UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

DIOGO MOURY FERNANDES IZIDIO

Uma Abordagem Evolutiva para Combinação de Modelos Aplicada a Previsão de Consumo de Energia

Recife
2021

DIOGO MOURY FERNANDES IZIDIO

Uma Abordagem Evolutiva para Combinação de Modelos Aplicada a Previsão de Consumo de Energia

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 Mestre em Ciência da Computação.

Área de Concentração: Inteligência Computacional

Orientador (a): Paulo Salgado Gomes de Mattos Neto

Recife

2021

Catalogação na fonte
Bibliotecária Monick Raquel Silvestre da S. Portes, CRB4-1217

I98a Izidio, Diogo Moury Fernandes
Uma abordagem evolutiva para combinação de modelos aplicada a previsão de consumo de energia / Diogo Moury Fernandes Izidio. – 2021.
79 f.: il., fig., tab.

Orientador: Paulo Salgado Gomes de Mattos Neto.
Dissertação (Mestrado) – Universidade Federal de Pernambuco. CIn, Ciência da Computação, Recife, 2021.
Inclui referências.

1. Inteligência computacional. 2. Séries temporais. 3. Redes neurais artificiais. I. Mattos Neto, Paulo Salgado Gomes de (orientador). II. Título.

006.31 CDD (23. ed.) UFPE - CCEN 2021 – 110

Diogo Moury Fernandes Izidio

“Uma Abordagem Evolutiva para Combinação de Modelos Aplicada a Previsão de Consumo de Energia”

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação.

Aprovado em: 29/01/2021.

BANCA EXAMINADORA

_____

Prof. Dr. Luciano de Andrade Barbosa
Centro de Informática/ UFPE

_____

Prof. Dr. João Fausto Lorenzato de Oliveira
Escola Politécnica de Pernambuco / UPE

_____

Prof. Dr. Paulo Salgado Gomes de Mattos Neto
Centro de Informática / UFPE
(**Orientador**)

Aos familiares e amigos, pelo incentivo e apoio constantes.

AGRADECIMENTOS

Agradeço ao Prof. Orientador Paulo Salgado Gomes de Mattos Neto, por acompanhar o desenvolvimento da pesquisa, assim como dar sugestões, propor melhorias e debater ideias durante o andamento do mestrado, tal assistência se mostrou essencial ao progresso deste trabalho.

Aos Profs. João Fausto Lorenzato de Oliveira (UPE) e ao Prof. Luciano de Andrade Barbosa (UFPE) por também contribuírem de forma excepcional com críticas construtivas, assim como auxiliar, junto com o Prof. Orientador, na elaboração do artigo entitulado *An evolutionary approach for hybridization of ARIMA and machine learning models for energy consumption forecasting*.

Aos amigos, pelo companheirismo e por proporcionar momentos de contentamento durante o decorrer do curso. Finalmente, agradeço aos meus pais e familiares, pelo apoio constante e pelos conselhos.

RESUMO

O estudo sobre o consumo de energia elétrica vem crescentemente atraindo atenção da comunidade científica devido aos grandes impactos econômicos e ambientais. Dentro desse contexto, a introdução de medidores inteligentes vem sendo proposta como uma alternativa para aumentar a eficiência e melhorar o gerenciamento do consumo de energia em edifícios. Tais benefícios são geralmente associados à uma melhor compreensão acerca do consumo, assim como melhorias na detecção de fraudes e no planejamento da malha elétrica, tarefas diretamente ligadas à capacidade de prever o consumo de energia de forma acurada. Esse trabalho propõe um sistema híbrido que combina modelos estatísticos e de Aprendizado de Máquina (AM) para modelar padrões lineares e não-lineares das séries temporais de consumo de energia registradas por medidores inteligentes. O sistema híbrido é composto de três partes: uma técnica linear é utilizada para modelar os padrões lineares e sazonais presentes nas séries de consumo; uma técnica de AM é empregada para modelar os padrões não-lineares presentes nas séries de resíduos do modelo linear; e a combinação das previsões lineares e não-lineares é feita por um terceiro modelo de AM. Para realizar seleção de variáveis de entrada de cada técnica e superar problemas como underfitting, overfitting e má-especificação de parâmetros, um Algoritmo Genético foi usado nas etapas de modelagem não-linear e combinação das previsões. A avaliação experimental utilizou cinco métricas largamente utilizadas, mostrando que o sistema híbrido proposto alcança melhorias estatisticamente significantes nos resultados quando comparados com modelos estatísticos, híbridos e de AM da literatura.

Palavras-chaves: Séries Temporais. Sistemas Híbridos. Redes Neurais Artificiais. Algoritmos Genéticos. Aprendizado de Máquina.

ABSTRACT

The investigation of energy consumption patterns has increasinglly attracted interest from the scientific community due to its major economic and environmental effects. In that context, the introduction of smart metering has been proposed as an alternative to increase efficiency and improve energy consumption management in buildings. Such benefits are usually associated with a better understanding of energy consumption patterns and improvements in fraud detection and grid planning, which are directly linked to the ability of accurately forecasting energy demand. This work proposes a hybrid system which combines statistical and Machine Learning (ML) models to model linear and non-linear patterns of a energy consumption time series registered by smart meters. The proposed system is composed by three stages: a linear model is utilized to model the linear and seasonal patterns of the time series; a ML technique is then applied to the first model residual series to account for non-linear patterns; finally, a second ML model is responsible for the combination of both linear and non-linear forecasts. To perform feature selection and avoid problems like underfitting, overfitting and bad choice of parameters, a Genetic Algorithm is utilized in the non-linear and combination stages. Experimental evaluation is performed using five (5) metrics widely utilized in the literature, indicating that the proposed hybrid system reaches statistically significant improvements when compared to other statistical, hybrid, and ML approaches from the literature.

Keywords: Time Series. Hybrid Systems. Artificial Neural Networks. Genetic Algorithms. Machine Learning.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SÍMBOLOS

Z_t	Valor final da série temporal.
L_t	Componente linear da série temporal.
N_t	Componente não linear da série temporal.
c_r	Razão da população do algoritmo genético que será composta por crossover.
n_{tries}	Quantidade de vezes que a MLP é treinada no algoritmo genético.
n_{max}	Número máximo de mutações em uma iteração do algoritmo genético.
n_0	Número máximo de mutações na geração inicial.
n_m	Número de mutações para um determinado indivíduo em uma determinada iteração.
g_n	Geração limite onde o número máximo de mutações começa a incrementar.
n_{step}	Quantidade do incremento no número máximo de mutações.
l_{max}	Quantidade máxima do número de camadas da MLP.
ϕ_i	Coeficientes do modelo AR para o tempo $t - i$.
B	Operador de defasagem.
ϵ_t	Ruído aleatório de modelos lineares no tempo t.
θ_i	Coeficientes do modelo MA para o tempo $t - i$.
Δ^d	Operador da d-ésima diferença.
Φ_i	Coeficientes do modelo AR sazonal para o tempo $t - i$.
Θ_i	Coeficientes do modelo AR sazonal para o tempo $t - i$.
w_k	Peso da entrada no neurônio k.
v_t	Momento de segunda ordem do gradiente.
m_t	Momento de primeira ordem do gradiente.
θ_t	Parâmetros finais do treinamento em AM.
a_j^l	Ativação do neurônio j na camada l na MLP.

$C(\cdot)$	Função custo.
δ_j^l	Erro do neurônio j na camada l.
C	Constante de regularização do SVM.
$y^{(i)}$	Saída do SVM.
w^T	Vetor com coeficientes do hiperplano do SVM.
ξ	Representação de classificações erradas no SVM.
Ψ	Mapeamento utilizado para espaço de maior dimensionalidade no SVM.
Ω	Espaço de maior dimensionalidade.
X	Espaço de características original.
$K(x, x')$	Função de kernel.
p_i	Probabilidade de um indivíduo ser selecionado no método da roleta.
ML_R	Modelo de resíduos.
ML_C	Modelo de combinação.
$\hat{y}$	Valor previsto pelo modelo de AM.
r	Correlação de Pearson.

SUMÁRIO

1 INTRODUÇÃO

O estudo sobre o consumo de energia em edifícios está crescentemente atraindo atenção devido aos grandes impactos econômicos e ambientais causados na sociedade. O setor de construção, especificamente, se mostrou responsável por aproximadamente 40% do consumo de energia global e 30% das emissões de CO2 (ALLOUHI et al., 2015). Esses números vêm mostrando constante crescimento devido à continua evolução da urbanização (ZHAO; MAGOULÈS, 2012). Portanto, mudanças no consumo e na eficiência em edifícios estão sujeitos a impactar dramaticamente a sociedade atual, incluindo em assuntos socioeconômicos e ecológicos primordiais como aquecimento global e mudanças climáticas (CHOU; NGO, 2016a).

Dentro desse contexto, o *Smart Metering* é um conceito que vem sendo proposto como uma alternativa para aumentar a eficiência e melhorar o gerenciamento do consumo de energia em edifícios. Ao permitir um monitoramento próximo do consumo de energia dos consumidores uma melhor observabilidade e controlabilidade da malha energética, o *smart metering* pode ser utilizado como uma ferramenta para ganhar eficiência em cada passo da relação entre o consumidor e as empresas de geração, transmissão e distribuição de energia.

O conceito de *smart metering* consiste na instalação de um medidor inteligente no edifício, que pode ser acessado localmente ou remotamente e é capaz de registrar, processar e prover informações referentes ao consumo de energia do ambiente (GERWEN; JAARSMA; WILHITE, 2006).

A ampla utilização dos medidores inteligentes permite o registro e a fácil disponibilização do histórico de consumo de energia em diversas situações. Sendo assim, o acesso a esses dados em grande escala torna possível a extração, através de um processo de mineração de dados, de conclusões acerca dos padrões que regem o comportamento de consumo.

Em geral, tais informações podem ser utilizadas para obter uma visão mais aprofundada acerca do consumo de energia em cada região. O consumidor final pode utilizar tais informações para controlar e entender seu comportamento frente ao consumo de energia em cada ambiente.

Além disso, com o entendimento dos padrões de consumo, permite-se uma melhor previsão da demanda, o que pode ser utilizado para fins de planejamento e manutenção da malha elétrica. A previsão do consumo de energia permite ao fornecedor reagir a mudanças em qualquer parte do processo de geração, transmissão e distribuição, permitindo assim identificar atividades suspeitas e padrões de mudanças e fraude, por exemplo (GERWEN; JAARSMA; WILHITE, 2006; KABALCI, 2016).

Portanto, a tarefa de prever o consumo de energia se mostra uma tarefa de grande valor para auxiliar os ganhos de eficiência provenientes do *smart metering*. A habilidade de

prever o consumo de energia de forma acurada permite ao provedor de energia uma melhor compreensão acerca do consumo de energia, assim como leva a melhorias na detecção de fraudes, e no planejamento e manutenção da malha elétrica.

No entanto, a capacidade de prever adequadamente o consumo de energia representa uma tarefa complexa. Em geral, modelos utilizados amplamente para previsão de séries temporais na literatura possuem performance limitada em séries do mundo real. Esse resultado deve-se ao fato de que os padrões presentes em séries temporais reais seguem padrões não lineares que não são capturados por modelos lineares como AR, MA, ARMA e ARIMA (ZHANG, 2003).

Por outro lado, modelos não lineares baseados em Aprendizado de Máquina (AM), como Redes Neurais Artificiais, vêm sendo propostos como uma alternativa capaz de modelar padrões não lineares (ZHANG; PATUWO; HU, 1998).

Porém, em cenários do mundo real, a utilização de um modelo de aprendizado de máquina individual pode resultar em problemas originados por má especificação de parâmetros, under-fitting e overfitting. Sendo assim, modelos híbridos, que combinam as previsões de modelos estatísticos lineares e modelos de aprendizado de máquina vêm sendo propostos como uma abordagem que permite a modelagem dos padrões lineares e não lineares da série de forma mais eficiente (DOMINGOS; OLIVEIRA; NETO, 2019).

Portanto, esse trabalho propõe uma metodologia para prever o consumo de energia que utiliza a combinação de modelos estatísticos lineares e de AM. O sistema híbrido é composto de três partes:

1. Uma técnica linear é utilizada para modelar os padrões lineares e sazonais presentes nas séries de consumo.

2. Uma técnica de AM é empregada para modelar os padrões não lineares presentes nas séries de resíduos do modelo linear.

3. A combinação das previsões lineares e não lineares é feita por um terceiro modelo de AM.

O modelo estatístico linear é treinado utilizando uma metodologia largamente utilizada na literatura proposta por Box & Jenkins (BOX et al., 2015). Para realizar seleção de variáveis de entrada de cada técnica de AM e superar problemas como underfitting, overfitting e má-especificação de parâmetros, um algoritmo genético é utilizado para selecionar as entradas e realizar a otimização dos parâmetros dos modelos de AM nas etapas de modelagem não linear e combinação das previsões.

O método proposto é aplicado em dados reais de consumo de energia extraídos de um edifício de três andares localizado em Taiwan e comparado com métodos propostos existentes na literatura. A comparação será realizada através de cinco métricas largamente utilizadas. Assim, busca-se investigar se o sistema híbrido proposto alcança melhorias

estatisticamente significantes nos resultados quando comparados com modelos estatísticos, híbridos e de AM da literatura.

2 REVISÃO DA LITERATURA

O desenvolvimento de sistemas baseados em modelos de AM vem ganhando grande destaque na área de energia (AHMAD; CHEN, 2020). Neste campo, a previsão do consumo vem recebendo uma atenção crescente devido à sua relação com problemas ambientais e de geração e demanda (DEB et al., 2017; Heydari et al., 2018).

Em geral, a previsão da carga energética possui um impacto significativo no planejamento, operação e monitoramento de sistemas elétricos de potência. A acurácia dos modelos de previsão pode impactar os custos de operação dado que uma superestimação pode aumentar o número de geradores empregados e produzir uma reserva desnecessária de energia. Da mesma forma, a subestimação da carga energética pode colocar em risco a confiabilidade do sistema devido à produção insuficiente para atender o mercado consumidor (Chan et al., 2012). Sendo assim, modelos de previsão de consumo de energia pode aumentar a eficiência energética e a sustentabilidade em diversos setores como o residencial (CULABA et al., 2020; GAO et al., 2020; PINTO et al., 2021) e o industrial (WALTHER; WEIGOLD, 2021; BARZOLA-MONTESES et al., 2020).

Com o intuito de atingir uma melhor acurácia na previsão de consumo de energia, diversos modelos de AM vem sendo utilizados para essa tarefa (RANA; KOPRINSKA, 2016; ZOLFAGHARI; GOLABI, 2021; EL-HENDAWI; WANG, 2020). Métodos como Redes Neurais Artificiais (RNA) baseados em Wavelets (RANA; KOPRINSKA, 2016), Long Short-Term Memory (LSTM) (MARINO; AMARASINGHE; MANIC, 2016) e Random Forests (ZOLFAGHARI; GOLABI, 2021) vem sendo investigados por pesquisadores.

Da mesma forma, modelos de previsão baseado na combinação com modelos de AM vem sendo utilizados na literatura. Culaba et al. (CULABA et al., 2020) implementa um modelo híbrido baseado em clusterização e regressão utilizando K-means e Regressão por Vetores de Suporte (SVR), respectivamente. Modelos de aprendizado profundo como Redes Neurais Convolucionais (CNN) foram implementados por (GAO et al., 2020) para previsão do consumo de energia no contexto de prédios com poucos dados históricos. Pinto et al. (PINTO et al., 2021) utilizou combinação de modelos para prever o consumo em prédios comerciais. Em Walther & Weigold (WALTHER; WEIGOLD, 2021) pode-se encontrar uma revisão sistemática da literatura sobre modelos de previsão de consumo de energia na indústria.

Considerando os trabalhos da literatura focados em medidores inteligentes, diversos métodos de AM também foram investigados. Nesse contexto, Gajowniczek & Zabkowski (GAJOWNICZEK; ZĄBKOWSKI, 2014) utilizou modelos do tipo Multilayer Perceptron (MLP) e SVR para a previsão de medidores inteligentes individuais. Para isso, a solução extrai características relacionadas ao consumo histórico do medidor como os valores de consumo médio, máximos e mínimos e a temperatura dentro da casa. Argumenta-se que não é feita uma previsão de série temporal da forma tradicional devido à alta volati-

lidade dos dados.

Em Zhukov et al. (ZHUKOV; SIDOROV; FOLEY, 2016), o efeito da deriva de conceito na análise da rede de medidores inteligentes. Um algoritmo de Random Forest para deriva de conceito foi implementado, e uma combinação utilizando o voto majoritário ponderado foi utilizado para combinar a saída dos modelos individuais. O método proposto foi comparado com outros algoritmos no contexto de detecção de deriva de conceito, obtendo resultados promissores.

Precificação de eletricidade e previsão da carga energética são tarefas importantes em redes de medidores inteligentes devido á seus ganhos de eficiência e gerenciamento de sistemas elétricos (HEYDARI et al., 2020; HEYDARI et al., 2019; Heydari et al., 2018). Nesse cenário, Heydari et al. (HEYDARI et al., 2020) propõe um modelo híbrido baseado em decomposição de modo variacional (VMD), algoritmo de busca gravitational (GSA), e redes neurais de regressão. A VMD realiza a decomposição da série em diversos funções de modo intrínseco (IMFS), enquanto o GSA realia seleção das *features* na série temporal. Ademais, considerando a importância da previsão do consumo de energia em sistemas elétricos, essa tarefa pode ser também realizada para edifícios individuais através da implementação de medidores inteligentes (YU; MIROWSKI; HO, 2016; FEKRI et al., 2021). Dessa forma, Li et al. (LI et al., 2021) empregou uma rede neural LSTM Convolucional com *features* autoregressivas selecionadas para melhorar a acurácia da previsão. Fekri et al. (FEKRI et al., 2021) utiliza modelos de *deep learning* baseados em modelos online de redes neurais recorrentes adaptivas, considerando que os padrões de consumo de energia podem variar ao longo do tempo. Além disso, diversas aplicações de previsão de consumo de energia vem sendo tratadas, como alertas de pico (KOMATSU; KIMURA, 2020), onde uma Regressão por Vetores de Suporte (SVR) modificado é utilizado, por meio de dados meteorológicos e de medidores inteligentes.

Outro trabalho que lida com previsões de medidores inteligente (LUSIS et al., 2017) investigou o efeito de fatores como sazonalidade e clima na previsão do consumo de energia utilizando vários modelos de AM: árvore de regressão, MLP e SVR. As conclusões obtidas por esse trabalho indicam que árvores de regressão atingem o menor valor de Raiz do Erro Médio Quadrático (RMSE) em praticamente todos os cenários avaliados. Além disso, a adição de dados meteorológicos não melhora os resultados, e o período histórico de um ano é suficiente para realizar o treinamento dos modelos e atingir baixo erro de previsão.

Sajjad et al. (SAJJAD et al., 2020) propõe a utilização de um modelo de *deep learning* para a previsão do consumo de energia de eletrodomésticos e residências hora a hora. Os dados de entrada são processados utilizando normalização do tipo min-max ou padronização z-score, o que é introduzido numa Rede Neural Convolucional (CNN), seguido por uma Rede Neural Recorrente (RNN), especificamente uma Gated Recurrent Unit (GRU). Finalmente, uma camada densa de neurônios após a GRU é responsável pela previsão final. Não é provido, no entanto, qualquer detalhes sobre a estratégia de seleção

de hiper-parâmetros em sua rede.

De forma similar, Wang et al. (WANG et al., 2019) implementa uma rede do tipo Long Short-Term Memory (LSTM) que possui como saída a previsão de probabilidade de determinados quantis. Para treinamento, a rede minimiza o erro médio para todos os quantis. E entrada da rede consiste no histórico de consumo, o dia da semana que os dados foram previstos. Assim como em (SAJJAD et al., 2020), o processo de seleção dos nós e das camadas da rede não é apresentado.

Adicionalmente, sistemas híbridos vem ganhando atenção devido a sua habilidade de incrementar a acurácia de um modelo simples de AM (ZHANG et al., 2021; DEB et al., 2017). Esses sistemas são desenvolvidos com o objetivo de superar as limitações de modelos individuais de AM relacionados a má especificação de parâmetros, overfitting e underfitting (TASKAYA-TEMIZEL; CASEY, 2005). Nesse sentido Somu et al. (SOMU; Raman M R; RAMAMRITHAM, 2021) utiliza redes neurais convolucionais (CNN) baseados no algoritmo de clusterização K-Means e no LSTM (KCNN-LSTM) para prever o consumo de energia utilizando dados de medidores inteligentes. Nesse trabalho, o K-means é utilizado para identificar tendência e padrões sazonais nas séries temporais, enquanto a CNN-LSTM é utilizada no processo de previsão.

Chou & Truoung (CHOU; TRUONG, 2021a) propõe um sistema híbrido composto por quatro etapas: uma modelagem linear de série temporal, uma modelagem não linear dos resíduos, combinação e otimização. A seleção de parâmetros para o modelo utilizada nas três primeiras etapas consiste num algoritmo de otimização conhecido como Busca Jellyfish (JS) (CHOU; TRUONG, 2021b). Bouktif et al. (BOUKTIF et al., 2020) utiliza um algoritmo genético (GA) e Otimização Enxame de Partículas (PSO) para buscar os hiperparâmetros da LSTM tarefa de previsão da carga energética.

O sistema híbrido proposto difere dos sistemas híbridos propostos na literatura dado que implementa um GA para realizar a otimização do modelo de resíduos e do modelo de combinação em um modelo híbrido de séries temporais com combinação não-linear. Ademais, a otimização também seleciona os lags mais importantes em cada etapa para reduzir a complexidade do modelo e aumentar a acurácia da previsão.

3 FUNDAMENTAÇÃO TEÓRICA

Uma série temporal consiste numa série de observações ordenadas no tempo. A tarefa de previsão de séries temporais consiste no entendimento do processo gerador da série e a utilização de valores passados para extrapolar e estimar valores futuros de uma determinada variável de interesse (MORETTIN; TOLOI, 2006). A habilidade de prever uma série temporal de forma acurada possui diversas aplicações científicas e industriais, com exemplos que se estendem desde a medicina, engenharia e meteorologia até previsão de variáveis econômicas.

Neste capítulo, será descrito brevemente a utilização de modelos estatísticos e de aprendizado de máquina (AM) para a previsão de séries temporais. A partir de modelos mais simples como os Modelos Autoregressivos (AR) e Modelos de Médias Móveis (MA), será desenvovida uma explicação sobre o modelo SARIMA, utilizado neste trabalho. Por fim, será feita uma introdução sobre modelos híbridos e a combinação de modelos estatísticos lineares e modelos de AM para previsão de séries temporais de forma conjunta. Por fim, realizaremos uma breve descrição sobre os métodos utilizados nesse trabalho para avaliação de modelos de previsão de séries temporais.

3.1 MODELOS LINEARES

Uma abordagem clássica de previsão de séries temporais consiste no uso de modelos SARIMA (Seasonal Autoregressive Integrated Moving Average), devido à sua flexibilidade e à existência de uma metodologia bem definida proposta por Box & Jenkins (BOX et al., 2015).

Para entendermos o método de previsão do SARIMA, necessita-se a compreensão sobre Modelos Autoregressivos (AR) e modelos de Médias Móveis (MA), e por fim o modelo ARIMA, que é resultado da combinação dos dois primeiros modelos anteriores de forma integrada.

3.1.1 Modelos Autoregressivos (AR)

Os Modelos Autoregressivos (AR) são um tipo de abordagem que consistem na utilização dos valores passados da série temporal em questão para prever valores futuros. Assim, a previsão da série é calculada utilizando-se da combinação linear dos valores passados da mesma série (BROCKWELL; DAVIS; CALDER, 2002).

O número de *lags* utilizado para a previsão de um valor é definido como a ordem do modelo AR. Assim, utilizando-se os p valores passados para previsão, pode-se definir um modelo do tipo $AR(p)$, cuja representação matemática pode ser visualizada na Equação

3.1.

$$Z_t = \phi_1 Z_{t-1} + \phi_2 Z_{t-2} + ... + \phi_p Z_{t-p} + \epsilon_t \tag{3.1}$$

Dado que Z_t representa o valor da série temporal no tempo t, temos que Z_{t-1}, Z_{t-2}, ... , Z_{t-p} representam os p valores passados da série. ϕ_1 , ϕ_2 , ... , ϕ_p são coeficientes reais e ϵ_t representa um ruído aleatório representado por uma variável normalmente distribuída com média 0 e variância igual a σ^2.

Podemos também representar o modelo AR introduzindo o operador de defasagem B. Assim, temos que $Z_{t-1} = BZ_t$. Assim, a Equação 3.1 pode ser reescrita como a Equação 3.2.

$$\phi_p(B)Z_t = \epsilon_t \tag{3.2}$$

Onde $\phi_p(B)$ trata-se do operador de AR(p), definido como $\phi_p(B) = 1 - \phi_1 B - \phi_2 B^2 - ...\phi_p B^p$.

3.1.2 Modelos de Médias Móveis (MA)

Modelos de médias moveis (MA) representam as séries utilizando os valores passados dos resíduos. O resíduo é calculado considerando a diferença entre o valor previsto e o valor real da série, sendo portanto um termo de erro da previsão (BROCKWELL; DAVIS; CALDER, 2002).

Assim, a fórmula geral de um modelo MA(q) pode ser visualizada na Equação 3.3.

$$Z_t = \epsilon_t - \theta_1 \epsilon_{t-1} + ... + \theta_q \epsilon_{t-q} \tag{3.3}$$

Utilizando o operador de defasagem B, temos que a Equação 3.3 pode ser reescrita na forma da Equação 3.4.

$$Z_t = \theta_q(B)\epsilon_t \tag{3.4}$$

3.1.3 Modelos Autorregressivos e de Médias Móveis (ARMA)

Modelos Autorregressivos e de Médias Móveis (ARMA) combinam as características dos modelos AR e MA. Sendo assim, para este tipo de modelo, são considerados tantos os valores passados da série assim como os valores dos resíduos passados para a previsão (BROCKWELL; DAVIS; CALDER, 2002).

A equação resultante deste modelo é uma combinação das Equações 3.1 e 3.3, cujo resultado pode ser visualizado na Equação 3.5.

$$Z_t = \phi_1 Z_{t-1} + ... + \phi_p Z_{t-p} + \epsilon_t - \theta_1 \epsilon_{t-1} - ... - \theta_q \epsilon_{t-q} \tag{3.5}$$

Utilizando o operador de defasagem B, temos que um modelo ARMA(p,q) pode ser representado pela Equação 3.6.

$$\phi_p(B) Z_t = \theta_q(B) \epsilon_t \tag{3.6}$$

3.1.4 Modelos Autorregressivos, Integrados e de Médias Móveis (ARIMA)

No entanto, os modelos explorados acima possuem a premissa implícita de que a série analisada é estacionária. Um processo é considerado estacionário caso suas características se mantenham constantes ao longo do tempo. Ou seja, independente da escolha de τ, temos que as características de $Z(t)$ são as mesmas de $Z(t + \tau)$ (BROCKWELL; DAVIS; CALDER, 2002).

Em geral, essa definição de estacionariedade implica que estatísticas básicas calculadas da série como média, variância e covariância são constantes ao longo do tempo. (HANKE; REITSCH; WICHERN, 2001)

Em muitas aplicações, no entanto, as séries não apresentam comportamento estacionário. Uma das abordagens para transformar séries não estacionárias em estacionárias consiste na introdução do operador diferença (MORETTIN; TOLOI, 2006).

$$\Delta Z_t = Z_t - Z_{t-1} \tag{3.7}$$

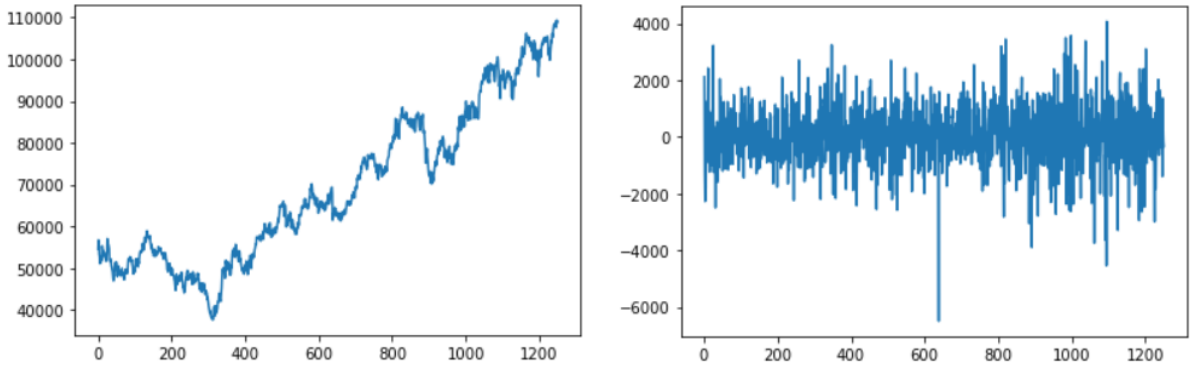
Tal operador resulta na construção de uma nova série cujo os valores finais são as diferenças em relação ao ponto anterior. Além disso, podemos estender o operador diferença para qualquer ordem d, que representam sucessivas diferenciações nas séries resultantes. Em geral, a d-ésima diferença de $Z(t)$ é dada por

$$\Delta^d Z_t = \Delta[\Delta^{d-1} Z_t] \tag{3.8}$$

Como pode-se visualizar na Figura 1, a operação de diferenciação de uma série temporal é capaz de transformar uma série não estacionária em uma série estacionária. Uma abordagem proposta por Box & Jenkins (BOX et al., 2015) para estacionarização de séries temporais consiste na utilização de um operador diferença repetidamente até as observações apresentarem propriedades de uma série estacionária (BROCKWELL; DAVIS; CALDER, 2002).

Combinando o modelo ARMA com a operação de diferenciação, obtemos os modelos do tipo ARIMA. Sendo assim, um modelo ARIMA(p,d,q) é composto por

Figura 1 – Diferenciação de primeira ordem de uma série temporal.



Fonte: Elaborado pelo Autor.

- Um modelo AR de ordem p - AR(p)
- Um modelo MA de ordem q - MA(q)
- Operação de diferença de ordem d - Δ^d

Matematicamente, o modelo ARIMA pode ser representado através da Equação 3.9, onde $\Delta^d Z_t = (1 - B)^d Z_t$ representa a operação de diferenciação de ordem d.

$$\phi_p(B)\Delta^d Z_t = \theta_q(B)\epsilon_t \tag{3.9}$$

Sendo assim, temos que no modelo ARIMA, consiste numa evolução do modelo ARMA para lidar com séries não estacionárias. Nesse caso, a série pode ser transformada em uma série estacionária por meio do processo de diferenciação dos dados. Os modelos AR(p), MA(q) e ARMA(p,q) podem ser considerados casos especiais do ARIMA, sendo representados como ARIMA(p,0,0), ARIMA(0,0,q) e ARIMA(p,0,q), respectivamente.

3.1.5 Modelos Autorregressivos, Integrados e de Médias Móveis Sazonais (SARIMA)

Em muitos casos, no entanto, a série temporal analisada pode apresentar um comportamento sazonal. Considerando séries de consumo de energia, por exemplo, o consumo pode possuir padrões cíclicos que se repetem durante o dia de acordo com a rotina do usuário. Com o intuito de lidar com variações sazonais nos valores da série, uma extensão do modelo ARIMA foi proposto por Box & Jenkins (BOX et al., 2015), denominado SARIMA.

O modelo SARIMA possui três parâmetros (P, D, Q) novos em relação ao modelo ARIMA tradicional. Tais parâmetros possuem comportamento análogo aos parâmetros representados por suas letras minúsculas equivalentes no modelo ARIMA, porém, sua

ação se refere apenas à atrasos múltiplos de S, que é definido como o período sazonal (BROCKWELL; DAVIS; CALDER, 2002).

Sendo assim, um modelo $SARIMA(0,0,0)(1,0,1)_{12}$ pode ser representado matematicamente de acordo com a Equação 3.10. Neste exemplo, temos que o efeito sazonal se repete à cada 12 observações ($S = 12$), sendo esse comportamento típico de uma série mensal com comportamento cíclico anual, por exemplo.

$$Z_t = \phi Z_{t-12} + \epsilon_t + \theta \epsilon_{t-12} \tag{3.10}$$

Sendo assim, o modelo ARIMA representa também um caso especial do modelo SARIMA, dado por $SARIMA(p,d,q)(0,0,0)_S$. Em geral, o modelo $SARIMA(p,d,q)(P,D,Q)_S$ pode ser representado de acordo com a Equação 3.11, onde $\Phi_P(B^S) = 1 - \Phi_1(B^S) - \Phi_2(B^{2S}) - ... - \Phi_P(B^{PS})$, referente à um modelo AR nos *lags* sazonais e $\Theta_Q(B^S) = 1 - \Theta_1(B^S) - \Theta_2(B^{2S}) - ... - \Theta_Q(B^{QS})$, referente à um modelo MA nos *lags* sazonais.

$$\phi_p(B)\Phi_P(B^S)\Delta^d\Delta_S^D Z_t = \theta_q(B)\Theta_Q(B^S)\epsilon_t \tag{3.11}$$

3.1.6 Metodologia Box & Jenkins

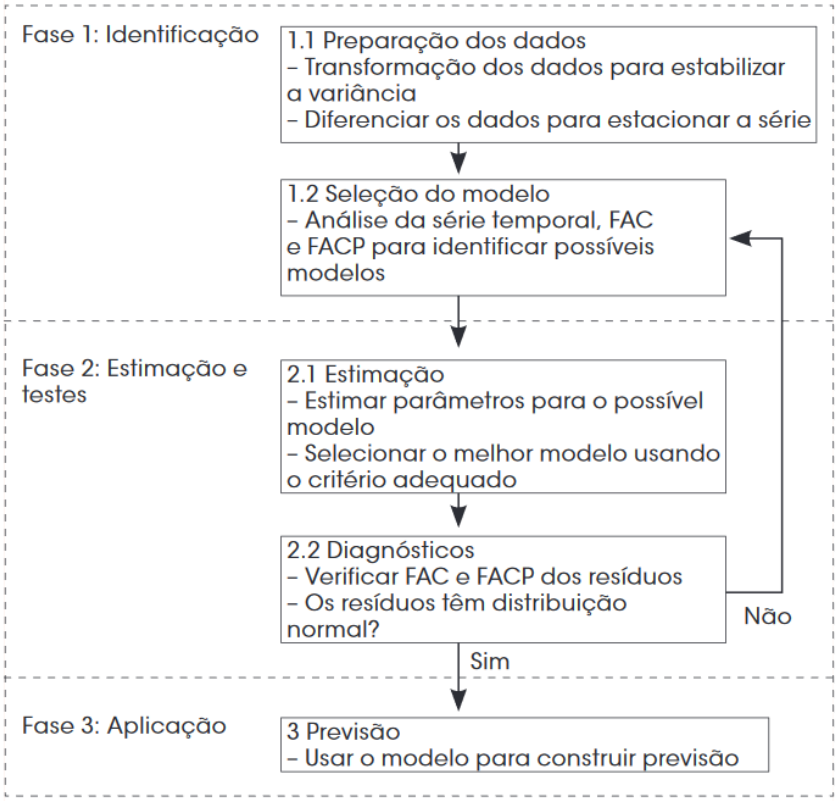
A metodologia Box & Jenkins (BOX et al., 2015) consiste numa metodologia amplamente utilizada para construção de modelos lineares do tipo ARIMA. O diagrama de todo o processo, que é composto por três etapas, pode ser visualizado na Figura 2.

Na primeira fase, é realizada a transformação dos dados da série até a mesma exibir propriedades de uma série estacionária. Durante essa etapa, testes de hipótese como o Teste de Dickey-Fuller Aumentado (SCHWERT, 2002) podem ser utilizados para verificar a estacionariedade da série. Em seguida, a função de autocorrelação (FAC) e a função de autocorrelação parcial (FACP) são utilizadas para especificação do modelo.

Na segunda fase, são estimados os coeficientes responsáveis pela previsão da série. Tal estimação pode ser realizada através de algoritmos de otimização como Mínimos Quadrados ou outros métodos de estimação não lineares (ASTERIOU; HALL, 2007). Em seguida, pode-se observar o comportamento dos resíduos do modelo, que não devem apresentar sinais de autocorrelação e devem seguir uma distribuição normal.

Por fim, na última fase, o modelo construído pode ser utilizada para prever os valores da série. O valor previsto é calculado utilizando os últimos valores em conjunto com os coeficientes calculados na fase anterior.

Figura 2 – Diagrama geral da metodologia Box & Jenkins.



Fonte: Adaptado de: (MAKRIDAKIS; WHEELWRIGHT; HYNDMAN, 2008).

3.2 MODELOS NÃO LINEARES

No entanto, a abordagem descrita na Seção 3.1 possui uma premissa implícita que o processo gerador da série temporal é linear, e portanto, não possui uma performance ótima em processos não lineares. Embora a linearidade seja em geral uma premissa útil, tem mostrado ser falsa em diversos problemas do mundo real desde a década de 80 (GOOIJER; HYNDMAN, 2006).

Como consequência, modelos de aprendizado de máquina vêm sendo apresentados como uma alternativa que consegue lidar com a não linearidade (BONTEMPI; TAIEB; BORGNE, 2012). Dentre os modelos de aprendizado de máquina comumente utilizados, pode-se citar as redes neurais do tipo multilayer perceptron (MLP) e as máquinas de vetores de suporte (SVM).

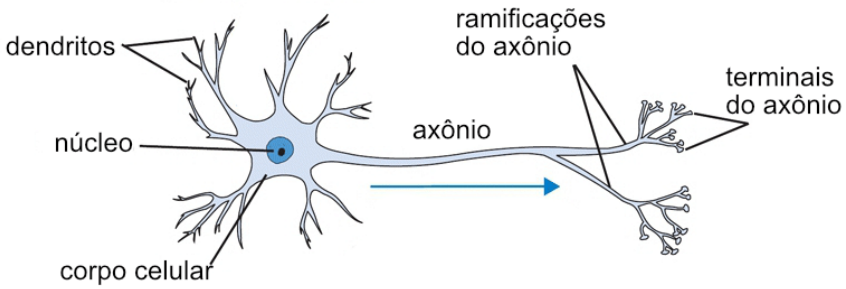
3.2.1 Redes Neurais

3.2.1.1 Neurônio Biológico

O elemento básico do sistema nervoso humano é o neurônio, cuja estrutura básica pode ser visualizada na Figura 3. Este elemento básico trata-se de uma célula excitável

capaz de propagar informações pelo cérebro.

Figura 3 – Modelo de neurônio biológico



Fonte: (BEZERRA, 2016).

Em repouso, um neurônio biológico possui uma diferença de potencial de aproximadamente $-70mV$ entre o meio intracelular e o meio extracelular, resultado da diferença entre as concentrações externas e internas de íons, que podem entrar e sair da membrana celular através de canais iônicos (KANDEL et al., 2000).

Os neurônios recebem estímulos eletroquímicos em seus dendritos, e caso a soma desses estímulos ultrapasse um determinado limiar, pode-se gerar uma variação no potencial elétrico do interior da célula, chamado de potencial de ação. Essa variação de potencial será responsável pela propagação desta informação através do axônio para a próxima célula (KANDEL et al., 2000).

A transmissão de informação para o próximo neurônio ocorre através das ligações sinápticas entre os terminais do axônio de um neurônio e os dendritos do neurônio seguinte. Quando um neurônio dispara um potencial de ação, ele transmite substâncias químicas (neurotransmissores) para o próximo grupo de neurônios.

Os neurotransmissores formam as entradas para o próximo neurônio, e constituem as mensagens enviadas de um neurônio ao outro.

Essas mensagens podem ser:

- Excitatórias - Neurotransmissores excitatórios aumentam a probabilidade do próximo neurônio disparar um potencial de ação.
- Inibitórias - Neurotransmissores inibitórios diminuem a probabilidade do próximo neurônio disparar um potencial de ação.

O cérebro humano é composto por cerca de 86 bilhões de neurônios (AZEVEDO et al., 2009). Estes neurônios estão interligados de modo a formar uma rede capaz de processar uma quantidade enorme de informações de forma simultânea. O processamento das informações realizada pelo cérebro é resultado da interação e da complexidade de interconexão entre os neurônios, que representam unidades de processamento individuais.

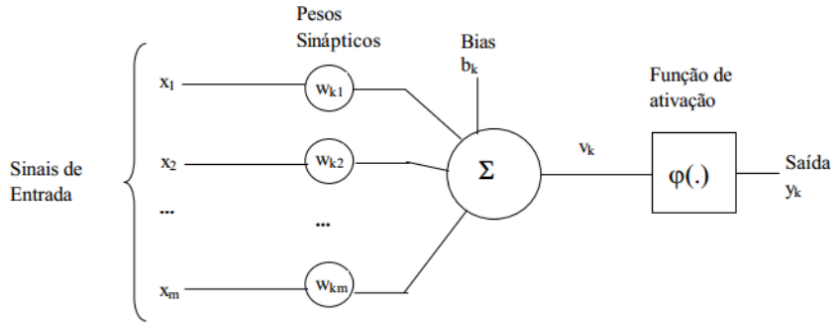
Sendo assim, os padrões presentes nas conexões entre os neurônios e a organização dos circuitos neurais são responsáveis diretos pelo funcionamento cognitivo. Todo processamento sensorial e comportamento, desde simples reflexos até tarefas cognitivas complexas, são produto da sinalização entre neurônios apropriadamente interconectados (KANDEL et al., 2000).

3.2.1.2 Redes Neurais Articiais

As redes neurais artificiais são modelos matemáticos utilizados em aprendizagem de máquina baseadas no funcionamento do cérebro humano. O elemento básico de uma rede neural consiste num neurônio artificial, uma formulação matemática do seu equivalente biológico descrito na Seção 3.2.1.1.

O modelo matemático para representação de um neurônio biológico foi proposto primeiramente por McCulloch e Pitts em 1943 (MCCULLOCH; PITTS, 1943). Esta estrutura pode ser visualizada na Figura 4.

Figura 4 – Modelo de neurônio artificial



Fonte: (HAYKIN, 2009).

Neste modelo, pode-se observar diversos elementos que buscam representar a estrutura e a funcionalidade de um neurônio biológico. Os sinais de entradas, representados por x_1, x_2, ..., x_n são equivalentes aos estímulos eletroquímicos nos dendritos do neurônio biológico.

Cada entrada x_m será multiplicada pelo seu respectivo peso w_{km}. Os pesos sinápticos são referentes às variações nas ligações sinápticas entre cada neurônio, e determinará a influência dessa entrada como estímulo (MCCULLOCH; PITTS, 1943).

Ao final, os estímulos multiplicados pelos seus respectivos pesos serão somados com um "bias", representado por b_k, que é responsável por determinar o limiar de ativação.

A soma resultante será a entrada de uma função de ativação, que é responsável por representar o comportamento não linear encontrado em neurônios biológicos (NIELSEN, 2015). Uma função de ativação consistente com o comportamento do tipo "tudo ou nada"encontrados em neurônios biológicos trata-se da função degrau, que pode ser de-

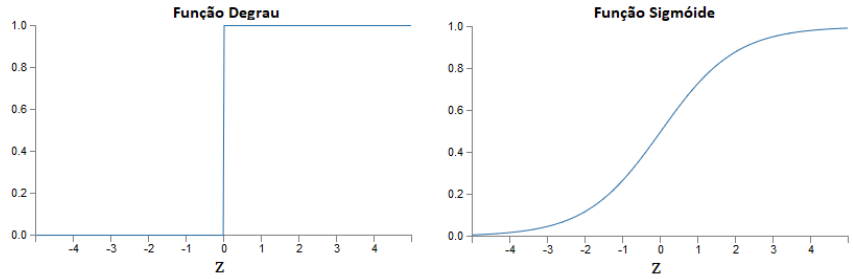
finida como $f(z) = 1$ caso sua entrada (z) seja positiva e $f(z) = 0$ caso sua entrada (z) seja negativa.

No entanto, a função degrau possui propriedades indesejáveis como função de ativação. Devido à sua descontinuidade em $x = 0$, uma variação pequena nos pesos pode causa uma grande variação na sua saída. Sendo assim, normalmente são utilizadas versões mais suaves de ativação (NIELSEN, 2015), como por exemplo a função sigmóide, que pode ser representada como

$$f(z) = \frac{1}{1 + e^{-z}} \tag{3.12}$$

A função sigmóide trata-se de uma versão suave da função de ativação do tipo degrau. Ambas as funções podem ser visualizadas na Figura 5.

Figura 5 – Funções de ativação utilizadas em neurônios artificiais



Fonte: (NIELSEN, 2015).

Sendo assim, para um neurônio k com função de ativação sigmóide, a saída deste neurônio será dada por

$$y_k = \frac{1}{1 + exp(-\sum_m (w_{km} \cdot x_m + b_k))} \tag{3.13}$$

Sendo assim, quando z é um número grande positivo, e^{-z} se aproximará de zero, e portanto $y \approx 1$. Da mesma forma, caso z seja um número alto negativo, $e^{-z} \to \infty$, assim, $y \approx 0$.

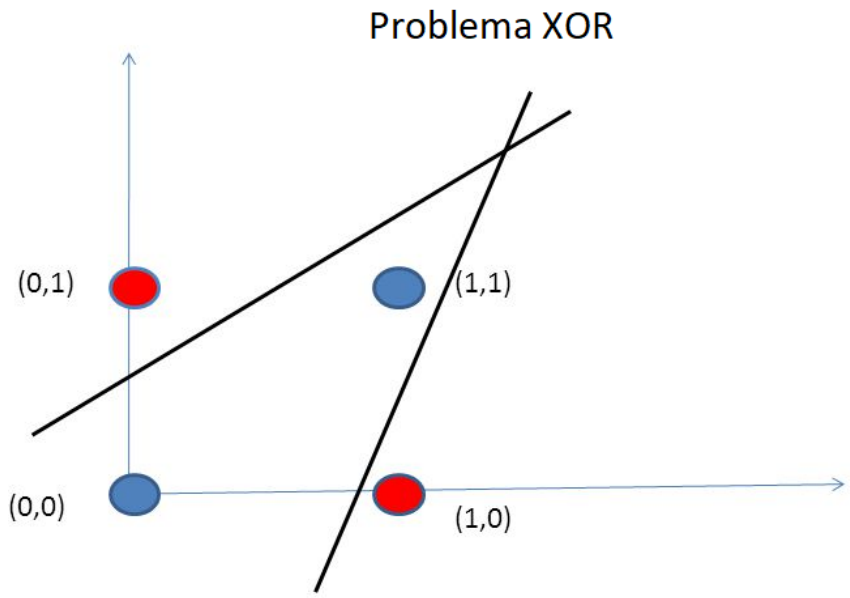
No entanto, um neurônio individual pode apenas lidar com problemas linearmente separáveis. Supondo um neurônio individual implementando uma função do tipo degrau, a saída do neurônio é resultado do seguinte processamento

$$\text{y} = \begin{cases} 0 & \text{se } w \cdot x + b \leq 0 \\ 1 & \text{se } w \cdot x + b > 0 \end{cases} \tag{3.14}$$

Neste caso, pode-se observar que um neurônio é capaz de ponderar sobre suas entradas a fim de comparar com um limiar, dado pelo bias, que causará ou não a ativação do mesmo. Sendo assim, um neurônio individual é capaz de realizar separações lineares entre suas variáveis de entrada através de um hiperplano. Por exemplo, não é possível utilizar um

neurônio para solucionar um problema do tipo XOR, já que não é possível separar as saídas do sistema através de um único hiperplano (MINSKI; PAPERT, 1969), como pode-se observar na Figura 6.

Figura 6 – Separação por hiperplanos do problema do tipor XOR



Fonte: Adaptado de (TAVARES, 2015) [1].

No entanto, esse problema pode ser resolvido através da introdução de novas camadas na rede. O cérebro humano, por exemplo, é composto por aproximadamente 86 bilhões de neurônios (AZEVEDO et al., 2009), onde cada neurônio liga-se à aproximadamente 100 outros através de sinapses, formando uma rede neural biológica de estrutura complexa.

Baseando-se nisso, pode-se desenvolver redes neurais artificiais (RNAs) que resultam dos neurônios artificiais conectados entre si em diversas camadas.

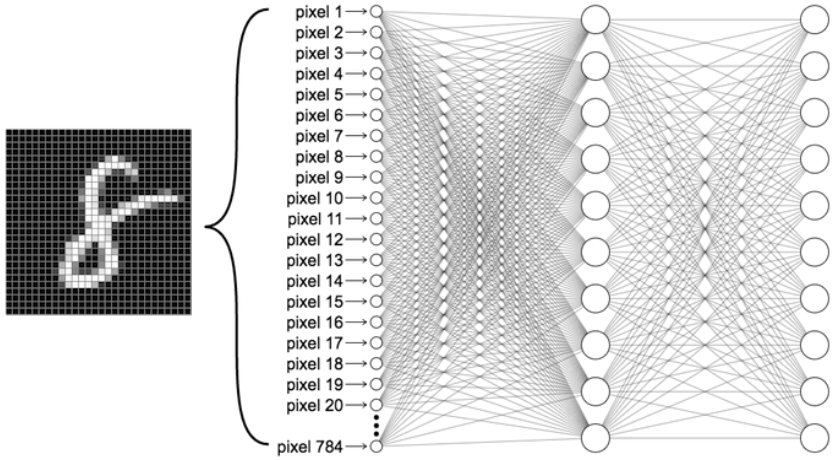
Um exemplo deste tipo de rede neural artificial é o modelo que pode ser visualizado na Figura 7.

Neste modelo, os pixels de uma imagem são inseridas como entrada da rede, que tem como função classificar o dígito presente na imagem, sendo assim um sistema de identificação de caracteres. Assim, cada neurônio age como um detector de atributos. Os neurônios da primeira camada utilizarão os pixels da imagem para realizar decisões ponderadas sobre sua entrada.

Sendo assim, contrariamente ao neurônio individual, redes neurais de múltiplas camadas são capazes de solucionar problemas que não são linearmente separáveis.

[1] Disponível em: <https://slideplayer.com.br/slide/2261325/>. Acesso em Jul. 2020.

Figura 7 – Estrutura de rede neural para reconhecimento de dígitos



Fonte: (NIELSEN, 2015).

3.2.1.3 Gradiente Descendente

Uma característica essencial dos neurônios artificiais com função de ativação sigmoidal é que uma pequena alteração nos seus pesos resulta numa pequena alteração na saída do neurônio. Sendo assim, a suavidade da curva garante a relação dada pela equação

$$\Delta y_k \approx \sum_j \frac{\partial y_k}{\partial w_j} \Delta w_j + \frac{\partial y_k}{\partial b} \Delta b \tag{3.15}$$

Assim, isso implica que a variação na saída é aproximadamente uma função linear da variação nos pesos e do bias. Logo, uma variação realizada nos pesos e no bias causará um efeito muito mais previsível, devido a sua suavidade, do que a função de ativação do tipo degrau (NIELSEN, 2015).

Após a definição das características da rede, torna-se natural a necessidade de um algoritmo que determine os melhores pesos/bias para um determinado problema. Para topologias complexas de redes neurais, a busca exaustiva torna-se computacionalmente inviável. Um algoritmo popular para encontra-se os pesos corretos para um determinado problema é conhecido como gradiente descendente. O algoritmo backpropagation se utiliza do método de gradiente descendente para determinar o valor mínimo de uma função custo. Uma função custo comumente utilizada por ser introduzida como

$$C(w, b) = \frac{1}{2n} \sum_x ||y(x) - a||^2 \tag{3.16}$$

Esta função custo é denominada Erro Quadrático Médio (Mean Squared Error, ou MSE), e trata-se de uma função que depende do conjunto dos pesos e bias da rede e cuja saída será sempre positiva (HAYKIN, 2009).

A função custo torna-se pequena quando a saída desejada da rede $y(x)$ se aproxima de uma saída a. Da mesma forma, caso tenhamos uma função custo com valor grande, isso

significa que as saídas desejadas $y(x)$ não são próximas das saídas da ativação a. Sendo assim, o objetivo do algoritmo em questão é minimizar a função custo variando os pesos e os bias da rede.

O uso da função quadrática deve-se ao fato de sua suavidade permitir que variações pequenas nos parâmetros da rede causem um efeito muito mais previsível. Além disso, como o algoritmo backpropagation procura o valor mínimo da função custo através do gradiente da mesma em cada passo, deve-se utilizar uma função de ativação que seja diferenciável e suave em todos os pontos. Daí torna-se evidente a vantagem de utilizar uma função como a função sigmóide.

Pode-se minimizar a função custo $C(w, b)$ calculando o gradiente

$$\nabla C = \left(\frac{\partial C}{\partial w_1}, \frac{\partial C}{\partial w_2}, ..., \frac{\partial C}{\partial b_l} \right)^T \tag{3.17}$$

Como o gradiente de uma função representa a inclinação no espaço n-dimensional, tem-se que para minimizar a função desejada, seus parâmetros podem ser alterados através da seguinte relação

$$\Delta w_i = -\eta \frac{\partial C}{\partial w_i} \qquad \forall \quad i = 1, ..., k \tag{3.18}$$

Onde o parâmetro η representa uma constante de aprendizagem. Portanto, os pesos após uma iteração são dados por

$$w'_k = w_k - \eta \frac{\partial C}{\partial w_k} \tag{3.19}$$

$$b'_k = b_k - \eta \frac{\partial C}{\partial b_k} \tag{3.20}$$

Assim, sem perda de generalidade, pode-se definir a atualização de um parâmetro arbitrário θ em um passo t de acordo com a seguinte relação:

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta C \tag{3.21}$$

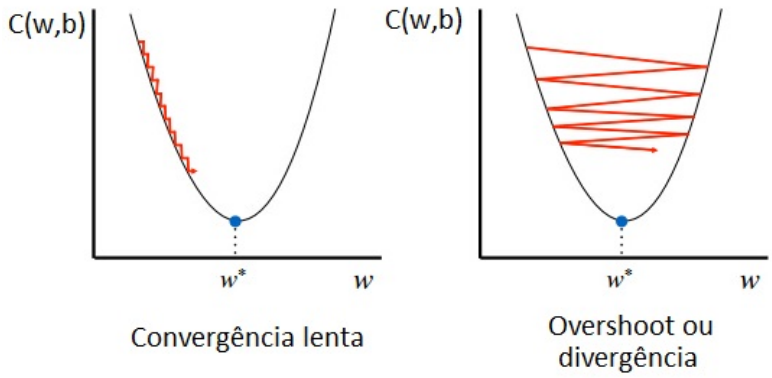
Sendo assim, o problema de calcular os pesos onde a saída se aproxima de uma referência y se tornou um problema de encontrar o mínimo da função custo.

A constante de aprendizagem é responsável por determinar o tamanho do passo realizado em cada iteração do algoritmo. Assim, para uma constante de aprendizagem pequena, a convergência será lenta. No lado oposto, caso se tenha uma constante de aprendizagem muito alta, o comportamento será errático ou até mesmo divergente. Assim, a escolha da taxa de aprendizagem ideal consiste em uma importante tarefa para garantir uma convergência estável e rápida.

Além disso, existem alguns problemas em implementar este algoritmo. Um desses problemas deve-se ao fato da função custo ser resultado de um somatório que envolve

todos os exemplos de treinamento. Na prática, para calcular o gradiente ∇C, deve-se que calcular o gradiente individual para cada exemplo e fazer a média aritmética. Quando tem-se um problema com muitos exemplos de treinamento, esse processo pode tornar-se extremamente lento. Esse tipo de comportamento pode ser observado na Figura 8.

Figura 8 – Comportamento do algoritmo de gradiente descendente para situação de taxa de aprendizado pequena (à esquerda) e grande (à direita)



Fonte: Adaptado de (VARMA; DAS, 2018).

Portanto, uma solução para este problema é realizar o **gradiente descendente estocástico** , onde apenas um exemplo de treinamento é escolhido de forma aleatória e utilizado para calcular uma aproximação de ∇C.

Como meio-termo, é possível utilizar o método de **aprendizado em lotes (batch)**, onde é utilizado um subgrupo dos exemplos de treinamento para calcular uma aproximação ∇C. O tamanho de exemplos utilizados trata-se de um parâmetro definido como *tamanho do lote*.

Assim, neste caso, ∇C pode ser aproximado da forma

$$\nabla C \approx \frac{\sum_{j=1}^{m} \nabla C_{X_j}}{m} \tag{3.22}$$

Onde o somatório em C_{X_j} representa o somatório nos exemplos inclusos no **lote** e m representa o tamanho escolhido para o *lote*. Assim, o gradiente descendente estocástico consiste em um caso especial do gradiente descendente baseado em lotes, onde $m = 1$.

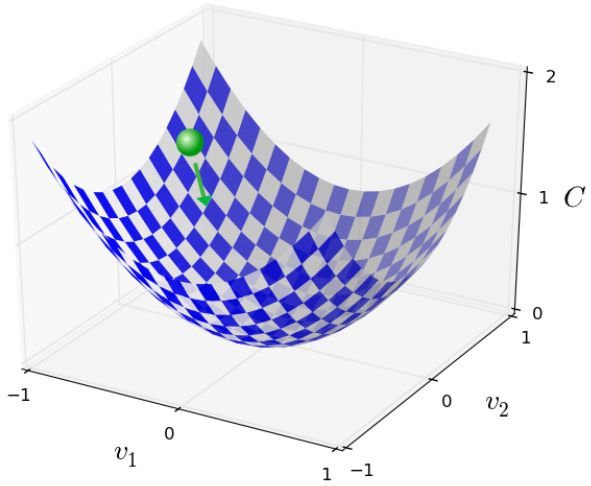
Assim, a atualização dos pesos pode ser feito da seguinte forma

$$w'_k = w_k - \frac{\eta}{m} \sum_j \frac{\partial C_{X_j}}{\partial w_k} \tag{3.23}$$

$$b'_k = b_k - \frac{\eta}{m} \sum_j \frac{\partial C_{X_j}}{\partial b_k} \tag{3.24}$$

As equações de atualização dos pesos produzirão uma série de passos que irão direcionar os pesos para uma posição de mínimo. Sendo assim, utilizando o **batch** para obter uma estimativa do gradiente e atualizando os pesos iterativamente, estaremos nos aproximando do mínimo local da função custo, como pode-se observar na Figura 9.

Figura 9 – Atualização dos pesos e dos bias de acordo com o gradiente da função custo



Fonte: (NIELSEN, 2015).

3.2.1.4 Algoritmos de Otimização do Gradiente Descendente

3.2.1.4.1 Momento

Um problema do algoritmo do Gradiente Descendente deve-se ao fato de que este tipo de algoritmo pode ter uma performance reduzida em locais onde o gradiente possui vales, levando à oscilações em torno de um mínimo local (SUTTON, 1986). Portanto, o algoritmo padrão de gradiente descendente pode levar à uma convergência lenta após os passos iniciais.

Um dos métodos utilizados para suavizar tais comportamentos erráticos e garantir uma convergência mais rápida consiste em registrar o histórico de atualizações dos pesos e bias e incorporá-lo nas regras de atualização.

Sendo assim, pode-se substituir a regra do gradiente descendente dadas pelas Equações 3.23 e 3.24 para incorporar um termo denominado *momento*. Neste caso, a atualização dos pesos torna-se uma combinação linear do gradiente calculado e da última atualização realizada.

Matematicamente,

$$w'_k = w_k + v'_k \tag{3.25}$$

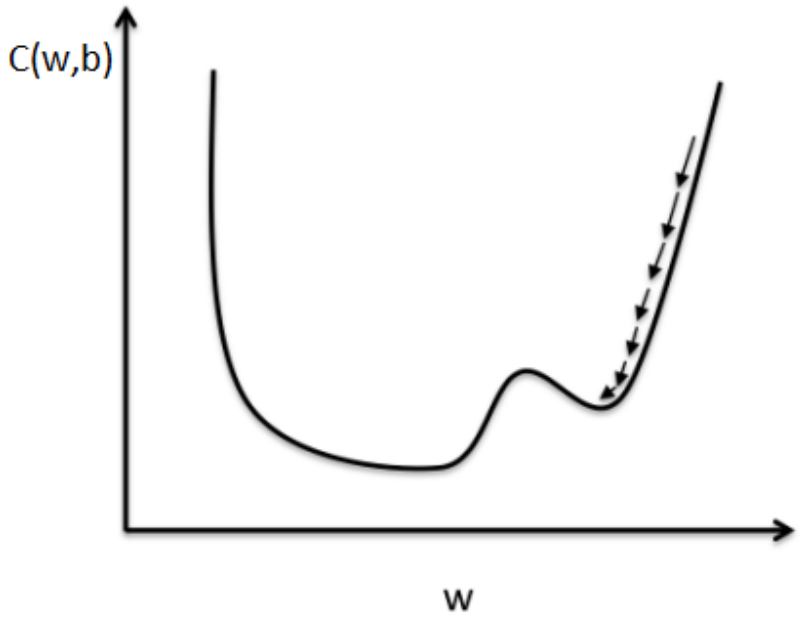
Onde

$$v'_k = \mu v_k - \eta \frac{\partial C}{\partial w_k} \tag{3.26}$$

A técnica se diferencia do gradiente descendente padrão por introduzir um vetor v, que representa os valores de velocidade atual em cada passo realizado. A relação visualizada nas Equações 3.25 e 3.26 permitem com que a variação dos pesos possuam uma inércia, aproximando-se assim da imagem física de uma esfera deslizando por um vale, visualizada na Figura 9. Assim, ao deslizar por um vale na direção correta, a esfera acumulará momento. Isso é responsável por aumentar a velocidade de convergência quando estivemos andando na direção correta.

Além disso, esse tipo de abordagem, ao realizar alterações na velocidade, ao invés de diretamente na posição, permite com que oscilações devido à estimação do gradiente de forma estocástica sejam reduzidas e evita com que os pesos encontrem-se presos em pequenos vales de mínimo local (HAGAN et al., 1996). Caso tenha-se um pequeno vale, como visualizado na Figura 10, a esfera terá momento acumulado para ultrapassar o mínimo local.

Figura 10 – Pequeno vale de mínimo local evitado pela implementação do momento



Fonte: iT State Help - Deep learning[2].

Assim, o termo de momento agirá como um filtro passa baixa para irregularidades locais do gradiente, reduzindo assim as oscilações locais da trajetória (HAGAN et al., 1996).

3.2.1.4.2 Adagrad

O algoritmo de gradiente descendente padrão realiza a atualização dos parâmetros utilizando-se da mesma taxa de aprendizado para todos.

O algoritmo **Adagrad**(DUCHI; HAZAN; SINGER, 2011) utiliza uma taxa de aprendizado diferente para cada parâmetro em cada iteração de aprendizagem. Assim, o Adagrad

[2] Disponível em: <https://ithelp.ithome.com.tw/articles/10189086>. Acesso em Jul. 2020.

modifica as Equações 3.23 e 3.24 para cada iteração do parâmetro w_i baseando-se nos gradientes computados anteriormente. Matematicamente,

$$w'_k = w_k - \frac{\eta}{\sqrt{\sum_{\tau=1}^{t} g_\tau^2 + \epsilon}} \frac{\partial C}{\partial w_k} \tag{3.27}$$

$$b'_k = b_k - \frac{\eta}{\sqrt{\sum_{\tau=1}^{t} g_\tau^2 + \epsilon}} \frac{\partial C}{\partial b_k} \tag{3.28}$$

Onde g_i representa o gradiente do parâmetro na iteração atual i. Além disso, o termo ϵ trata-se de uma constante utilizada para evitar a divisão por zero. Assim, o algoritmo mantém uma taxa da aprendizado por parâmetro que melhora a performance em problemas de gradiente esparsos, como nos casos de linguagem natural e visão computacional (DUCHI; HAZAN; SINGER, 2011).

3.2.1.4.3 RMSProp

Um dos problemas do algoritmo Adagrad consiste no fato de que ao aumentar-se o número de iterações, a soma dos gradientes ao quadrado torna-se um valor grande, o que faz com que a taxa de aprendizado efetiva torne-se muito pequena.

O algoritmo RMSProp (TIELEMAN; HINTON, 2012) resolve este problema acumulando apenas os valores de gradiente de uma janela fixa que contém as últimas k iterações. Para isso, utiliza-se um sistema de média móvel, que pode ser definido como

$$E[g^2]_t = \beta E[g^2]_{t-1} + (1 - \beta) g_t^2 \tag{3.29}$$

Assim, o RMSProp, ao contrário do AdaGrad, utiliza apenas os últimos valores de gradiente no seu histórico, com o intuito de resolver o problema da diminuição da taxa de aprendizado encontrado na técnica AdaGrad.

Assim, a relação de atualização dos pesos pode ser expressa, de forma generalizada para um parâmetro θ, como

$$g_{t+1} = \beta g_t + (1 - \beta)(\nabla_{\theta_t} C)^2 \tag{3.30}$$

$$\theta_{t+1} = \theta_t - \frac{\eta \nabla_\theta C}{\sqrt{g_{t+1} + \epsilon}} \tag{3.31}$$

Onde g_t é denominado momento de segunda ordem de ∇C.

3.2.1.4.4 Adam - Adaptive Moment Estimation

O algoritmo Adam (KINGMA; BA, 2014) trata-se de um algoritmo similar ao RMSProp, com a diferença que além de se manter uma média móvel dos gradientes ao quadrado,

denominado momento de segunda ordem do gradiente, também é mantida uma média do gradiente em si, definido como o momento de primeira ordem.

Assim, m_t e v_t representam os momentos de primeira e segunda ordem do gradiente, respectivamente, e são dados por

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \tag{3.32}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \tag{3.33}$$

Como os momentos de primeira e segunda ordem do gradiente são inicializados com valor zero, um termo de correção é utilizado para neutralizar o bias resultante em favor do ponto zero, especialmente nos steps iniciais ou quando os fatores β são muito próximos de 1 (taxas de decaimento baixas). Assim, os valores de momento finais são dados por

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \tag{3.34}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \tag{3.35}$$

E assim, a regra de atualização dos parâmetros utilizada pelo Adam é dada por

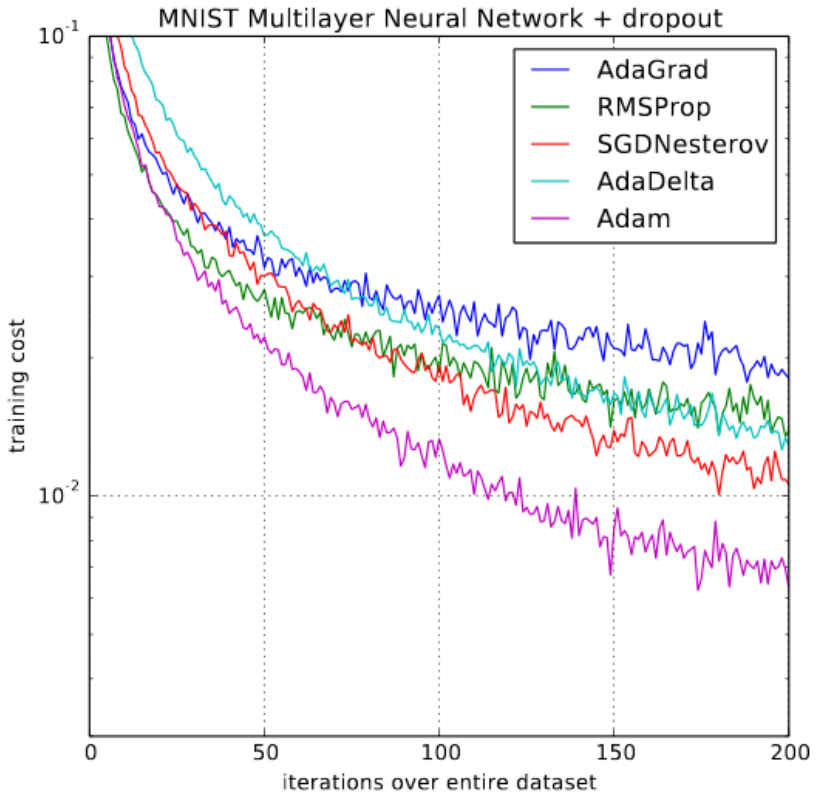
$$\theta_{t+1} = \theta_t - \frac{\eta \hat{m}_{t+1}}{\sqrt{\hat{v}_{t+1} + \epsilon}} \tag{3.36}$$

Utilizando elementos do RMSProp e do Adagrad, o algoritmo Adam mostra-se empiricamente superior à outros algoritmos de otimização com parâmetros adaptativos (KINGMA; BA, 2014), como pode-se observar na Figura 11.

Assim, o algoritmo final utilizado neste trabalho consiste no gradiente descendente com o otimizador Adam. O algoritmo final de treinamento consiste em

1. Inicializar momentos de primeira e segunda ordem do gradiente como $m_0 = v_0 = 0$.
2. Inicializar o contador de passo $t = 0$
3. Enquanto não há convergência:
    a) Incrementar o valor de passo ($t = t + 1$)
    b) Calcular o gradiente baseado nos parâmetros θ atuais $g_t = \nabla_\theta C(\theta_{t-1})$
    c) Calcular o momento de primeira ordem do gradiente $m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$
    d) Calcular o momento de segunda ordem do gradiente $v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$
    e) Computar a correção do bias para o momento de primeira ordem $\hat{m}_t = m_t/(1 - \beta_1^t)$
    f) Computar a correção do bias para o momento de segunda ordem $\hat{v}_t = v_t/(1 - \beta_2^t)$

Figura 11 – Comparação do algoritmo de otimização Adam com outros algoritmos de otimização do gradiente descendente



Fonte: (KINGMA; BA, 2014).

g) Atualizar os parâmetros $\theta_t = \theta_{t-1} - (\eta\hat{m}_t)/(\sqrt{\hat{v}_t} + \epsilon)$

4. Retornar vetor de parâmetros finais θ_t

3.2.1.5 O Algoritmo Backpropagation

Para aplicar o algoritmo de gradiente descendente, deve-se primeiramente ser capaz de calcular o gradiente ∇C em relação à cada peso e bias. Para isso, pode-se utilizar o algoritmo backpropagation (RUMELHART; HINTON; WILLIAMS, 1986), esse algoritmo é responsável por determinar a influência de cada um desses parâmetros no erro da função custo, dado que esta função é uma cadeia de funções. Sendo assim, utiliza-se a regra da cadeia para determinar o gradiente propagando pelas camadas de forma inversa.

Primeiramente, será utilizado w^l_{jk} para denominar o peso da camada do neurônio k na camada $l-1$ para o neurônio j da camada l. Da mesma forma, b^l_j se refere ao bias do neurônio j da camada l.

Assim, a ativação do neurônio j na camada l é dada por

$$a^l_j = \sigma\left(\sum_k w^l_{jk} a^{l-1}_k + b^l_j\right) \tag{3.37}$$

Para facilitar, pode-se utilizar uma forma matricial e determinar w^l e b^l como uma matriz que contém os pesos e bias de uma camada l, respectivamente.

Assim, na forma matricial, definindo $z^l \equiv w^l a^{l-1} + b^l$, pode-se escrever

$$a^l = \sigma\left(w^l a^{l-1} + b^l\right) = \sigma(z^l) \tag{3.38}$$

Para um exemplo de treinamento, pode-se calcular a função custo da seguinte forma

$$C = \frac{1}{2}||y - a^L||^2 \tag{3.39}$$

Assim, para possibilitar a derivação deste gradiente, utiliza-se a regra da cadeia na camada de saída para encontrar a cadeia de funções que irá compor a saída da camada final.

Primeiramente, definimos o erro do neurônio j na camada l da forma

$$\delta_j^l \equiv \frac{\partial C}{\partial z_j^l} \tag{3.40}$$

Assim, δ^l representa o vetor que contem os erros dos neurônios da camada l. Assim, aplicando a regra da cadeia nessa definição para a última camada, obtém-se

$$\delta_j^L = \frac{\partial C}{\partial a_j^L}\frac{\partial a_j^L}{\partial z_j^L} \tag{3.41}$$

Na forma matricial, pode-se descrever o erro da última camada como

$$\delta^L = \nabla_a C \odot \sigma'(z^L) \tag{3.42}$$

Onde ∇_a representa o gradiente da função custo que contém as derivadas $\partial C/\partial a_j^L$ e o símbolo $\odot$ representa o produto de Hadamard entre as matrizes.

Assim, obtendo o valor de δ^L, torna-se necessário derivar uma relação que estabeleça os valores de δ^l para as camadas anteriores(NIELSEN, 2015). Isso pode ser feito através da regra da cadeia, fazendo-se

$$\delta_j^l \equiv \frac{\partial C}{\partial z_j^l} = \sum_k \frac{\partial C}{\partial z_k^{l+1}}\frac{\partial z_k^{l+1}}{\partial z_k^l} = \sum_k \frac{\partial z_k^{l+1}}{\partial z_k^l}\delta_k^{l+1} \tag{3.43}$$

Notando-se que

$$z_k^{l+1} = \sum_j w_{kj}^{l+1} a_j^l + b_k^{l+1} = \sum_j w_{kj}^{l+1}\sigma(z_j^l) + b_k^{l+1} \tag{3.44}$$

Diferenciando essa relação, obtém-se

$$\frac{\partial z_k^{l+1}}{\partial z_j^l} = w_{kj}^{l+1}\sigma'(z_j^l) \tag{3.45}$$

Substituindo esta relação na equação 3.43, obtém-se

$$\delta_j^l = \sum_k w_{kj}^{l+1} \delta_k^{l+1} \sigma'(z_j^l) \tag{3.46}$$

Portanto, na forma matricial,

$$\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l), \tag{3.47}$$

Assim, pode-se calcular o erro de cada camada começando pela última camada, utilizando a equação 3.42 e posteriormente utilizando a Equação 3.47 para calcular o erro das camadas anteriores. Esse fato explica o motivo do algoritmo se chamar **backpropagation**, pois começamos a calcular os erros a partir da ultima camada e propagamos até retornar à primeira.

Assim, obtendo os elementos das matrizes δ^l, é possível calcular o gradiente da função custo utilizando as seguintes relações

$$\frac{\partial C}{\partial w_{jk}^l} = a_k^{l-1} \delta_j^l \tag{3.48}$$

$$\frac{\partial C}{\partial b_j^l} = \delta_j^l \tag{3.49}$$

Assim, algoritmo pode ser sintetizado da seguinte forma:

1. Calcular a ativação da primeira camada a^1
2. Para cada $l = 2, 3, ..., L$ calcular $z^l = w^l a^{l-1} + b^l$ e $a^l = \sigma(z^l)$
3. Calcular o erro da última camada $\delta^L = \nabla_a C \odot \sigma'(z^L)$
4. Propagar o erro para as camadas anteriores $l = L-1, L-2, ..., 2$ fazendo $\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l)$
5. Obter o gradiente da função custo através das relações $\frac{\partial C}{\partial w_{jk}^l} = a_k^{l-1} \delta_j^l$ e $\frac{\partial C}{\partial b_j^l} = \delta_j^l$

3.2.1.6 Long Short-Term Memory (LSTM)

Os modelos de redes neurais apresentados até esta seção consistem em modelos do tipo *feedforward*. Nesse caso, o fluxo de informação segue em camadas de forma unidirecional e uma entrada única é responsável por determinar a saída final. No entanto, existem problemas onde a saída deve ser determinada não somente pela entrada atual mas por ativações passadas na rede (NIELSEN, 2015).

Uma classe específica de redes neurais, conhecidas como *redes neurais recorrentes (RNN)*, introduzem conexões de *feedback* na arquitetura, o que permite a variação do

comportamento da rede no tempo. Tal abordagem é direcionada majoritariamente para modelagem de processos que variam no tempo, como reconhecimento de fala, modelos de linguagem e séries temporais (DENG; LIU, 2018).

Um problema que afeta as soluções iniciais baseadas em RNNs é que instabilidades nos gradientes podem causar maiores dificuldades na etapa de treinamento, comparando-se com redes do tipo *feedforward*. O problema se torna pior pois os gradientes não são propagados apenas pelas camadas mas também pelo tempo, o que pode tornar o treinamento extremamente instável (GOODFELLOW et al., 2016).

Assim, redes neurais do tipo recorrente possuem enorme dificuldade em estabelecer relações de longo prazo pois sofrem do problema do desaparecimento do gradiente (*vanishing gradient*), onde o gradiente se torna pequeno ao longo do *backpropagation* no tempo e não contribui muito para o aprendizado (NIELSEN, 2015).

Uma das soluções propostas para ligar com o problema da instabilidade do gradiente consiste na introdução de arquiteturas do tipo Long Short-Term Memory (LSTM), idealizado por Hochreiter & Schmidhuber em 1997 (HOCHREITER; SCHMIDHUBER, 1997). Tais redes são um tipo especial de rede neural recorrente projetadas para lidar melhor com relações à longo prazo (DENG; LIU, 2018).

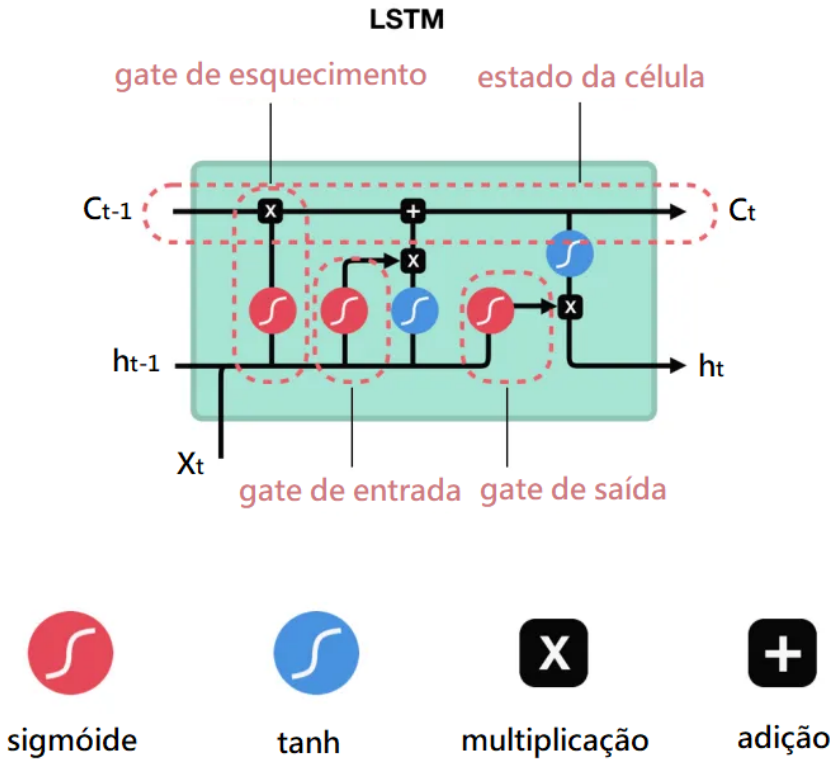
A estrutura básica de uma rede LSTM consiste na introdução de duas linhas, denominadas**estado da célula** e **estado escondido**, como pode-se visualizar na Figura 12. O LSTM tem a capacidade de remover ou adicionar informações ao estado da célula, com a utilização de *gates* que controlam a passagem de informação entre dois canais. Sendo assim, um dos valores de entrada do *gate* controla qual porcentagem de cada valor deve passar pela porta.

Sendo assim, a informação X_t é concatenada com o estado escondido do tempo anterior h_{t-1} e sua ativação controla o gate de esquecimento, responsável por definir quais partes do estado da célula continuam armazenados a partir do estado da célula anterior (C_{t-1}). O gate de entrada define quais informações da memória de curto prazo devem ser adicionadas ao estado da célula e o gate de saída define quais partes do estado da célula serão relevantes para a saída final (C_t). A ativação do estado escondido final (h_t) também é passado para a o próximo passo e armazenado na memória.

Com essa estrutura, o armazenamento de informações transforma-se em um comportamento padrão, permitindo assim lidar melhor com relações de longo prazo. Sendo assim, redes do tipo LSTM vêm ganhando popularidade na previsão de séries temporais (SIAMI-NAMINI; TAVAKOLI; NAMIN, 2018). No entanto, deve-se observar que ainda assim, a sua implementação mostra-se extremamente mais custosa, demandando um maior poder computacional para realizar a etapa de treinamento.

[3] Disponível em: <http://www.deeplearningbook.com.br>. Acesso em Mar. 2021.

Figura 12 – Estrutura básica de uma célula LSTM.



Fonte: Adaptado de Data Science Academy, 2019[3]

3.2.2 Máquinas de Vetores de Suporte (SVM)

Máquinas de Vetores de Suporte (Support Vector Machines - SVM) são uma técnica de aprendizado de máquina que obtiveram bastante sucesso por exigir poucos parâmetros e serem robustos à outliers, além de possuírem uma forte fundamentação na teoria de aprendizado estatístico (STEINWART; CHRISTMANN, 2008). Foram introduzidas por Vapnik para solucionar problemas de classificação, baseando-se em uma generalização do algoritmo *Generalized Portrait* (VAPNIK, 1963). Porém, este tipo de algoritmo também pode ser utilizado para problemas de regressão.

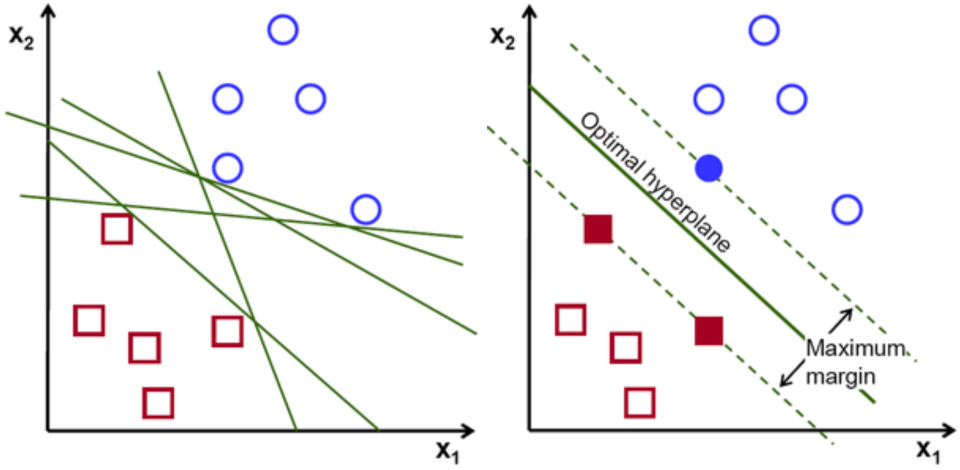
O algoritmo consiste na utilização de um hiperplano de dimensão $n-1$ para separação das classes em um espaço n-dimensional de características. Um hiperplano pode ser descrito pela relação linear da Equação 3.50.

$$a_1x_1 + a_2x_2 + ...a_nx_n = b \tag{3.50}$$

Portanto, para um determinado conjunto de dados, existem infinitas soluções possíveis para as equações do hiperplano, dada pelas possibilidades de escolha dos parâmetros $a_1, a_2, ..., a_n$ e b. No caso de um problema com duas características (features), por exemplo, as classes podem ser divididas por infinitas possibilidades de linhas, conforme mostra a

Figura 13.

Figura 13 – Diferentes possibilidades de hiperplanos de separação entre as classes.



Fonte: (GANDHI, 2018) [4]

Na figura acima, temos duas classes (círculos azuis e quadrados vermelhos), que são apresentadas como pontos em um gráfico de dispersão, onde as coordenadas cartesianas representam os valores das *features* de cada exemplo. Assim, temos que apesar de todas as linhas presentes na Figura 13 separarem as classes corretamente, para novos pontos, haverá um comportamento diferente nas regiões de fronteiras. Assim, um algoritmo que deseja realizar a otimização deve determinar um critério para escolher a linha que melhor se ajusta ao problema. No caso do SVM, a melhor linha é definida como a linha que maximiza a margem entre as classes e o hiperplano. A margem é definida como a distância entre o hiperplano e o ponto mais próximo de cada classe. Assim, o hiperplano ótimo consiste no hiperplano que maximiza a distância entre si e os exemplos de fronteira de cada classe.

$$\begin{cases} min_{\mathbf{w},b} \quad \frac{1}{2}||w||^2 \\ y^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + b) \geq 1 \end{cases} \tag{3.51}$$

Com o intuito de adaptar este modelo para situações mais genéticas, introduz-se o conceito de margem suave, onde permite-se que alguns exemplos possam violar a Equação 3.51. Isso é feito através da introdução de uma variável de folga ξ e uma constante de regularização C.

Assim, suavizamos as margens do classificador, permitindo assim a ocorrência de alguns erros na classificação (SMOLA et al., 2000). A otimização de um hiperplano pode, portanto, ser sumarizada em um problema de otimização conforme a Equação 3.52. A presença do segundo termo pode ser vista como uma minimização dos erros, dado que um

[4] Disponível em: <https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47>. Acesso em Jan. 2021.

valor $\xi \in (0, 1]$ indica um dado dentro das margens (SMOLA et al., 2000). O parâmetro C determina o impacto das classificações erradas na função custo durante o treinamento.

$$\begin{cases} min_{\mathbf{w},b,\xi} \quad \frac{1}{2}||w||^2 + C\sum_{i=1}^{m}\xi^{(i)} \\ y^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + b) \geq 1 - \xi^{(i)} \\ \xi^{(i)} \geq 0 \text{ para } \ i = 1, ..., m \end{cases} \tag{3.52}$$

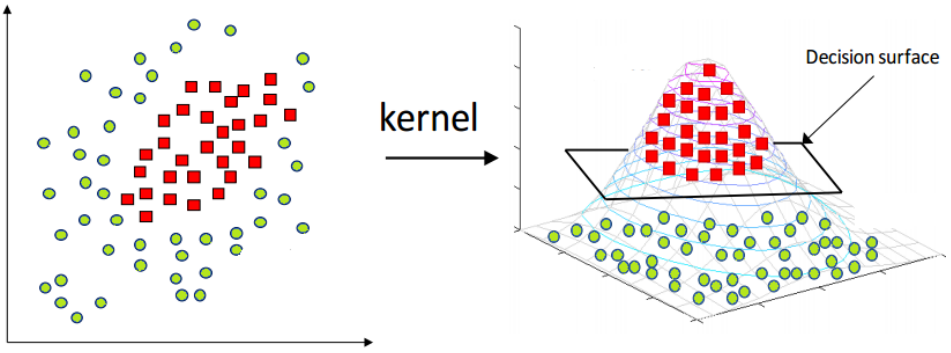
3.2.2.1 O Truque do Kernel

Um dos problemas do modelo SVM, similar ao problema encontrado ao perceptron, é que tal abordagem apenas consegue lidar com problemas linearmente separáveis, pois utiliza um hiperplano para separação das classes. Com o intuito de solucionar esse problema, torna-se essencial a introdução do conceito de Kernel no SVM.

O "truque"consiste no mapeamento do espaço original de *features* de uma dimensão menor, onde os dados não são linearmente separáveis, para uma maior dimensão onde é possível realizar uma separação linear (HERBRICH, 2001). Portanto, é realizada uma projeção apropriada $\Psi : X \rightarrow \Omega$ do espaço original X para um espaço de maior de dimensionalidade Ω, com o objetivo de permitir a separação linear do conjunto de dados, como pode-se observar na Figura 14. No novo espaço Ω, pode-se aplicar, portanto, um SVM Linear.

Esse procedimento é motivado pelo teorema de Cover (HAYKIN, 2009). Tal teorema afirma que o espaço de características X pode ser transformado em um espaço de características Ω onde os dados serão linearmente separáveis com alta probabilidade.

Figura 14 – Visualização mapeamento para dimensões maiores, permitindo a separação entre classes de forma linear.



Fonte: (SHARMA,2019) [5]

No entanto, para realizar o treinamento de um SVM, não é necessário o cálculo explícito do mapeamento Ψ (CRISTIANINI; SHAWE-TAYLOR et al., 2000), sendo assim utilizadas

[5] Disponível em: <https://medium.com/analytics-vidhya/how-to-classify-non-linear-data-to-linear-data-bb2df1a6b781>. Acesso em Jan. 2021.

as funções Kernel. Por definição, um Kernel consiste numa função que computa o produto escalar entre os pontos x e x' no espaço de características (HERBRICH, 2001), podendo ser definido como

$$K(x, x') = \Psi(x) \cdot \Psi(x') \tag{3.53}$$

Sendo assim, os Kernels possuem uma representação da transformação no espaço de features de forma simplificada (CRISTIANINI; SHAWE-TAYLOR et al., 2000). Na definição dos Kernels, também é possível definir alguns parâmetros que devem ser determinados pelo usuário.Os Kernels mais utilizados na prática são

- Linear
$$K(x, x') = (x \cdot x')$$
- Polinomial
$$K(x, x') = (\gamma(x \cdot x') + r)^d$$
- RBF
$$K(x, x') = e^{-\gamma||x-x'||^2}$$
- Tangente Hiperbólica
$$K(x, x') = tanh(\gamma(x \cdot x') + r)$$

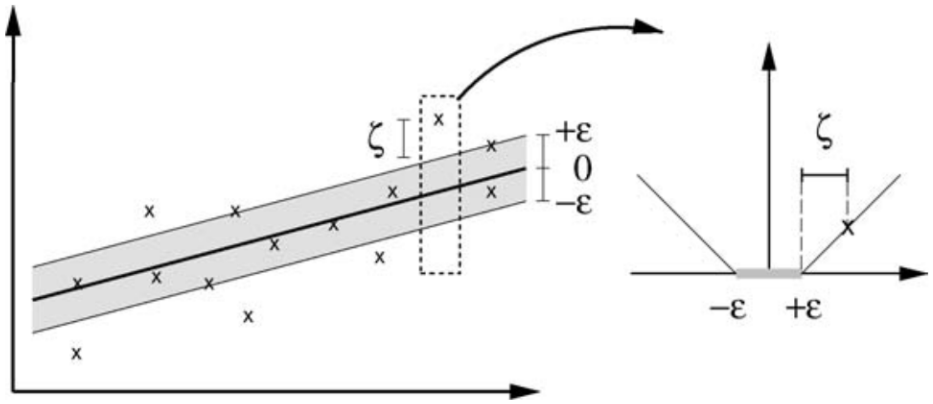
3.2.2.2 Regressão por Vetores de Suporte (SVR)

Máquinas de vetores de suporte também podem ser utilizados para solucionar problemas de regressão, onde o conjunto de dados é formado por pares de valores contínuos $(x^{(i)}, y^{(i)})$. Nesse caso, o SVM funciona de maneira similar ao problema de classificação, porém com algumas diferenças importantes no problema de otimização a ser resolvido e na implementação da função custo.

Na classificação, o objetivo trata-se de maximizar a margem entre a fronteira das classes e o hiperplano responsável pela separação das mesmas. Na Regressão por Vetores de Suporte (SVR), o objetivo trata-se de minimizar o desvio entre os pontos da curva a ser modelada e o hiperplano final. Nesse caso, também é utilizado o conceito de margem, definido como ε, onde exemplos que se encontram dentro da margem não possuem impacto na função custo final (ALPAYDIN, 2020).

Em outras palavras, temos que o objetivo é encontrar uma previsão que possua no máximo um desvio de ε com os valores verdadeiros obtidos (SMOLA; SCHÖLKOPF, 2004). Logo, não há impacto na função custo desde que eles se encontrem menor do que ε. A Figura 15 demonstra visualmente o impacto dos pontos fora da região escura na função custo.

Figura 15 – Definição da margem ε e impacto dos pontos fora da margem na função custo.



Fonte: (SMOLA; SCHÖLKOPF, 2004)

De forma análoga ao conceito de margem suave utilizado no SVM para classificação, pode-se definir um valor C que representa um *trade-off* entre a maximização da margem e a quantidade de desvio acima de ε que é permitido. Portanto, o problema de otimização a ser resolvido é transformado na Equação 3.54.

$$\begin{cases} min_{w,b,\xi}\ \frac{1}{2}||w||^2 + C\sum_t(\xi_+ + \xi_-) \\ y^{(i)} - (w^T x^{(i)} + b) \leq \epsilon + \xi_+^t \\ (w^T x^{(i)} + b) - y^{(i)} \leq \epsilon + \xi_-^t \\ \xi_+^t, \xi_-^t \geq 0 \end{cases} \tag{3.54}$$

3.3 MODELOS HÍBRIDOS

Porém, devido à sua flexibilidade superior, modelos de aprendizado de máquina são propensos sofrer problemas decorrentes da má especificação de parâmetros (DOMINGOS; OLIVEIRA; NETO, 2019), como *overfitting* e *underfitting*. Ainda, existem resultados mistos referentes à performance de modelos não lineares comparando-se com modelos clássicos como o ARIMA (TEALAB, 2018). A conclusão obtida por trabalhos de revisão da literatura como Paliwal & Kumar (PALIWAL; KUMAR, 2009) não apresentam evidências estatisticamente significantes a favor dos modelos não lineares considerando-se a performance de previsão de séries temporais em relação aos modelos lineares. Dado esse fato, a melhor metodologia para previsão de séries temporais com processos geradores não lineares permanece um problema em aberto, o que justifica uma maior necessidade de pesquisa na área.

Utilizando uma perspectiva derivada da Teoria do Caos, Tong (TONG, 2009) argumenta que o benefício da utilização de modelos não lineares podem ser dependentes de um período restrito de tempo onde a previsão está sendo realizada, e que há a possibili-

dade que hajam "janelas de oportunidade"para redução de erro em situações específicas. Esse comportamento pode ocorrer em situações onde a não linearidade apenas torna-se relevante em cenários restritos, como por exemplo, quando a variável resposta atinge um determinado limiar (CLEMENTS; FRANSES; SWANSON, 2004).

Assim, muitos esforços vêm sendo tomados na direção de utilizar uma combinação de preditores de forma que obtêm-se vantagem de diferentes modelos em diferentes tipos de situação. Quando considera-se métricas de erro como o Erro Médio Quadrático (MSE), Zhang (ZHANG, 2003) observou que a combinação de preditores pode resultar em previsões melhores do que as realizadas por cada preditor separadamente. A combinação de modelos lineares com outros modelos não lineares foram avaliadas em trabalhos anteriores em problemas como a previsão da qualidade da água (FARUK, 2010), preço de ações (PAI; LIN, 2005) e outros (KHASHEI; BIJARI, 2011; DOMINGOS; OLIVEIRA; NETO, 2019; PANIGRAHI; BEHERA, 2017). Assim, há na literatura um conjunto de evidências significativas referente à melhoria das previsões de séries temporais ao combinar-se modelos preditivos com aplicações nas mais diversas áreas.

Portanto, uma alternativa consiste na utilização de sistemas híbridos, onde o método linear é utilizado em conjunto com um modelo de AM (não linear). Com essa metodologia, o modelo linear é responsável por modelar o componente linear da série, e os modelos de aprendizado de máquina são otimizados para prever os resíduos, a fim de prever componentes não lineares no processo gerador da série (ZHANG, 2003). Essa abordagem pode ser adequada em casos onde a introdução de não linearidade no sistema de previsão apenas melhora a performance em situações específicas, por exemplo. A previsão final da série temporal, portanto, pode ser decomposta de acordo com a Equação 3.55, onde L_t representa o componente linear da previsão e N_t representa o componente não linear.

$$Z_t = L_t + N_t \tag{3.55}$$

Como modelos de aprendizado de máquina como redes neurais foram demonstrados ter capacidade de representar qualquer função contínua (LESHNO et al., 1993), essa abordagem tem o poder de combinar as propriedades estatísticas dos modelos ARIMA com a flexibilidade dos modelos de aprendizado de máquina para aprender quando não há uma relação linear entre valores passados e futuros da série temporal (ZHANG, 2003).

No entanto, ao propor a combinação de modelos, muitas escolhas são introduzidas. Por exemplo, o desenvolvedor deve ser responsável por decidir qual a estratégia de combinação que deve ser utilizada para combinar as diferentes saídas dos modelos. Métodos diferentes foram propostos para combinação das saídas de cada modelo, porém não há uma metodologia consistente para determinação da combinação ótima em um determinado caso. Em geral, uma mistura linear das previsões individuais são utilizadas (CLEMENTS; FRANSES; SWANSON, 2004) devido à sua simplicidade de implementação, embora não tenha sido

claramente demonstrado que a combinação entre os fatores lineares e não lineares de um processo gerador possem qualquer tipo de propriedade aditiva.

Em trabalhos mais recentes, foram propostos métodos alternativos para combinação das previsões de cada modelo (KHASHEI; BIJARI, 2011; ZHU; WEI, 2013; DOMINGOS; OLIVEIRA; NETO, 2019), onde a previsão final pode ser definida como a combinação não linear das componentes individuais da previsão da série, o que pode ser representado de acordo com a Equação 3.56.

$$Z_t = f(L_t, N_t) \tag{3.56}$$

O ganho de complexidade dessa abordagem introduz a possibilidade de uma melhor hibridização de sistemas de previsão sem a premissa implícita que os componentes possuem propriedades aditivas. Por outro lado, tal abordagem também apresenta desvantagens significantes.

- Uma metodologia consistente que define a função de combinação adequada para a combinação não linear ainda não foi proposta. Isso impacta significantemente a possibilidade de uso em massa desse método, pois introduz um problema onde não há uma solução fechada.

- Como em todas as aplicações de aprendizado de máquina, não há um consenso referente à quais parâmetros são melhor adaptados para um determinado problema. Como resultado, a otimização de parâmetros é comumente feita por tentativa e erro ou por força bruta.

- Assumindo a não linearidade dos dados, a seleção de parâmetros não pode ser feita calculando correlação entre os *lags*. Portanto, deve-se decidir manualmente quais features são relevantes para que o modelo realize a previsão adequadamente.

A primeira desvantagem apontada pode ser endereçada ao utilizar-se uma rede neural como um aproximador de uma função não linear genérica, ao invés da definição analítica de uma expressão matemática. Em Kashei (KHASHEI; BIJARI, 2011), uma rede neural artificial é utilizada para aproximar a função de combinação, dado que é sabido que uma rede neural com apenas uma camada é capaz de aproximar qualquer função não linear contínua (LESHNO et al., 1993). No entanto, a habilidade de definir os parâmetros corretos e realizar a seleção de variáveis utilizadas para a previsão permanecem um problema em aberto.

Esse trabalho aborda esses problemas, ao utilizar uma combinação de um modelo ARIMA e outros modelos de aprendizado de máquina como Redes Neurais Artificiais (ANN) e Máquinas de Vetores de Suporte (SVM). Será utilizado um algoritmo genético para realizar a otimização dos parâmetros de cada modelo, assim como realizar a seleção de variáveis e a combinação dos resultados de forma simultânea. Sendo assim, o método

descrito nesse trabalho pode ser utilizado como uma abordagem geral para a construção de modelos híbridos de previsão de séries temporais.

3.4 ALGORITMOS GENÉTICOS

Algoritmos Genéticos são uma classe de algoritmos de busca heurísticos que possuem como inspiração a Teoria da Evolução Natural. Foram introduzidos por Holland em 1975 (HOLLAND et al., 1975), como uma possível abordagem para solucionar problemas de busca.

Um problema de busca pode ser caracterizado como uma otimização onde deseja-se encontrar a melhor solução possível dentro de um conjunto de possibilidades. Algoritmos tradicionais de busca se dividem em grandes grupos, sendo os mais comuns a busca aleatória e a busca exaustiva. No entanto, em problemas onde o espaço de busca se torna extenso, tais métodos se mostram impraticáveis (SIVANANDAM; DEEPA, 2008).

Por outro lado, algoritmos genéticos vêm mostrado um ganho considerável comparando-se com outros métodos de busca locais e aleatórios, devido ao fato de utilizarem informações sobre a performance de possíveis soluções para enviesar a busca para uma direção promissora (JONG, 1988).

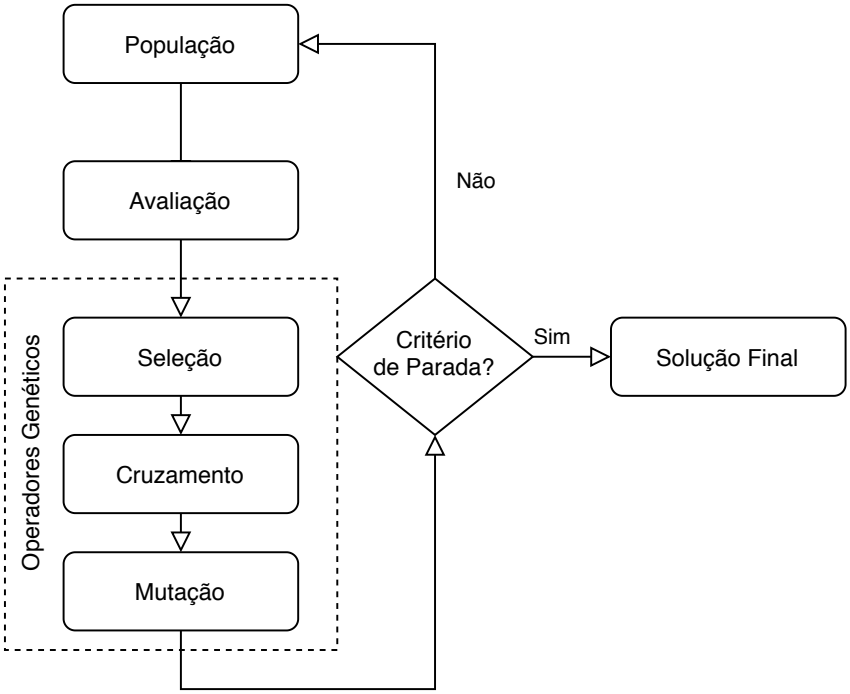
Nas últimas décadas, esse tipo de abordagem vem sido aplicada em diversos problemas do mundo real de alta complexidade (MCCALL, 2005), destacando-se especialmente em problemas onde o conhecimento *a priori* sobre o domínio do problema se mostra escasso.

De acordo com a teoria da evolução natural, os seres vivos atuais são resultados de adaptações que ocorreram durante milhões de anos em resposta às demandas do ambiente em que se encontram. Indivíduos que possuem maior aptidão a sobreviver e competir por recursos em um determinado ambiente são capazes de gerar um número maior de descendentes, e portanto se tornam mais numerosos. Por outro lado, indivíduos menos adaptados possuem menores chances de procriar e portanto tendem a diminuir sua população. Assim, todos os indivíduos que coexistem estão inseridos num processo competitivo onde sobrevivem os mais adaptados para aquele ambiente (SIVANANDAM; DEEPA, 2008).

Os algoritmos genéticos realizam a abstração desse raciocínio. Assim, essa classe de abordagens consiste em submeter uma população de indivíduos (que representam soluções individuais à um determinado problema) à uma seleção de acordo com uma determinada métrica. Assim, é possível produzir a próxima geração populacional, performando-se operações como mutações e combinações entre os cromossomos dos indivíduos. Esse procedimento é repetido até a convergência, onde em muitas ocasiões será num ótimo global (LEE; ANTONSSON, 2000).

A descrição de um algoritmo genético pode ser feito considerando três determinações: cromossomo, métrica de avaliação (fitness) e reprodução. Tais características serão abordada com mais detalhes nas próximas seções.

Figura 16 – Diagrama básico de um algoritmo genético.



Fonte: Elaborada pelo autor.

3.4.1 Cromossomo

O cromossomo representa as características de um indivíduo e determina uma possível solução. Dentro do cromossomo, haverá informações sobre os as características daquele indivíduo. Tais características são codificadas em **genes**, cuja representação pode ser feita de diversas formas (ROTHLAUF, 2006).

A representação binária representa a característica de forma tudo ou nada. Ou seja, tal representação pode ser utilizada para indicar a ausência ou a presença de uma determinada característica no indivíduo.

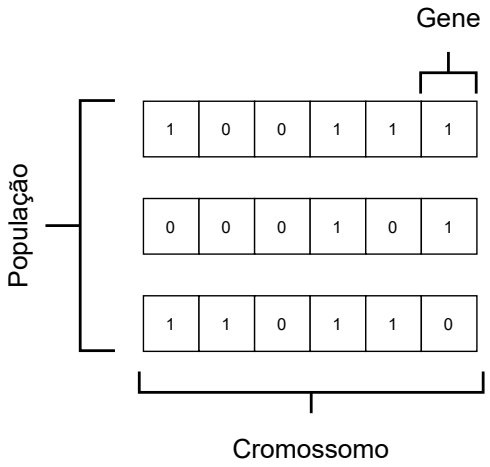
Uma representação textual pode representar uma característica utilizando códigos alfanuméricos. Nesse caso, a característica apresentada num indivíduo pode ser uma dentro de uma lista de possibilidades finitas. Cada possibilidade é representada como uma categoria.

Uma representação visual de um cromossomo pode ser visto na Figura 17. Nesse exemplo, três indivíduos são representados por três cromossomos, onde cada característica é representado por um valor binário, denominado de gene.

3.4.2 Métrica de Avaliação (fitness)

Durante o processo evolutivo, torna-se necessário realizar a avaliação de uma determinada solução, buscando-se obter uma análise da performance de cada indivíduo. A

Figura 17 – Representação de um cromossomo em uma população de três indivíduos.



Fonte: Elaborada pelo autor.

métrica de avaliação consiste na função utilizada para avaliar cada indivíduo dentro de uma iteração. Em geral, são utilizadas funções matemáticas relacionados ao objetivo da otimização.

Portanto, a função *fitness* é responsável por avaliar a aptidão do indivíduo em realizar a tarefa que deseja-se otimizar, e portanto se mostra dependente da tarefa. Caso a otimização esteja relacionado com a minimização de um erro de previsão de uma série, a métrica de avaliação pode estar relacionado com o erro médio de cada possível solução.

A função de avaliação deve obedecer diversas restrições

- A métrica de avaliação deve ser facilmente definida e calculada para cada indivíduo.
- A métrica de avaliação deve diferenciar os melhores e os piores indivíduos dentro de cada geração.
- O valor da métrica deve aumentar a medida que a solução se aproxima da solução ideal.

3.4.3 Reprodução

A reprodução representa a etapa onde indivíduos são selecionados e geram descendentes para a próxima iteração. As características da reprodução possuem alto impacto sobre a convergência final do algoritmo (KATOCH; CHAUHAN; KUMAR, 2020).

3.4.3.1 Seleção

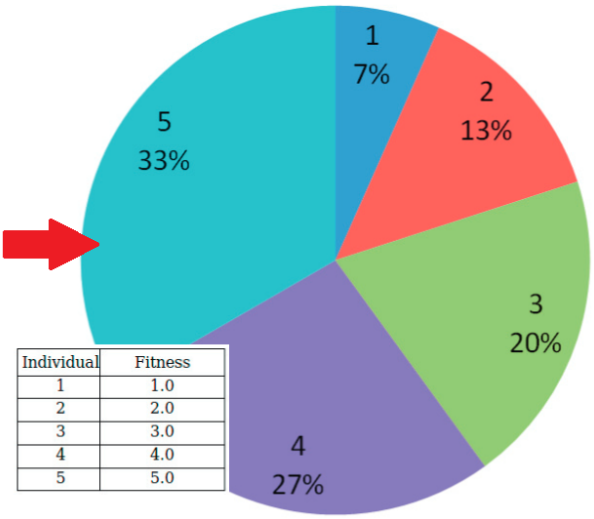
A etapa de seleção consiste na escolha de indivíduos de interesse para a próxima iteração e para cruzamentos. Em geral, uma das estratégias mais comuns de seleção consiste

na roleta. Matematicamente, considerando-se N indivíduos que possuem *fitness* maiores que zero, onde o *fitness* do indivíduo i é dado por w_i, a probabilidade de seleção de um determinado indivíduo é dado por

$$p_i = \frac{w_i}{\sum_{i=1}^{N} w_i} \tag{3.57}$$

Essa abordagem é equivalente à uma roleta fictícia girada, onde a porção da roleta que representa cada indivíduo é proporcional à sua aptidão, como pode-se visualizar na Figura 18. Sendo assim, indivíduos que possuem maior aptidão tem maior chance de serem selecionados.

Figura 18 – Representação visual do método de seleção por roleta.



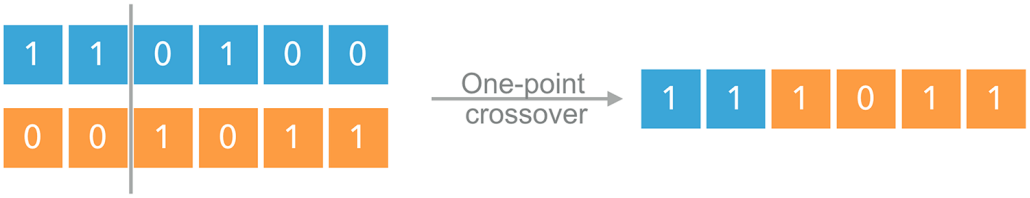
Fonte: Adaptada de: (XAVIER et al., 2013)

3.4.3.2 *Cruzamento*

O cruzamento representa a mistura do material genético entre mais de um indivíduo. Dado que indivíduos foram selecionados utilizando um algoritmo de seleção, a etapa de cruzamento é responsável pela geração de indivíduos filhos a partir do material genético dos pais.

Um método simples de realizar a mistura do material genético consiste na utilização do método de *crossover* de único ponto. Nessa abordagem, temos que uma determinada posição do cromossomo é escolhido aleatoriamente como ponto de corte (KLUG; PARK; KRUG, 2019). A partir desse ponto, os cromossomos dos indivíduos pais são seccionados e recombinados em um novo indivíduo filho. Sendo assim, o indivíduo resultante possui o cromossomo do primeiro pai antes do ponto de corte, e o cromossomo do segundo pai após o ponto de corte. Pode-se observar o processo de cruzamento na Figura 19.

Figura 19 – Representação de um *crossover* de ponto único.



Fonte: Adaptada de: (KLUG; PARK; KRUG, 2019)

3.4.3.3 Mutação

A etapa de mutação consiste em modificações no cromossomo de forma estocástica, imitando o comportamento das mutações naturais que ocorrem durante o processo natural de cópia das instruções genéticas. Sendo assim, a probabilidade de mutação ou o número de mutações em uma determinada iteração é definido como parâmetro do algoritmo genético (KATOCH; CHAUHAN; KUMAR, 2020).

O processo de mutação garante diversidade à nova população e permite a exploração de regiões do espaço de busca novos, evitando assim a convergência prematura em mínimos locais (COLEY, 1999).

O processo de mutação pode ser definido de várias formas. Para valores binários, pode-se definir uma mutação como uma operação de inversão no valor *booleano*. Para valores textuais, a mutação pode ser efetuada simplesmente trocando o valor atual do gene por outro dentro da lista finita de possibilidades.

3.4.4 Discussão

A combinação entre algoritmos genéticos e modelos de aprendizado de máquina se mostram na literatura nas mais diversas formas, dentre elas: otimização de hiper-parâmetros (YOUNG et al., 2015), seleção de *features* (VAFAIE; JONG, 1992) e busca de arquitetura em redes neurais profundas (ELSKEN; METZEN; HUTTER, 2018).

Em geral, problemas de otimização de parâmetros em modelos de aprendizado de máquina possuem pouca fundamentação teórica e espaço de busca extenso, o que torna métodos tradicionais de busca como a busca extensiva impraticáveis. Sendo assim, algoritmos genéticos são comumente propostos como uma solução alternativa adequada à este grupo de problemas.

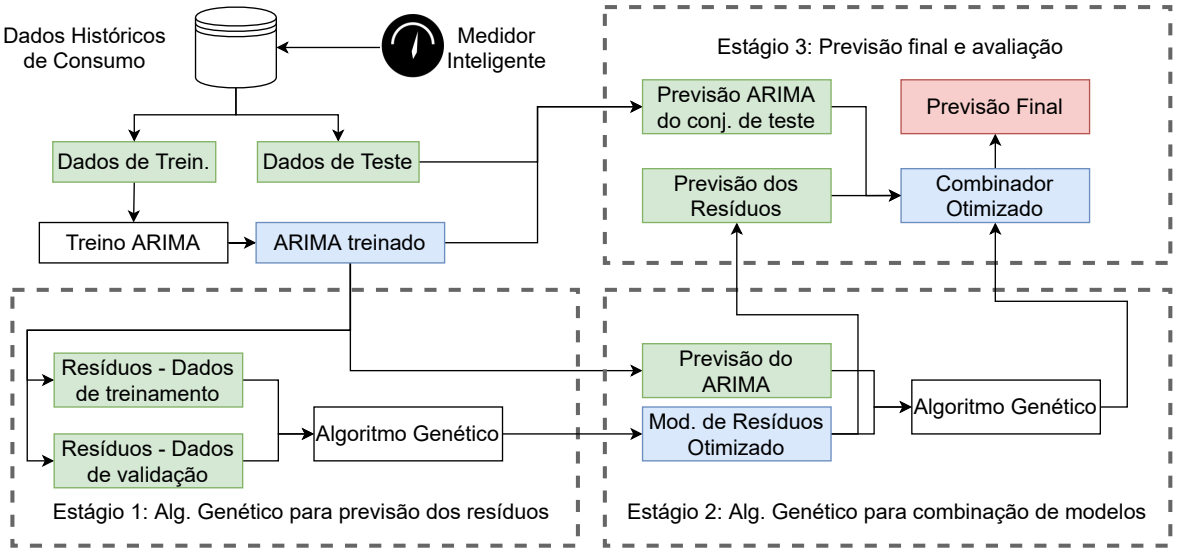
4 MÉTODO PROPOSTO

O método proposto nesse trabalho consiste numa abordagem evolucionária para a combinação de modelos de aprendizado de máquina e modelos do tipo SARIMA para uma previsão conjunta. Nossa proposta pode ser vista como um método de busca e otimização heurística que procura encarar os principais desafios encontrados ao realizar previsões de séries temporais através de modelos híbridos: seleção de *lags*, parametrização do modelos e otimização da combinação da previsão entre os modelos.

O método proposto, diferente de outros métodos da literatura, realiza a otimização de todas as etapas da modelagem de forma conjunta. Ou seja, a seleção de parâmetros dos modelos utilizados é feita em conjunto com a seleção dos *lags* de entrada, resolvendo assim dois grandes desafios ao aplicar modelos de ML. Além disso, o algoritmo genético apresentado como parte do método proposto é flexível, e portanto é capaz de ser utilizado para otimização em todas as etapas, permitindo assim a busca de soluções de forma automática e fluida.

O processo completo do método proposto pode ser visualizado na Figura 20. Para fins didáticos, pode-se dividir a metodologia proposta em duas etapas: treinamento e previsão. A etapa de treinamento será responsável pelo treinamento e otimização dos modelos lineares e de AM para a previsão da série. Em seguida, a etapa de previsão utilizará os modelos encontrados na etapa anterior para realizar a previsão do valor final da série temporal.

Figura 20 – Diagrama do treinamento método proposto para previsão de consumo de energia.

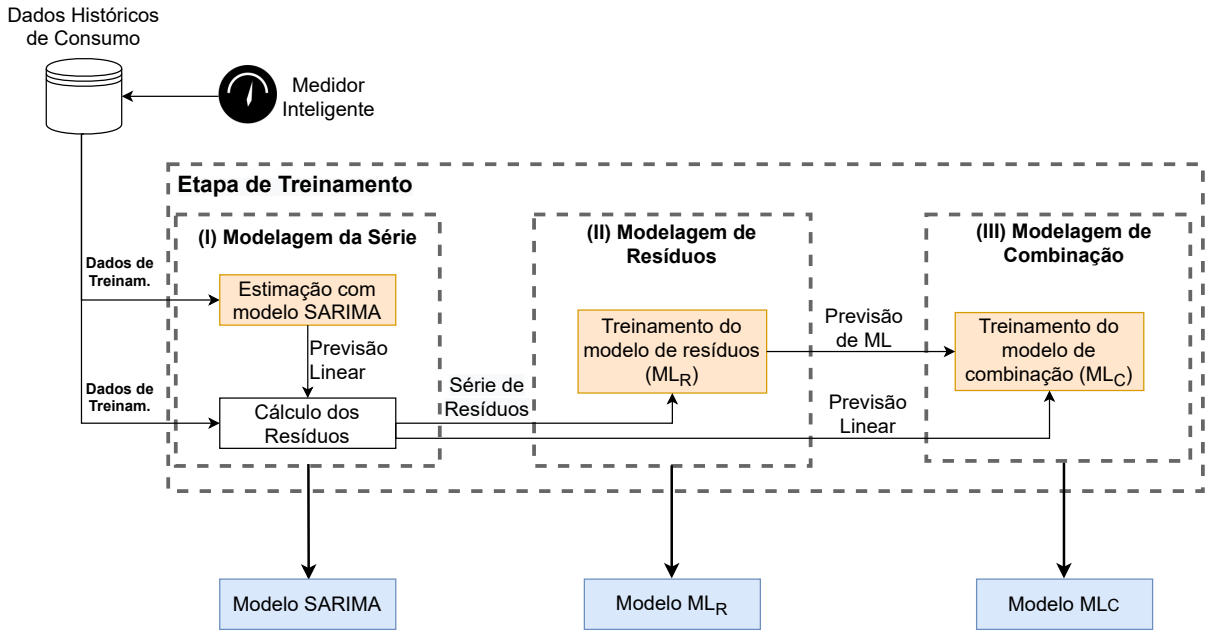


Fonte: Elaborada pelo autor.

4.1 TREINAMENTO

A etapa de treinamento pode ser visualizada com mais detalhes na Figura 21. Considerando N pontos dos dados históricos de consumo de energia obtidos pelo *smart meter*, temos que tal conjunto é dividido em treino, validação e teste. Após a divisão, o conjunto composto pelos dados de treinamento e validação é utilizado para ajustar um modelo SARIMA utilizando a metodologia Box & Jenkins (BOX et al., 2015).

Figura 21 – Diagrama da etapa de treinamento do proposto para previsão de consumo de energia.



Fonte: Elaborada pelo autor.

Em seguida, é introduzido um modelo de AM responsável pela previsão dos resíduos do SARIMA (denominado **Modelo de Resíduos**). Os resíduos podem ser definidos como a diferença entre o valor real da série temporal e o valor previsto pelo SARIMA. Tal relação é dada pela Equação 4.1, onde onde L_t representa o componente linear da previsão e Z_t representa o valor real da série.

$$N_t = Z_t - L_t \tag{4.1}$$

O intuito desse modelo consiste na utilização de modelos de aprendizado de máquina para detecção de padrões não lineares na série temporal de consumo de energia, dado que esses padrões não são capturados através de modelos lineares.

No entanto, a utilização de modelos de AM enfrentam diversos problemas que foram citados no Capítulo 3. Portanto, o Algoritmo Genético apresentado na Seção 4.3 é utilizado para realizar a busca dos parâmetros e seleção dos *lags* utilizados pelo modelo de AM. Além disso, no caso da rede neural, também é encontrado o melhor otimizador (de treinamento) para uma determinada série.

Sendo assim, a segunda etapa consiste na busca de um modelo de aprendizado de máquina otimizado para prever o componente não linear (resíduos) da série temporal. Para esse processo, o conjunto de treinamento é utilizado para ajuste dos modelos de AM e o conjunto de validação é utilizado para avaliar os indivíduos. O algoritmo genético é responsável por treinar, portanto, diversos indivíduos (que representam possíveis modelos de aprendizado de máquina com parâmetros e entradas diferentes) utilizando os dados de treinamento. Para cada indivíduo, a avaliação para seleção é feita através do Erro Médio Quadrático (MSE) para o conjunto de validação.

Portanto, a saída da segunda etapa do método proposto consiste em um modelo otimizado para a previsão do componente não linear da série temporal, que pode ser tanto um modelo do tipo MLP como um modelo do tipo SVR.

Na terceira etapa, o modelo otimizado na etapa anterior é combinado com as previsões do SARIMA para a previsão final da série. Isso é feito através de um segundo modelo de AM (denominado **Modelo de Combinação**).

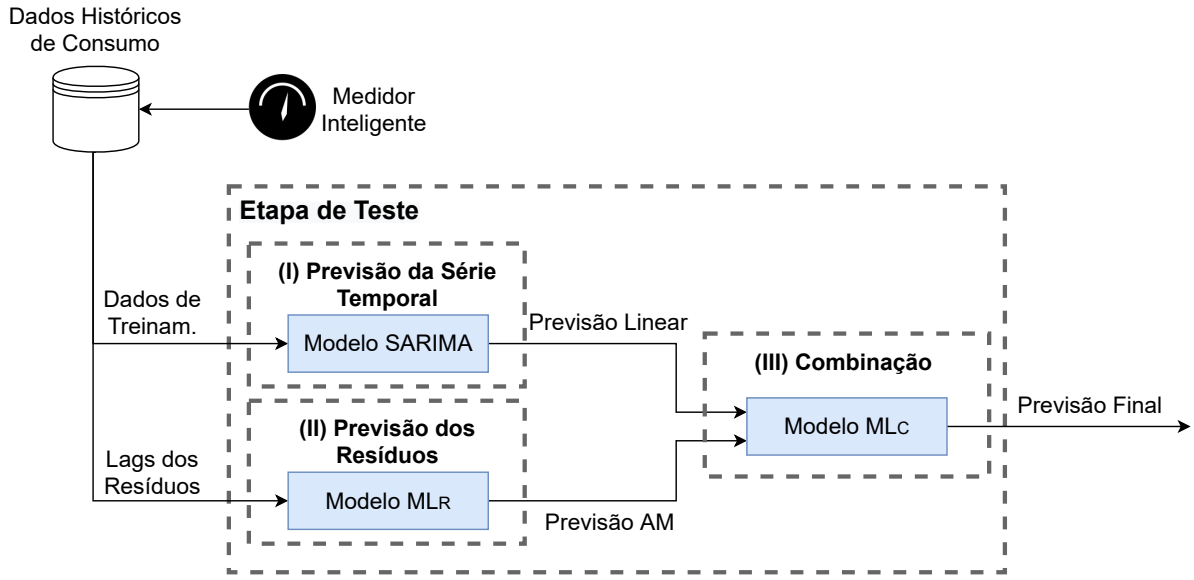
O modelo de combinação possui como entrada os valores previstos pelo SARIMA e os valores previstos como resíduo pelo Modelo de Resíduo, a sua variável resposta consiste no consumo real aferido pelo *smart meter*. Sua otimização é realizada através do mesmo Algoritmo Genético da etapa anterior (Seção 4.3), que será responsável por encontrar o melhor modelo de aprendizado de máquina capaz de aproximar a função de combinação da forma $Z_t = f(L_t, N_t)$, utilizando os mesmos conjuntos de treinamento e teste. Portanto, a aproximação da função Z_t é dada pela saída do modelo de combinação (ML_C), que recebe as previsões lineares e não-lineares como entrada. Assim, as melhores soluções são avaliadas de acordo com o Erro Médio Quadrático (MSE) para o conjunto de validação.

4.2 PREVISÃO

O processo de previsão final da série pode ser visualizado de forma detalhada na Figura 22. Na etapa de previsão, temos que os *lags* da série temporal são dados como entrada para o modelo SARIMA e os resíduos passados do modelo SARIMA são dados como entrada para o Modelo de Resíduos (ML_R).

Assim, o modelo SARIMA realizará uma previsão linear do valor da série, e o Modelo de Resíduos (ML_R) realizará a previsão dos resíduos de forma não linear. Por fim, o Modelo de Combinação (ML_C) é responsável por combinar a saída dos dois modelos anteriores e realizar a previsão final do valor da série, sendo a previsão do método proposto a saída do Modelo de Combinação (ML_C). Sendo assim, a previsão final é composta pela combinação da previsão dos componentes lineares e não lineares.

Figura 22 – Diagrama da etapa de teste do método proposto para previsão de consumo de energia.



Fonte: Elaborada pelo autor.

4.3 ALGORITMO GENÉTICO

Conforme descrito no Capítulo 3, um algoritmo genético consiste na evolução de um grupo de soluções candidatas, denominada população, que passa por um processo de seleção e mutação similar ao ocorrido na seleção natural. Essa abordagem pode ser brevemente definida descrevendo três prontos principais: o cromossomo, a métrica de avaliação (fitness) e o mecanismo de reprodução, que serão abordados nas seções a seguir.

4.3.1 Cromossomo

O cromossomo trata-se de uma representação de uma possível solução ao problema que deseja ser otimizado. Para o nosso método proposto, a representação de cada indivíduo é feita através de um vetor de tamanho fixo que pode ser divido em duas partes distintas.

No primeiro bloco, temos informações sobre os *lags* a serem utilizados pelo modelo preditivo. Sendo assim, a primeira parte do genoma possui um tamanho correspondente ao número máximo de possíveis entradas, onde cada posição no vetor contém um valor binário indicando se uma determinada entrada vai ser utilizada ou não para aquele indivíduo.

A variável l_{max} determina o último *lag* que pode ser utilizados para cada série utilizada como entrada. Sendo assim, considerando o exemplo $l_{max} = 10$, temos que o modelo a ser otimizado pode selecionar até o décimo *lag* de cada série utilizada. No caso do Modelo de Resíduos, a única série presente é o valor passado dos resíduos. Para o Modelo de combinação, temos duas séries possíveis: a previsão do SARIMA e os valores previstos pelo Modelo de Resíduos, acrescidas do valor atual de cada série (sem atraso). Sendo

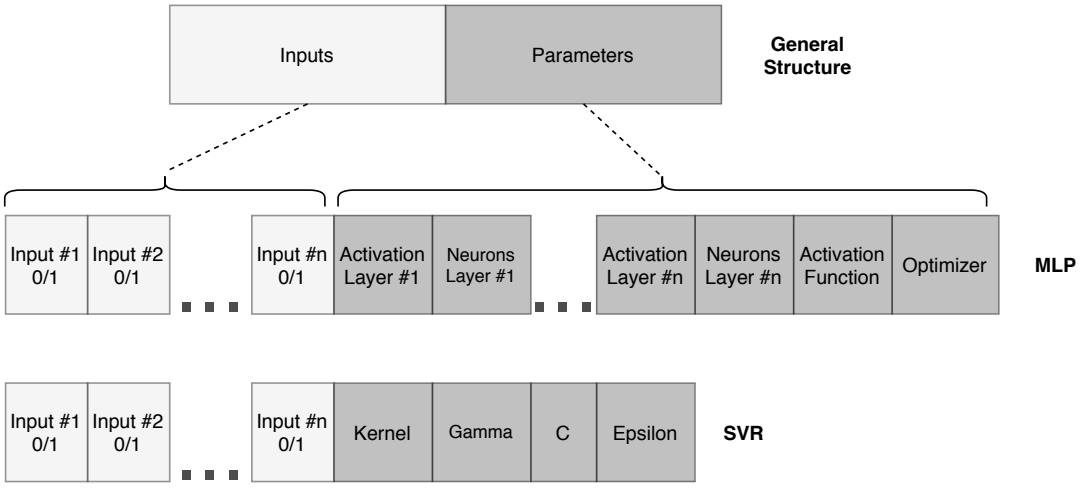
assim, no Modelo de Combinação, o tamanho do primeiro bloco de cromossomo é maior que o Modelo de Resíduos, já que no primeiro caso, por possuir duas séries de entrada e a inclusão dos valores sem *lag*, o número máximo de entradas utilizada é dado por $2 * (l_{max} + 1)$.

No segundo bloco, temos informações sobre os parâmetros de cada modelo. Para uma MLP, além dos parâmetros, um conjunto de genes é responsável por determinar a ativação das camadas e a quantidade de neurônios de cada respectiva camada, o que são determinados a partir de uma lista fixa contendo todas as possibilidades. Assim, temos que caso uma determinada camada seja desativada, a camada anterior terá uma conexão direta com a próxima camada, sendo esta camada efetivamente ignorada.

Uma restrição importante utilizada para esse cromossomo é que a primeira camada da rede neural deve estar sempre ativada, que representa a camada de entrada da MLP, isso evita a possibilidade de um cromossomo com todas as camadas desativadas, resultando em uma rede neural que não possui nenhuma camada.

Nota-se que, utilizando essa formulação, o número máximo de camadas de uma rede neural (c_{max}) deve ser limitado previamente, definindo assim o tamanho final do cromossomo. Após a caracterização das camadas, a parte final do segundo bloco do cromossomo para uma rede neural é composta por informações sobre a função de ativação utilizada em cada camada e o otimizador a ser utilizado para a definição dos pesos da rede. A estrutura final do cromossomo para uma rede neural pode ser visualizado na Figura 23.

Figura 23 – Cromossomo básico de um indivíduo.



Fonte: Elaborada pelo autor.

No caso de um modelo do tipo SVR, a segunda parte possui informação referente à configuração dos parâmetros de um modelo SVR como o *gamma* da função Kernel e a constante C, que representa o impacto de cada ponto individual e o *trade-off* entre a classificação correta de todos os pontos e a maximização da margem.

Como resultado, o cromossomo final contem informação referentes aos *lags* utilizados

da série temporal para realizar a previsão final, assim como os parâmetros e a arquitetura de cada modelo. Os estados possíveis do segundo bloco são representados por valores textuais que representam uma possibilidade dentro de uma lista pré-definida.

4.3.2 Métrica de Avaliação (fitness)

A função custo utilizada para avaliar cada indivíduo é o inverso do Erro Médio Quadrático (MSE) para um conjunto de dados de validação. Tal função calculará para cada ponto a diferença entre o valor previsto $\hat{y}$ e o valor real y, e retornará o inverso da média do erro quadrático para todos os pontos do conjunto. Sendo assim, quanto menor o erro médio obtido por um determinado modelo, maior será sua aptidão. Assim, temos

$$fitness = \frac{1}{1 + MSE}, \tag{4.2}$$

onde MSE representa o Erro Quadrático Médio para a saída desejada do modelo sendo otimizado. Assim, temos

$$MSE = \frac{1}{n}\sum(y_t - \hat{y}_t)^2. \tag{4.3}$$

O MSE, devido ao termo quadrático, possui uma alta penalização para erros altos conforme descrito no Capítulo 3, e portanto, mostra-se adequado ao caso onde deseja-se minimizar erros grosseiros na previsão da série temporal.

Como o conjunto de validação não possui intersecção com o conjunto de treinamento, temos que a métrica é responsável por avaliar a capacidade de generalização do modelo, diminuindo assim a possibilidade de *overfitting* no modelo final.

Para redes neurais, a inicialização dos pesos pode afetar o resultado final. Portanto, para garantir a avaliação do modelo em diferentes estados iniciais, o modelo é treinado separadamente n_{tries} vezes, onde n_{tries} representa um parâmetro do algoritmo. Assim, garante-se uma melhor avaliação do modelo, caracterizado como estocástico. A partir dos n_{tries} treinamentos, o modelo que obteve o melhor *fitness* é utilizado como o representante final do indivíduo.

Sendo assim, o objetivo final do algoritmo genético trata-se de otimizar a previsão de um determinado valor. O treinamento é feito utilizando o conjunto de treinamento do algoritmo genético, e a validação é realizada com o conjunto de validação do algoritmo genético, garantindo a capacidade de generalização dos modelos fora do treinamento. Sendo assim, a função objetivo consiste na minimização do erro médio quadrático global.

Claramente, o valor tido como saída desejada ao calcular o MSE dependerá da série onde o algoritmo genético está sendo executada. Para o Modelo de Resíduos, o MSE será

calculado considerando a previsão dos resíduos e o resíduo verdadeiro. Sendo assim, a métrica utilizada pode ser representada pela Equação 4.4.

$$MSE = \frac{1}{n}\sum(N_t - \hat{N}_t)^2. \tag{4.4}$$

Para o Modelo de Combinação, temos que o MSE será calculado utilizando os valores finais previstos e o valor real da série de consumo, sendo portanto dado pela Equação 4.5.

$$MSE = \frac{1}{n}\sum(Z_t - \hat{Z}_t)^2. \tag{4.5}$$

4.3.3 Reprodução

Após a avaliação da iteração atual, a criação da próxima iteração é realizada em duas etapas, que representam dois grupos distintos, cada um representando metade da nova população.

No primeiro grupo, os indivíduos são formados por cópias de indivíduos da iteração anterior (embora possam sofrer mutações posteriormente). Os indivíduos que vão ser selecionados para a próxima iteração são selecionados através do método da roleta viciada (BLICKLE; THIELE, 1996), onde indivíduos com melhor *fitness* possuem uma maior chance de serem selecionados. A probabilidade de um indivíduo ser selecionado é proporcional à sua performance na previsão da série, medido pelo MSE.

Sendo assim, indivíduos que representam uma melhor solução para o problema e obtiveram melhor capacidade de generalização no conjunto de validação, possuem uma maior probabilidade de serem selecionados e passar seus genes para a iteração seguinte.

O segundo grupo é formado por cruzamentos entre indivíduos da população anterior. Novamente, os indivíduos são escolhidos através de uma roleta viciada, onde a chance de um indivíduo ser escolhido para um cruzamento é proporcional à sua performance. Sendo os cromossomos pai/mãe escolhidos, o cruzamento é realizado realizando um *crossover* de ponto único (*single point crossover*).

A proporção de indivíduos do primeiro grupo e do segundo grupo na população final é determinado por um parâmetro c_r (ou *crossover ratio*, que representa o percentual de indivíduos na nova população que serão formados por *crossover* entre indivíduos da iteração passada.

Logo, a fração $(1-c_r)$ da população final é composta por indivíduos da última iteração, e a outra fração c_r são indivíduos resultantes de um cruzamento entre indivíduos da população anterior. Em ambos os casos, indivíduos com boa performance possuem maior chance de serem escolhidos. O processo de reprodução utilizado, portanto, garante que indivíduos com bons resultados possuem uma alta probabilidade de serem escolhidos para

a próxima iteração. Sendo assim, o algoritmo genético garante uma convergência rápida ao realizar uma busca direcionada no espaço de busca.

4.3.4 Mutações

Após o processo de reprodução, todos os indivíduos possuem chances de mutação em seus genes para garantir variabilidade na população, e portanto, expandir o espaço de busca para evitar uma convergência prematura.

Em cada iteração, o números de mutações para um determinado indivíduo é escolhido aleatoriamente utilizando uma variável aleatória que segue uma distribuição de probabilidade uniforme. A distribuição uniforme atua em um range específico, definindo como $[0, n_{max}]$. Sendo assim, para cada indivíduo é atribuído um número de mutações n_m, onde todos os indivíduos possuem probabilidade de igual de possuírem um número de mutações entre 0 e n_{max}.

Dado que uma mutação irá ocorrer para um determinado indivíduo, é realizada a seleção do gene a ser modificado. Nesse caso, todos os genes possuem a mesma chance de serem escolhidos. Para os genes binários, a mutação resulta na inversão do estado original (de 0 para 1 ou vice-versa), a única exceção trata-se da ativação da primeira camada da MLP, já que esta não deve ser desativada nunca. Caso esse gene seja escolhido, o valor não sofre mudança e a ativação é mantida como um (1). Nos outros genes, a mutação irá substituir o estado atual com uma nova opção escolhida aleatoriamente da lista finita de possíveis estados. Sendo assim, caso o número de mutações do indivíduo seja três ($n_m = 3$), temos que três genes serão escolhidos para serem modificados, podendo o mesmo gene ser escolhido mais de uma vez.

Nas primeiras iterações, o número de mutações máximo (n_{max}) é fixo, determinado por um parâmetro n_0, o que é mantido até a iteração g_n. Após a iteração n, o número máximo de mutações irá ser acrescido de uma unidade a cada n_{step} iterações.

Sendo assim, à medida que a busca do algoritmo genético progride e leva à convergência, o método proposto aumenta o número máximo de mutações para garantir a adição de variabilidade nas soluções, evitando assim que a busca realize a convergência prematura em um mínimo local.

5 EXPERIMENTOS

5.1 DADOS

A série temporal utilizada para avaliar o método proposto foi originada de uma medição do consumo de energia apresentada em (CHOU; NGO, 2016b) do período entre 22 de Junho de 2015 e 26 de Julho de 2015. Uma rede de *smart meters* foi responsável por realizar a medição em tempo real dos padrões de consumo de energia de um edifício. O edifício utilizado nos nossos experimentos possui três andares e está localizado em Taiwan. Sua ocupação, durante as medições, era composta por três crianças e seus pais, com uma área total de $350m^2$.

O primeiro andar é composto por uma área de escritório, onde funciona um negócio que é administrado pelo morador. No segundo andar, a sala de estar se apresenta em conjunto com áreas de estudo, cozinha e sala de jantar. O terceiro andar é composto pelos quartos dos moradores, assim como uma sala de hóspedes e de estudo. A Figura 24 apresenta a planta do edifício onde foram instalados medidores inteligentes responsáveis por registrar o consumo de energia em cada cômodo.

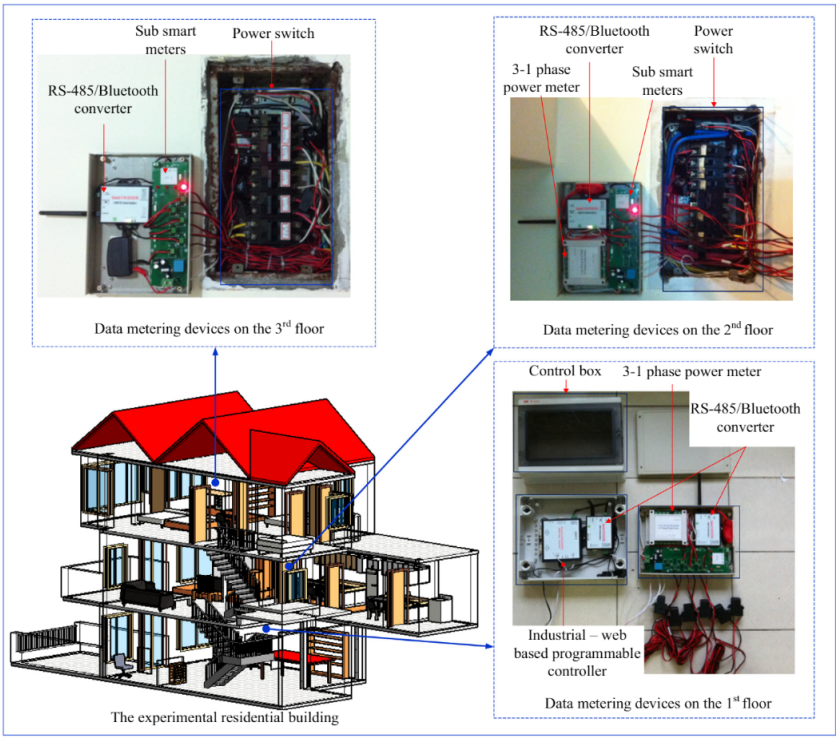
Figura 24 – Planta do edifício onde foram realizadas as medições.

Fonte: (CHOU; NGO, 2016b)

Os dados de consumo de energia foram registrados pelos *smart meters* e transferidos para um servidor dedicado. A visualização dos medidores inteligentes utilizados para esse experimento podem ser visualizados na Figura 25. Um fluxo de dados é enviado a cada 1 minuto, o que resulta num conjunto de 1440 pontos para cada dia. Em seguida, os minutos foram convertidos em intervalos de 15 minutos, resultando portanto em 96 pontos diari-

amente. Além disso, outros fatores externos como dia da semana e temperatura externa foram registrados em conjunto com o consumo de energia.

Figura 25 – Medidores inteligentes instalados no edifício.



Fonte: (CHOU; NGO, 2016b)

Tal série foi utilizada para avaliação do modelo de forma isolada devido à dificuldade de encontrar outros conjuntos de dados públicos referentes à medições de medidores inteligentes com o mesmo rigor experimental e que possuam *baselines* de comparação na literatura.

5.2 CONFIGURAÇÃO EXPERIMENTAL

Neste trabalho, foi considerado apenas o consumo total do edifício, não sendo realizados experimentos considerando o consumo de cada cômodo individualmente. Além disso, não foram utilizadas informações externas como temperatura como entrada dos modelos preditivos. Sendo assim, o objetivo do nosso trabalho consiste em otimizar a capacidade de gerar previsões do consumo de energia de um edifício a partir da série temporal do consumo.

Para isto, o método apresentado no Capítulo 4 foi executado de acordo com sua descrição. O conjunto de dados total possui 2879 pontos, onde divisão entre treino, validação e teste utilizada foi realizada utilizando as proporções $61,3\% - 15,3\% - 23,3\%$, respectivamente, respeitando a ordem temporal. Seguindo o método descrito no Capítulo 4, foi

treinado um modelo do tipo SARIMA utilizando os dados de treinamento e validação. Tal modelo é responsável por realizar previsões do componente linear da série temporal.

5.2.1 Algoritmo Genético

O segundo e o terceiro passo consistiram na otimização do Modelo de Resíduos e do Modelo de Combinação.

O Modelo de Resíduos é responsável por prever o resíduo resultante do modelo linear. Neste caso, a entrada foi constituída pelos resíduos passados calculados com a previsão do SARIMA. Para o Modelo de Combinação, as entradas foram compostas pelas previsões dos resíduos, o valor previso pelo SARIMA e seus valores passados.

Assim, o algoritmo genético proposto foi executado duas vezes, para cada modelo, utilizando suas respectivas entradas e saídas. No entanto, os parâmetros utilizados para os dois casos permaneceram constantes:

- O número da geração onde o número de mutações começa a incrementar uma unidade é doze ($g_n = 12$)
- O número inicial de mutações em cada geração é três ($n_0 = 3$)
- O número máximo de camadas de uma rede neural é quatro ($c_{max} = 4$)
- A proporção da população formada por crossovers de indivíduos da população anterior é 0.5 ($c_r = 0.5$).
- O número máximo de lags utilizado para cada série de entrada é dez ($l_{max} = 10$).
- O número de vezes que uma rede neural é treinada para avaliar esse tipo de modelo é três ($n_{tries} = 3$)

Os valores possíveis para modelos do tipo SVR são descritos abaixo.

Support Vector Regression (SVR)

- input_active = [0, 1]
- kernel = ['linear', 'poly2', 'poly3', 'poly4', 'poly5', 'poly6', 'rbf', 'sigmoid']
- gamma = ['scale','auto']
- C = [1e-5, 1e-4, 1e-3, 1e-2, 1e-1, 1, 10, 100, 1000, 1e4]
- epsilon = [0.0, 1e-5, 1e-4, 1e-3, 1e-2, 1e-1, 1, 10]

Nesse caso, o kernel *linear* consiste na implementação padrão do SVR, o kernel *sigmoid* consiste na implementação da função sigmóide e o kernel *rbf* consiste num kernel baseado em Radial Basis Function (RBF). Por fim, os kernels do tipo *poly2,poly3,poly4* representam kernels com polinômios de grau 2,3 e 4, respectivamente.

O valor de *gamma* é calculado de acordo com a implementação oficial da biblioteca *scikit-learn*, do Python, que permite duas possibilidades:

- se $gamma = auto$, então o valor de gamma adotado é $1/n_{features}$, onde $n_{features}$ representa a quantidade de variáveis de entrada do modelo.
- se $gamma = scale$, então o valor de gamma adotado é $1/(n_{features} \cdot \sigma^2)$ onde σ representa o desvio padrão da matriz de entrada.

Para uma rede neural, os valores possíveis dentro do espaço de busca podem ser visualizados na lista abaixo.

Multilayer Perceptron (MLP)

- input_active = [0, 1]
- activation = ['identity', 'logistic', 'tanh', 'relu']
- solver = ['sgd', 'adam', 'lbfgs']
- layer_active = [0, 1]
- layer_nodes = [1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048]

Onde a variável *activation* representa a função de ativação de cada neurônio, a variável *solver* representa o algoritmo de otimização utilizado para o ajuste dos pesos. As camadas da rede neural são ativadas ou desativadas pelo parâmetro *layer_active*, e possuem o número de neurônios dentro das possibilidades do parâmetro *layer_nodes*. Sendo assim, o número máximo de neurônios de uma camada individual é 2048.

Assim, ao final do algoritmo genético, obtivemos um modelo otimizado para a previsão dos resíduos provenientes do modelo SARIMA. Ao final da execução, os modelos obtidos podem ser visualizados na Tabela 1.

Tabela 1 – Modelos finais obtidos pelo algoritmo genético.

Modelo	Tipo	Parâmetros	Lags
Resíd.	SVR	C=0.01, epsilon=1e-05, gamma='scale', kernel='rbf'	Resíd.: -1,-5,-8,-9
Comb.	MLP	neurons= (32,), activ.='relu',optim.='adam'	SARIMA: 0,-2,-5,-6,-7;
			Resíd.: -3,-5,-7,-8

Sendo assim, pode-se observar que o Modelo de Resíduos otimizado consiste num SVR com Kernel do tipo RBF. Além disso, o Modelo de Combinação final possui apenas uma camada com 32 neurônios, sendo esta arquitetura representada pela tupla $(32,)$. Os números negativos indicados na coluna de *lags* representam a posição dos valores utilizados em relação ao tempo atual. Assim, o valor -1 representa que o último valor de resíduo é utilizado para a previsão do próximo resíduo, e assim por diante. Como o Modelo de Combinação também possui como entrada o valor atual do SARIMA e do resíduo, temos que é possível utilizar o índice zero (0), que representa o valor sem *lag*.

5.3 MÉTRICAS

Após a execução do método proposto, as métricas referentes aos erros de previsão do método final foram calculados, as métricas utilizadas são listadas abaixo.

- Raiz do Erro Médio Quadrático (RMSE)

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}}$$

- Erro Médio Absoluto (MAE)

$$MAE = \frac{1}{n} \sum_{j=1}^{n} |y_i - \hat{y}_i|$$

- Erro Médio Absoluto Percentual (MAPE)

$$MAPE = \frac{1}{n} \sum_{j=1}^{n} \left| \frac{y_i - \hat{y}_i}{y_i} \right|$$

- Correlação de Pearson (R)

$$r = \frac{\sum_{j=1}^{n} (y_i - \bar{y})(\hat{y}_i - \bar{\hat{y}})}{\sqrt{\sum_{j=1}^{n} (y_i - \bar{y})^2 \sum_{j=1}^{n} (\hat{y}_i - \bar{\hat{y}})}}$$

- Erro Máximo Absoluto (MaxAE)

$$MaxAE = max\ |y_i - \hat{y}_i|$$

A RMSE, calculada como a raiz quadrática da MSE, possui uma alta penalização para diferenças altas nos erros de previsão devido ao seu termo quadrático e portanto trata-se adequando quando deseja-se avaliar erros grosseiros nos modelos propostos. Tal efeito é minimizado quando utilizamos o MAE, que utiliza apenas os valores absolutos das diferenças.

No entanto, as duas métricas anteriores possuem sensibilidade à escala dos dados. O que não acontece ao utilizar métricas como MAPE e R. Por fim, o erro máximo absoluto consiste na verificação do maior erro atingido pelo modelo no conjunto de teste.

Tais métricas foram escolhidas devido à sua utilização recorrente na literatura, com o intuito de comparar nosso modelo com outros modelos propostos utilizando as métricas reportadas pelos seus respectivos trabalhos.

Para efeito de comparação com o método proposto, também foi executado o treinamento de um modelo de aprendizado de máquina LSTM (Long Short Term Memory) para a previsão de série completa de consumo de energia. Tal modelo representa um modelo mais robusto e recente de aprendizado de máquina, e portanto representa uma ótima comparação com a metodologia proposta.

No entanto, devido à sua alta complexidade e alto custo computacional, torna-se difícil realizar uma otimização de parâmetros utilizando o algoritmo genético, e portanto a seleção de parâmetros deve ser feita da forma tradicional para modelos de AM. Além disso, nos experimentos realizados, a quantidade modesta de dados pode resultar em *overfitting*, já que sua alta complexidade requer uma grande quantidade de dados para treinamento (JAMES et al., 2013).

O método proposto também foi executado substituindo o Algoritmo Genético pelo Optuna (AKIBA et al., 2019) para otimização dos parâmetros dos Modelo de Resíduo e do Modelo de Combinação. Assim, busca-se investigar o impacto do algoritmo de otimização na solução final. O Optuna consiste em um framework de otimização de hiper-parâmetros de AM de código aberto, que promete uma busca eficiente de hiper-parâmetros em grandes espaços de busca.

Para otimização dos parâmetros, o Optuna utiliza o método de Estimador de Parzen com Estrutura em Árvore Multivariado (MTPE) (BERGSTRA et al., 2011), que consiste em uma forma de otimização Bayesiana. Para reduzir o tempo de busca, é utilizado um método de *pruning* baseado na mediana, onde o treinamento é interrompido caso o resultado intermediário seja pior do que a mediana dos resultados dos modelos anteriores na mesma iteração.

Em comparação com o método proposto, o Optuna é baseado em otimização Bayesiana, que se mostra eficiente para otimizar poucos hiper-parâmetros mas degrada em performance consideravelmente quando o espaço de busca aumenta (LI et al., 2017). Por outro lado, algoritmos genéticos em geral se mantêm eficientes quando o espaços de busca se tornam extensos (GODEFROID; KHURSHID, 2002).

Como por padrão, a biblioteca Optuna não permite seleção de *lags*, foram utilizadas todos os *lags* até l_{max} para cada série como entrada dos modelos. Assim, no caso do Modelo de Resíduos, foram utilizados os l_{max} valores passados dos resíduos; Para o Modelo de combinação, foram utilizados os l_{max} valores passados das duas séries disponíveis: a previsão do SARIMA e os valores previstos pelo Modelo de Resíduos.

Finalmente, a comparação das métricas de erros gerais entre o método propostos e outros métodos pode ser visualizado na Tabela 2.

Tabela 2 – Comparação entre resultados de erros médios entre modelo proposto e literatura.

Model	R	RMSE (kWh)	MAE (kWh)	MAPE (%)	MaxAE (kWh)	p-value
SARIMA	0.435	0.116	0.091	60.52	0.321	7.9e-05
ANN	0.556	0.092	0.057	36.834	0.353	1.6e-06
SVR	0.441	0.101	0.070	43.224	0.33	1.4e-10
LSTM	0.832	0.049	0.029	14.317	0.277	0.0087
LR	0.445	0.101	0.062	38.517	0.395	1.2e-12
Class. & Regress. Tree	0.381	0.107	0.074	48.204	0.362	5.8e-17
SVR + LR	0.512	0.164	0.104	42.75	0.564	0.0259
Bagging with ANNs	0.607	0.094	0.049	42.31	0.338	0.0149
SARIMA-MetaFA- LSSVR	0.796	0.164	0.028	15.66	**0.179**	3.9e-18
SARIMA-PSO-LSSVR	0.757	0.182	0.035	16.19	0.220	–
MLP(SARIMA,SVR)	**0.854**	**0.046**	**0.025**	**12.74**	0.278	1.2e-07
Optuna: MLP(SARIMA,SVR)	0.850	0.047	0.028	14.32	0.264	0.3976

Assim, podemos concluir que o modelo proposto, representado pela combinação final *MLP(SARIMA,SVR)*, ou seja, um modelo combinador do tipo MLP e um modelo de resíduos do tipo SVR, resultaram em melhores métricas de erro do que todos os outros métodos analisados. Com exceção do MaxAE (Máximo Erro Absoluto).

Após a análise apresentada acima, foi realizada também uma comparação por dia da semana, a fim de identificar padrões do comportamento do modelo preditivo em relação a outras abordagens. Novamente, o método proposto obteve melhores resultados usando como referência quase todas as métricas, conforme a Tabela 3.

Outra maneira de comparar a melhoria do modelo proposto comparando-se com outros modelos da literatura consiste no cálculo do percentual do valor atingido por outros métodos tendo como referência as métricas de erro do modelo proposto. Tais valores foram calculados de acordo com as Equações 5.1 (para as métricas de erro) e 5.2 (para a métrica de correlação) e estão dispostos na Tabela 4.

$$PD = 100 \times \frac{Model_{LM} - Model_{PR}}{Model_{LM}}, \tag{5.1}$$

$$PD_R = 100 \times \frac{Model_{PR} - Model_{LM}}{Model_{LM}}, \tag{5.2}$$

Nas equações acima, $Model_{LM}$ representa o modelo da literatura e $Model_{PR}$ representa o modelo proposto. Assim, valores positivos de *PD (Diferencial Percentual)* representam uma melhor performance do método considerado, enquanto valores negativos indicam que o método considerado obteve piores resultados.

Finalmente, com o intuito de avaliar a significância estatística dos resultados atingidos, foi utilizado um teste de hipótese. O teste escolhido consiste no teste de Diebold-Mariano

Tabela 3 – Comparação das métricas entre o modelo proposto e literatura por dia da semana.

Approach	Model	Measure	Mon.	Tue.	Wed.	Thur.	Fri.	Sat.	Sun.	Max	Min
Single Model	SARIMA	R	0.192	0.519	0.332	0.602	0.733	0.527	0.138	0.733	0.138
		RMSE (kWh)	0.146	0.081	0.176	0.109	0.074	0.129	0.098	0.176	0.074
		MAE (kWh)	0.103	0.058	0.161	0.073	0.055	0.102	0.085	0.161	0.055
		MAPE (%)	49.79	26.23	103.20	47.52	34.01	80.35	82.53	103.20	26.23
		MaxAE (kWh)	**0.180**	0.226	0.983	0.265	0.178	0.204	0.213	0.983	0.178
	ANN	R	0.373	0.729	0.518	0.697	0.645	0.651	0.276	0.729	0.276
		RMSE (kWh)	0.097	0.068	0.116	0.105	0.073	0.068	0.116	0.116	0.068
		MAE (kWh)	0.058	0.045	0.068	0.083	0.051	0.052	0.044	0.083	0.044
		MAPE (%)	35.06	19.04	34.95	56.69	29.29	40.82	41.99	56.69	19.04
		MaxAE (kWh)	0.340	0.360	0.529	0.375	0.226	0.177	0.464	0.529	0.177
	SVR	R	0.339	0.371	0.345	0.725	0.535	0.584	0.19	0.725	0.19
		RMSE (kWh)	0.094	0.109	0.134	0.089	0.077	0.095	0.108	0.134	0.077
		MAE (kWh)	0.064	0.082	0.085	0.062	0.058	0.065	0.072	0.085	0.058
		MAPE (%)	34.78	37.25	34.53	36.96	34.76	49.67	74.62	74.62	34.53
		MaxAE (kWh)	0.308	0.371	0.526	0.335	0.215	0.279	0.276	0.526	0.215
	LSTM	R	0.745	0.839	0.874	0.751	0.825	0.879	0.911	0.911	0.745
		RMSE (kWh)	0.052	0.050	0.058	0.052	**0.042**	0.044	0.047	0.058	0.042
		MAE (kWh)	0.030	0.033	0.037	0.032	0.020	0.028	0.022	0.037	0.020
		MAPE (%)	14.18	13.74	16.71	16.81	**11.65**	14.84	12.28	16.81	11.65
		MaxAE (kWh)	0.307	0.239	0.308	0.289	0.310	0.166	0.318	0.318	0.166
	LR	R	0.296	0.326	0.525	0.593	0.57	0.563	0.239	0.593	0.239
		RMSE (kWh)	0.099	0.103	0.109	0.111	0.105	0.063	0.119	0.119	0.063
		MAE (kWh)	0.069	0.068	0.069	0.064	0.076	0.039	0.046	0.076	0.039
		MAPE (%)	42.62	28.15	37.32	42.89	51.92	22.79	43.93	51.92	22.79
		MaxAE (kWh)	0.348	0.363	0.390	0.390	0.355	0.264	0.655	0.655	0.264
	C&R Tree	R	0.392	0.519	0.381	0.355	0.425	0.512	0.080	0.519	0.080
		RMSE (kWh)	0.097	0.103	0.125	0.116	0.099	0.124	0.087	0.125	0.087
		MAE (kWh)	0.058	0.083	0.068	0.089	0.078	0.085	0.056	0.089	0.056
		MAPE (%)	35.06	40.41	26.47	65.49	48.96	63.43	57.61	65.49	26.47
		MaxAE (kWh)	0.409	0.262	0.538	0.347	0.327	0.346	0.305	0.538	0.262
Voting	SVR + LR	R	0.545	0.573	0.528	0.363	0.535	0.658	0.384	0.658	0.363
		RMSE (kWh)	0.087	0.131	0.126	0.167	0.362	0.101	0.174	0.362	0.087
		MAE (kWh)	0.048	0.064	0.058	0.056	0.178	0.171	0.153	0.178	0.048
		MAPE (%)	38.76	45.89	34.65	35.36	47.25	43.58	53.74	53.74	34.65
		MaxAE (kWh)	0.215	0.391	0.429	0.534	0.641	0.486	1.253	1.253	0.215
Bagging	ANN	R	0.551	0.616	0.658	0.623	0.734	0.651	0.416	0.734	0.416
		RMSE (kWh)	0.058	**0.016**	0.072	0.171	0.125	0.149	0.067	0.171	**0.016**
		MAE (kWh)	0.042	0.068	0.031	0.061	0.034	0.039	0.069	0.069	0.031
		MAPE (%)	34.82	54.95	38.19	35.31	28.07	42.62	62.19	62.19	28.07
		MaxAE (kWh)	0.277	0.549	**0.157**	**0.232**	0.451	0.381	0.319	0.549	0.157
Hybrid System	SARIMA-MetaFA-LSSVR	R	**0.817**	0.855	**0.910**	0.813	**0.867**	0.820	0.492	0.910	0.492
		RMSE (kWh)	0.179	0.180	0.180	0.170	0.172	0.168	0.100	0.180	0.100
		MAE (kWh)	0.032	0.032	0.032	0.029	0.030	0.028	**0.010**	0.032	**0.010**
		MAPE (%)	16.56	13.98	14.93	16.55	18.42	18.97	**10.19**	18.97	10.19
		MaxAE (kWh)	0.182	0.213	0.249	0.261	**0.151**	0.166	**0.033**	**0.261**	**0.033**
	SARIMA-PSO-LSSVR	R	0.731	0.632	0.845	**0.908**	**0.867**	0.825	0.493	0.908	0.493
		RMSE (kWh)	0.240	0.239	0.182	0.180	0.171	0.162	0.100	0.240	0.100
		MAE (kWh)	0.057	0.057	**0.033**	0.032	0.029	0.026	**0.010**	0.057	**0.010**
		MAPE (%)	33.85	22.45	14.28	14.95	18.00	17.40	**10.20**	33.85	**10.20**
		MaxAE (kWh)	0.240	0.458	0.234	0.244	**0.151**	0.182	**0.033**	0.458	**0.033**
	MLP(SARIMA,SVR)	R	0.773	0.856	0.880	0.842	0.819	0.892	**0.919**	**0.919**	**0.773**
		RMSE (kWh)	**0.049**	0.048	**0.056**	**0.042**	**0.042**	0.043	**0.045**	0.056	0.042
		MAE (kWh)	**0.028**	**0.030**	**0.031**	**0.024**	**0.019**	**0.026**	0.020	**0.031**	0.019
		MAPE (%)	14.36	**13.12**	**13.99**	**12.60**	11.80	**12.04**	11.27	**14.36**	11.27
		MaxAE (kWh)	0.326	0.236	0.302	0.277	0.315	0.190	0.302	0.326	0.190
	Optuna: MLP(SARIMA,SVR)	R	0.766	**0.870**	0.887	0.802	0.818	**0.907**	0.897	0.907	0.766
		RMSE	0.050	0.045	0.055	0.046	**0.042**	**0.039**	0.050	**0.055**	0.039
		MAE	0.029	0.031	0.034	0.028	0.022	**0.026**	0.024	0.034	0.022
		MAPE	**14.17**	14.05	16.33	15.46	13.27	12.74	14.19	16.33	12.74
		MaxAE	0.311	**0.170**	0.306	0.292	0.308	**0.141**	0.317	0.317	0.141

Tabela 4 – Percentual em relação ao método proposto.

Model	R	RMSE (kWh)	MAE (kWh)	MAPE (%)	MaxAE (kWh)
SARIMA	96.41	60.01	72.15	78.95	13.33
ANNs	53.67	49.58	55.54	65.41	21.18
SVR	93.74	54.07	63.79	70.53	15.69
LR	92.00	54.07	59.12	66.92	29.56
C&R Tree	124.25	56.64	65.75	73.57	23.14
SVR + LR	66.88	71.71	75.63	70.20	50.67
Bagging with ANNs	40.76	50.65	48.28	69.89	17.69
SARIMA-MetaFA-LSSVR	7.34	71.71	9.49	18.63	-55.43
SARIMA- PSO-LSSVR	12.87	74.51	27.59	21.30	-26.46

(DIEBOLD; MARIANO, 2002), capaz de avaliar diferenças na acurácias de dois modelos. Definindo os erros de previsão de um modelo i como $e_{it} = \hat{y_{it}} - y_t$, temos que o diferencial entre dois modelos é dado por $d_t = g(e_{1t}) - g(e_{2t})$, onde $g(\cdot)$ é uma função de custo, geralmente associada ao erro quadrático. A hipótese nula desse teste considera o valor esperado para o diferencial nulo, conforme visto na Equação 5.3.

$$\begin{cases} H_0 : E(d_t) = 0 \\ H_1 : E(d_t) \neq 0 \end{cases} \tag{5.3}$$

Considerando as hipóteses apresentadas, foi calculado o *p-value* para cada par de modelos. Considerando um teste com um intervalo de confiança de 95%, valores menores que 0.05 indicam uma diferença significativa entre os modelos. Os resultados obtidos pelo teste de hipótese utilizado podem ser visualizados na Tabela 5.

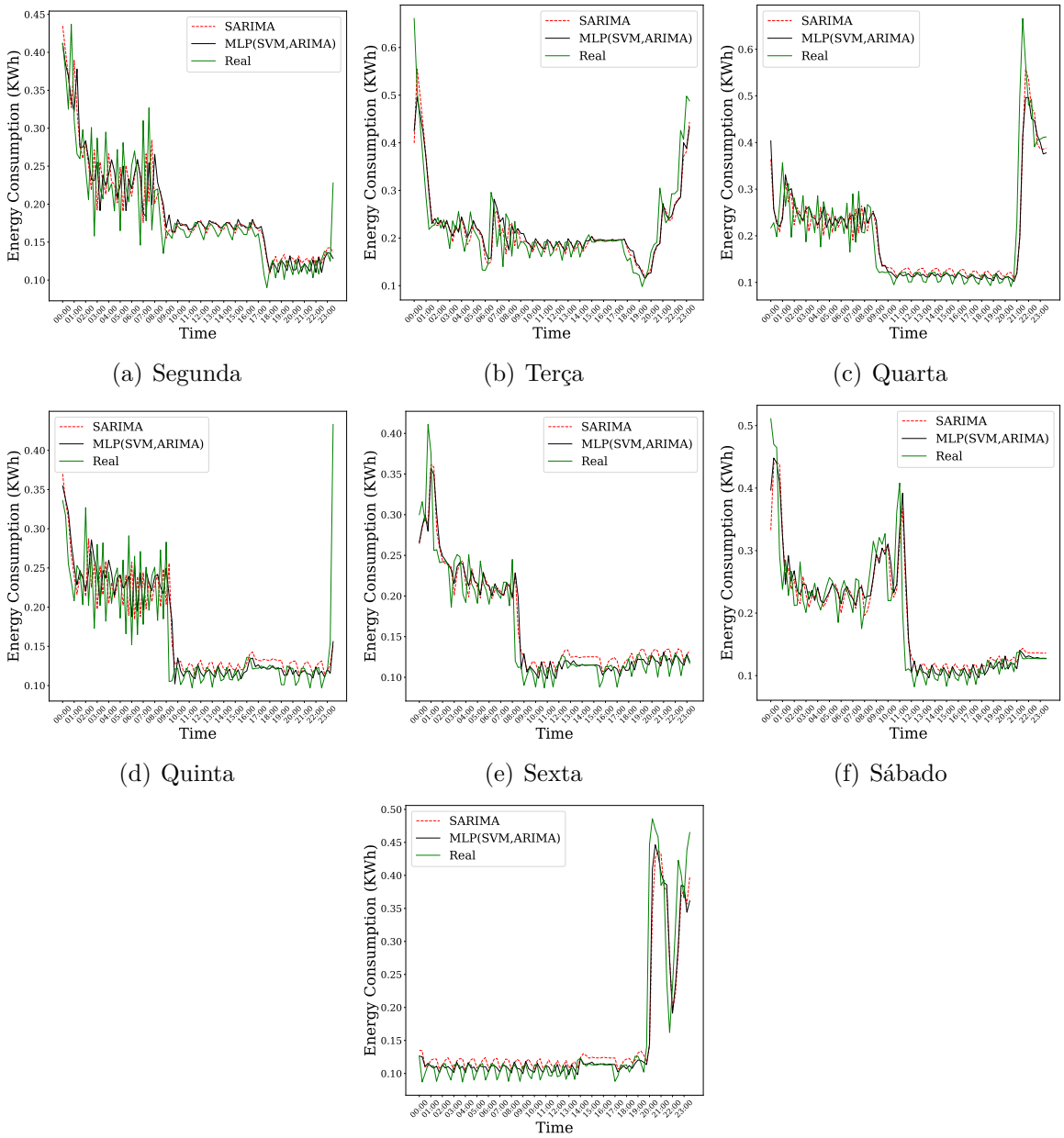
Tabela 5 – P-Values do teste de hipótese de Diebold-Mariano.

Models	P-Value
SARIMA	7.9e-05
MLP	1.6e-06
SVR	1.4e-10
LSTM	0.0087
LR	1.2e-12
C&R Tree	5.8e-17
SVR + LR	0.0259
Bagging with MLPs	0.0149
SARIMA-MetaFA-LSSVR	3.9e-18
SARIMA-PSO-LSSVR	1.2e-07
Optuna: MLP(SARIMA,SVR)	0.3976

Por fim, para avaliar o efeito do modelo de resíduos no resultado final da previsão da série de consumo, foram plotados para cada dia semana, um gráfico contendo os valores

reais da série em conjunto com as previsões realizadas pelo modelo ARIMA e pelo modelo final. Assim, pode-se observar a influência dos modelos de resíduo e de combinação nos padrões não lineares apresentados na série de consumo de energia. Os resultados para cada dia da semana podem ser visualizados na Figura 26.

Figura 26 – Comparação entre o valor real da série temporal, do método proposto e das previsões do modelo ARIMA.



(a) Segunda (b) Terça (c) Quarta

(d) Quinta (e) Sexta (f) Sábado

(g) Domingo

Fonte: Elaborada pelo autor.

6 CONCLUSÕES

De acordo com os experimentos realizados, se pôde observar que o método proposto possui grande valor na construção de modelos híbridos para previsão de séries de consumo de energia. A combinação de modelos lineares e modelos de AM se mostrou capaz de modelar os padrões lineares e não-lineares da série temporal, e o algoritmo genético se mostrou capaz de realizar a otimização dos modelos não-lineares de previsão de resíduos e de combinação.

De acordo com as cinco métricas utilizadas na avaliação experimental, indica-se que o sistema proposto alcança melhorias estatisticamente significantes nos resultados quando comparados com modelos estatísticos, híbridos e de AM da literatura.

6.1 TRABALHOS FUTUROS

6.1.1 Aplicações

O trabalho proposto realizou previsões referentes ao consumo geral do edifício. Com a presença de dados em cada cômodo, é possível estender a abordagem proposta para prever o consumo em cada cômodo individualmente. Tal abordagem permitiria entender os padrões de consumo em cada cômodo de forma individualizada. Assim, poderia-se detalhar mais especificamente a dinâmica de consumo em uma determinado edifício que não compartilha das mesmas características e estrutura física.

Outro avanço possível a ser considerado neste trabalho consiste na introdução de variáveis externas como temperatura e dia da semana como variáveis de entrada. Como descrito no Capítulo 5, os experimentos descritos nesse trabalho apenas consideraram os valores anteriores da série para a previsão do consumo de energia. Possivelmente, a introdução de variáveis exógenas pode aumentar a assertividade da previsão.

O método descrito por este trabalho pode também ser adaptado para outros problemas de previsão de séries temporais e de aprendizado de máquina em geral. Portanto, um possível trabalho futuro consiste na extensão do método utilizado neste trabalho para solucionar problemas de outra natureza. Sendo assim, pode-se aplicar o método descrito para aplicações financeiras, médicas, sociais e etc, ou qualquer outra situação onde busca-se uma otimização da construção de modelos híbridos com modelos de aprendizado de máquina para uma maior assertividade na previsão.

6.1.2 Método Proposto

Conforme descrito no Capítulo 3, algoritmos genéticos pertencem à uma classe de algoritmos onde pode-se encontrar na literatura diversas estratégias de seleção, mutação e reprodução. Mudanças nessas estratégias podem afetar dramaticamente a convergência e o resultado final do algoritmo genético.

O processo de *crossover* realizado nesse trabalho, por exemplo, pode ser substituído por um *crossover* de ponto duplo, onde são selecionados dois pontos diferentes do cromossomo para realizar a recombinação das partes. Algoritmos de seleção também podem ser modificados, podendo-se utilizar métodos diferentes do método da roleta, como a seleção por torneio ou seleção truncada, dentre diversas outras possibilidades de modificações.

Durante o algoritmo genético, cada gene no cromossomos representa uma possibilidade dentro de uma lista finita definida previamente. Assim, dado as possíveis configurações utilizadas nos experimentos deste trabalho, e é possível realizar uma expansão para incluir novas possibilidades. O número de neurônios de cada camada da MLP nos nossos experimentos é realizado em incrementos exponenciais de base 2 e limitado em 2048, por exemplo. Tal limitação deve-se principalmente ao crescimento do tempo de execução do algoritmo para redes neurais mais complexas, o que também requer uma maior capacidade de processamento. A inclusão de números de neurônios intermediários ou o aumento do número máximo de neurônios pode ser feita em uma máquina com maior poder de processamento, possibilitando assim modelos mais complexos de serem testados para cada problema.

Pode-se também introduzir novas funções de ativações mais recentes como GELU, SiLU, Softplus, PReLU e LeakyReLU. Esta última, por exemplo, soluciona problemas presentes em funções de ativação do tipo ReLU que foi utilizada nos experimentos, como o desaparecimento do gradiente.

Por fim, os parâmetros utilizados do algoritmo genético foram definidos empiricamente e atingiram resultados satisfatórios. No entanto, devido à uma grande possibilidade de valores de parâmetros um estudo sobre como os parâmetros afetam o resultado final e a convergência do modelo trata-se de um problema em aberto que pode ser explorado.

A extensão dos parâmetros possíveis de cada modelo também pode ser expandido. O valor máximo de neurônios em cada camada deve-se especialmente à capacidade de processamento limitado nos experimentos. Sendo assim, a execução do algoritmo em ambientes com maior capacidade de processamento permitirá a avaliação de modelos MLP mais complexos como soluções candidatas.

REFERÊNCIAS

AHMAD, T.; CHEN, H. A review on machine learning forecasting growth trends and their real-time applications in different energy systems. *Sustainable Cities and Society*, v. 54, p. 102010, 2020. ISSN 2210-6707.

AKIBA, T.; SANO, S.; YANASE, T.; OHTA, T.; KOYAMA, M. Optuna: A next-generation hyperparameter optimization framework. In: *Proceedings of the 25rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. [S.l.: s.n.], 2019.

ALLOUHI, A.; FOUIH, Y. E.; KOUSKSOU, T.; JAMIL, A.; ZERAOULI, Y.; MOURAD, Y. Energy consumption and efficiency in buildings: current status and future trends. *Journal of Cleaner production*, Elsevier, v. 109, p. 118–130, 2015.

ALPAYDIN, E. *Introduction to machine learning*. [S.l.]: MIT press, 2020.

ASTERIOU, D.; HALL, S. G. Applied econometrics: a modern approach, revised edition. *Hampshire: Palgrave Macmillan*, v. 46, n. 2, p. 117–155, 2007.

AZEVEDO, F. A.; CARVALHO, L. R.; GRINBERG, L. T.; FARFEL, J. M.; FERRETTI, R. E.; LEITE, R. E.; LENT, R.; HERCULANO-HOUZEL, S. et al. Equal numbers of neuronal and nonneuronal cells make the human brain an isometrically scaled-up primate brain. *Journal of Comparative Neurology*, Wiley Online Library, v. 513, n. 5, p. 532–541, 2009.

BARZOLA-MONTESES, J.; ESPINOZA-ANDALUZ, M.; MITE-LEÓN, M.; FLORES-MORÁN, M. Energy consumption of a building by using long short-term memory network: A forecasting study. In: IEEE. *2020 39th International Conference of the Chilean Computer Science Society (SCCC)*. [S.l.], 2020. p. 1–6.

BERGSTRA, J.; BARDENET, R.; BENGIO, Y.; KÉGL, B. Algorithms for hyper-parameter optimization. In: NEURAL INFORMATION PROCESSING SYSTEMS FOUNDATION. *25th annual conference on neural information processing systems (NIPS 2011)*. [S.l.], 2011. v. 24.

BEZERRA, S. *Reservoir Computing com Hierarquia para Previsão de Vazões Médias Diárias*. Dissertação (Mestrado) — Universidade de Pernambuco, 2016.

BLICKLE, T.; THIELE, L. A comparison of selection schemes used in evolutionary algorithms. *Evolutionary Computation*, MIT Press, v. 4, n. 4, p. 361–394, 1996.

BONTEMPI, G.; TAIEB, S. B.; BORGNE, Y.-A. L. Machine learning strategies for time series forecasting. In: SPRINGER. *European business intelligence summer school*. [S.l.], 2012. p. 62–77.

BOUKTIF, S.; FIAZ, A.; OUNI, A.; SERHANI, M. Multi-sequence lstm-rnn deep learning and metaheuristics for electric load forecasting. *Energies*, v. 13, n. 2, 2020.

BOX, G. E.; JENKINS, G. M.; REINSEL, G. C.; LJUNG, G. M. *Time series analysis: forecasting and control*. [S.l.]: John Wiley & Sons, 2015.

BROCKWELL, P. J.; DAVIS, R. A.; CALDER, M. V. *Introduction to time series and forecasting*. [S.l.]: Springer, 2002. v. 2.

Chan, S. C.; Tsui, K. M.; Wu, H. C.; Hou, Y.; Wu, Y.; Wu, F. F. Load/price forecasting and managing demand response for smart grids: Methodologies and challenges. *IEEE Signal Processing Magazine*, v. 29, n. 5, p. 68–85, 2012.

CHOU, J.-S.; NGO, N.-T. Smart grid data analytics framework for increasing energy savings in residential buildings. *Automation in construction*, Elsevier, v. 72, p. 247–257, 2016.

CHOU, J.-S.; NGO, N.-T. Time series analytics using sliding window metaheuristic optimization-based machine learning system for identifying building energy consumption patterns. *Applied Energy*, v. 177, p. 751 – 770, 2016.

CHOU, J.-S.; TRUONG, D.-N. Multistep energy consumption forecasting by metaheuristic optimization of time-series analysis and machine learning. *International Journal of Energy Research*, v. 45, n. 3, p. 4581–4612, 2021.

CHOU, J.-S.; TRUONG, D.-N. A novel metaheuristic optimizer inspired by behavior of jellyfish in ocean. *Applied Mathematics and Computation*, v. 389, p. 125535, 2021.

CLEMENTS, M. P.; FRANSES, P. H.; SWANSON, N. R. Forecasting economic and financial time-series with non-linear models. *International Journal of Forecasting*, Elsevier, v. 20, n. 2, p. 169–183, 2004.

COLEY, D. A. *An introduction to genetic algorithms for scientists and engineers*. [S.l.]: World Scientific Publishing Company, 1999.

CRISTIANINI, N.; SHAWE-TAYLOR, J. et al. *An introduction to support vector machines and other kernel-based learning methods*. [S.l.]: Cambridge university press, 2000.

CULABA, A. B.; ROSARIO, A. J. R. D.; UBANDO, A. T.; CHANG, J.-S. Machine learning-based energy consumption clustering and forecasting for mixed-use buildings. *International Journal of Energy Research*, Wiley Online Library, v. 44, n. 12, p. 9659–9673, 2020.

DEB, C.; ZHANG, F.; YANG, J.; LEE, S. E.; SHAH, K. W. A review on time series forecasting techniques for building energy consumption. *Renewable and Sustainable Energy Reviews*, v. 74, p. 902–924, 2017.

DENG, L.; LIU, Y. *Deep learning in natural language processing*. [S.l.]: Springer, 2018.

DIEBOLD, F. X.; MARIANO, R. S. Comparing predictive accuracy. *Journal of Business & economic statistics*, Taylor & Francis, v. 20, n. 1, p. 134–144, 2002.

DOMINGOS, S. d. O.; OLIVEIRA, J. F. de; NETO, P. S. de M. An intelligent hybridization of arima with machine learning models for time series forecasting. *Knowledge-Based Systems*, Elsevier, v. 175, p. 72–86, 2019.

DUCHI, J.; HAZAN, E.; SINGER, Y. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, v. 12, n. Jul, p. 2121–2159, 2011.

EL-HENDAWI, M.; WANG, Z. An ensemble method of full wavelet packet transform and neural network for short term electrical load forecasting. *Electric Power Systems Research*, v. 182, p. 106265, 2020.

ELSKEN, T.; METZEN, J. H.; HUTTER, F. Neural architecture search: A survey. *arXiv preprint arXiv:1808.05377*, 2018.

FARUK, D. Ö. A hybrid neural network and arima model for water quality time series prediction. *Engineering applications of artificial intelligence*, Elsevier, v. 23, n. 4, p. 586–594, 2010.

FEKRI, M. N.; PATEL, H.; GROLINGER, K.; SHARMA, V. Deep learning for load forecasting with smart meter data: Online adaptive recurrent neural network. *Applied Energy*, v. 282, p. 116177, 2021.

GAJOWNICZEK, K.; ZĄBKOWSKI, T. Short term electricity forecasting using individual smart meter data. *Procedia Computer Science*, Elsevier, v. 35, p. 589–597, 2014.

GAO, Y.; RUAN, Y.; FANG, C.; YIN, S. Deep learning and transfer learning models of energy consumption forecasting for a building with poor information data. *Energy and Buildings*, v. 223, p. 110156, 2020.

GERWEN, R. V.; JAARSMA, S.; WILHITE, R. Smart metering. *Leonardo-energy. org*, v. 9, 2006.

GODEFROID, P.; KHURSHID, S. Exploring very large state spaces using genetic algorithms. In: SPRINGER. *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. [S.l.], 2002. p. 266–280.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A.; BENGIO, Y. *Deep learning*. [S.l.]: MIT press Cambridge, 2016. v. 1.

GOOIJER, J. G. D.; HYNDMAN, R. J. 25 years of time series forecasting. *International journal of forecasting*, Elsevier, v. 22, n. 3, p. 443–473, 2006.

HAGAN, M. T.; DEMUTH, H. B.; BEALE, M. H. et al. *Neural network design*. [S.l.]: Pws Pub. Boston, 1996. v. 20.

HANKE, J. E.; REITSCH, A. G.; WICHERN, D. W. *Business forecasting*. [S.l.]: Prentice Hall New Jersey, 2001. v. 9.

HAYKIN, S. S. *Neural networks and learning machines*. [S.l.]: Pearson Upper Saddle River, NJ, USA, 2009. v. 3.

HERBRICH, R. *Learning kernel classifiers: theory and algorithms*. [S.l.]: MIT press, 2001.

HEYDARI, A.; GARCIA, D. A.; KEYNIA, F.; BISEGNA, F.; SANTOLI, L. D. Hybrid intelligent strategy for multifactor influenced electrical energy consumption forecasting. *Energy Sources, Part B: Economics, Planning, and Policy*, Taylor & Francis, v. 14, n. 10-12, p. 341–358, 2019.

Heydari, A.; Keynia, F.; Garcia, D. A.; De Santoli, L. Mid-term load power forecasting considering environment emission using a hybrid intelligent approach. In: *2018 5th International Symposium on Environment-Friendly Energies and Applications (EFEA)*. [S.l.: s.n.], 2018. p. 1–5.

HEYDARI, A.; NEZHAD, M. M.; PIRSHAYAN, E.; GARCIA, D. A.; KEYNIA, F.; SANTOLI, L. D. Short-term electricity price and load forecasting in isolated power grids based on composite neural network and gravitational search optimization algorithm. *Applied Energy*, Elsevier, v. 277, p. 115503, 2020.

HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. *Neural computation*, MIT Press, v. 9, n. 8, p. 1735–1780, 1997.

HOLLAND, J. H. et al. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. [S.l.]: MIT press, 1975.

JAMES, G.; WITTEN, D.; HASTIE, T.; TIBSHIRANI, R. *An introduction to statistical learning*. [S.l.]: Springer, 2013. v. 112.

JONG, K. D. Learning with genetic algorithms: An overview. *Machine learning*, Springer, v. 3, n. 2-3, p. 121–138, 1988.

KABALCI, Y. A survey on smart metering and smart grid communication. *Renewable and Sustainable Energy Reviews*, Elsevier, v. 57, p. 302–318, 2016.

KANDEL, E. R.; SCHWARTZ, J. H.; JESSELL, T. M.; SIEGELBAUM, S. A.; HUDSPETH, A. J. et al. *Principles of neural science*. [S.l.]: McGraw-hill New York, 2000. v. 4.

KATOCH, S.; CHAUHAN, S. S.; KUMAR, V. A review on genetic algorithm: past, present, and future. *Multimedia Tools and Applications*, Springer, p. 1–36, 2020.

KHASHEI, M.; BIJARI, M. A novel hybridization of artificial neural networks and arima models for time series forecasting. *Applied Soft Computing*, Elsevier, v. 11, n. 2, p. 2664–2675, 2011.

KINGMA, D. P.; BA, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

KLUG, A.; PARK, S.-C.; KRUG, J. Recombination and mutational robustness in neutral fitness landscapes. *PLoS computational biology*, Public Library of Science, v. 15, n. 8, p. e1006884, 2019.

KOMATSU, H.; KIMURA, O. Peak demand alert system based on electricity demand forecasting for smart meter data. *Energy and Buildings*, v. 225, p. 110307, 2020. ISSN 0378-7788. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0378778820307854>.

LEE, C.; ANTONSSON, E. Variable length genomes for evolutionary algorithms. In: *GECCO*. [S.l.: s.n.], 2000. v. 2000, p. 806.

LESHNO, M.; LIN, V. Y.; PINKUS, A.; SCHOCKEN, S. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural networks*, Elsevier, v. 6, n. 6, p. 861–867, 1993.

LI, L.; JAMIESON, K.; DESALVO, G.; ROSTAMIZADEH, A.; TALWALKAR, A. Hyperband: A novel bandit-based approach to hyperparameter optimization. *The Journal of Machine Learning Research*, JMLR. org, v. 18, n. 1, p. 6765–6816, 2017.

LI, L.; MEINRENKEN, C. J.; MODI, V.; CULLIGAN, P. J. Short-term apartment-level load forecasting using a modified neural network with selected auto-regressive features. *Applied Energy*, v. 287, p. 116509, 2021. ISSN 0306-2619.

LUSIS, P.; KHALILPOUR, K. R.; ANDREW, L.; LIEBMAN, A. Short-term residential load forecasting: Impact of calendar effects and forecast granularity. *Applied Energy*, Elsevier, v. 205, p. 654–669, 2017.

MAKRIDAKIS, S.; WHEELWRIGHT, S. C.; HYNDMAN, R. J. *Forecasting methods and applications*. [S.l.]: John wiley & sons, 2008.

MARINO, D. L.; AMARASINGHE, K.; MANIC, M. Building energy load forecasting using deep neural networks. In: IEEE. *IECON 2016-42nd Annual Conference of the IEEE Industrial Electronics Society*. [S.l.], 2016. p. 7046–7051.

MCCALL, J. Genetic algorithms for modelling and optimisation. *Journal of computational and Applied Mathematics*, Elsevier, v. 184, n. 1, p. 205–222, 2005.

MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, Springer, v. 5, n. 4, p. 115–133, 1943.

MINSKI, M. L.; PAPERT, S. A. Perceptrons: an introduction to computational geometry. *MA: MIT Press, Cambridge*, 1969.

MORETTIN, P. A.; TOLOI, C. Análise de séries temporais. In: *Análise de séries temporais*. [S.l.: s.n.], 2006. p. 538–538.

NIELSEN, M. A. *Neural Networks and Deep Learning*. [S.l.]: Determination Press, 2015.

PAI, P.-F.; LIN, C.-S. A hybrid arima and support vector machines model in stock price forecasting. *Omega*, Elsevier, v. 33, n. 6, p. 497–505, 2005.

PALIWAL, M.; KUMAR, U. A. Neural networks and statistical techniques: A review of applications. *Expert systems with applications*, Elsevier, v. 36, n. 1, p. 2–17, 2009.

PANIGRAHI, S.; BEHERA, H. S. A hybrid ets–ann model for time series forecasting. *Engineering Applications of Artificial Intelligence*, Elsevier, v. 66, p. 49–59, 2017.

PINTO, T.; PRAçA, I.; VALE, Z.; SILVA, J. Ensemble learning for electricity consumption forecasting in office buildings. *Neurocomputing*, v. 423, p. 747–755, 2021. ISSN 0925-2312.

RANA, M.; KOPRINSKA, I. Forecasting electricity load with advanced wavelet neural networks. *Neurocomputing*, v. 182, p. 118–132, 2016.

ROTHLAUF, F. Representations for genetic and evolutionary algorithms. In: *Representations for Genetic and Evolutionary Algorithms*. [S.l.]: Springer, 2006. p. 9–32.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning representations by back-propagating errors. *nature*, Nature Publishing Group, v. 323, n. 6088, p. 533, 1986.

SAJJAD, M.; KHAN, Z. A.; ULLAH, A.; HUSSAIN, T.; ULLAH, W.; LEE, M. Y.; BAIK, S. W. A novel cnn-gru-based hybrid approach for short-term residential load forecasting. *IEEE Access*, IEEE, v. 8, p. 143759–143768, 2020.

SCHWERT, G. W. Tests for unit roots: A monte carlo investigation. *Journal of Business & Economic Statistics*, Taylor & Francis, v. 20, n. 1, p. 5–17, 2002.

SIAMI-NAMINI, S.; TAVAKOLI, N.; NAMIN, A. S. A comparison of arima and lstm in forecasting time series. In: IEEE. *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*. [S.l.], 2018. p. 1394–1401.

SIVANANDAM, S.; DEEPA, S. Genetic algorithms. In: *Introduction to genetic algorithms*. [S.l.]: Springer, 2008.

SMOLA, A. J.; BARTLETT, P.; SCHÖLKOPF, B.; SCHUURMANS, D. Introduction to large margin classifiers. MIT Press, 2000.

SMOLA, A. J.; SCHÖLKOPF, B. A tutorial on support vector regression. *Statistics and computing*, Springer, v. 14, n. 3, p. 199–222, 2004.

SOMU, N.; Raman M R, G.; RAMAMRITHAM, K. A deep learning framework for building energy consumption forecast. *Renewable and Sustainable Energy Reviews*, v. 137, p. 110591, 2021.

STEINWART, I.; CHRISTMANN, A. *Support vector machines*. [S.l.]: Springer Science & Business Media, 2008.

SUTTON, R. S. Two problems with backpropagation and other steepest-descent learning procedures for networks. In: *Proceedings of Eightth Annual Conference of the Cognitive Science Society, 1986*. [S.l.: s.n.], 1986.

TASKAYA-TEMIZEL, T.; CASEY, M. C. A comparative study of autoregressive neural network hybrids. *Neural Networks*, v. 18, n. 5, p. 781 – 789, 2005. ISSN 0893-6080. IJCNN 2005.

TEALAB, A. Time series forecasting using artificial neural networks methodologies: A systematic review. *Future Computing and Informatics Journal*, Elsevier, v. 3, n. 2, p. 334–340, 2018.

TIELEMAN, T.; HINTON, G. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, v. 4, n. 2, p. 26–31, 2012.

TONG, H. A personal overview of non-linear time series analysis from a chaos perspective. In: *Exploration Of A Nonlinear World: An Appreciation of Howell Tong's Contributions to Statistics*. [S.l.]: World Scientific, 2009. p. 183–229.

VAFAIE, H.; JONG, K. A. D. Genetic algorithms as a tool for feature selection in machine learning. In: *ICTAI*. [S.l.: s.n.], 1992. p. 200–203.

VAPNIK, V. Pattern recognition using generalized portrait method. *Automation and remote control*, v. 24, p. 774–780, 1963.

VARMA, S.; DAS, S. *Deep Learning*. [S.l.]: Bookdown, 2018.

WALTHER, J.; WEIGOLD, M. A systematic review on predicting and forecasting the electrical energy consumption in the manufacturing industry. *Energies*, v. 14, n. 4, 2021.

WANG, Y.; GAN, D.; SUN, M.; ZHANG, N.; LU, Z.; KANG, C. Probabilistic individual load forecasting using pinball loss guided lstm. *Applied Energy*, Elsevier, v. 235, p. 10–20, 2019.

XAVIER, C. R.; SANTOS, E. P. dos; VIEIRA, V. da F.; SANTOS, R. W. dos. Genetic algorithm for the history matching problem. *Procedia Computer Science*, Elsevier, v. 18, p. 946–955, 2013.

YOUNG, S. R.; ROSE, D. C.; KARNOWSKI, T. P.; LIM, S.-H.; PATTON, R. M. Optimizing deep learning hyper-parameters through an evolutionary algorithm. In: *Proceedings of the Workshop on Machine Learning in High-Performance Computing Environments*. [S.l.: s.n.], 2015.

YU, C.-N.; MIROWSKI, P.; HO, T. K. A sparse coding approach to household electricity demand forecasting in smart grids. *IEEE Transactions on Smart Grid*, IEEE, v. 8, n. 2, p. 738–748, 2016.

ZHANG, G.; PATUWO, B. E.; HU, M. Y. Forecasting with artificial neural networks:: The state of the art. *International journal of forecasting*, Elsevier, v. 14, n. 1, p. 35–62, 1998.

ZHANG, G. P. Time series forecasting using a hybrid arima and neural network model. *Neurocomputing*, Elsevier, v. 50, p. 159–175, 2003.

ZHANG, L.; WEN, J.; LI, Y.; CHEN, J.; YE, Y.; FU, Y.; LIVINGOOD, W. A review of machine learning in building load prediction. *Applied Energy*, v. 285, p. 116452, 2021.

ZHAO, H.-x.; MAGOULÈS, F. A review on the prediction of building energy consumption. *Renewable and Sustainable Energy Reviews*, Elsevier, v. 16, n. 6, p. 3586–3592, 2012.

ZHU, B.; WEI, Y. Carbon price forecasting with a novel hybrid arima and least squares support vector machines methodology. *Omega*, Elsevier, v. 41, n. 3, p. 517–524, 2013.

ZHUKOV, A. V.; SIDOROV, D. N.; FOLEY, A. M. Random forest based approach for concept drift handling. In: SPRINGER. *International Conference on Analysis of Images, Social Networks and Texts*. [S.l.], 2016. p. 69–77.

ZOLFAGHARI, M.; GOLABI, M. R. Modeling and predicting the electricity production in hydropower using conjunction of wavelet transform, long short-term memory and random forest models. *Renewable Energy*, v. 170, p. 1367–1381, 2021.