{\mathbf{Y}_t\}$ é estacionário. A estatística do teste de Dickey-Fuller aumentado é a estatística t do coeficiente

estimado, obtido por regressão de mínimos quadrados. A tabela com os valores críticos pode ser vista em FULLER (2009).

C

Otimização por Exame de Partículas

C.1 O Algoritmo PSO

A otimização por exame de partículas, ou *Particle Swarm Optimization* (PSO), é uma técnica de busca aleatória desenvolvida por James Kennedy e Russell C. Eberhart (KENNEDY, 2010). Esse método de otimização é inspirado nos princípios de interação social. O algoritmo PSO possui uma diferença básica em relação aos métodos tradicionais de busca aleatória: em vez de atualizar uma única solução candidata $x^{(k)}$ a cada iteração, é feita a atualização de uma população (ou conjunto) de soluções candidatas. O conjunto é chamado de enxame e cada solução candidata é chamada de partícula. Podemos pensar em um enxame como sendo uma população de indivíduos que se movem de forma agrupada de acordo com uma tendência, mas com cada indivíduo movendo-se de forma aleatória dentro do agrupamento. O algoritmo PSO tenta imitar o comportamento social de animais e insetos, como os enxames de abelhas ou bandos de aves.

Suponha que queremos minimizar uma função objetivo sobre o espaço $\mathbb{R}^n$. No PSO, começamos com uma população gerada aleatoriamente com pontos definidos em $\mathbb{R}^n$. Associado a cada ponto da população existe um vetor de velocidade. Cada ponto pode ser visto como a posição da partícula no espaço de busca, com a sua movimentação sendo associada à uma certa velocidade. Em seguida, avaliamos a função objetivo em cada ponto da população. De acordo com a avaliação, criamos uma nova população de pontos com um novo conjunto de velocidades. Cada partícula mantém o registro de sua melhor posição em respeito à função objetivo. A melhor posição individual de cada partícula é chamada de *pbest* (do inglês *personal best*). De forma similar, a melhor posição entre todas as partículas da população é chamada de *gbest* (do inglês *global best*). As partículas "interagem" entre si, atualizando suas velocidades de acordo com os seus valores de *pbest* e *gbest*. Na versão básica do PSO, a velocidade de cada partícula é alterada a cada instante de tempo através da combinação das localizações de *pbest* e *gbest*. A velocidade é ponderada por um termo aleatório, gerado a partir das localizações de *pbest* e *gbest*. Assim, as partículas são atraídas tanto para as suas melhores posições individuais quanto para a melhor posição de todo o enxame.

Formalizando a discussão anterior, seja $f : \mathbb{R}^n \to \mathbb{R}$ a função objetivo que queremos minimizar. Seja d o tamanho da população e i o índice das partículas, de forma que $i = 1, ..., d$. Denote a posição da partícula i por $\mathbf{x}_i \in \mathbb{R}^n$ e sua velocidade por $\mathbf{v}_i \in \mathbb{R}^n$. Seja $\mathbf{p}_i$ o valor de *pbest* da partícula i e g o valor de *gbest*. O algoritmo PSO possui três parâmetros constantes: ω, c_1 e c_2. Os passos do algoritmo são descritos a seguir:

1. Faça $k = 0$. Para $i = 1, ..., d$, gere aleatoriamente as posições iniciais $\mathbf{x}_i^{(0)}$ e as velocidades iniciais $\mathbf{v}_i^{(0)}$. Faça $\mathbf{p}_i^{(0)} = \mathbf{x}_i^{(0)}$. Faça $\mathbf{g}^{(0)} = \underset{\mathbf{x} \in \left\{\mathbf{x}_1^{(0)}, ..., \mathbf{x}_d^{(0)}\right\}}{\arg\min} f(\mathbf{x})$.

2. Para $i = 1, ..., d$, gere aleatoriamente n vetores $\mathbf{r}_i^{(k)}$ e $\mathbf{r}_i^{(k)}$ com elementos uniformemente distribuídos no intervalo $(0, 1)$ e faça

$$v_i^{(k+1)} = \omega v_i^{(k)} + c_1 r_i^{(k)} \left(p_i^{(k)} - x_i^{(k)}\right) + c_2 s_i^{(k)} \left(g^{(k)} - x_i^{(k)}\right)$$
$$x_i^{(k+1)} = x_i^{(k)} + v_i^{(k+1)}$$

3. Para $i = 1, ..., d$, se $f\left(x_i^{(k+1)}\right) < f\left(p_i^{(k)}\right)$, então faça $p_i^{(k+1)} = x_i^{(k+1)}$. Senão, faça $p_i^{(k+1)} = p_i^{(k)}$.

4. Se existe $i \in \{1, \ldots d\}$, tal que $f\left(x_i^{(k+1)}\right) < f\left(g^{(k)}\right)$, então faça $g^{(k+1)} = x_i^{(k+1)}$. Senão, faça $g^{(k+1)} = g^{(k)}$.

5. Se o critério de parada for satisfeito, pare a execução.

6. Faça $k = k + 1$ e retorne ao passo 2.

No algoritmo do PSO, o parâmetro ω é chamado de constante de inércia. Em KENNEDY (2010), é sugerido a utilização de valores menores que 1 para esse parâmetro. Os parâmetros c_1 e c_2 são constantes que determinam o quanto cada partícula deve se mover para "boas" posições. Eles representam os componentes cognitivo e social, respectivamente. Em KENNEDY (2010) recomenda-se adotar $c_1, c_2 \approx 2$.