RODRIGO DA SILVA CUNHA

Previsão de carga utilizando redes neurais LSTM

São Paulo
(2019)

RODRIGO DA SILVA CUNHA

Previsão de carga utilizando redes neurais LSTM

Projeto de Formatura apresentado à Escola Politécnica da Universidade de São Paulo para obtenção do título de Bacharel em Engenharia

Orientador: Prof. Dr. Edson Satoshi Gomi

São Paulo

(2019)

RODRIGO DA SILVA CUNHA

Previsão de carga utilizando redes neurais LSTM

Projeto de Formatura apresentado à Escola Politécnica da Universidade de São Paulo para obtenção do título de Bacharel em Engenharia

Área de Concentração: Engenharia Elétrica

Orientador: Prof. Dr. Edson Satoshi Gomi

São Paulo

(2019)

De acordo com o depósito da monografia.

04/07/2019

Catalogação-na-publicação

Cunha, Rodrigo
Previsão de carga utilizando redes neurais LSTM / R. Cunha -- São Paulo, 2019.
88 p.

Trabalho de Formatura - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Energia e Automação Elétricas.

1.REDES NEURAIS 2.CARGA ELÉTRICA I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Energia e Automação Elétricas II.t.

ESCOLA POLITÉCNICA DA UNIVERSIDADE DE SÃO PAULO

PEA3507 Projeto de Formatura II

TERMO DE ORIGINALIDADE[1]

Eu abaixo qualificado como aluno regularmente matriculado no Curso de Engenharia de Energia e Automação Elétricas da Escola Politécnica da USP, declaro que o relatório final "Previsão de carga utilizando redes neurais LSTM" entregue em versão eletrônica, orientada pelo professor Dr. Edson Satoshi Gomi, no primeiro semestre de 2019, como requisito para obtenção do título de Bacharel em Engenharia, é produto de minha elaboração própria, desenvolvida com base em referências devidamente indicadas com base nas Normas Técnicas vigentes para indicação de citações (diretas e indiretas) e referências, e não representa plágio de qualquer material existente e disponível em qualquer meio.

Estou cônscio: (a) do que é plágio; (b) que copiar, total ou parcialmente, um fruto de trabalho realizado por outra pessoa é inaceitável, mesmo que seja de domínio público; pelo Código de Ética da USP, Resolução 4871 de 22/10/2001, Art. 23 – "É vedado aos membros do corpo discente e demais alunos da Universidade lançar mão de meios e artifícios que possam fraudar a avaliação do desempenho, seu ou de outrem, em atividades acadêmicas, culturais, artísticas, desportivas e sociais, no âmbito da Universidade, e acobertar a eventual utilização desses meios"; (c) dos requisitos expostos Lei 9.610, de 19/02/1998, referente a Direitos Autorais; (d) das penalidades previstas nos artigos 297-299 do Código Penal Brasileiro.

São Paulo, 04 de julho de 2019.

Aluno

assinatura: Rodrigo Cunha

Nome
RG e CPF Rodrigo da Silva Cunha
RG: MG-13.991.553
CPF: 104.512.486-94

[1] Adaptado a partir do Termo de Originalidade utilizado na Disciplina PQI-2000 do Departamento de Engenharia Química da EPUSP.

À minha mãe, meu pai e meu irmão, que nunca mediram esforços para me ajudar e sempre estiveram ao meu lado.

AGRADECIMENTOS

Agradeço aos Professores Dr. Edson Gomi e Dr. Giovanni Manassero pelo apoio na escolha do tema e ao longo do desenvolvimento do trabalho.

RESUMO

A evolução das redes elétricas inteligentes promete flexibilidade sem precedentes quanto à eficiência da gestão da energia. Para isso modelos capazes de fazer previsão de demanda energética em diferentes níveis de agregação são elementos cruciais. Embora a literatura sobre de previsão de carga seja extensa, com os primeiros artigos publicados sobre assunto na década de 1960, o problema possui solução difícil até os dias de hoje. Por outro lado, o grande sucesso obtido por modelos de aprendizado máquina profundo em outras áreas, como no campo de visão computacional e de processamento de linguagem natural, chamou a atenção para aplicação desses modelos para a previsão de carga. Nesse trabalho serão apresentadas aplicações de diferentes modelos baseados em redes neurais LSTM no problema da previsão de carga, considerando um conjunto de dados que representa consumo elétrico de uma única residência.

Palavras-Chave: Aprendizado Profundo. Redes Neurais Recorrentes. LSTM. Previsão de carga.

ABSTRACT

The evolution of smart grids promises great flexibility in energy management. Therefore, models that are capable of forecasting power consumption in different levels of aggregation are crucial elements. Despite the literature on load forecasting being extremelly vast, with the first articles being published in the decade of 1960, the problem persists to have very difficult solution. On the other side, the great success achieved by deep learning models on other areas, such as computer vision and natural language processing, has caught the attention for the suitability of theses models for the load forecasting problem. This work presentes the results of different models based on LSTM neural networks on the load forecasting problem, considering the point of view of a single household.

Keywords: Deep Learning. Recurrent Neural Networks. LSTM. Load Forecasting.

SUMÁRIO

1 INTRODUÇÃO 12
2 REVISÃO DA LITERATURA. 13
2.1 PREVISÃO DE CARGA 14
2.1.1 A previsão de carga ao longo do século XX e início do século XX . 14
2.1.2 A importância da previsão de carga 17
2.2 CLASSIFICAÇÃO DAS TÉCNICAS DE PREVISÃO DE CARGA . . 18
2.2.1 Previsão determinística e probabilística 18
2.2.2 Níveis de agregação 19
2.2.3 Horizonte de previsão 20
2.2.4 Objetivo de previsão 22
2.2.5 Modelos lineares e não-lineares 23
2.3 MODERNIZAÇÃO DO SISTEMA ELÉTRICO 25
2.3.1 *Smart grids* 25
2.3.2 Geração distribuída 25
2.3.3 Virtual power plants 26
2.3.4 *Microgrids* 26
2.3.5 *Smart buildings* e *smart environments* 27
2.4 *DEEP LEARNING* 28
2.4.1 Aprendizado de máquina 28
2.4.1.1 Algoritmos de aprendizado 28
2.4.1.1.1 A tarefa, T 29
2.4.1.1.2 A performance, P 30
2.4.1.1.3 A experiência, E 31
2.4.1.2 *Overfitting e underfitting* 31
2.4.1.3 Construindo um algoritmo de aprendizado 33
2.4.2 Redes neurais 33
2.4.2.1 O neurônio artificial: perceptron 35
2.4.2.2 Funções de ativação 36
2.4.2.3 *Multilayer perceptron* 38
2.4.2.4 Treinamento por gradiente e *backpropagation* 39
2.4.2.4.1 Função de custo 40

2.4.2.4.2 Gradiente descendente estocástico 41
2.4.2.4.3 Backpropagation 42
2.4.3 Redes neurais recorrentes 43
2.4.3.1 O problema das dependências de longo prazo 44
2.4.3.2 Redes LSTM 45
2.4.3.2.1 A ideia central por trás das redes LSTM 46
2.4.3.2.2 Passo a passo de uma rede LSTM 47
3 MATERIAL E MÉTODOS 50
3.1 MATERIAL 50
3.1.1 Artigo de referência 51
3.1.2 *Datasets* 52
3.1.2.1 *Individual household electric power consumption* 52
3.1.2.2 *Smartmeter energy consumption data in London households* 53
3.1.2.3 Lista de séres utilizadas. 54
3.1.3 Computação na nuvem 55
3.1.3.1 *Cloud Datalab* 56
3.1.3.2 *Cloud AI Platform* 57
3.1.3.3 *Cloud Storage* 57
3.1.3.4 Nível gratuito da GCP 58
3.1.4 Keras 58
3.1.5 Pandas 58
3.2 MÉTODOS 59
3.2.1 Pré-processamento de dados. 59
3.2.1.1 Tratamento de dados faltantes 60
3.2.1.2 Reamostragem 61
3.2.1.3 Criação de novos de atributos 62
3.2.1.4 Divisão dos dados de treino, validação e teste 62
3.2.1.5 Geração de sequencias 63
3.2.1.6 Normalização dos dados 65
3.2.2 Redes 65
3.2.2.1 Rede A: *standard* LSTM com auto regressão 66
3.2.2.2 Rede B: *standard* LSTM sem auto regressão 67
3.2.2.3 Rede C: *sequence-to-sequence* LSTM 68

3.2.2.4 Função de custo 69
3.2.2.5 Técnicas de regularização: *Dropout* 70
3.2.3 Projeto Pajé 71
4 RESULTADOS 73
4.1 REDE A: *STANDARD* LSTM COM AUTO REGRESSÃO 73
4.1.1 Treinamento 73
4.1.2 Resultados 74
4.2 REDE B: *STANDARD* LSTM SEM AUTO REGRESSÃO 76
4.2.1 Treinamento 76
4.2.2 Resultados 77
4.3 REDE C: *SEQUENCE-TO-SEQUENCE LSTM* 78
4.3.1 Treinamento 78
4.3.1.1 Horizonte de previsão de 60 minutos 78
4.3.1.2 Horizonte de previsão de 60 horas 79
4.3.2 Resultados 81
4.3.2.1 Horizonte de previsão de 60 minutos 81
4.3.2.2 Horizonte de previsão de 60 horas para a série hpc_h 81
4.3.2.3 Horizonte de previsão de 60 horas para as demais séries 82
5 DISCUSSÃO 85
6 CONCLUSÃO 86

1 INTRODUÇÃO

A previsão de carga não é um assunto novo dentro da engenharia elétrica, sendo que um dos primeiros artigos sobre técnicas computacionais para previsão de carga foi publicado no fim da década de 1960. Nos dias de hoje, a previsão de carga é parte integral do planejamento energético, indo além das necessidades operacionais das instalações de energia, como subestações e usinas de geração; operadores do sistema, fornecedores de energia, instituições financeiras e todo o mercado relacionado à geração, transmissão, distribuição e comercialização de energia têm interesse no aumento da acurácia das técnicas de previsão de carga. Sob um ponto de vista mais amplo, a evolução das técnicas de previsão de carga está relacionada com um horizonte tecnológico que envolve sistemas elétricos inteligentes, já que tais técnicas estão intimamente ligadas à capacidade de coordenar diferentes tipos de cargas e fontes de geração de maneira eficiente, segura e automatizada.

Durante os últimos 50 anos, diversos métodos para previsão de carga foram aplicados, havendo uma vasta literatura relacionada ao tema. Durante as primeiras décadas, os métodos lineares foram preferência para a solução do problema, em especial as técnicas baseadas em modelos auto regressivos de médias móveis e suas variações.

A predominância dos modelos lineares se deu sobretudo pela complexidade dos modelos não-lineares, que em geral exigiam um esforço computacional que os tornava inviáveis. No entanto, o aumento expressivo da capacidade computacional e da disponibilidade de dados permitiu uma mudança de paradigma, dando espaço aos modelos não-lineares. Considerando o cenário atual em que redes elétricas inteligentes, geração distribuída, carros elétricos, dentre outros avanços estão cada vez mais nítidos no horizonte tecnológico, há grande motivação para o estudo do tema de previsão de carga e as possíveis aplicações dessas novas tecnologias na solução de problemas característicos de sistemas elétricos.

Diante desse contexto, o objetivo final desse projeto é replicar técnicas propostas em um artigo de referência para o problema de previsão de carga utilizando redes neurais LSTM.

2 REVISÃO DA LITERATURA

Nesse tópico serão introduzidas as principais referências que serviram de base para a o desenvolvimento desse projeto. A revisão da literatura foi realizada a partir do estudo de diversos artigos, sobretudo da base IEEE Xplore. A pesquisa foi conduzida de modo a compreender três principais aspectos: entender o problema de previsão de carga e seu desenvolvimento ao longo da história, situá-lo no momento tecnológico atual e escolher um caso de estudo relevante. Os assuntos abordados serão discutidos ao longo dos seguintes subtópicos:

2.1 **Previsão de carga**: uma perspectiva histórica da evolução da previsão de carga será apresentada, assim como uma discussão sobre a importância da previsão de carga e seu papel nas redes elétricas atualmente.

2.2 **Classificação das técnicas de previsão**: introdução às diferentes técnicas e modelos empregados, assim como os critérios utilizados para sua classificação.

2.3 **Modernização dos sistemas elétricos**: contextualização do problema de previsão de carga no cenário tecnológico atual, ressaltando as novas oportunidades promovidas pelo advento de medidores inteligentes e redes elétricas inteligentes, além das tecnologias de processamento distribuído e aprendizado profundo.

2.4 **Deep Learning**: introdução ao assunto de aprendizado de máquina, com uma breve discussão do tema e dos tipos de algoritmos; em seguida serão discutidos o campo de aprendizado profundo e os métodos baseados em Redes Neurais Artificiais (RNA) que serão utilizados na implementação do modelo de previsão.

2.1 PREVISÃO DE CARGA

O sistema elétrico pode ser dividido em três partes fundamentais: geração, transmissão e distribuição e comercialização. Com exceção das atividades referentes à comercialização, a operação de todo o sistema envolve a solução de problemas do ponto de vista elétrico. No entanto, independente do ponto de vista pelo qual o sistema elétrico é interpretado, há uma dependência elementar de sua operação à demanda, dado que esse é o seu próprio fim. À medida que os sistemas elétricos se tornam maiores e mais complexos, aumenta o desafio para garantir a estabilidade e eficiência do sistema, já que a diversidade e volume das cargas e a probabilidade de eventuais problemas também aumentam, de modo que a importância da capacidade de antecipar o comportamento da demanda torna-se cada vez maior.

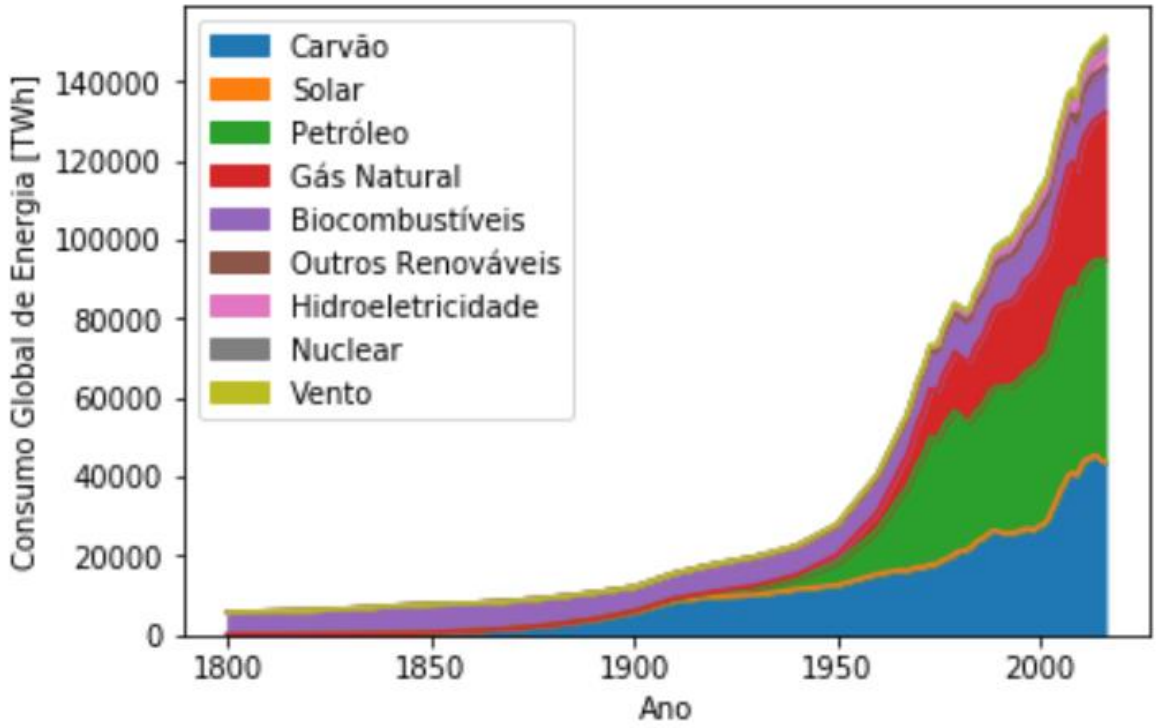
Embora essa realidade seja parte dos sistemas elétricos desde a sua concepção, no início do século XX, a complexidade e a gradual aplicação de inteligência em diferentes níveis do sistema elétrico nos dias de hoje, assim como o avanço acelerado da tecnologia, tornam o problema de previsão de carga muito mais importante e factível.

2.1.1 A previsão de carga ao longo do século XX e no início do século XXI

Desde o surgimento dos primeiros sistemas elétricos, o comportamento da demanda energética é essencial para a operação do sistema. Devido à dependência da demanda a fatores pouco previsíveis, sobretudo ao clima, variações bruscas de demanda ao longo do dia e variações sazonais ao longo do ano sempre fizeram parte da operação de sistemas elétricos de potência. No entanto, no início do século XX a energia elétrica era emprega em sua maior parte para iluminação e aquecimento durante o inverno, de modo que o perfil de carga era relativamente estável; além disso, a complexidade dos sistemas elétricos era menor e boa parte do seu funcionamento era conduzido por operadores humanos.

A grande expansão econômica que se deu após o fim da Segunda Guerra Mundial teve grande impacto sobre o consumo e produção de energia globais. A rápida expansão econômica elevou o consumo mundial de energia expressivamente, além de provocar grandes mudanças nos hábitos de consumo da população, adicionando volatilidade ao comportamento da demanda por energia elétrica.

Figura 2 – Consumo global de energia nos últimos dois séculos



Fonte: ourworldindata.org/energy-production-and-changing-energy-sources, (2018)

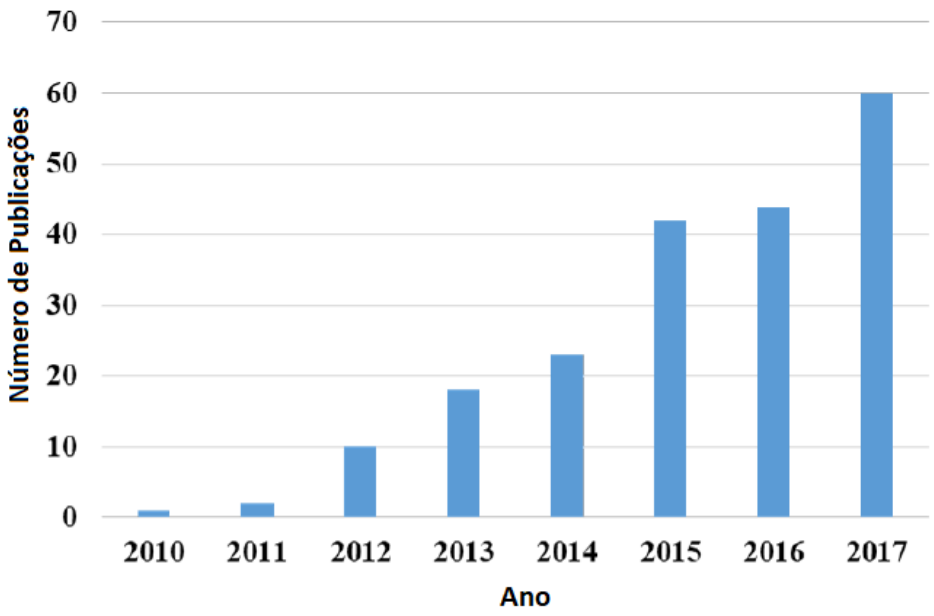
A partir da década de 1950, os hábitos de consumo de energia mudaram muito ao redor do mundo, lançando novos desafios para a operação dos sistemas elétricos. De fato, um dos primeiros artigos sobre aplicação de técnicas computacionais para previsão de carga foi publicado no fim da década de 1960 (HEINEMANN, NORDMIAN & PLANT, 1966), motivado pelo impacto do uso de ar condicionado na carga de pico durante o verão.

A evolução das técnicas de previsão de carga prosseguiu ao longo das décadas da segunda metade do século, num cenário em que as técnicas lineares protagonizavam, devido à limitada capacidade computacional e à pequena disponibilidade de dados, fatores que impossibilitavam aplicações reais usando algoritmos não-lineares.

Embora as técnicas lineares tenham se consagrado na solução do problema de previsão de demanda elétrica ao longo do século XX, as técnicas não-lineares ganharam espaço a partir da década de 2000, em especial por conta do avanço tecnológico e do aumento notável da disponibilidade de dados. Com a gradual integração de inteligência e comunicação nas redes elétricas e dos avanços em *big data* e aprendizado de máquina, a disponibilidade de dados sobre demanda elétrica e

a capacidade para processar essa informação evoluíram muito rapidamente, abrindo novos horizontes. Com efeito, ao longo século XX a previsão de carga foi realizada sob um ponto de vista da alta tensão, já que fazer a medição em baixa tensão era impraticável, além do fato que processar e armazenar grandes volumes de dados não era um obstáculo transponível. Um dos principais artigos de revisão na área de análise de dados de medidores inteligentes, publicado em 2018 por pesquisadores chineses, ilustra o aumento do número de publicações indexados pela *Web of Science* (WoF) envolvendo o assunto a partir do início dos anos 2010, assim como o seu franco crescimento até os dias de hoje (WANG, CHEN, HONG & KANG, 2018).

Figura 3 – Número de publicações relacionadas à análise de dados de medidores inteligentes indexadas pelo Web of Science (WoS)



Fonte: WANG, CHEN, HONG & KANG, 2018

Atualmente, os sistemas elétricos estão passando por uma transição tecnológica em direção à total integração com tecnologias de comunicação e de inteligência computacional, não só pelo grande avanço e interesse da sociedade nessas áreas, como também pelo aumento e mudança da complexidade da demanda por energia, além dos desafios cada vez maiores a respeito do consumo mundial de enrgia, como o aquecimento global e a própria expansão econômica. De um modo geral, esse cenário abre muito espaço para o desenvolvimento da pesquisa na área, que tende a continuar evoluindo ao longo das próximas décadas.

2.1.2 A importância da previsão de carga

Sistemas elétricos de potência são essencialmente dependentes da demanda, de modo que a capacidade de prever o seu comportamento é extremamente importante, não só do ponto de vista operacional, mas também sob uma perspectiva econômica e financeira. Historicamente o setor de energia elétrica foi extremamente regulado, de modo que sua operação era estabelecida por regras bem definidas e sob o monopólio estatal; por muito tempo, a previsão de carga de curto prazo foi utilizada para garantir confiabilidade da oferta de energia e previsões de longo prazo para planejamento e investimento em aumento de capacidade.

No entanto, nas últimas décadas houve uma crescente iniciativa mundial na direção da desregulação do mercado de energia, inciativa a qual se apoia na dificuldade em gerir sistemas elétricos cada vez mais complexos a partir de regras pré-definidas, e em como esse cenário prejudica diretamente o consumidor final, que não tem escolha sob o produto que consome. De fato, a popularização e a maior acessibilidade a fontes renováveis dão ao consumidor final maior autonomia na hora de escolher de quem comprar a energia. Essa mudança de paradigma no mercado de energia tornou a previsão de carga um assunto cada vez mais importante, que passou a ser interessante não só do ponto de vista operacional e econômico, mas também sob um ponto de vista de modernização dos sistemas elétricos e da própria competitividade entre os agentes do mercado de energia.

2.2 CLASSIFICAÇÃO DAS TÉCNICAS DE PREVISÃO DE CARGA

O estudo de previsão de carga é extremamente amplo e se apoia em uma vasta literatura, de modo que as técnicas para previsão de carga são bastante diversas. Nesse tópico serão discutidos os diferentes modelos utilizados, assim como a classificação que lhes é dada segundo os artigos de revisão mais recentes sobre o tema.

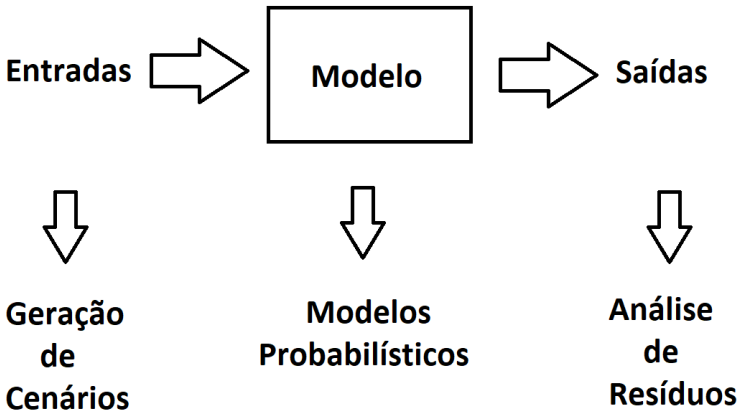
2.2.1 Previsão determinística e probabilística

Grande parte da literatura disponível sobre previsão de demanda se concentra na solução do problema de um ponto de vista determinístico, de modo que o objetivo das técnicas determinísticas é prever um valor específico de demanda para um dado momento ou período futuros. De um modo geral, as técnicas de previsão determinísticas podem ser divididas em três partes básicas: um conjunto de dados de entrada, um modelo matemático e um conjunto de dados de saída (previsões). Em suma, a previsão determinística procura modelar a relação de causa e efeito associada ao comportamento da carga. No entanto, a aplicação de técnicas de previsão de carga está intimamente ligada à tomada de decisão, de modo que há uma crescente demanda para a diminuição da incerteza que separa a previsão da tomada de decisão. Nesse sentido a previsão probabilística é muito útil, pois fornece mais informação a respeito da incerteza associada aos valores previstos.

As técnicas de previsão probabilística podem ser subdivididas em três partes fundamentais: previsão probabilística de cenários a partir dos dados de entrada, modelos de previsão probabilística, que podem variar entre adaptações de técnicas determinísticas e técnicas de estimação de densidade de probabilidade, e pós-processamento através da combinação de diferentes previsões ou através de análise de resíduos, ou seja, dos erros associados a cada previsão (HONG & FAN, 2016).

De um modo geral, as técnicas probabilísticas são derivadas das técnicas determinísticas, e em geral procuram explorar dois pontos de vista distintos: melhorar a capacidade de tomada de decisão ou avaliar a performance das técnicas de previsão determinísticas. A Figura 4 ilustra a relação entre os modelos de previsão determinística e probabilística.

Figura 4 – Previsão determinística e probabilística



Fonte: WANG, CHEN, HONG & KANG, 2018

2.2.2 Níveis de agregação

A previsão de carga em níveis altos de agregação, isto é, do ponto de vista da alta tensão, é uma área bastante madura. Quando as primeiras técnicas computacionais para previsão de carga começaram a ser estudadas, o nível tecnológico não permitia o acesso a dados com alta granularidade, cenário o qual prevaleceu até o início do século XXI. A rápida evolução tecnológica permitiu mudanças substanciais aos sistemas elétricos de potência, que se traduziram em maiores capacidades de comunicação e inteligência. Tais mudanças permitiram um aumento significativo do volume e da granularidade dos dados acessíveis, o que possibilitou outras perspectivas para a previsão de carga que passou a ser possível sob pontos de vista de menor agregação.

Na literatura mais recente, o termo *load forecasting (LF)* se refere à previsão da demanda elétrica esperada para níveis de alta agregação. A previsão realizada a nível local ou de equipamento é referida como *spacial load forecasting (SLF)*. Devido à aplicação de medidores inteligentes na última década, a indústria obteve acesso a um volume de dados extremamente massivo e granular, tanto do ponto de vista espacial como temporal. No entanto, o comportamento da demanda de pontos de vista de menor agregação pode apresentar características muito mais aleatórias, de modo que as técnicas convencionais utilizadas para previsão de carga não se adaptam ao novo cenário. O grande avanço da tecnologia computacional e das técnicas de previsão permitiu grandes avanços nesse aspecto, transformando SLF em uma área

emergente referida como *hierarquical load forecasting (HLF)*. HLF aborda previsão em diferentes níveis de agregação, desde o nível doméstico a níveis de alta agregação em alta tensão, sob vários horizontes diferentes (HONG & FAN, 2016), sendo o campo de maior atividade dado a sua correlação com redes elétricas inteligentes, ou *smart grids (SG)*.

2.2.3 Horizonte de previsão

De acordo com a literatura disponível, a previsão de carga pode ser classificada com base no período o qual se deseja prever (HERNANDEZ *et al*., 2014), comumente referido como horizonte de previsão. Embora atualmente não haja um conjunto de regras consolidado na literatura sobre a classificação de modelos quanto ao horizonte de previsão, é possível dividi-los nas seguintes classes:

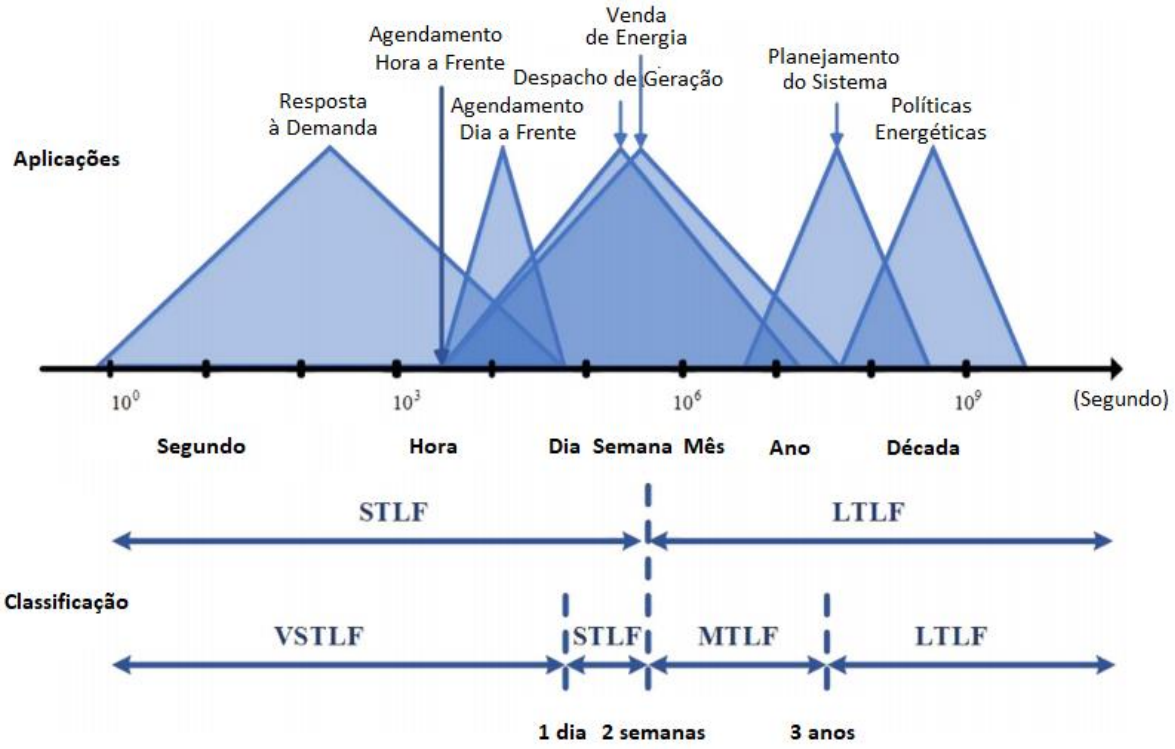
- ***Very Short-Term Load Forecasting (VSTLF)***: a previsão de carga de prazo muito curto tem como horizonte de previsão períodos que vão de segundos a um dia. Modelos de previsão VSTLF em geral são aplicados para controle de fluxo de potência.
- ***Short-Term Load Forecasting (STLF)***: a previsão de carga de curto prazo tem como horizonte de previsão períodos que vão de um dia a duas semanas. Esses modelos são úteis para ajustar geração e demanda, sendo aplicados também no mercado de energia. As aplicações de previsão de carga de curto prazo são as mais abundantes na literatura.
- ***Medium-Term Load Forecasting e Long-Term Load Forecasting (MTLF e LTLF)***: a previsão de carga de médio prazo tem como horizonte de previsão períodos que vão de duas semanas a três anos; os modelos de previsão de longo prazo tem como horizonte de previsão períodos superiores a três anos. Os modelos para previsão de carga para longos períodos são aplicados no planejamento e expansão de infraestrutura.

Não há um consenso estabelecido na literatura, e diferentes referências apresentam perspectivas diferentes. Uma classificação mais superficial dos modelos de previsão de carga poderia dividi-los entre STLF e LTLF, com um horizonte de distinção de duas

semanas. Ocasionalmente, os termos VSTLF e MTLF são utilizados para referir-se a aplicações específicas desses modelos.

Como já discutido anteriormente, modelos de previsão de carga são aplicados no setor de energia elétrica em diversas áreas. Os horizontes de previsão mais importantes são os horários, diários e semanais. Produzir previsões precisas para um curto espaço de tempo é essencial para o planejamento da carga e das unidades de geração, assim como para o mercado de venda de energia. A previsão de carga de longo prazo em geral tem aplicações menos imediatas, associadas ao planejamento diante de horizontes maiores, e são as menos abundantes na literatura.

Figura 5 – Classificações e aplicações de previsão de carga



Fonte: HONG & FAN, 2016

Independentemente do modelo, a principal diferença entre as classes de horizonte de previsão citadas são o tipo e o escopo das variáveis consideradas. Os modelos VSTLF em geral consideram apenas parâmetros elétricos, já que dados climáticos com alta granularidade temporal são mais difíceis de se obter e muitas vezes não estão disponíveis. Modelos STLF utilizam parâmetros elétricos, climáticos e de calendário, embora haja modelos que não utilizem dados climáticos, por motivos semelhantes aos

modelos VSTLF. Os modelos MTLF e LTLF utilizam, além de parâmetros elétricos, climáticos e de calendário, parâmetros econômicos e sociais. De um modo geral, não há regras que definam quais os parâmetros ou mesmo o escopo dos dados utilizados, já que tais fatores dependem de particularidades de cada problema. Além disso, o avanço das técnicas de *deep learning* pode mudar paradigmas, à medida que processar volumes de dados cada vez maiores deixar de ser um obstáculo.

2.2.4 Objetivo de previsão

A classificação dos modelos de previsão de carga também pode ser feita de acordo com o objetivo de previsão. É possível separar os modelos em dois grupos: o primeiro grupo é formado pelos modelos que fazem a previsão de apenas um valor (carga da próxima hora, carga de pico do próximo dia, carga total do próximo dia, etc.); o segundo grupo é formado por modelos que fazem a previsão de múltiplos valores, como por exemplo a carga das próximas horas ou a previsão de carga horária pra os próximos dias – também conhecida como perfil de carga, ou *load profile*.

Os modelos do primeiro tipo são aplicados há mais tempo, sobretudo pela limitação computacional e pela pequena disponibilidade de dados do passado. Os modelos previsão para múltiplos valores se desenvolveram da década de 1990, com a aplicação de conjuntos de modelos. As estratégias disponíveis na literatura são bastante vastas; uma possível estratégia é dividir o dia em 24 horas e utilizar uma RNA para prever cada hora (MCMENAMIN & MONFORTE, 1998). De um modo geral, diversos artigos sugerem que agrupamentos de RNA apresentam bons resultados. A partir de meados da década de 2000, com a popularização das técnicas de *deep learning*, a vastidão de modelos na literatura cresceu ainda mais. Atualmente, as técnicas aplicadas para solução dos problemas de previsão de carga vão desde aplicações de conjuntos de redes neurais híbridas – modelos baseados na associação de conjuntos de redes neurais e outros modelos de aprendizado de máquina, a modelos com arquiteturas mais sofisticadas e profundas, como redes convolucionais e recorrentes.

2.2.5 Modelos lineares e não-lineares

As seções anteriores trataram de possíveis classificações para a previsão de carga sem levar em consideração os modelos utilizados. Essas classificações estão diretamente relacionadas ao tipo de problema a ser resolvido, porém, independentemente do problema, modelos lineares ou não-lineares – ou até mesmo combinações de ambos – podem ser utilizados (HERNANDEZ *et al*., 2014).

Os modelos lineares são baseados na síntese das variáveis do problema em equações para as quais encontrar a solução é um problema viável. O comportamento da demanda de maneira alguma é um problema trivial, envolvendo uma grande variedade de não-linearidades que precisam ser detectadas e posteriormente transferidas para equações que as modelem, envolvendo um grande esforço e conhecimento do problema para a construção do modelo. Os modelos lineares mais aplicados na solução do problema de previsão de carga são modelos dinâmicos, isto é, que consideram variáveis climáticas além das elétricas, baseadas em um algoritmo conhecido como *Auto Regressive Moving Average (ARMA)* ou *Box-Jenkins*. Embora os modelos lineares sejam difíceis de construir, apresentam elevada interpretabilidade, o que nem sempre ocorre com modelos não-lineares.

A partir da década de 80, pesquisadores começaram a notar que modelos não-lineares podiam ser utilizadas para previsão de carga com acurácia satisfatória, alguns inclusive enfatizando as limitações dos modelos lineares em relação aos modelos não-lineares, sobretudo pelas características do comportamento da demanda. A partir da década de 1990, modelos baseados em RNA passaram a ser ostensivamente testados pela comunidade científica, que obteve resultados superiores aos alcançados pelos modelos utilizados previamente. Um tipo de modelo não-linear que obteve sucesso na solução do problema de carga a partir da década de 1990 foram as *Support Vector Machines (SVM)*, no entanto elas foram superadas por modelos baseados em RNA, com o desenvolvimento das técnicas de aprendizado profundo a partir da década de 2000. Em resumo, uma disputa entre modelos lineares e não-lineares se deu a partir da década de 1980, com expoentes de ambos os lados obtendo bons resultados até a década de 2000; a partir daí, com a popularização das técnicas de aprendizado profundo, os modelos lineares perderam o protagonismo.

Figura 6 – Modelos lineares e não-lineares

	Modelos lineares	**Modelos não-lineares**
Horizonte de previsão	VSTLF/STLF/MTLF/LTLF	VSTLF/STLF/MTLF/LTLF
Objetivo de previsão	Valores múltiplos e únicos	Valores múltiplos e únicos
Interpretabilidade	Sim	Às vezes
Facilidade para modelar não-linearidades	Não	Sim
Necessidade de ajuste inicial	Sim	Sim
Custo computacional	Médio/Alto (depende da complexidade do modelo)	Baixo/Médio/Alto (depende do modelo)
Possibilidade de implementação de modelos híbridos	Sim	Sim
Resposta para um número alto de parâmetros históricos	Alta	Alta
Resposta para um número baixo de parâmetros históricos	Média	Alta
Trabalhos e artigos antes de 1990	Muitos	Poucos
Trabalhos e artigos entre 1990 e 2000	Muitos	Aumentando
Trabalhos e artigos após 2000	Poucos	Muitos

Fonte: HERNANDEZ et al., 2014

2.3 MODERNIZAÇÃO DO SISTEMA ELÉTRICO

Como já comentado anteriormente, o setor elétrico passou por diversas mudanças nas últimas décadas, que se deu sobretudo através da incorporação de tecnologias que permitiram maior inteligência e comunicação. A seguir, serão discutidos os principais fatores que impulsionam essa evolução e estão definindo o futuro dos sistemas elétricos (HERNANDEZ et al., 2014).

2.3.1 *Smart grids*

As redes elétricas inteligentes, ou *smart grids*, são definidas como *hardware* e *software* adicionados ao sistema elétrico para alcançar maior autonomia em relação a eventos que o impactem e garantir operação eficiente na distribuição de energia elétrica. As principais características das redes elétricas podem ser listadas como: (1) flexível: entregar a energia necessária aos usuários e responder a eventos aleatórios ao mesmo tempo; (2) acessível: o acesso a todos os usuários do sistema elétrico deve ser garantido, incluindo as unidades de geração de energia renovável e as unidades de geração locais; (3) confiável: qualidade e confiabilidade da energia, garantindo o necessário para economias cada vez mais dependentes da energia elétrica; (4) inovativa: de uma perspectiva econômica, uma regulação que permita competição entre os agentes do mercado impulsiona o avanço das redes elétricas, de modo que inovação é peça chave para gerenciamento eficiente de energia (FERREIRA et al., 2010).

As SGs devem apresentar a inteligência distribuída necessária para operarem eficientemente num contexto globalizado. Portanto, a importância de aspectos financeiros e de risco, assim como o comportamento dos consumidores finais, tornarão os modelos de previsão de carga essenciais para garantir a operação do sistema com máxima eficiência e menor custo.

2.3.2 Geração distribuída

Um sistema com geração distribuída é construído sob a estrutura proporcionada pelas SGs, e envolve novas oportunidades para energias renováveis, já que permite a sua alocação em lugares mais próximos aos pontos de consumo. A ideia consiste em

utilizar tecnologias de geração de baixa potência colocadas mais próximas ao consumidor final, diminuindo a dependência das redes de transmissão e distribuição. O papel da GD será definitivo no planejamento urbano, considerando o impacto de sistemas desse tipo da rede elétrica como um todo. Nesse sentido não só a previsão de carga, mas também a previsão de geração de fontes renováveis, serão indispensáveis para a operação do sistema, dado que a GD adiciona incerteza ao seu funcionamento.

2.3.3 *Virtual power plants*

O conceito de VPP está relacionado ao surgimento de um novo modelo de produção de energia em que a unidade de geração não é uma instalação única, estando distribuída ao longo da rede, sendo, portanto, uma combinação de elementos inteligentes menores que cooperam entre si (WILLE-HAUSSMANN et al., 2010). Exemplos de VPPs poderiam ser um parque industrial ou uma área residencial com módulos solares distribuídos.
No entanto o gerenciamento de VPPs é um desafio, já que os elementos que a compõe devem ser capazes de resolver problemas locais e interagir com o sistema de modo que este funcione em unidade. De um modo geral, VPPs implicam no controle e coordenação de unidades de geração de baixa potência, aplicações para as quais a previsão de carga seria muito útil.

2.3.4 *Microgrids*

As *microgrids* podem ser definidas como uma agregação de cargas e unidades de geração operando como um sistema único para prover energia elétrica e calor. As *microgrids* podem existir em diversos níveis de potência e tensão, servindo para diferentes aplicações, porém independente disso há uma necessidade clara de controle sobre a demanda e a geração, de modo que modelos de previsão de carga são uma prioridade.

2.3.5 *Smart buildings* e *smart environments*

A partir do conceito de *microgrids* novos modelos de redes surgiram. A aplicação de inteligência, além da integração de fontes renováveis, criou novos conceitos como *smart buildings* e *smart environments*. Esses ambientes envolvem a aplicação de sensores interconectados e outros dispositivos inteligentes, melhorando a eficiência do consumo de energia e a vida dos consumidores. A aplicação de modelos de previsão de carga também se faz necessária nesses conceitos.

2.4 *DEEP LEARNING*

Embora o termo *deep learning*, ou aprendizado profundo, esteja relacionado a um campo emergente, suas raízes datam da primeira metade do século passado, com o primeiro modelo artificial de um neurônio inspirado pelo funcionamento do neurônio biológico (MCCULLOCH & PITTS, 1943). Ademais, as técnicas conhecidas como *deep learning* são técnicas particulares de aprendizado de máquina em que redes neurais artificiais profundas – isto é, com grande número de camadas e conexões – são utilizadas. Inicialmente será feita nesse tópico uma apresentação de conceitos básicos de aprendizado de máquina, seguido de uma sessão sobre redes neurais profundas, que serão utilizadas na construção do modelo de previsão e carga. O conteúdo abordado nesse tópico utiliza como fonte o livro texto de *deep learning* utilizado pelo MIT (GOODFELLOW et al., 2016), disponível gratuitamente na internet.

2.4.1 *Aprendizado de máquina*

Como mencionado anteriormente, *deep learning* é um caso particular de aprendizado de máquina, de modo que para se ter um bom entendimento sobre o assunto, é necessário um conhecimento sólido dos conceitos básicos de aprendizado de máquina. Nessa seção serão abordados aspectos de aprendizado de máquina que influenciaram o desenvolvimento do *deep learning*.

2.4.1.1 Algoritmos de aprendizado

Um algoritmo de aprendizado de máquina é um algoritmo que é capaz de aprender através dos dados disponíveis sobre um determinado problema. Uma definição formal disponível na literatura assume que “um computador é capaz de aprender através de uma experiência E com respeito a uma classe de tarefas T e uma medida de performance P se sua performance medida por P nas tarefas em T melhora com a experiência E” (MITCHELL, 1997). Há uma grande variedade de possíveis experiências, tarefas e medidas de desempenho possíveis, sendo que o objetivo não é apresentar uma definição formal para esses elementos, mas dar uma visão intuitiva sobre as possibilidades de aplicação de algoritmos de aprendizado de máquina.

2.4.1.1.1 A Tarefa, T

O aprendizado de máquina nos permite resolver tarefas que são difíceis demais para serem resolvidas por programas escritos e desenvolvidos explicitamente por humanos. De um ponto de vista científico e filosófico, o aprendizado de máquina é muito interessante, pois nos permite desenvolver o entendimento à cerca de mecanismos que de certo modo simulam o aprendizado e a inteligência. Nesse sentido, a palavra “tarefa” está associada ao problema que se quer resolver, e não o aprendizado em si. Muitos tipos diferentes de tarefas podem ser resolvidos com algoritmos de aprendizado de máquina. Os mais comuns são listados a seguir:

- **Classificação**: nesse tipo de tarefa, exige-se do programa que especifique à qual das k classes do problema pertence uma entrada específica. Para resolver esse problema, o algoritmo de aprendizado busca produzir uma função que mapeie as entradas às saídas. Um exemplo desse tipo de problema seria um algoritmo capaz de classificar e-mails *spam*.
- **Regressão**: nesse tipo de tarefa, o computador deve prever um valor numérico dada uma entrada, sendo necessária uma função que relacione as entradas a um conjunto de valores reais. Um exemplo desse tipo de tarefa é a previsão de carga, utilizada na operação de sistemas elétricos.
- **Transcrição**: nesse tipo de tarefa o computador deve ser capaz de observar uma representação de algum dado de maneira não-estruturada e transcrever para uma representação discreta. Um exemplo seria um algoritmo capaz de identificar números ou outros caracteres em imagens e transcrevê-los para texto, como o que o *Google Street View* utiliza para processar os números dos endereços através da imagem da fachada.
- **Tradução**: numa tarefa de tradução, o programa deve ser capaz de receber uma sequência de símbolos em uma determinada linguagem e traduzi-la para outra linguagem. Esse tipo de tarefa é característico de problemas de processamento de linguagem natural, ou *natural language processing* (NLP). O *Google Translate* utiliza algoritmos que realizam esse tipo de tarefa.
- **Detecção de anomalias**: nesse tipo de tarefa, o programa monitora uma série de eventos e objetos e classifica os que apresentarem comportamento atípico

ou anômalo. Um exemplo desse tipo de aplicação é a detecção de fraudes em cartões de crédito.

- **Síntese e amostragem**: nesse tipo de tarefa o programa é requisitado a produzir uma série de novos exemplos que se assemelhem àqueles disponíveis nos dados de treino. Um exemplo desse tipo de aplicação é utilizado em jogos de *video game*, em que programas são utilizados para gerar texturas para o cenário, dispensando a necessidade de um artista produzir a imagem do zero.
- **Estimação de densidade e massa de probabilidade**: nesse tipo de problema, o algoritmo deve produzir uma função que represente uma densidade de probabilidade para uma determinada variável. Um exemplo desse tipo de aplicação é a previsão probabilística de carga.

Evidentemente, muitas outras tarefas são possíveis. As tarefas listadas aqui têm por objetivo apenas dar um entendimento geral do que algoritmos de aprendizado de máquina são capazes de resolver.

2.4.1.1.2 A Performance, P

De modo a avaliar as habilidades de um algoritmo de aprendizado de máquina, é necessário desenvolver uma medida quantitativa da sua performance. Normalmente, a performance P é específica para a tarefa T.

Tarefas de classificação e transcrição em geral utilizam como medida de performance a **acurácia**, que é basicamente a proporção de exemplos para os quais o modelo produz uma resposta correta. Tal medida é equivalente à **taxa de erro**, que é a proporção de exemplos classificados incorretamente pelo modelo. Há também medidas para penalizar diferentes tipos de erro, ou seja, falsos positivos ou falsos negativos. Para tarefas como estimação de densidade de probabilidade ou regressão, por exemplo, não faz sentido em considerar a acurácia ou taxa de erro do problema. Em geral, para problemas de estimação de densidade se usa a **máxima verossimilhança** ou **entropia cruzada,** e para problemas de regressão utiliza-se **erro quadrático médio**.

De um modo geral, há diversas maneiras de produzir uma medida quantitativa de performance para uma determinada tarefa T, no entanto a escolha da medida de performance é particular à tarefa.

2.4.1.1.3 A Experiência, E

Os algoritmos de aprendizado de máquina podem ser classificados de acordo com o tipo de aprendizado, ou experiência, em duas grandes classes: aprendizado supervisionado e não-supervisionado. Essa classificação é referente ao tipo de experiência ao qual o algoritmo é exposto durante o processo de aprendizado.

- **Aprendizado não-supervisionado**: algoritmos de aprendizado não-supervisionado exploram um *dataset* com diversas características com o objetivo de aprender propriedades úteis da estrutura que o compõe. Em tarefas características de algoritmos não-supervisionados em geral queremos estimar uma densidade de probabilidade que gerou os dados, identificar ruídos ou realizar agrupamentos.
- **Aprendizado supervisionado**: algoritmos de aprendizado supervisionado exploram um *dataset* com diversas características, porém a diferença para os algoritmos não-supervisionados reside no fato de que nesse caso cada amostra componente do *dataset* é associado com uma variável objetivo, ou rótulo. O objetivo nesse caso é construir uma função capaz de associar os dados aos seus respectivos rótulos, como por exemplo numa tarefa de classificação ou regressão.

Embora essa classificação seja possível, os termos não-supervisionado e supervisionados não são definições formais, sendo que a distinção entre as duas classes é tênue. Em muitos casos, algoritmos utilizam estratégias de aprendizado que podem ser composições entre os dois tipos de aprendizado.

2.4.1.2 *Overfitting* e *Underfitting*

O principal desafio em aprendizado de máquina é que a performance de um algoritmo seja adequada em dados novos, previamente não observados, e não apenas nos dados em que o modelo foi previamente treinado. A habilidade de performar bem em dados desconhecidos é chamada **generalização**.

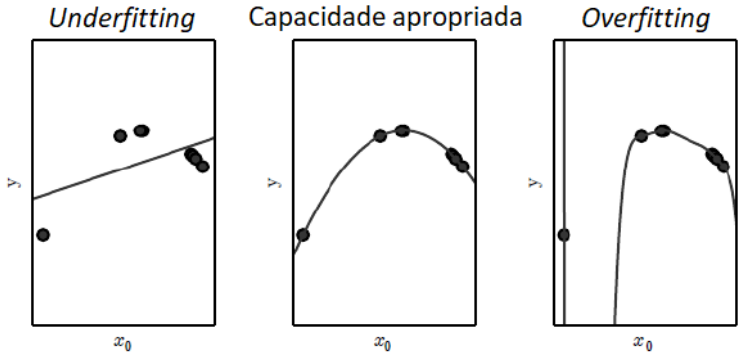
Tipicamente, durante o treinamento de um algoritmo de aprendizado de máquina, temos acesso aos dados de treino, de modo que é possível comparar o valor predito pelo modelo com o valor real, computar um erro relativo e agir de modo a diminuir esse erro. Sob essa perspectiva, o treinamento pode ser interpretado como um

processo de otimização; a diferença entre o processo de aprendizado e um processo de otimização reside no fato de que no processo de treinamento o objetivo é a generalização do problema, ou seja, há interesse na diminuição do erro nos dados de teste, o que só é atingível através dos dados de treino. Portanto, o que define se um algoritmo irá performar bem ou mal é a sua habilidade de:

1. Diminuir o erro nos dados de treino;
2. Diminuir a distância entre o erro de treino e o erro de teste.

Esses fatores correspondem a dois desafios centrais em aprendizado de máquina: *underfitting* e *overfitting*. *Underfitting* ocorre quando a complexidade do modelo não é adequada para um problema, não sendo possível obter um erro de treino suficientemente pequeno. *Overfitting* ocorre quando a complexidade do modelo ou o número de épocas de treinamento são grandes demais para o problema, de modo que o erro de treino atinge valores extremamente baixos, no entanto o erro de teste é alto, não havendo no caso capacidade de generalização.

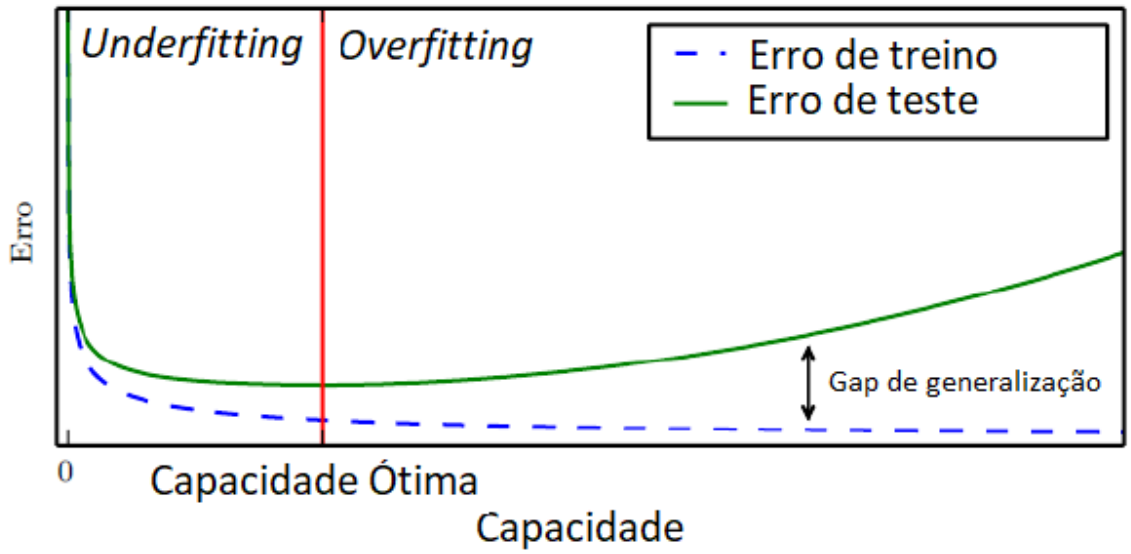
Figura 7 – *Underfitting* e *Overfitting*



Fonte: GOODFELLOW et al., 2016

É possível controlar o quanto um modelo tende ao *overfitting* ou *underfitting* alterando a complexidade do modelo, ou também chamada de capacidade. Modelos com alta capacidade são capazes de memorizar um número maior de propriedades, de modo que tendem ao *overfitting*. Modelos com baixa capacidade, por outro lado, tendem ao *underfitting*. A questão central envolvendo esses dois fatores é a generalização, que consiste na diminuição do erro de teste.

Figura 8 – *Gap de generalização*



Fonte: GOODFELLOW et al., 2016

2.4.1.3 *Construindo um algoritmo de aprendizado de máquina*

De um modo geral, os algoritmos de aprendizado de máquina são formados por uma combinação de componentes básicos: um *dataset* que descreva o problema, uma função de custo que avalie o erro em relação aos dados de treino, um procedimento para otimização e um modelo. A possibilidade de substituição de cada um desses componentes de maneira independente um dos outros permite que uma grande variedade de algoritmos de aprendizado de máquina seja possível. Por outro lado cada problema tem as suas próprias particularidades, de modo que o desenvolvimento de algoritmos de aprendizado de máquina sempre envolve estudo profundo do problema e uma boa quantidade de testes e ajustes.

Os algoritmos baseados em redes neurais também seguem a mesma receita básica. Nas sessões seguintes o assunto será direcionado para redes neurais, e os seus aspectos de construção serão abordados com mais detalhes.

2.4.2 *Redes neurais*

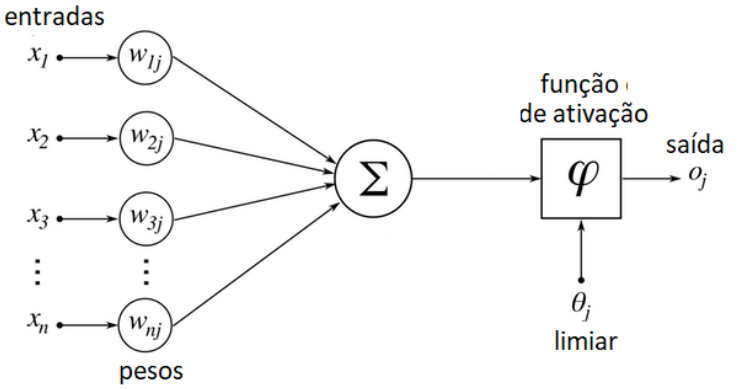
Embora o sucesso dos modelos de aprendizado de máquina baseados em redes neurais seja relativamente recente, as suas raízes datam da primeira metade do século passado, com o modelo de neurônio artificial proposto na década de 1940

(MCCULLOCH & PITTS, 1943). Inicialmente, os primeiros modelos baseados em redes neurais foram desacreditados, pois acreditava-se que as redes neurais não eram capazes de resolver alguns problemas básicos que não eram linearmente separáveis – como por exemplo a função XOR. Além disso, via-se um grande obstáculo quanto ao uso desses modelos por conta da dificuldade em treiná-los, necessitando grande capacidade computacional e abundância de dados. As redes neurais ressurgiram durante a década de 1980 com o desenvolvimento do algoritmo de *backpropagation*, utilizado no treinamento das redes neurais, sendo que nas décadas seguintes os algoritmos baseados em redes neurais ganhariam cada vez mais espaço na comunidade científica. Nas sessões seguintes serão apresentados os princípios básicos dos algoritmos baseados em redes neurais, assim como as redes neurais recorrentes, que serão utilizadas nesse projeto para análise de séries temporais.

2.4.2.1 O neurônio artificial: Perceptron

A base para o desenvolvimento dos algoritmos baseados em redes neurais é o modelo de neurônio artificial proposto em 1943 por McCulloch e Piits. O modelo de neurônio artificial proposto em 1943 consiste basicamente em uma função matemática que aplica uma função de ativação ao somatório das entradas, funcionando como uma porta lógica. Posteriormente, o modelo inicial do neurônio artificial foi revisado e aprimorado, e um novo modelo chamado *Perceptron* foi proposto em 1969 (MINSKY & PAPERT, 1969). A principal diferença entre o modelo proposto por McCulloch e o *perceptron* consiste na introdução de pesos associados à cada uma das entradas e um limiar, também chamado *bias*, adicionado à soma ponderada das entradas. As modificações propostas no *perceptron* permitiram que problemas mais diversos fossem abordados, fazendo as redes neurais mais úteis e generalizáveis.

Figura 9 – *Neurônio Artificial ou Perceptron*



Fonte: ResearchGate, 2018

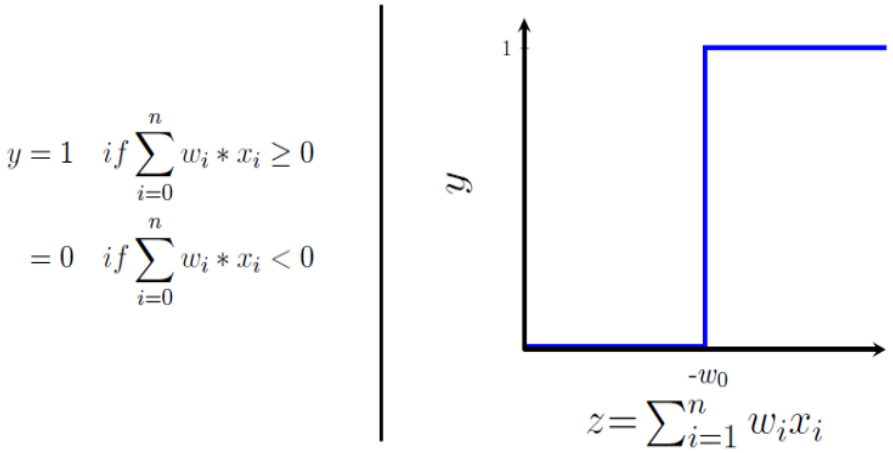
O funcionamento do neurônio artificial é extremamente simples, sendo que o componente mais importante do arranjo é a função de ativação, que dá ao *perceptron* suas principais características. Embora os modelos de redes neurais utilizados atualmente sejam baseados no neurônio *perceptron*, as funções de ativação são distintas. A importância da função de ativação está relacionada principalmente ao processo de aprendizado da rede, o qual é realizado pelo algoritmo de *backpropagation* (ver seção 3.4.2.3). O algoritmo de *backpropagation* depende do cálculo de derivativos que envolvem a função de ativação, de modo que a função de ativação devem ser contínuas e diferenciáveis em seu domínio.

2.4.2.2 Funções de ativação

Como citado anteriormente, os neurônios utilizados nos modelos de redes neurais atualmente diferem ligeiramente do neurônio *perceptron*, em especial por conta da função de ativação. O modelo inicial do *perceptron* utilizava como função de ativação uma função degrau centrada no limiar definido para o neurônio. A utilização da função degrau como função de ativação do neurônio impõe uma série de problemas no que diz respeito à abstração do problema pelo modelo, já que a função não é diferenciável em zero. Tal característica causa problemas, pois torna o modelo menos sensível a detalhes: supondo um exemplo hipotético em que variáveis são consideradas em um *perceptron*, ponderando se um determinado filme é bom ou não considerando um limiar de probabilidade de 0,5. Nesse caso, se dois filmes apresentarem respectivamente valores de 0,49 e 0,51 à função de ativação, um deles será classificado como bom e o outro não, já que a saída é binária, embora os valores

anteriores à função de ativação sejam bastante próximos. Isso evidentemente causa problemas durante o aprendizado, o que motivou o uso de funções de ativação com características mais suaves, diferenciáveis em todo o domínio.

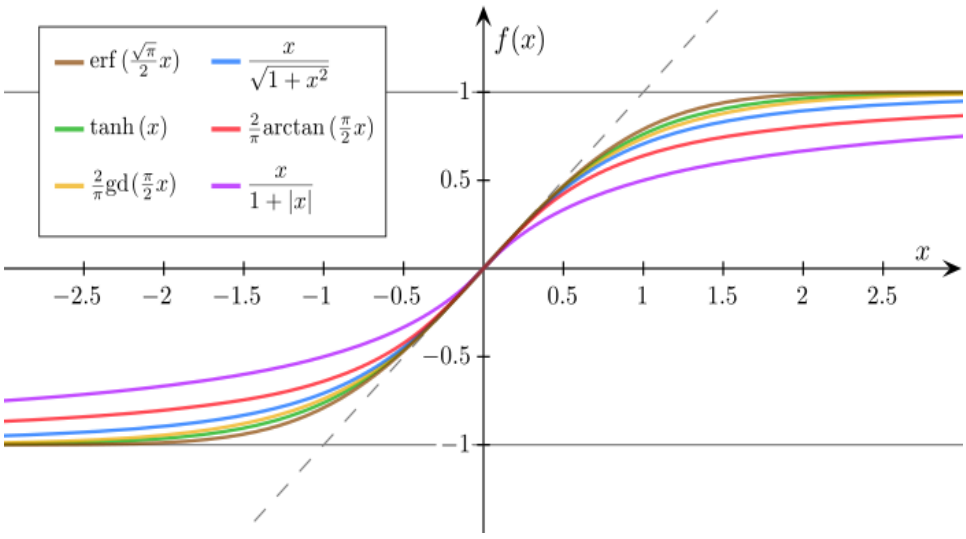
Figura 10 – *Função de ativação do tipo degrau*



Fonte: https://towardsdatascience.com/perceptron-the-artificial-neuron-4d8c70d5cc8d

A introdução de funções de ativação com perfil mais suave permitiu que o *perceptron* produzisse uma saída real, não mais limitada a valores binários, permitindo ao modelo maior capacidade de abstração. Os modelos *perceptron* passaram a utilizar funções sigmoidais como função de ativação, que consistem basicamente de funções que apresentam característica em forma de “S”. Há uma série de funções que podem ser utilizadas com esse fim, tal como ilustrado na Figura 11. Dentre essas funções, podemos citar a função logística e a função tangente hiperbólica.

Figura 11 – *Funções de ativação suaves*

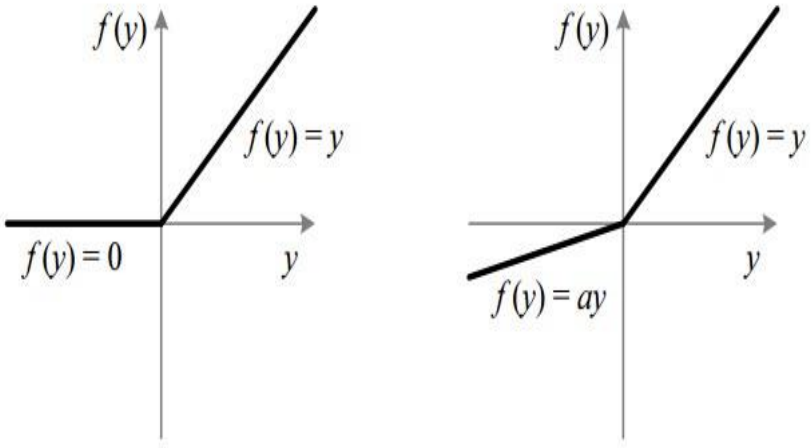


Fonte: Wikipedia, 2018

Cada função de ativação apresenta características próprias, que podem ser mais adequadas a depender do tipo de problema abordado, isto é, classificação, regressão etc. As características mais importantes da função de ativação estão relacionadas à amplitude de sua saída – de 0 a 1, no caso da função logística, e de -1 a 1 para a função *tanh* – e a característica de seus derivativos. O cálculo de derivativos é parte essencial do aprendizado de redes neurais, e, portanto, a característica dos derivativos da função de ativação é muito importante. Nesse caso, funções que não apresentam derivativos monotônicos podem apresentar problemas durante o treinamento.

Atualmente, a função de ativação mais utilizada em redes neurais profundas é função ReLU. A função ReLU é extremamente simples, consistindo em uma reta $y=x$ para $x > 0$ e $y=0$ para $x < 0$. A grande vantagem dessa função de ativação é que ela é contínua e diferenciável em todo o domínio (com excessão do ponto $x=0$, no entanto como computacionalmente os valores de x raramente são exatamente iguais a zero e não há uma mudança brusca no valor de y em torno de x), apresentando derivativo monotônico e constante. Tais características são muito favoráveis ao funcionamento das redes neurais, pois facilitam muito os cálculos necessários durante o treino. Um ponto a ser citado a respeito da função ReLU é que ela ignora valores negativos, o que pode ser um problema; para contornar esse problema, há uma variação da função ReLU, chamada *Leaky ReLU*, em que os valores menores que zero são considerados de acordo com uma segunda reta para $x < 0$.

Figura 11 – *ReLU vs. Leaky ReLU*

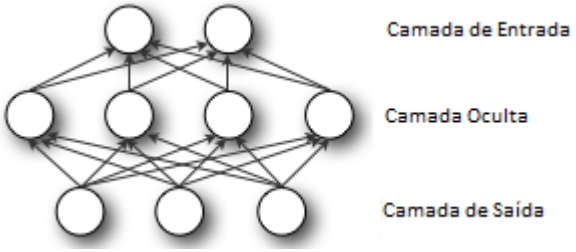


Fonte: Wikipedia, 2018

2.4.2.3 *Multilayer perceptron*

O desenvolvimento do *perceptron* no fim da década de 1960 permitiu que novos modelos baseados em redes neurais fossem criados. Basicamente, a utilidade de uma rede neural seria a de um aproximador universal de funções. No entanto, a capacidade dos *perceptrons* para resolver problemas não linearmente separáveis – como o problema XOR, por exemplo, era um obstáculo para a sua aplicação efetiva. A solução encontrada para o problema foi utilizar redes de *perceptrons*, que em conjunto eram capazes de modelar características não-lineares em uma função. Como o principal bloco de construção das redes eram os *perceptrons*, esse modelo de rede neural recebeu o nome de *multilayer perceptron*, ou MLP. Uma rede neural MLP consiste basicamente na conexão em série de no mínimo três camadas de neurônios: uma camada de entrada, uma camada oculta e uma camada de saída. Uma rede MLP simples com apenas uma capada oculta pode ser representada graficamente como ilustrado na Figura 12.

Figura 12 – *Rede MLP*



Fonte: http://deeplearning.net/tutorial/mlp.html, 2018

Formalmente, uma rede MLP é uma função $f: R^D \rightarrow R^L$, onde D é o tamanho do vetor de entradas e L é o tamanho do vetor de saídas. Considerando um vetor x de entradas e um vetor $f(x)$ de saídas, temos a seguinte equação:

$$f(x) = G_2(b_2 + W_2(G_1(b_1 + W_1 x)))$$

Onde G_i são as funções de ativação, b_i são os vetores de valores de limiar, ou vetor de *bias*, W_i são as matrizes de peso e os índices 1 e 2 são correspondentes respectivamente à camada oculta e à camada de saída. Evidentemente, a rede poderia ter mais camadas ocultas, sendo esse um dos hiperparâmetros da rede MLP. As redes MLP são parte de uma classe de redes neurais chamada *feedforward neural networks*, que se refere especificamente ao sentido de propagação da informação dentro da rede, que segue unicamente da camada de entrada para a camada de

saída, ou seja, refere-se à classe de redes que não formam um ciclo. Redes neurais cíclicas, ou redes neurais recorrentes, fazem parte de uma outra classe de redes neurais, chamada *recurrent neural networks*, ou RNN. Durante o treinamento da rede, o algoritmo de *backpropagation* transfere os derivativos da função de custo para os pesos da rede.

O desenvolvimento das técnicas de *deep learning* descendem essencialmente desse tipo de rede neural, pois conforme a capacidade computacional e abundância de dados cresceu, redes cada vez mais complexas foram necessárias para a generalização da solução. De fato, a premissa sob a qual as técnicas de *deep learning* foram desenvolvidas é a de que utilizar redes profundas, ou seja, com várias camadas, é mais vantajoso que utilizar poucas camadas com muitos neurônios. Outros problemas surgiram com a aplicação de redes neurais profundas, como a dificuldade do algoritmo de *backpropagation* para transferir informações para neurônios no início da rede, problema conhecido como *vanishing gradiente*. No entanto, diversas outras estratégias foram desenvolvidas para contornar esses problemas. Atualmente, as redes neurais profundas são utilizadas em uma grande variedade de aplicações, alcançando resultados surpreendentes em diversos campos, como visão computacional por exemplo.

2.4.2.4 *Treinamento por gradiente e Backpropagation*

Assim como a maioria dos algoritmos de aprendizado de máquina, os algoritmos baseados em redes neurais profundas podem ser interpretados como uma associação de um procedimento de otimização, uma função de custo e um modelo. Da mesma forma, a ideia por trás do treinamento de redes neurais é semelhante ao de outros algoritmos de aprendizado de máquina, que consiste em minimizar o erro em relação aos dados de treino, dado pela função de custo, através de um método de otimização. No entanto, a não-linearidade inerente às redes neurais profundas faz com que a função de custo apresente característica não-convexa, de modo que a otimização não pode ser feita por equações lineares, como no caso de uma regressão linear, sendo necessário um método iterativo, que em geral é o gradiente descendente. A cada iteração, o algoritmo de gradiente descendente faz alterações nos parâmetros otimizáveis no sentido oposto ao gradiente da função de custo; o gradiente é obtido a partir do algoritmo de *backpropagation*, que calcula os derivativos da função de custo

em relação a todos os parâmetros otimizáveis da rede. A seguir, serão apresentados aspectos importantes do treinamento de redes neurais, essenciais para a compreensão do seu funcionamento.

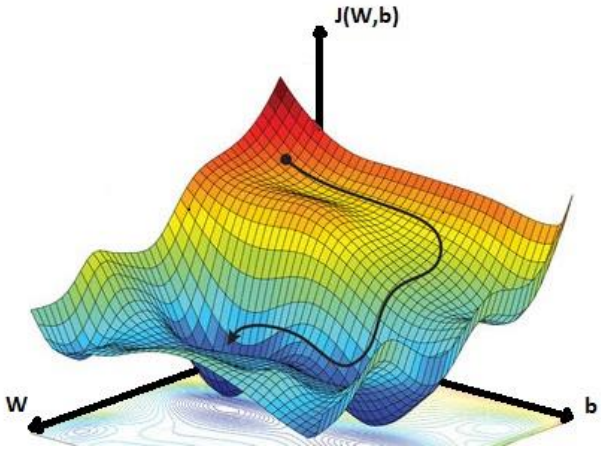
2.4.2.4.1 Função de custo

Um aspecto importante do treinamento de redes neurais é a escolha da função de custo. A função de custo está diretamente relacionada ao processo de aprendizado, já que ela quantifica o erro da rede em relação ao dado de treino, e por esse motivo a função de custo é intimamente ligada ao tipo de problema. Para um problema de regressão, como por exemplo previsão de carga, uma possível função de custo seria o erro quadrático médio, conhecido na literatura como *mean squared error* (MSE).

$$MSE = \frac{1}{n}\sum^{n}(Y_i - \widehat{Y_i})^2$$

Onde Y_i é o valor real atribuído ao dado de treino e $\widehat{Y_i}$ é o valor predito pela rede. Outros problemas, como classificação, utilizam outras funções de custo, como entropia cruzada entre os dados reais e os dados da predição. Como já demonstrado anteriormente, a saída da rede neural é computada através de diversas multiplicações matriciais envolvendo matrizes de pesos W e vetores de limiar b, que são os parâmetros otimizáveis da rede. Dessa forma, a função de custo pode ser calculada em função desses parâmetros, permitindo a minimização da função a partir do método do gradiente descendente. Em geral, a função de custo determina uma superfície não-convexa em função dos parâmetros, e o que o gradiente descendente faz é procurar o ponto mais baixo dessa superfície, como uma bola rolando ao centro de uma bacia. Na literatura de redes neurais, a função de custo muitas vezes representada pelas letras J ou C.

Figura 13 – *Superfície não-convexa definida por uma função de custo*



Fonte: https://www.sciencemag.org, 2018

As funções de custo podem ainda apresentar termos de regularização. Técnicas de regularização são comumente utilizadas para adaptar a função de custo ao problema, como por exemplo a normalização de escalas para facilitar a convergência.

2.4.2.4.2 Gradiente descendente estocástico

O método utilizado para minimização da função de custo é o algoritmo de gradiente descendente (GD). O GD funciona iterativamente, a partir de pequenas alterações nos pesos, proporcionais ao oposto do gradiente da função de custo.

$$w \rightarrow w - \gamma \nabla J(w)$$

Onde w é matriz de pesos, γ é a taxa de aprendizado, ou *learning rate*, e $\nabla J(w)$ é o gradiente da função de custo em relação aos pesos. O grande desafio em se aplicar o GD para otimização de redes neurais é que, a cada iteração, o custo deve ser recalculado para todos os pontos da base de treino, o que inviabiliza a sua aplicação para *datasets* muito grandes, os quais são o foco de aplicação das redes neurais. A alternativa para esse problema é o algoritmo de gradiente descente estocástico, ou *stochastic gradiente descente* (SGD). Nesse algoritmo, a cada iteração o custo não é calculado para todos os pontos do *dataset*, mas para uma amostra ou lote de dados, referido na literatura como *batch*. A ideia é calcular uma estimativa da função de custo, que embora seja uma estimativa ruim, permite que o desempenho do SGD seja muito superior ao do GD, pois permite que a convergência seja muito mais rápida e eficiente, evitando mínimos locais e obtendo resultados muito melhores.

O algoritmo SGD, assim como o GD, permitem a aplicação de técnicas de regularização, que otimizam o funcionamento do algoritmo. Alterações na taxa de aprendizado ou adição de termos extras são algumas das práticas comuns.

2.4.2.4.3 Backpropagation

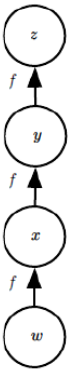
O processo iterativo de aprendizado é baseado na minimização da função de custo, que como apresentado anteriormente, é realizada a partir do algoritmo de gradiente descendente e suas variações. O algoritmo de gradiente descendente depende do cálculo dos derivativos da função de custo em relação aos parâmetros da rede, e essa tarefa é realizada pelo algoritmo de *backpropagation*.

O cálculo da previsão de uma rede neural é realizado por uma composição de funções, de modo que função de custo é função de todos os parâmetros da rede. O cálculo dos derivativos da função de custo em relação a todos os parâmetros da rede pode ser extremamente custoso do ponto de vista computacional, dado que para isso são necessárias diversas multiplicações de matrizes. O aspecto mais importante do algoritmo de *backpropagation* é a sua capacidade de calcular os derivativos usando a regra da cadeia, utilizada para cálculo de derivativos de funções compostas. Supondo por exemplo que $y = g(x)$ e $z = f(g(x))$, a expressão para os derivativos de z em relação a x pode ser expressa como:

$$\frac{dz}{dx} = \frac{dz}{dy}\frac{dy}{dx}$$

Tal propriedade do Cálculo permite que o algoritmo de *backpropagation* calcule os derivativos da saída da rede até a entrada aproveitando os cálculos previamente realizados, o que permite eficiência durante o treinamento.

Figura 14 – *Rede neural na forma de grafo*



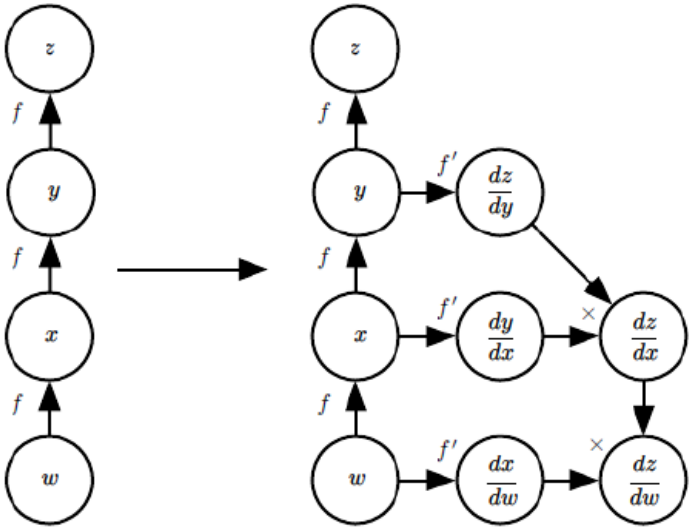
Fonte: GOODFELLOW et al., 2016

A Figura 14 ilustra uma rede neural na forma de grafo, onde $x = f(w)$, $y = f(x)$ e $z = f(y)$. Para calcular $\frac{\partial z}{\partial w}$, podemos aplicar a regra da cadeia:

$$\frac{\partial z}{\partial w} = \frac{\partial z}{\partial y}\frac{\partial y}{\partial x}\frac{\partial x}{\partial w}$$

O cálculo dos derivativos é feito por partes, de modo que os derivativos sejam aproveitados. Assim, uma representação do algoritmo de *backpropagation* para a rede da Figura 14 pode ser ilustrada como:

Figura 15 – *Algoritmo de backpropagation*



Fonte: GOODFELLOW et al., 2016

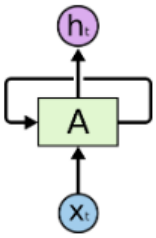
Essa característica do algoritmo de *backpropagation* permite que o treinamento de redes neurais seja muito mais eficiente, sendo esse um dos principais fatores que impulsionaram a popularização das redes neurais.

2.4.3 *Redes neurais recorrentes*

A maneira como humanos pensam é baseada na persistência dos pensamentos, no sentido em que não começamos do início sempre que pensamos em algo, pois somos capazes de relacionar fatos do presente com fatos do passado. Ao assistir a um filme ou ler um livro, nosso entendimento de uma determinada cena depende do conhecimento acumulado ao longo do tempo e não só da cena em si. Ou seja,

problemas dependentes do tempo exigem capacidade de persistência do conhecimento. Redes neurais tradicionais como as redes MLP não possuem essa capacidade, limitando a sua capacidade para resolver problemas como o de previsão de carga. Felizmente, as redes neurais recorrentes (RNN) foram desenvolvidas para solucionar esse problema. Tais redes possuem *loops* que permitem que a informação do passado tenha persistência.

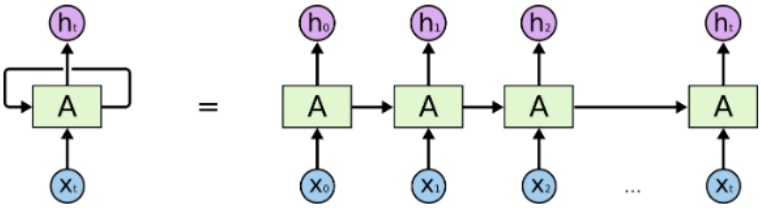
Figura 16 – *Rede neural recorrente*



Fonte: http://colah.github.io/posts/2015-08-Understanding-LSTMs, 2015

No diagrama apresentado acima, uma rede neural A observa uma entrada x_t em um instante t e retorna um valor h_t, e o laço presente na rede permite que a informação do instante t seja utilizada para o instante $t + 1$. Uma representação mais lúdica de uma rede neural recorrente pode ser obtida se abrirmos o laço, tal como no diagrama abaixo.

Figura 17 – Laços em uma rede neural recorrente



Fonte: http://colah.github.io/posts/2015-08-Understanding-LSTMs, 2015

A arquitetura em forma de corrente de uma RNN torna evidente a sua intimidade com problemas relacionados a listas e sequências, como séries temporais ou processamento de linguagem natural.

2.4.3.1 O problema das dependências de longo prazo

A característica mais interessante das RNN é a sua capacidade de relacionar informações do passado na solução de um problema no presente, o que permite uma grande variedade de possíveis aplicações. No entanto, a performance de uma RNN

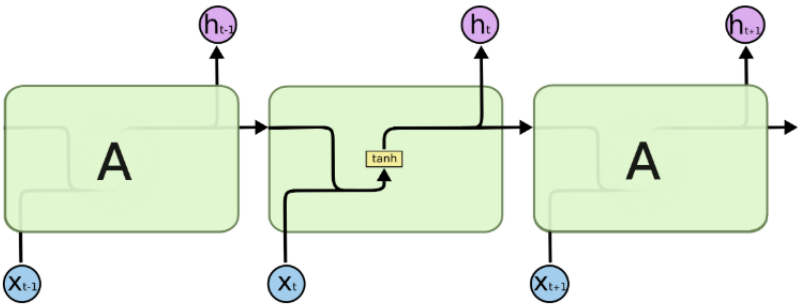
está diretamente ligada a quanto do passado ela precisa considerar para resolver um problema. Considerando o exemplo em que se deseja prever a última palavra de uma frase, como por exemplo “a cor do céu é *azul,*” não há necessidade de entendimento do contexto além da frase, pois a resposta é bastante óbvia. Em problemas desse tipo, ou seja, em que o *gap* entre a informação e o lugar em que ela é necessária é pequeno, redes neurais recorrentes apresentam desempenho satisfatório. No entanto, para problemas em que a informação necessária para o entendimento do contexto vá muito longe no passado, as redes neurais recorrentes apresentam grande dificuldade de aprendizado. Em teoria, RNN’s são capazes de aprender dependências de um passado distante, no entanto o treinamento dessas redes através do gradiente da função de custo se torna extremamente ineficiente, já que os derivativos para pontos no passado ficam cada vez menores, dificultando o aprendizado para informações muito distantes no passado. Além disso, as redes neurais recorrentes não têm a capacidade de decidir se uma informação é relevante ou não, de modo que nem sempre a informação que é importante permanece na rede. Para solucionar esse problema, uma variação das redes neurais recorrentes foi desenvolvida, chamada LSTM, ou *long short term memory*.

2.4.3.2 Redes LSTM

As redes LSTM são um tipo especial de redes neurais recorrentes, capazes de aprender dependências de longo prazo, introduzidas incialmente em 1997 (HOCHREITER & SCHMIDHUBER, 1997). Tais redes obtiveram muito sucesso na solução de problemas com longa dependência no tempo, sendo vastamente utilizadas atualmente.

As redes LSTM foram desenvolvidas para resolver o problema das dependências de longo prazo. Sendo uma variação das redes neurais recorrentes, as LSTM também apresentam a arquitetura em forma de corrente, no entanto a diferença está no bloco componente dessa corrente. Considerando as redes neurais recorrentes tradicionais, cada bloco da arquitetura é composto por uma estrutura muito simples, em geral uma camada de neurônios com uma função tangente hiperbólica como função de ativação.

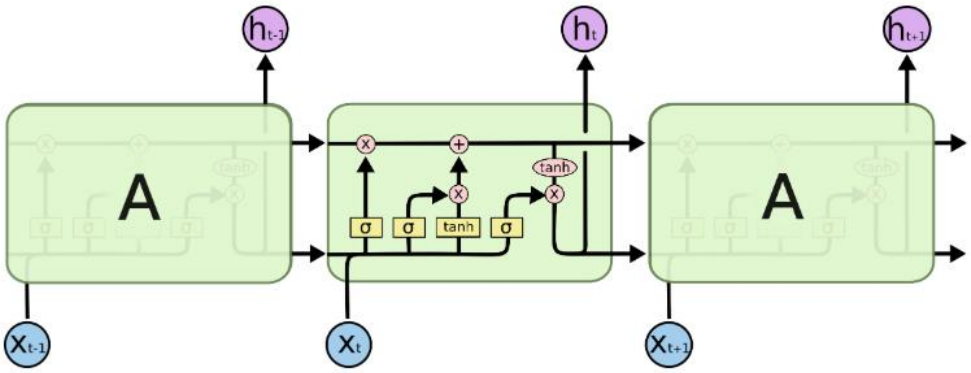
Figura 18 – RNN com camada única



Fonte: http://colah.github.io/posts/2015-08-Understanding-LSTMs, 2015

A estrutura de uma LSTM é semelhante, no entanto o bloco componente da arquitetura de corrente é diferente, contendo uma estrutura mais complexa, com mais camadas.

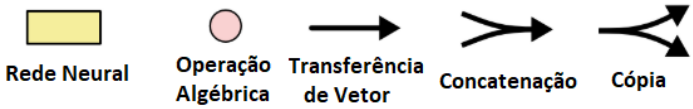
Figura 19 – Estrutura interna de uma LSTM



Fonte: http://colah.github.io/posts/2015-08-Understanding-LSTMs, 2015

A notação utilizada nos diagramas está representada a seguir.

Figura 20 – Notação utilizada nos diagramas

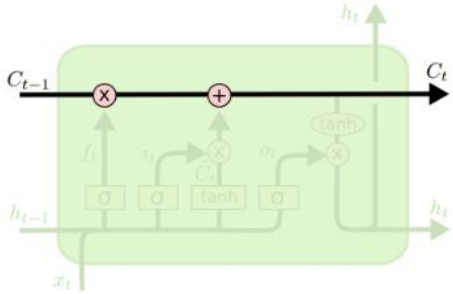


Fonte: http://colah.github.io/posts/2015-08-Understanding-LSTMs, 2015

2.4.3.2.1 A ideia central por trás das redes LSTM

A ideia central do funcionamento de uma rede LSTM reside na sua capacidade de decidir se uma determinada informação deve ou não ser persistida. A chave para que tal capacidade seja possível é o estado da célula, ou *cell state*, que consiste na linha horizontal superior que percorre a rede por todos os seus blocos componentes.

Figura 21 – Estado da célula LSTM, ou *cell state*



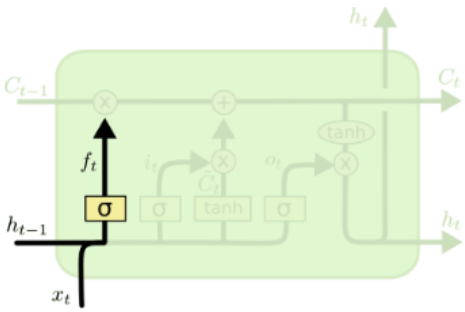
Fonte: http://colah.github.io/posts/2015-08-Understanding-LSTMs, 2015

O grande trunfo das redes LSTM é a sua capacidade de adicionar e remover informação do estado da célula, o que é regulado por estruturas chamadas *gates*, ou portões. *Gates* são compostos de uma rede neural sigmoide e uma operação algébrica. A camada sigmoide é utilizada pois essa função de ativação retorna valores entre 0 e 1, o que permite descrever o quanto de cada elemento deve ser mantido no estado da célula, ou seja, 1 seria equivalente a transferir toda a informação e 0 equivalente a não transferir informação alguma. As redes LSTM possuem três *gates*, utilizados para controlar o estado da célula.

2.4.3.2.2 Passo a passo de uma rede LSTM

O processamento da informação dentro de uma LSTM acontece em etapas. Num primeiro momento, a rede decide o que esquecer ou manter no estado da célula através do primeiro *gate*, chamado *forget gate*. O resultado da rede sigmoide do *forget gate* é então aplicado ao estado da célula, através de uma operação de multiplicação. O *forget gate* utiliza como entrada o vetor x_t, correspondente ao instante atual o qual se deseja prever, e a previsão da rede para o instante anterior, h_{t-1}.

Figura 22 – *Forget gate*

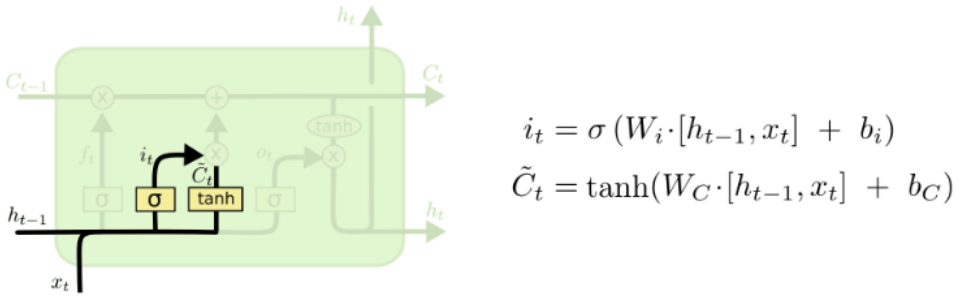


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Fonte: http://colah.github.io/posts/2015-08-Understanding-LSTMs, 2015

O próximo passo da rede LSTM é decidir qual informação deve ser salva no estado da célula. Inicialmente, uma rede neural sigmoide, chamada *input gate*, decide quais valores deverão ser atualizados. Em seguida, uma camada tanh produz um vetor com possíveis valores, o qual é multiplicado pelo resultado obtido no *input gate*, de modo que apenas os elementos escolhidos no *input gate* passem para o estado da célula.

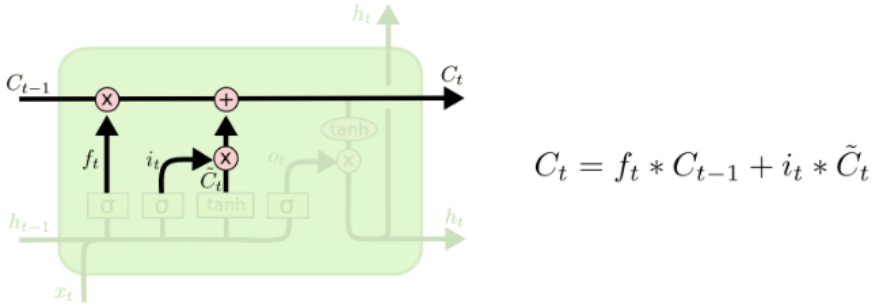
Figura 23 – *Input e Update gates*



Fonte: http://colah.github.io/posts/2015-08-Understanding-LSTMs, 2015

O resultado obtido da multiplicação de i_t e $\tilde{C}_t$ é adicionado ao estado da célula através de uma operação de soma.

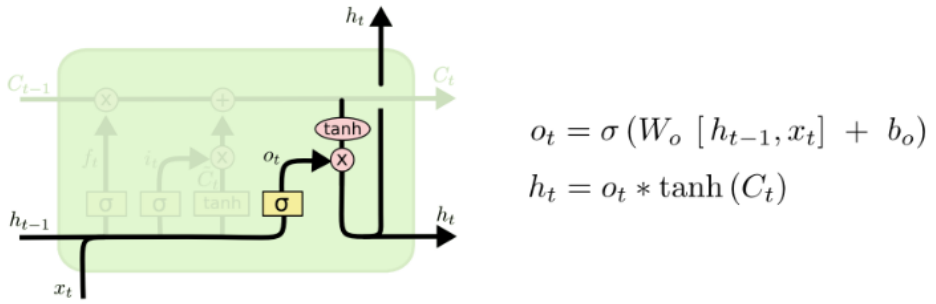
Figura 24 – Atualização do estado da célula



Fonte: http://colah.github.io/posts/2015-08-Understanding-LSTMs, 2015

Finalmente, a rede decide o qual será o seu valor de saída. A saída da rede é baseada no estado da célula, no entanto a informação é filtrada por uma camada tanh. O resultado da camada tanh é então multiplicado pelo resultado de uma outra camada sigmoide, chamada *output gate*, que decide o que deve ser apresentado como saída, selecionando as informações mais relevantes.

Figura 25 – *Output gate*



Fonte: http://colah.github.io/posts/2015-08-Understanding-LSTMs, 2015

A estrutura da LSTM apresentada pode ser considerada o padrão desse tipo de rede recorrente, no entanto há variações dessa arquitetura, em que os *gates* são utilizados de maneira diferente. De um modo geral, as redes LSTM têm obtido sucesso em uma grande variedade de problemas.

3 MATERIAL E MÉTODOS

Apresentada toda a base teórica utilizada no desenvolvimento desse trabalho, discutiremos os materiais e os métodos utilizados para abordar o problema de previsão de carga. A estratégia utilizada foi inicialmente descrever o problema de previsão de carga, ou seja, qual a relevância e necessidade do tema em termos práticos, assim como quais as soluções utilizadas atualmente que são consideras o estado-da-arte nessa área. Dado o necessário entendimento sobre o tema e quais as ferramentas necessárias, o próximo passo foi encontrar um artigo que servisse de base para o trabalho, ou seja, algum artigo que aplicasse uma técnica conhecida a um *dataset* acessível, de modo que a replicação da técnica tivesse uma referência de comparação. O passo seguinte foi compreender as técnicas utilizadas no artigo de referência, de modo a poder aplicá-las corretamente ao problema. Finalmente, o projeto chegou em sua parte prática, que envolveu a aplicação das técnicas a fim de se obter resultados semelhantes aos do artigo de referência.

3.1 MATERIAL

O desenvolvimento desse trabalho foi baseado no em um artigo em que são apresentados resultados obtidos com duas redes LSTM diferentes aplicadas a um *dataset* de domínio público. São estudadas a rede *standard* LSTM e a rede *sequence-to-sequence* (S2S) LSTM, onde a segunda rede apresenta os melhores resultados. Procurou-se reproduzir os resultados apresentados nesse artigo, assim como aplicar as redes para séries diferentes, a fim de avaliar seu desempenho de uma maneira mais ampla. Para a implementação do modelo, adotou-se a linguagem de programação Python. Sendo a linguagem mais utilizada para aplicações de aprendizado de máquina atualmente, há uma grande variedade de bibliotecas disponíveis, o que permite que menos esforço seja necessário para a implementação dos modelos. Para a implementação dos modelos utilizou-se a biblioteca Keras, que consiste em um *framework* para construção de modelos de aprendizado de máquina que usa em plano de fundo outras bibliotecas, como o TensorFlow ou Theano, especializadas nos cálculos necessários na implementação das redes neurais. Entretanto, embora haja vasta literatura disponível sobre aprendizado de máquina, a implementação e treinamento de um modelo exigem uma infraestrutura computacional

robusta, muitas vezes não acessível em um computador pessoal. Para contornar esse problema, optou-se pela utilização de infraestrutura na nuvem. Utilizando a computação em nuvem, é possível ter acesso a tecnologias avançadas de maneira simples, facilitando a implementação e o treinamento do modelo, assim como a sua aplicação em condições reais. Há muitas empresas que possuem serviços e infraestrutura específica para aprendizado de máquina, sendo as maiores a Amazon Web Services (AWS), a Google Cloud Platform (GCS) e a Microsoft Azure. De um modo geral, todas fornecem serviços semelhantes, no entanto, optou-se pela GCP pela grande abundância de material de suporte disponível, o que permite uma curva de aprendizado mais suave. A seguir, serão discutidos maiores detalhes sobre o material utilizado.

3.1.1 *Artigo de referência*

Para o desenvolvimento e avaliação dos modelos utilizou-se como base o artigo "*Building energy load forecasting using Deep Neural Networks*" (MARINO, AMARASINGHE & MANIC, 2016). No artigo são propostas duas redes LSTM diferentes sob a proposta de avaliar a viabilidade desse tipo específico de rede recorrente na previsão de demanda elétrica. Os resultados apresentados no artigo levam à conclusão de que a rede *standard* LSTM não produz bons resultados, se comparada a outros modelos. Em contrapartida, a rede S2S LSTM apresenta desempenho satisfatório na previsão de carga, alcançando bons resultados na previsão de carga em baixo nível de agregação. Tal premissa serviu como base para o desenvolvimento desse projeto. A princípio, implementou-se a rede *standard* LSTM, a fim de confirmar os resultados apresentados pelo artigo, além de servir como uma introdução para a implementação da rede S2S LSTM, que apresenta maior complexidade.

De um modo geral, o artigo foi utilizado como guia, no sentido em que buscou-se replicar seus resultados. Os resultados obtidos nesse projeto foram muito próximos aos apresentados no artigo, considerando o mesmo *dataset*. Finalmente, o modelo foi aplicado a outras séries, a fim de aferir o desempenho do modelo de uma forma mais ampla.

3.1.2 *Datasets*

Ao longo da implementação dos modelos, utilizou-se o *dataset* utilizado pelo artigo de referência como objeto de estudo. No entanto, após a implementação do modelo, outras séries temporais representando a demanda elétrica foram utilizadas, com objetivo de avaliar o desempenho do modelo de um modo mais amplo. Nos tópicos a seguir mais detalhes sobre os *datasets* utilizados serão discutidos.

3.1.2.1 *Individual household electric power consumption*

O *dataset* utilizado no artigo de referência e no desenvolvimento do modelo é chamado *"Individual household electric power comsumption,"* (HEBRAIL & BERARD, 2013). Seus dados compreendem um período de 4 anos (47 meses) e representam o consumo global médio de potência ativa em kW por minuto, além de outras medidas como potência global reativa e tensão, para uma única residência localizada na cidade de Sceaux, na região metropolitana de Paris, e foi gerado como parte de um estudo realizado pela *Électricité de France* (EDF). O *dataset*, atualmente em domínio púbico, foi doado pela EDF à *University of California Irvine* (UCI) em 2013 e atualmente encontra-se disponível para acesso público no repositório de *datasets* de aprendizado de máquina da UCI. Os dados podem ser acessados através do link a seguir: https://archive.ics.uci.edu/ml/datasets/individual+household+electric+power+consumption

Abaixo estão listados os atributos que compõe o dataset:

- *date*: data no formato dd/mm/yyyy
- *time*: hora no formato hh:mm:ss
- *global_active_power*: potência ativa global média por minuto [kW]
- *global_reactive_power*: potência reativa global média por minuto em kilowatts [kW]
- *voltage*: tensão média por minuto [V]
- *global_intensity*: corrente global média por minuto [A]
- *sub_metering_1*: submedida de energia ativa média por minuto correspondente ao cômodo da cozinha [kWh]

- *sub_metering_2*: submedida de energia ativa média por minuto correspondente ao cômodo da lavanderia [kWh]
- *sub_metering_3*: submedida de energia ativa média por minuto correspondente ao ar condicionado e ao aquecedor [kWh]

Embora o *dataset* forneça diferentes atributos além potência ativa média do domicílio, a única medida a ser utilizada no modelo será a correspondente à potência ativa global. Em princípio, as outras séries seriam uteis ao modelo, que é favorecido pela abundância dos dados, no entanto adotou-se a mesma metodologia utilizada no artigo de referência a fim de que os resultados fossem mais facilmente comparáveis.

3.1.2.2 *Smartmeter energy consumption data in London households*

O *dataset "Smartmeter energy consumption data in London households"* (UK Power Networks, 2014) é composto por medidas do consumo de energia elétrica para 5567 residências entre novembro de 2011 e fevereiro de 2014, tomadas a cada meia hora. Os dados foram obtidos como parte de um projeto chamado *Lower Carbon London*, liderado pela *UK Power Networks*, uma operadora de distribuição britânica. O projeto tinha como objetivo avaliar o impacto de políticas de precificação dinâmicas sob o padrão de consumo dos clientes. As residências participantes do projeto foram selecionadas de modo que a amostra representasse a distribuição característica da região metropolitana de Londres, portanto há uma grande variedade de séries temporais que podem ser utilizadas para avaliar o modelo de forma ampla, com representantes de diferentes padrões de consumo.

Durante o experimento, parte dos consumidores foi exposta à tarifa baseada no tempo de uso, numa condição em que eram informados da tarifa um dia antes, através do próprio *smartmeter* localizado na residência ou através de mensagens de texto no celular. A outra parcela dos consumidores foi submetida à tarifa constante. Os atributos disponíveis nesse *dataset* estão listadas a seguir:

- *datetime*: data no formato yyyy-mm-dd hh:mm:ss
- *LCLid*: código único identificador da residência
- *stdorToU*: tipo de tarifa aplicada, dinâmica ou padrão
- *kWh/hh (per half hour)*: energia média consumida em meia hora [kWh]

- *Acorn*: classificação demográfica da residência segundo o padrão CACI Acorn (2010)
- *Acorn_grouped*: classificação demográfica agrupada da residência

Face à grande variedade de séries componentes do *dataset*, apenas algumas das séries foram escolhidas, dando preferência as séries com menos dados faltantes e maior comprimento possíveis. O dataset total possuí 168 milhões de linhas com um tamanho de 11Gb. Os dados podem ser acessados através do link a seguir: https://data.london.gov.uk/dataset/smartmeter-energy-use-data-in-london-households

3.1.2.3 Lista de séries utilizadas

O modelo foi desenvolvido para treinar em uma série temporal representando o consumo de potência elétrica para uma única residência. As séries temporais utilizadas nesse trabalho foram extraídas dos *datasets* apresentados anteriormente e estão listados na tabela a seguir. Para as séries extraídas do *dataset* sobre dados de consumo de potência de Londres, a coluna ***Série*** representa o campo *LCLid*, que contém o código único identificador da residência.

Tabela 1 – Lista de séries temporais

Origem	Fonte	Dataset	Número de séries	Série	Granularidade	Instantes
Électricité de France (EDF)	University of California Irvnie (UCI) Machine Learning Repository	Individual household electric power consumption	1	hpc_m	minuto	2.075.259
				hpc_h	hora	34.951
UK Power Networks	London Datastore	Smartmeter energy consumption data in London households	5.567	MAC000017	hora	19.586
				MAC000024	hora	19.526
				MAC000247	hora	19.623
				MAC000315	hora	17.367
				MAC000520	hora	16.861
				MAC000836	hora	16.144
				MAC001267	hora	15.710
				MAC001563	hora	15.515
				MAC002749	hora	15.809
				MAC004583	hora	19.020
				MAC004748	hora	16.717
				MAC004914	hora	18.496
				MAC004940	hora	18.447
				MAC005269	hora	17.968

3.1.3 *Computação na nuvem*

O grande avanço tecnológico das últimas décadas permitiu que os algoritmos de aprendizado profundo atingissem grande sucesso, não só pela grande abundância de dados disponível, mas principalmente pelo aumento da capacidade computacional. No entanto, o investimento necessário para se adquirir *hardware* de última geração pode se tornar um limitante, além do custo associado a pessoas capacitadas para gerir esse *hardware*.

Felizmente, o advento das tecnologias de computação em nuvem desconstrói tais condições. Grandes empresas como o Google, a Amazon e a Microsoft vendem as tecnologias que desenvolveram internamente para o público na forma de serviços, os

quais apresentam vantagens diante da opção de ter o *hardware* localmente. Os serviços de computação em nuvem tendem a ser totalmente auto gerenciados, o que permite ao desenvolvedor se concentrar no que realmente é importante, dispensando a necessidade de se preocupar com escalabilidade, configuração ou atualizações do sistema, aspectos que em geral atrasam projetos de desenvolvimento de *software*. Atualmente, a indústria da computação em nuvem é uma das que mais cresce mundialmente, movimentando bilhões de dólares por ano.

Diante disso, optou-se nesse projeto por utilizar computação em nuvem durante todo o desenvolvimento. As vantagens estão relacionadas não só à facilidade para o desenvolvimento, mas também com a praticidade com que se pode colocar um protótipo em execução em um problema real. Optou-se por utilizar os serviços fornecidos pelo Google através da *Google Cloud Platform* (GCP), embora a escolha de outra empresa fosse promover vantagens semelhantes. No contexto desse trabalho, a GCP se mostrou uma melhor opção diante da qualidade e abundância do suporte oferecido. A seguir serão apresentados os principais serviços da GCP utilizados durante o desenvolvimento do projeto.

3.1.3.1 *Cloud Datalab*

O *Cloud Datalab* é uma ferramenta interativa criada para facilitar a análise, transformação e visualização de dados e o desenvolvimento de modelos de aprendizado de máquina na GCP. O *Cloud Datalab* é baseado no *Jupyter* (conhecido anteriormente como *IPython*), uma aplicação *web* de código aberto que permite a criação de documentos que são na realidade ecossistemas de desenvolvimento, capazes de conter ao mesmo tempo código executável em diversas linguagens (como por exemplo Python, R, SQL, JavaScript dentre diversas outras), equações, visualizações de dados e texto narrativo, referidos normalmente como *notebooks*. A flexibilidade dos *notebooks* permite que se interaja com diferentes ferramentas em um mesmo documento, facilitando o desenvolvimento. Nesse sentido, o *Cloud Datalab* funciona da mesma forma que o *Jupyter*, no entanto possui ferramentas previamente instaladas que facilitam a interação com os serviços e ferramentas fornecidos pela GCP.

Através do console da GCP é possível instanciar uma máquina virtual contendo o *Cloud Datalab* instalado, na qual é possível se conectar através de um navegador

comum. O processo de configuração é solidamente documentado e extremamente simples, sendo possível concluí-lo sem dificuldades através de tutoriais em português, disponíveis na página oficial da GCP.

3.1.3.2 *Cloud AI Platform*

A *Cloud AI Platform* é a plataforma de inteligência da GCP, consistindo em um ambiente desenvolvimento de modelos de aprendizado de máquina. A plataforma fornece todas as ferramentas necessárias para o desenvolvimento de um modelo, além de rotinas automáticas de treinamento, teste e operação, o que permite que um modelo seja idealizado, desenvolvido e colocado em operação de maneira rápida e eficiente.

Através do *Cloud Datalab*, é possível interagir com a plataforma de inteligência artificial com facilidade, de modo que instanciar jobs de treinamento e colocar modelos em operação é extremamente prático. Isso favorece muito o desenvolvimento de um modelo de aprendizado de máquina, que em geral envolve muita experimentação.

Da mesma forma, a documentação disponível sobre o serviço é bastante rica, contendo vasta quantidade de tutoriais e exemplos disponíveis na página oficial da GCP.

3.1.3.3 *Cloud Storage*

O *Cloud Storage* é o serviço de armazenamento de arquivos do Google. No contexto desse projeto, foi utilizado para hospedar os *datasets* consumidos pelos modelos em treinamento, além de servir como repositório para os modelos treinados. O serviço oferece um explorador de arquivos, através do qual é possível interagir com os arquivos e pastas tal qual em um computador convencional. O *Cloud Storage* possui diversas APIs, através das quais é possível interagir com os arquivos hospedados de diversas formas, como por exemplo no caso de treinamento de modelos de inteligência artificial, que requerem a leitura de grandes quantidades de dado em alta velocidade.

3.1.3.4 Nível gratuito da GCP

A GCP oferece uma cota gratuita de trezentos dólares para consumo de serviços e qualquer conta do Google pode se candidatar, não havendo critérios de aprovação a não ser o que impede que uma mesma conta se candidate mais de uma vez ao nível gratuito. A cota oferecida gratuitamente é suficiente para treinar aprender o básico da utilização da GCP e até desenvolver pequenos projetos, o que favorece a democratização da tecnologia, facilitando o seu acesso através do serviço de nuvem.

3.1.4 *Keras*

O Keras é uma biblioteca *open source* para desenvolvimento de redes neurais escrita em Python. Desenvolvida para permitir a experimentação rápida de modelos baseados em redes neurais, com foco em fácil usabilidade, funcionamento modular e capacidade de extensão, o Keras é capaz de funcionar a partir de bibliotecas como TensorFlow e Theano. As bibliotecas como TensorFlow e Theano foram desenvolvidas para permitir flexibilidade na construção de redes neurais, funcionando basicamente como bibliotecas matemáticas que permitem a implementação de fluxos de dados complexos, compostos por grandes volumes de informação e álgebra matricial. No entanto, tais bibliotecas são complexas e a implementação de redes neurais a partir delas não é imediata, embora permita maior flexibilidade.
A biblioteca Keras é, portanto, ideal para experimentação de redes neurais, permitindo agilidade na depuração e teste dos modelos. Além disso, dado a grande aderência da comunidade ao Keras, há uma grande abundância de suporte e exemplos disponíveis na internet, o que diminui a curva de aprendizado necessária para a aplicação prática das redes neurais.

3.1.5 *Pandas*

O *Pandas* é uma biblioteca escrita na linguagem Python que fornece ferramentas para o processamento de dados. Através do *Pandas* é possível carregar dados de diversas fontes diferentes e manipulá-las através da sua transformação em um *dataframe*. O *dataframe* permite que os dados sejam representados como um objeto na forma de

grande, a partir do qual é possível realizar diversas funções como reamostragem, agrupamento e diversas outras funções necessárias para o processamento de dados. A biblioteca é vastamente utilizada pela comunidade e há um grande volume de material de suporte disponível na internet, tornando-a extremamente simples de usar.

3.2 MÉTODOS

O desenvolvimento do modelo foi executado procurando manter proximidade com o método utilizado no artigo de referência. Num primeiro momento, a arquitetura LSTM padrão foi implementada, e serviu como passo necessário para a implementação do modelo baseado na arquitetura S2S. Os testes e variações de hiperparâmetros de ambos os modelos foram executados tal qual aos do artigo de referência e serão apresentados em comparação aos obtidos. Além disso, serão apresentados no mesmo molde o resultado obtido com outras séries, diferentes da utilizada no artigo de referência.

De um modo geral, a metodologia aplicada no desenvolvimento de um modelo de aprendizado de máquina pode ser dividida nas seguintes etapas: 1) pré-processamento dos dados, 2) desenvolvimento do modelo e 3) treinamento, validação e teste. Nos tópicos a seguir serão apresentados detalhes de cada uma dessas etapas, além da apresentação do projeto Pajé, que percorre todas essas etapas no formato de um tutorial que aborda o problema da previsão de carga.

3.2.1 *Pré-processamento de dados*

Os *datasets* utilizados no desenvolvimento desse trabalho são provenientes de fontes autênticas e representam os hábitos de consumo de energia elétrica de residências reais. Portanto, os dados estão sujeitos às adversidades que existem na realidade e podem ser corrompidos, o que é fato para qualquer fonte de dados real. Desse modo, a etapa de pré-processamento de dados é mandatória no desenvolvimento de qualquer modelo de aprendizado de máquina, pois os dados são a matéria prima em si, de modo que a sua qualidade reflete no desempenho do modelo.

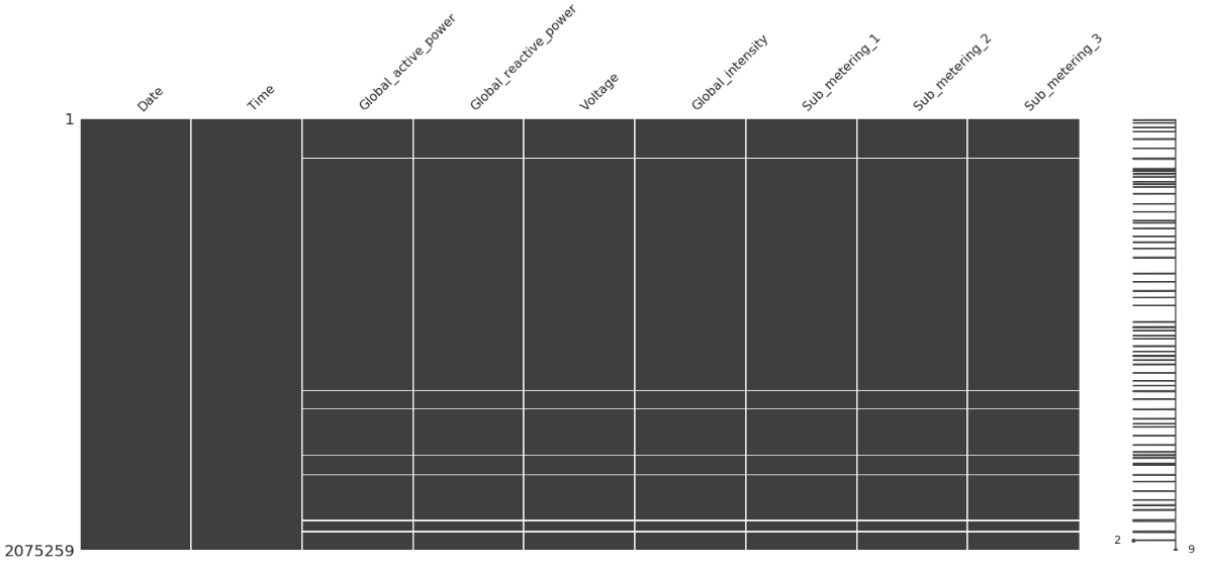
O pré-processamento de dados se refere ao processo de limpeza, reamostragem e transformação dos dados para adequá-lo ao treinamento do modelo, aspectos discutidos nos tópicos seguintes.

3.2.1.1 Tratamento de dados faltantes

Os dados de consumo de potência elétrica em geral são obtidos através de um *smartmeter*, sendo dessa forma dependentes da integridade desse dispositivo. No caso falha do dispositivo de medição, o dado referente ao instante da falha fica desconhecido para sempre, dado o caráter temporal da medição. Em geral esses instantes ficam associados a valores nulos, que não possuem representação matemática, de modo que se faz necessário reconstituir os dados para esses casos, referidos como dados faltantes.

Existem diversas formas para se lidar com dados faltantes, no entanto a forma ideal depende dos dados em si e no conhecimento que se tem sobre o problema ao qual se referem. Uma maneira possível de se reconstituir os dados seria recuperá-los através da interpolação, no entanto, dado às características na qual são feitas as medições de consumo de energia, as falhas nunca ocorrem para instantes isolados, mas para um período de instantes, como aconteceria no caso de ausência da fonte de energia do dispositivo de medição. Nesse caso, a interpolação não é uma solução prática, pois quando há dados faltantes, diversos instantes consecutivos apresentam o mesmo problema. A estratégia adotada para reconstituição dos dados reside no conhecimento adquirido sobre o problema de previsão de carga: sabe-se que o perfil de consumo de energia é estritamente ligado ao calendário, dependendo fortemente da época do ano, do dia da semana e da hora do dia, portanto faz sentido reconstituir o dado com o mesmo valor de um instante que se encontra exatamente a uma semana no passado do instante faltante. A função construída para reconstituição dos dados segue essa lógica de substituição, invertendo a direção de substituição caso ela não seja possível no passado (caso de dados faltantes no início da série), ou seja, substituindo com o valor de um instante a exatamente uma semana no futuro do instante faltante. A Figura 26 apresenta um gráfico com os dados faltantes em todas as séries contidas no *dataset "Individual household electric power comsumption"*, sendo representados na imagem pelas linhas brancas.

Figura 26 – Dados faltantes por coluna



Fonte: Própria, obtida com a biblioteca *missingno* do Python

3.2.1.2 Reamostragem

As séries temporais podem ser provenientes de diferentes *datasets*, os quais podem ter sido amostrados em frequências diferentes daquela adequada ao horizonte de previsão e, portanto, ao modelo. Nesse caso é necessário reamostrar os *datasets* na granularidade adequada. O problema de previsão abordado nesse trabalho considera a previsão do consumo de potência elétrica média por instante em dois horizontes: minutos e horas; os *datasets* devem ser reamostrados para se adequarem aos horizontes de previsão. O *dataset* apresentado em 4.1.2.1 apresenta a potência média em kW com um intervalo de 1 minuto, devendo ser reamostrado para a aplicação no horizonte de horas. Por outro lado, o *dataset* apresentado em 4.1.2.2 apresenta energia média em kWh com um intervalo de 30 minutos, de forma que não é possível utilizá-lo para o horizonte de minutos, fazendo-se necessária a reamostragem para o intervalo de hora e métrica em kW por hora para aplicação no horizonte de horas.
A reamostragem dos dados pode ser feita facilmente através da biblioteca *Pandas* citada anteriormente em 4.1.5.

3.2.1.3 Criação de novos de atributos

O bom desempenho de um modelo de aprendizado de máquina depende diretamente dos dados utilizados durante o seu treino, os quais representam a informação sobre a série para a qual se deseja realizar a previsão. Nesse sentido, o modelo desempenhará melhor caso informações importantes sobre o instante para o qual se pretende fazer a previsão lhe forem fornecidas, como por exemplo informações de calendário como dia da semana ou a hora do dia. Portanto, faz se necessária a criação de novos atributos que contenham tais informações adicionais. Essa prática é conhecida como *feature engineering*, e consiste na criação de novos atributos a partir da série temporal em si.

Nesse trabalho adotou-se o mesmo método apresentado no artigo utilizado como referência, onde os atributos de calendário são transformados em outras séries, defasadas em um instante para o futuro, de modo que a série final fornecida ao modelo contém os seguintes atributos:

- *day_of_year_t+1*: dia do ano para o instante seguinte
- *week_of_year_t+1*: semana do ano para o instante seguinte
- *day_t+1*: dia do mês para o instante seguinte
- *dayofweek_t+1*: dia da semana para o instante seguinte
- *hour_t+1*: hora do dia (0 a 23) para o instante seguinte
- *global_active_power*: potência média em kW por instante de tempo para o instante atual

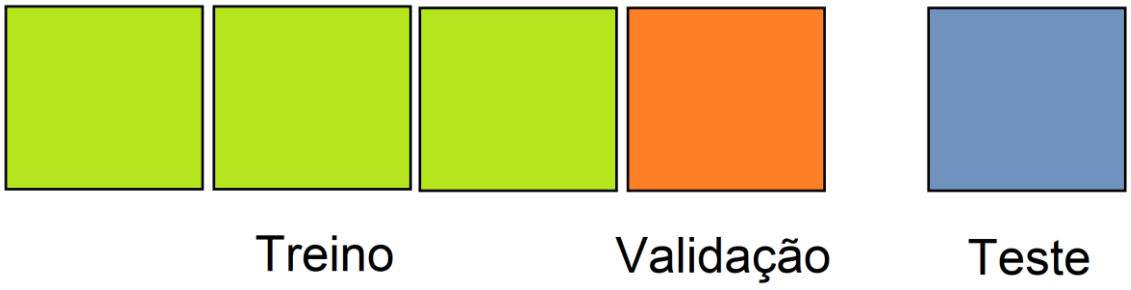
Outros atributos poderiam ser criados, como por exemplo um atributo referente ao dia ser ou não feriado, no entanto manteve-se nesse caso os mesmos atributos utilizados no artigo de referência para facilitar a comparação entre os resultados.

3.2.1.4 Divisão dos dados de treino, validação e teste

O processo de treinamento supervisionado exige que os dados disponíveis sejam divididos em três grupos, com finalidades diferentes: treino, validação e teste. Os dados de treino são aqueles aos quais o modelo é exposto durante as sessões de treinamento, sendo, portanto, a matéria prima para o treinamento do modelo. No entanto, o bom treinamento do modelo não pode ser garantido completamente a partir

dos dados de treinamento, pois é possível que ocorram problemas de treinamento, como *overfitting*. O modelo em treinamento percorre os dados de treino diversas vezes, sendo uma época equivalente ao percurso de todo os dados uma única vez. Para evitar o problema de *overfitting*, utiliza-se os dados de validação: ao final de cada época o modelo é avaliado nos dados de validação, não utilizados para treinamento, a fim de avaliar se o desempenho do modelo está de fato melhorando em dados desconhecidos, ou seja, se o modelo está conseguindo generalizar a solução do problema.

Figura 27 – Divisão do *dataset* em dados de treino, validação e teste



Fonte: Própria

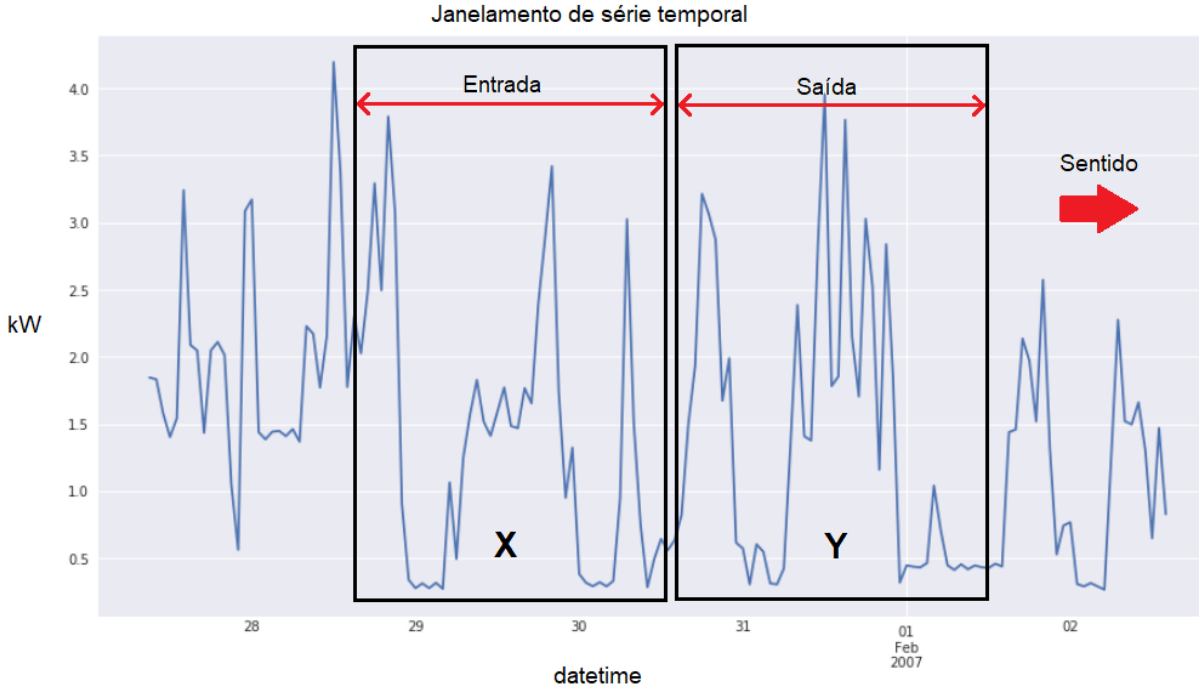
Os *datasets* de treino, validação e teste em geral são criados segundo uma proporção de aproximadamente 80%, 15% e 5%, respectivamente. A divisão é de certa forma arbitrária e não existe uma proporção correta, no entanto a ideia principal é sempre reservar a maior parcela dos dados disponíveis para o treinamento.

3.2.1.5 Geração de sequencias

Os modelos de aprendizado de máquina podem ser divididos quanto ao tipo de treinamento em dois grandes grupos: treinamento supervisionado e não-supervisionado, sendo o primeiro tipo o aplicado para problemas de regressão. O treinamento supervisionado consiste dividir o *dataset* em **atributos** e **rótulos**, ou **X** e **Y**, aos quais o modelo é exposto durante o treinamento: o modelo recebe os atributos e produz uma resposta **Y'**; o erro entre a resposta produzida pelo modelo e o rótulo, também denominada função de custo, é utilizado para calcular os derivativos necessários para o treinamento por *backpropagation*.

Dadas as características do treinamento supervisionado, faz-se necessária a transformação das séries temporais em um *dataset* para treinamento supervisionado, dividido entre atributos e rótulos. Espera-se do modelo uma previsão da carga para os próximos 60 instantes dado o consumo de energia para os últimos 60 instantes, de modo que os **atributos** do *dataset* de treino consistem na série que representa o consumo para os últimos 60 instantes e o **rótulo** consiste na série que representa os 60 instantes subsequentes. Portanto, o processo de transformação da série temporal em um *dataset* de treino supervisionado consiste no janelamento da série temporal, com uma janela definida pela soma do comprimento da sequência de entrada e do comprimento da sequência de saída.

Figura 28 – Janelamento de série temporal



Fonte: Própria

O processo de janelamento é realizado separadamente para cada atributo do *dataset*, e as sequências são salvas em arquivos separados, cada uma contendo um número pares entrada e saída igual ao tamanho do lote definido na criação do *dataset*, sendo este um hiperparâmetro do treinamento que precisa ser definido previamente.

3.2.1.6 Normalização dos dados

Durante o treinamento de um modelo baseado em redes neurais, algoritmos como o SGD são utilizados para calcular os derivativos dos parâmetros a partir dos dados conhecidos sobre o problema. Os algoritmos utilizados para otimização são sensíveis à escala dos dados aos quais são submetidos, pois dada a escala dos dados, os derivativos calculados podem se tornar números extremamente grandes, ou extremamente pequenos, fora do alcance numérico que o computador é capaz de representar, causando problemas na propagação do erro durante o treinamento.

Dessa forma faz-se necessária a normalização dos dados, escalando a série para valores entre 0 e 1. A normalização evita que erros como *exploding gradient,* ou explosão de gradiente, ocorram, impossibilitando o treinamento da rede, especialmente para *datasets* grandes. Os dados são escalados durante o processo de geração de sequências descrito anteriormente através de um modelo de normalização, sendo que cada atributo da série possui seu próprio modelo, o qual é salvo juntamente com os arquivos de sequências, para inversão da normalização necessária durante os procedimentos de teste.

O processo de normalização consiste basicamente em dividir todos os valores pelo valor máximo da série, de modo que todos se encontrem entre 0 e 1. Apenas os dados de treino e validação são utilizados para a construção do modelo de normalização; os dados de teste precisam ficar isolados do modelo em treinamento em todos os sentidos, para que o teste seja válido. Portanto, a normalização não é completamente inversível, dado que os dados de teste não são incluídos no cálculo do modelo de normalização, o que induz um aumento no erro do modelo, no entanto essa é uma concessão necessária para o seu treinamento.

3.2.2 *Modelos*

Nesse trabalho, estudou-se a performance em problemas de previsão de carga de modelos baseados em dois tipos de redes neurais LSTM. A primeira é a *standard* LSTM, que apresenta o funcionamento padrão das redes LSTM. A segunda rede é composta por dois blocos distintos de redes neurais LSTM, chamados *encoder* e *decoder*. Essa arquitetura foi aplicada com sucesso em problemas de tradução de texto, pois permite que o comprimento da sequência de saída seja arbitrário.

De um modo geral, o funcionamento de um modelo de aprendizado de máquina pode ser definido a partir de uma matriz de entradas X, um vetor de predições $\widehat{Y}$, onde Y é o vetor contendo os valores reais. Nos tópicos a seguir, serão apresentados em detalhes os modelos utilizados, elucidando o seu funcionamento e aplicação ao problema de previsão de carga.

3.2.2.1 Rede A: *standard* LSTM com auto regressão

Os modelos de auto regressão foram por muito tempo as assumem que o valor de uma função em um dado instante depende linearmente dos valores dos instantes passados, sendo possível, portanto, calcular o valor para um instante futuro a partir apenas de uma sequência de valores que o anteceda. Uma rede recorrente pode ser aplicada na solução desse problema, sendo os atributos a sequência de valores e o rótulo, o valor do instante subsequente à sequência de entrada. Dessa forma a rede consiste em um modelo que relaciona sequências a um valor escalar. Esse modelo pode ser utilizado para a previsão de uma sequência de valores, a partir da realimentação do modelo com suas próprias previsões, o que consiste na técnica de auto regressão. Dado um conjunto S contendo N sequências com M medidas anteriores ao instante t, representando cada uma um atributo do *dataset* de treino, deseja-se obter um escalar $\hat{y}$ que represente o valor da potência ativa para o instante seguinte t. A rede A pode então ser descrita de acordo com suas entradas como segue:

$$s_n(t) = \{ x_n(t-M), \dots, x_n(t-2), x_n(t-1) \}$$
$$S(t) = \{ s_0(t), s_1(t), \dots, s_N(t) \}$$
$$X(t) = S(t)$$

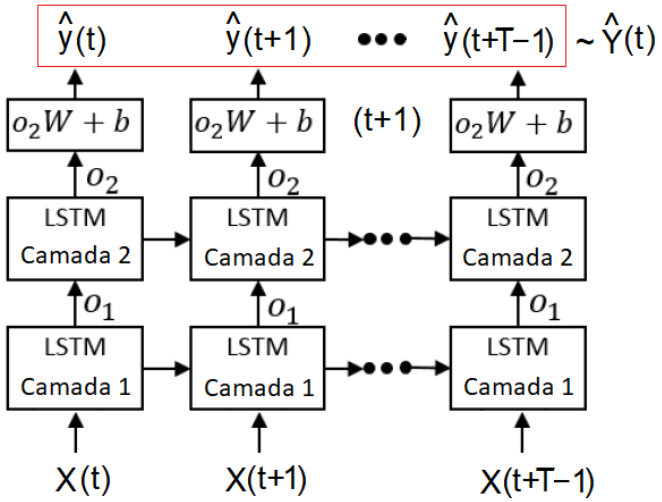
A previsão para uma sequência de T instantes pode então ser calculada iterativamente. Na primeira iteração, o conjunto $S(t)$ é fornecido ao modelo de modo a se obter $\hat{y}(t)$; na segunda iteração, as sequências de entrada são deslocadas um instante no futuro e o valor previsto $\hat{y}(t)$ é adicionado ao fim da sequência correspondente à potência ativa, assim como os valores para os outros atributos para esse instante, de modo que se obtém um conjunto de sequências $S(t+1)$. O processo segue iterativamente até que se obtenha $S(t+T-1)$, que contém a sequência $s_{\hat{y}}(t+T-1)$, correspondente à sequência a ser prevista. Cabe observar

que nesse caso os atributos associados à potência ativa devem necessariamente ser conhecidos para os instantes a serem previstos, o que é verdadeiro para os atributos de calendário. A saída do modelo pode então ser representada como se segue:

$$\hat{Y}(t) = \{\hat{y}(t), \hat{y}(t+1), \ldots, \hat{y}(t+T-1)\} \sim s_{\hat{y}}(t+T-1)$$

A rede resultante do modelo é uma associação de camadas compostas por células LSTM, seguidas de uma última camada simples para adensamento da resposta, tal como ilustrado na Figura 29.

Figura 29 – Rede *standard* LSTM com auto regressão



Fonte: MARINO, AMARASINGHE & MANIC (2016)

3.2.2.2 Rede B: *standard* LSTM sem auto regressão

A grande flexibilidade das redes neurais permite que elas sejam utilizadas de diversas formas na solução de um mesmo problema. Uma outra forma possível de utilizar a arquitetura padrão para previsão de carga consiste em treinar o modelo para prever uma sequência definida e não só um único instante. Nesse caso, dado um conjunto S contendo N sequências de M medidas anteriores ao instante t, deseja-se obter um vetor $\hat{Y}$ que represente a sequência de valores de potência ativa para os T instantes a partir de t. A rede B pode ser descrita de acordo com suas entradas da mesma forma que a rede A:

$$s_n(t) = \{x_n(t-M), \ldots, x_n(t-2), x_n(t-1)\}$$

$$S(t) = \{s_0(t), s_1(t), \ldots, s_{N-1}(t)\}$$

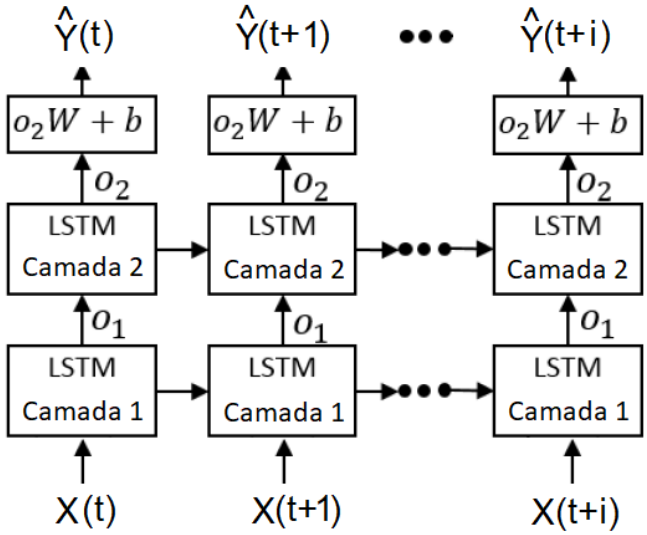
$$X(t) = S(t)$$

A diferença entre as redes A e B reside no treinamento do modelo, que nesse caso é treinado para estimar um vetor e não só um escalar. A sequência de previsão pode então ser obtida sem a necessidade de iteração, podendo ser representada como segue:

$$\hat{Y}(t) = \{\hat{y}(t), \hat{y}(t+1), \ldots, \hat{y}(t+T-1)\}$$

A rede resultante do modelo pode ser representada tal como ilustrado na Figura 30.

Figura 30 – Rede *standard* LSTM sem auto regressão



Fonte: MARINO, AMARASINGHE & MANIC (2016)

3.2.2.3 Rede C: *sequence-to-sequence* LSTM

A rede *sequence-to-sequence* LSTM foi proposta para mapear sequencias de diferentes tamanhos (SUTSKEVER, VINYALS & LE, 2014) e consiste em dois blocos de redes neurais LSTM: *encoder* e *decoder*. O primeiro bloco é responsável por codificar a sequência de entrada através do estado interno da rede LSTM, e o segundo bloco decodifica a saída a partir do estado interno fornecido pelo primeiro bloco e de outros atributos conhecidos para os instantes para os quais se deseja fazer a previsão. Para essa arquitetura, há duas camadas de entrada, sendo a camada de entrada do *encoder* responsável por receber a informação sobre a sequência que antecede o horizonte de predição, e a camada de entrada do *decoder* responsável por receber a informação referente aos T instantes para os quais se deseja fazer a previsão. Dado os conjuntos de sequências de entrada para o *encoder* e o *decoder* S_e e S_d, contendo respectivamente N_e e N_d sequências, M_e instantes de entrada anteriores ao instante

t e M_d instantes de saída, deseja-se obter um vetor $\hat{Y}$ que represente a sequência de valores de potência ativa para os T instantes seguintes, sendo $M_d = T$. A rede C pode ser descrita do ponto de vista de suas entradas como segue:

$$s_{e\,n}(t) = \{ x_{e\,n}(t - M_e), \ldots, x_{e\,n}(t-2), \ldots, x_{e\,n}(t-1) \}$$

$$s_{d\,n}(t) = \{ x_{d\,n}(t), x_{d\,n}(t+1), \ldots, x_{d\,n}(t + M_d) \}$$

$$S_e(t) = \{ s_{e_0}(t), s_{e_1}(t), s_{e_{N_e-1}}(t) \}$$

$$S_d(t) = \{ s_{d_0}(t), s_{d_1}(t), \ldots, s_{d_{N_d-1}}(t) \}$$

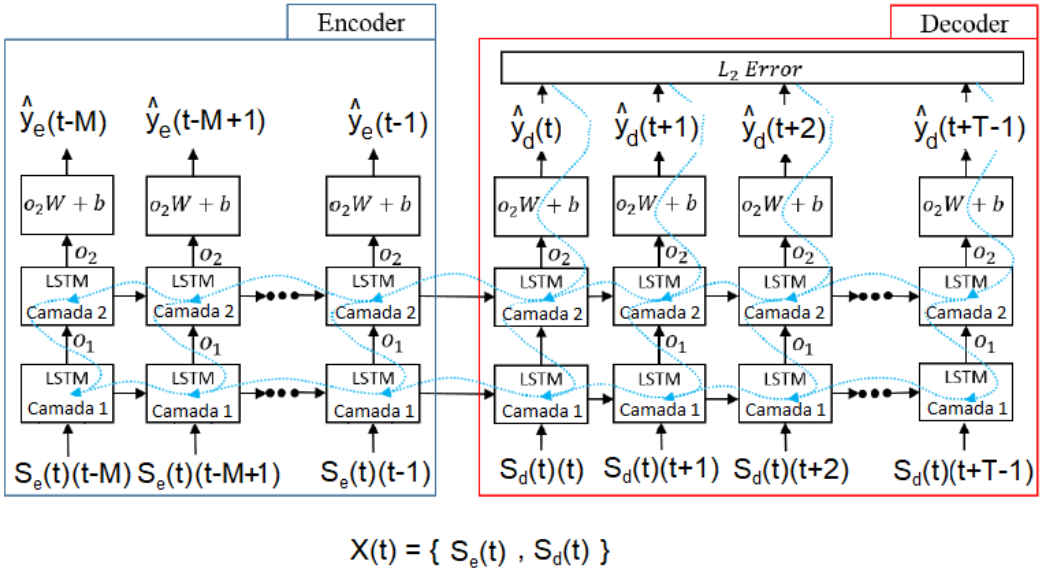
$$X(t) = \{ S_e(t), S_d(t) \}$$

O vetor de saída pode então ser expresso como:

$$\hat{Y}(t) = \{ \hat{y}_d(t), \hat{y}_d(t+1), \ldots, \hat{y}_d(t+T-1) \}$$

A rede resultante pode ser representada tal como ilustrado na Figura 31.

Figura 31 – Arquitetura LSTM S2S



Fonte: MARINO, AMARASINGHE & MANIC (2016)

É importante notar que a sequência correspondente às medidas de potência média não está contida no conjunto S_d, já que é a série a ser prevista, além do fato de que as séries contidas nesse conjunto devem ser conhecidas para os T instantes seguintes ao instante t.

3.2.2.4 Função de custo

O treinamento de um modelo de aprendizado de máquina baseado em redes neurais funciona a partir da propagação dos derivativos da função de custo em relação aos

parâmetros treináveis. Nesse caso, a função de custo deve se adequar ao tipo de problema, pois a função de custo é responsável por quantificar o erro das previsões.
Para um problema de regressão como no caso da previsão de carga, é necessário utilizar uma função capaz de avaliar o erro ao longo do tempo, de modo que uma das funções mais utilizadas para esse fim é o **erro quadrático médio**, ou *mean squared error* (MSE), descrito como segue:

$$MSE = \sum_{t=1}^{M} (y(t) - \hat{y}(t))^2$$

No entanto, devido à maior interpretabilidade dos resultados caso os erros sejam calculados na mesma dimensão que a grandeza para a qual se faz a previsão, optou-se por utilizar a raiz do MSE, também referido como *root mean squared error*, ou RMSE. O RMSE também é a métrica utilizada no artigo de comparação e em diversas outras referências disponíveis na literatura.

$$RMSE = \sqrt{\sum_{t=1}^{M} (y(t) - \hat{y}(t))^2}$$

A função de custo utilizada no treinamento do modelo é o MSE, pois a função RMSE não está disponível na biblioteca *Keras*. No entanto, como minimizar as duas funções nos levaria a um mesmo ponto, não há distinção entre se utilizar MSE ou RMSE como função de custo. Obtém-se daí as raízes dos resultados em MSE, os quais estão representados na mesma dimensão que função a ser prevista, no caso da previsão de carga, em kW.

3.2.2.5 Técnicas de regularização: *Dropout*

Entende-se por técnica de regularização qualquer técnica utilizada para impor maior capacidade de generalização ao modelo. Muitas técnicas diferentes de regularização existem, apresentando características diferentes e, portanto, sua efetividade depende diretamente do problema e do modelo em questão.
A técnica regularização utilizada no treinamento dos modelos apresentados é chamada *Dropout*, e foi introduzida em 2014 como uma solução para o problema de *overfitting* em redes neurais (SRIVASTAVA et al., 2014). O treinamento de redes neurais consiste basicamente na estimativa dos parâmetros treináveis, que podem ser

interpretados com uma representação do conhecimento adquirido durante o treinamento. Do ponto de vista do processo de treinamento humano, supõe-se que é possível se obter resultados melhores a partir de um treinamento mais rígido, que exponha o indivíduo a experiências mais intensas, que se assemelhem à realidade. A técnica de *dropout* segue a mesma lógica, impondo aos parâmetros do modelo uma probabilidade que define o seu comportamento quanto a estar presente ou não durante o cálculo das previsões. Nesse sentido, a técnica força o modelo a "aprender" a calcular a previsão mesmo que lhe falte alguma informação, o que o induz a uma melhor generalização para dados desconhecidos.

O *dropout* pode ser aplicado em qualquer camada da rede, mas no caso das redes neurais LSTM, o *dropout* será aplicado apenas na camada de entrada e na camada de recorrência.

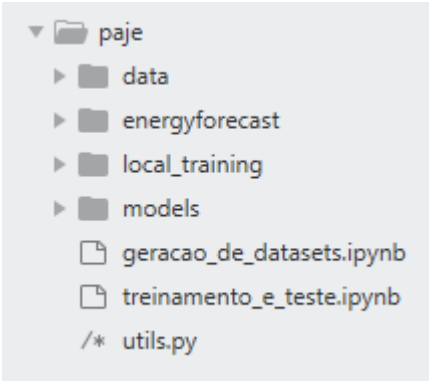
3.2.3 *Projeto Pajé*

A palavra de origem tupi-guarani **pajé,** que nomeia o projeto desenvolvido nesse trabalho**,** designa, nas comunidades indígenas brasileiras, aquele que tem o papel ou poder de se comunicar com entidades não humanas, sendo, portanto, capaz de praticar curas, prever o futuro dentre outras atribuições. Nesse sentido, o nome dado ao projeto se refere à capacidade de previsão dos modelos ao seu potencial na aplicação ao problema de previsão de carga.

O projeto tem por objetivo apresentar as etapas que envolveram o desenvolvimento desse trabalho na forma de tutorial, em que o usuário percorre todas as etapas descritas anteriormente até obter um modelo pronto e no final obtém os resultados do modelo para o *dataset* de teste.

De um modo geral, o projeto consiste em um aplicativo de treinamento, com qual se pode interagir através de documentos do *Cloud Datalab*, ou *notebooks*, apresentados na seção sobre computação em nuvem (4.1.3). O *notebook* **geracao_de_datasets.ipynb** contém os passos necessários para o pré-processamento dos dados e criação dos *datasets* de treino. O *notebook* **treinamento-_e_teste.ipynb** descreve os passos necessários para enviar o modelo para treinamento na nuvem e para obter os resultados no *dataset* de treino. A Figura 32 ilustra a estrutura de pastas do projeto Pajé.

Figura 32 – Estrutura de pastas do projeto Pajé



Fonte: Própria

Os diretórios e arquivos do projeto tem as funções descritas tal como segue:

- *data*: diretório contendo os *datasets* originais e os *datasets* de treino
- *energyforecast*: aplicativo de treinamento, que consiste em um pacote de Python, ou *python package*.
- *local_training*: diretório contendo os resultados das sessões de treinamento realizadas localmente, para corrigir eventuais problemas de código.
- *models*: diretório contendo os modelos treinados, assim como os seus logs de treinamento.
- *geracao_de_datasets.ipynb*: *notebook* contendo as etapas necessárias para pré-processamento de dados e geração dos *datasets* utilizados para treinamento e teste.
- *treinamento_e_teste.ipynb*: *notebook* contendo as etapas necessárias para corrigir, treinar e testar o modelo.
- *utils.py*: modulo contendo funções necessárias durante o tutorial.

Atualmente, o tutorial cobre apenas o modelo baseado na arquitetura S2S, não estando disponível para a arquitetura padrão. O objetivo maior do projeto Pajé é servir com introdução para iniciantes que desejem se aprofundar no problema de previsão de carga e na aplicação de redes neurais em problemas de previsão de séries temporais. O projeto pode ser acessado através do *GitHub*, disponível através do link https://github.com/rodra-go/paje. Qualquer pessoa pode então clonar o repositório e percorrer o tutorial, sendo que o único pré-requisito é usar a mesma infraestrutura de nuvem na GCP.

4 RESULTADOS

Nesse capítulo serão apresentados os resultados obtidos com os modelos A, B e C apresentados anteriormente, assim como a interpretação desses resultados. Os modelos serão apresentados separadamente, dado que as limitações, assim como o desempenho de cada modelo, são distintas. Os resultados para os modelos A e B serão considerados para o horizonte de previsão de 60 e 24 horas, respectivamente, utilizando a série **hpc_h** listada na Tabela 1; testes para o horizonte de 60 minutos, com a série **hpc_m**, e testes com as demais séries foram realizados apenas com o modelo C.

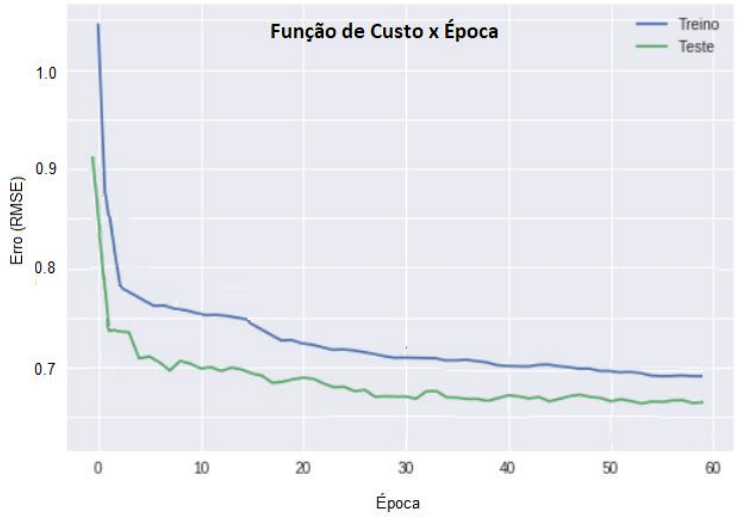
4.1 REDE A: STANDARD LSTM COM AUTO REGRESSÃO

A rede A funciona a partir da auto regressão, sendo que a previsão de uma sequência requer a realimentação com os valores previstos. Esse aspecto impõe algumas limitações ao desempenho do modelo, pois durante o treinamento o cálculo do erro é realizado para o valor previsto para um único instante e não para uma sequência completa, o que é obtido apenas posteriormente ao treinamento. Dessa forma, o modelo consegue obter bons resultados para a previsão de um único valor, mas falha completamente durante a previsão para uma série de instantes. A seguir serão discutidos os resultados de treinamento e teste para esse modelo.

4.1.1 Treinamento

O treinamento da rede A foi realizado para 60 épocas, com lotes de tamanho 100. Entende-se por lote a quantidade de linhas que será considerada a cada iteração durante o treinamento, antes que os parâmetros da rede sejam alterados pelo algoritmo de aprendizado; o número de épocas se refere a quantas vezes o algoritmo percorre todo o *dataset.* O gráfico a seguir ilustra a evolução da função de custo ao longo do treinamento.

Figura 32 – Treinamento da rede *standard* LSTM com auto regressão



Fonte: Própria

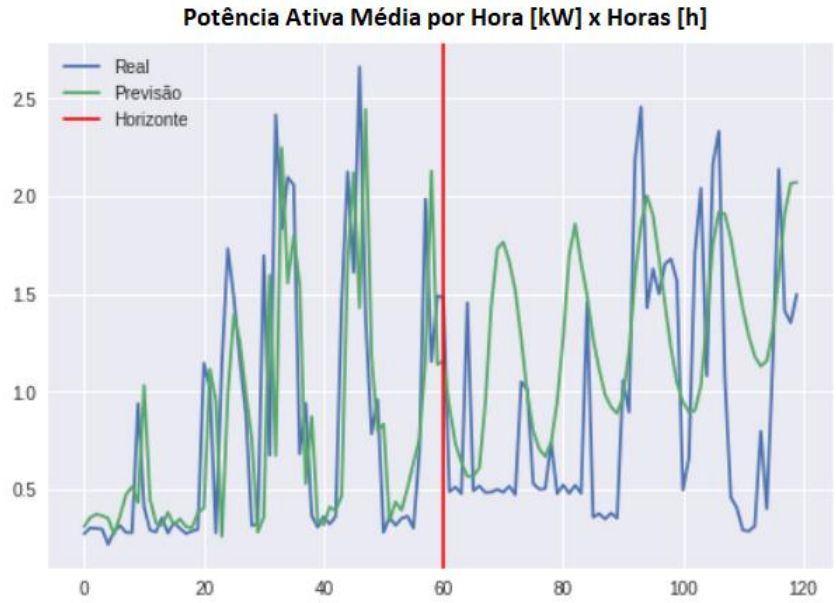
Nos dados e treinamento é possível observar que a curva de validação apresenta valores de perda menores que os apresentados no *dataset* de treino. Isso se dá pelo fato de que durante a validação, as camadas de *dropout* são desligadas. Além disso, nota-se que o erro alcançado durante o treinamento é razoavelmente baixo (em torno de 0.65), indicando um treinamento bem sucedido, o que não acontece com os dados de teste. Isso se explica pela diferença entre o problema ao qual o modelo é exposto durante o treinamento e durante o teste. De um modo geral, pode-se afirmar que o treinamento do modelo A foi realizado com sucesso para um único instante.

4.1.2 *Resultados*

Os resultados obtidos com a rede A foram os mesmos obtidos no artigo de referência, estando dentro do esperado. A rede *standard* LSTM apresenta desempenho razoável para os pontos anteriores ao horizonte, em que apenas dados reais são utilizados como entrada. No entanto, seu desempenho é péssimo para os valores obtidos para as 60 horas subsequentes, onde o modelo é realimenta com os dados valores previstos. A rede A, embora apresente desempenho fraco para a previsão para 60 horas, manteve-se dentro do esperado segundo os resultados apresentados no artigo de referência, que previa o baixo desempenho. O baixo desempenho da rede A pode ser interpretado segundo a perspectiva do treinamento: durante o treinamento, o modelo “aprende” a realizar a previsão para um único instante, objetivo distinto do que

se deseja obter com o modelo. Dessa forma, o modelo é aplicado para um problema para o qual não foi treinado para resolver, o que explica o seu desempenho.

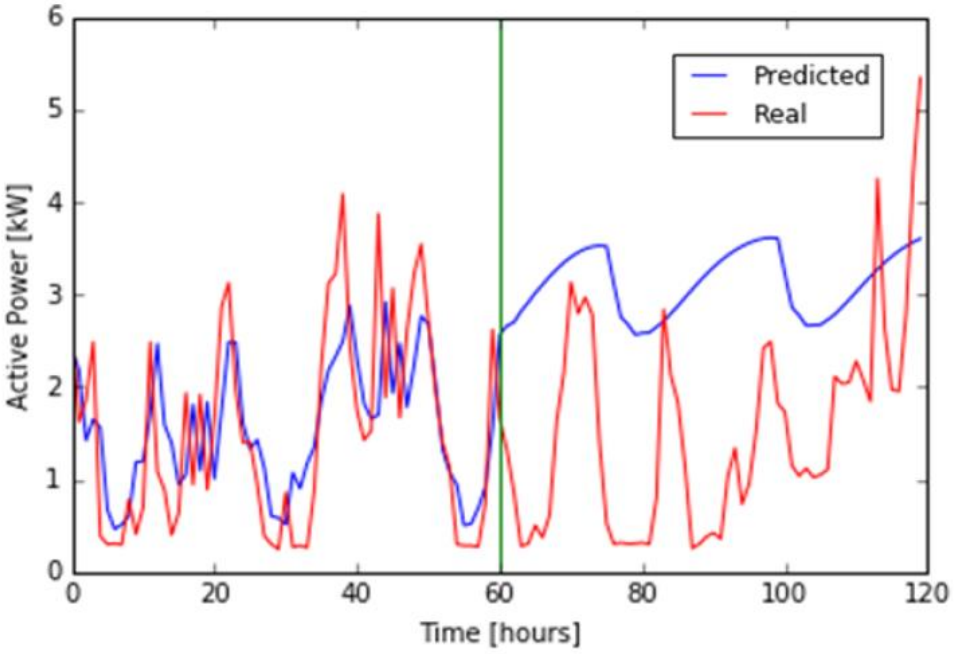
Figura 33 – Resultados da rede *standard* LSTM com auto regressão



Fonte: Própria

Os resultados obtidos são muito semelhantes aos resultados obtidos no artigo de referência, como ilustra a figura a seguir.

Figura 34 – Resultados da rede *standard* LSTM com auto regressão segundo o artigo de referência



Fonte: MARINO, AMARASINGHE & MANIC (2016)

Evidentemente, a rede A não foi capaz de realizar previsões com um nível de acurácia aceitável. O artigo de referência sugere que esse comportamento pode ser devido ao fato de que a rede tende a passar o valor do instante anterior diretamente para a saída, simplesmente o repetindo, aprendendo no caso uma representação fraca da curva de carga (MARINO, AMARASINGHE & MANIC, 2016), o que pode ser claramente observado nos resultados obtidos. A rede A atingiu um resultado final no *dataset* de treino com um RMSE de 2,45.

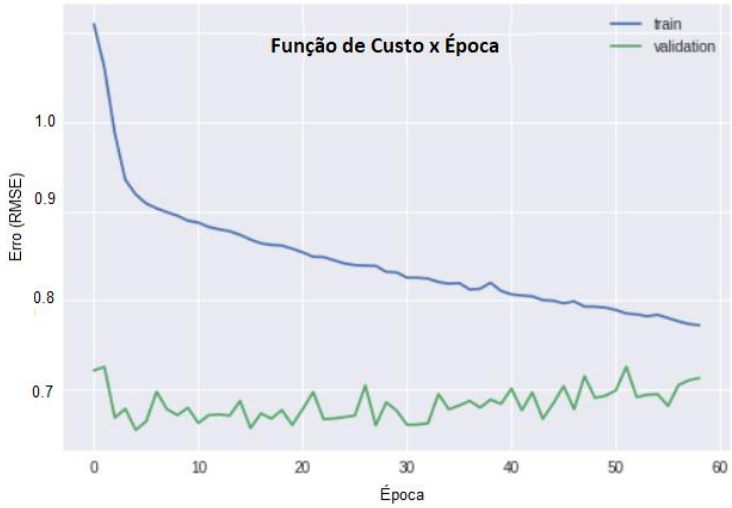
4.2 REDE B: STANDARD LSTM SEM AUTO REGRESSÃO

A rede B apresenta vantagens em relação à rede A, pois durante o treinamento a rede é treinada para realizar a previsão para uma sequência de instantes e não para um único instante, como na rede A. A rede B apresenta resultados melhores para a previsão para 60 horas, no entanto os resultados ainda são muito ruins. Uma limitação da rede B é o fato de que a resposta é calculada com apenas um ciclo da rede neural recorrente, de modo que o estado interno da rede LSTM para um instante determinado é utilizado para o cálculo de toda a sequência subsequente. Dessa forma, o estado da célula não se adapta para cada instante a ser previsto para a sequência de saída, o que afeta o desempenho da rede.

4.2.1 *Treinamento*

Assim como para a rede A, o treinamento da rede B foi realizado com 60 épocas e com um tamanho de lote igual a 100. A Figura 35 a seguir apresenta os dados de treinamento obtidos com a rede B. É possível notar que o treinamento da rede atinge a convergência rapidamente, no entanto há sinais de *overfitting* a partir da trigésima época, antes que haja uma melhora significativa do erro de validação.

Figura 35 – Resultados da rede *standard* LSTM sem auto regressão

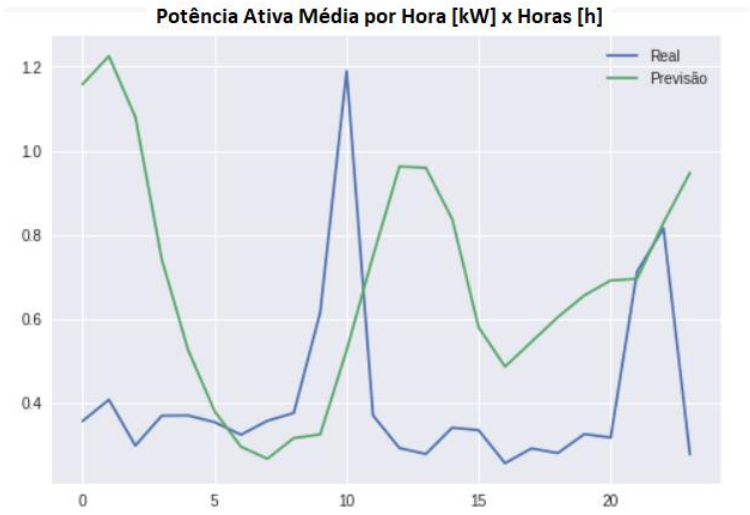


Fonte: Própria

4.2.2 *Resultados*

Os resultados obtidos com a rede B foram melhores que os obtidos com a rede A, mostrando que a rede LSTM é capaz de se adaptar razoavelmente à curva real, mesmo que de maneira ainda muito rudimentar. Os resultados são próximos aos apresentados no artigo de referência, atingindo um erro RMSE final igual a 1.235, o que ainda não representa um resultado aceitável. A Figura 36 a seguir apresenta a previsão obtida pelo modelo em comparação com a sequência real.

Figura 36 – Resultados da arquitetura LSTM padrão sem auto regressão



Fonte: Própria

O resultado da rede B pode ser explicado pela limitação do estado interno, já que, como comentado anteriormente, a sequência final é obtida em apenas um ciclo da rede recorrente. Embora os resultados obtidos com a rede B ainda sejam ruins, demonstram que a rede LSTM é capaz de se adaptar à curva, mesmo que ainda com um erro alto. Cabe ainda comentar que adoção da previsão para 24 instantes, e não 60 instantes como nas redes A e C, se deve à maior dificuldade de treinamento da rede para sequências longas, já que o estado da célula não é capaz de abstrair as sequências passadas de forma adequada.

4.3 REDE C: SEQUENCE-TO-SEQUENCE LSTM

A rede C apresenta grandes vantagens em relação às redes A e B, pois reúne pontos positivos de ambos os modelos. Durante o treinamento, a rede C é treinada a realizar a previsão para uma sequência de instantes, diferente da rede B, por exemplo. Por outro lado, a rede C é capaz de processar as sequências de entrada uma por uma, o que permite que o estado interno do bloco *encoder* represente adequadamente os dados passados. De um modo geral, a rede C mostrou-se capaz de se adaptar à curva real, atingindo os melhores resultados dentre todos as redes testadas. O seu desempenho foi comparável ao alcançado no artigo de referência, e sugere a viabilidade da aplicação da arquitetura S2S ao problema de previsão de carga para uma única residência.

Nos tópicos a seguir serão apresentados os resultados obtidos com a série **hpc_m**, em minutos, seguido dos resultados para a série **hpc_h**, em horas. Finalmente, serão apresentados os resultados para as demais séries descritas na Tabela 1.

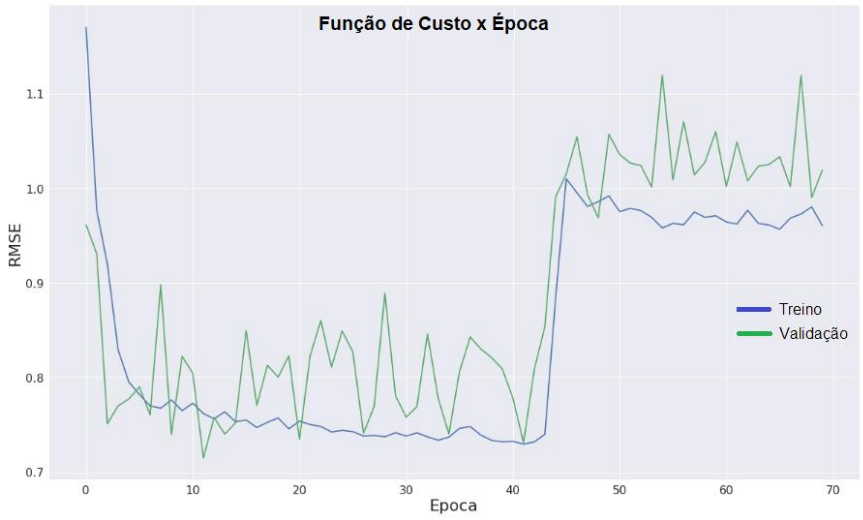
4.3.1 *Treinamento*

.3.1.1 Horizonte de previsão de 60 minutos

O treinamento da rede C para o horizonte de 60 minutos foi realizado a partir da série **hpc_m**, sendo essa a única série com granularidade de minutos. O treinamento foi realizado utilizando-se lotes de tamanho igual a 500 e duas camadas com 50 neurônios cada. O treinamento dessa rede mostrou-se muito lento devido ao tamanho do *dataset*, que contém mais de 2 milhões de linhas. Além disso, observou-se que a

rede atinge o *overfitting*, não conseguindo convergir de maneira adequada. A Figura 37 a seguir ilustra o treinamento da rede C na série com granularidade de minutos.

Figura 37 – Erro de treinamento o modelo C para o horizonte de 60 minutos



Fonte: Própria

O sinal que representa o consumo de potência por minuto é muito mais variável que no caso do consumo por horas, de modo que o modelo não possui capacidade suficiente para modelar o comportamento da onda. Para melhorar os resultados, outras combinações de hiperparâmetros poderiam ter sido experimentas, porém, devido à grande dificuldade para treinar o modelo, optou-se por realizar apenas uma sessão para uma avaliação superficial.

4.3.1.2 Horizonte de previsão de 60 horas

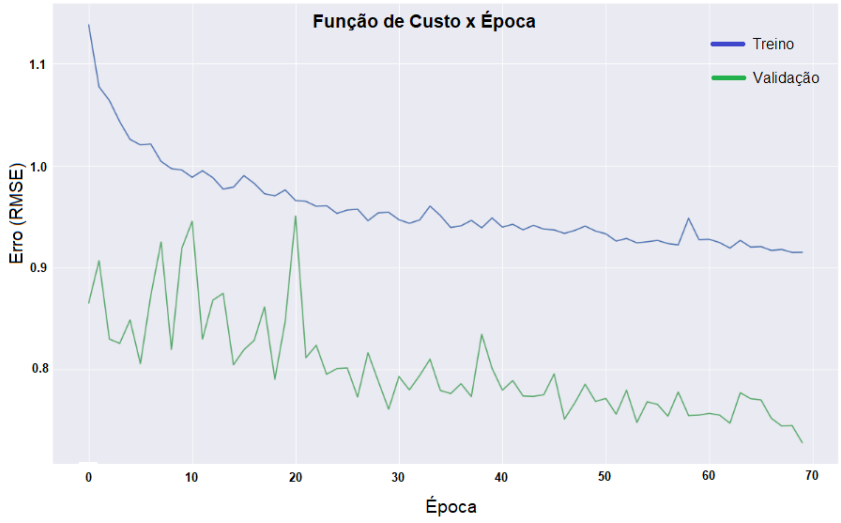
Em relação ao treinamento dos outros modelos, o treinamento da rede C para o horizonte de 60 horas passou por uma avaliação mais minuciosa do seu desempenho. Assim como no artigo de referência, experimentou-se o desempenho da rede *sequence-to-sequence* para diferentes combinações de camadas e de número de neurônios por camada. A Tabela 2 a seguir apresenta o desempenho da rede C com diferentes configurações durante o treinamento no *dataset* de validação para a série **hpc_h**.

Tabela 2 – Resultados de validação e teste para diferentes configurações da rede C na série hpc_h

Camadas	Neurônios por camada	Parâmetros treináveis	RMSE na validação [kW]	RMSE no teste [kW]
1	10	1.331	0,819	0,698
1	20	4.261	0,797	0,687
1	50	22.651	0,796	0,668
1	100	85.301	0,789	0,678
2	10	3.011	0,809	0,666
2	20	10.821	0,802	0,673
2	50	63.051	0,779	0,622
2	100	246.101	0,817	0,640
3	20	17.381	0,788	0,661
3	50	103.451	0,799	0,633

Os resultados apresentados condizem com o esperado da rede, de acordo com o artigo de referência, podendo-se concluir que o modelo atingiu convergência e o treinamento foi realizado com sucesso. A Figura 38 a seguir ilustra as curvas de treinamento para a rede C na série **hpc_h** com a configuração de 2 camadas com 50 neurônios.

Figura 38– Erro de treinamento do modelo C para o horizonte de horas



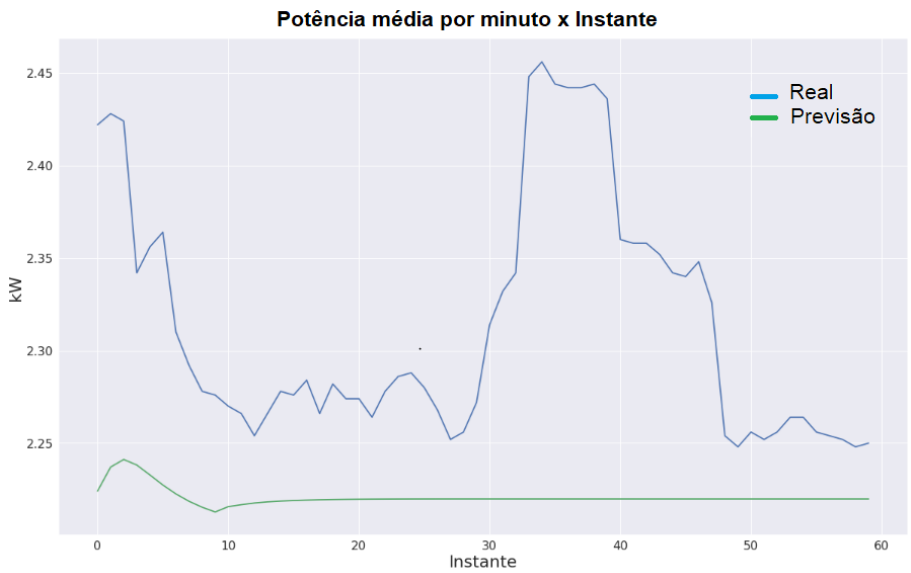
Fonte: Própria

4.3.2 *Resultados*

4.3.2.1 Horizonte de previsão de 60 minutos

Os resultados obtidos com a rede C para a série em minutos mostram que, ao menos com os hiperparâmetros utilizados, a rede não é capaz de se adaptar ao sinal. Os motivos para o resultado ruim estão relacionados à maior dificuldade para se realizar a previsão no horizonte de minutos, devido à maior imprevisibilidade do sinal.

Figura 37– Resultados de teste para a rede C para o horizonte de minutos



Fonte: Própria

4.3.2.2 Horizonte de previsão de 60 horas para a série hpc_h

Os resultados obtidos com a rede C durante o teste demonstram que a rede é capaz de modelar o comportamento real da curva de carga de uma residência, embora que ainda com um erro elevado. Os resultados demonstram que a rede C é capaz de modelar o comportamento médio da curva de carga, porém falha para picos de consumo, casos em que se encontram os maiores erros, sugerindo uma dificuldade da rede para modelar altas frequências.

Dentre todas as variações do modelo apresentadas anteriormente, a que obteve os melhores resultados foi a rede com 2 camadas com 50 neurônios. A Figura 37 a seguir apresenta as previsões obtidas com a rede C para a configuração de hiperparâmetros

descrita anteriormente, que atinge um erro RMSE de teste de 0.623, ficando muito próximo ao valor apresentado no artigo de referência, que é de 0.625.

Figura 37– Resultados de teste da rede C para o horizonte de horas

Fonte: Própria

Os resultados obtidos com a rede C sugerem a viabilidade da aplicação de redes neurais LSTM no problema de previsão de carga. Embora os resultados demonstrem que a rede tem dificuldades para modelar os picos, é notável que a curva prevista se adapta bem à curva real. É possível que a rede desempenhe melhor caso mais informação sobre a demanda por energia seja utilizada, no entanto, há referências que indicam que a associação do modelo recorrente com um modelo convolucional pode levar a resultados mais precisos, chegando a 0.437 para o mesmo *dataset* em que a rede C foi aplicada (KIM & CHO, 2018).

4.3.2.3 Horizonte de previsão de 60 horas para as demais séries

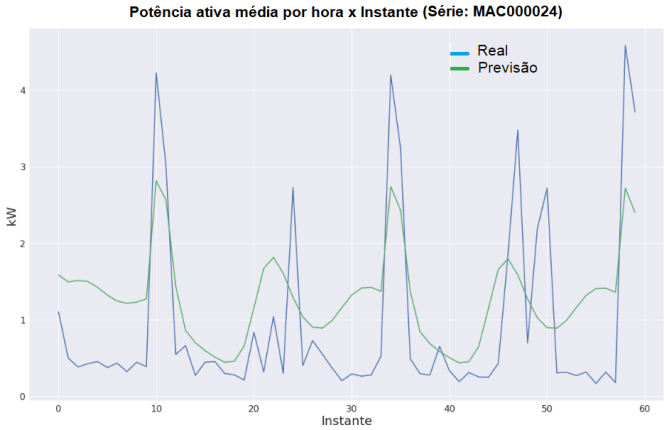
Para avaliar o desempenho do modelo em um caráter mais amplo, a rede C foi aplicada às demais séries apresentadas nesse trabalho, provenientes do *dataset* de Londres. *De* um modo geral, a rede alcançou bons resultados em todos os *datasets* em que foi aplicado, embora os resultados possam variar muito de um *dataset* para o outro. Além disso, a métrica utilizada para o cálculo do erro é dependente da amplitude dos dados, sendo que o resultado obtido para *datasets* diferentes não deve ser

comparado, embora os resultados demonstrem a viabilidade da rede C. A Tabela 3 a seguir apresenta os valores de erro RMSE obtidos para a rede C nas demais séries consideradas. A seguir, o resultado para algumas das séries é apresentado.

Tabela 3 – Resultados de validação e teste da rede C nas séries obtidas do *dataset* de Londres

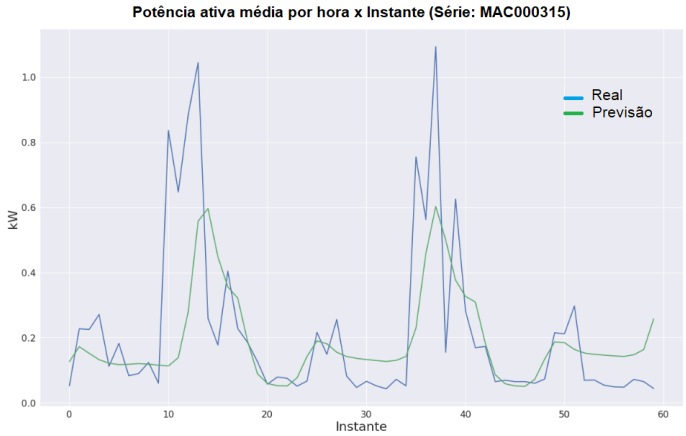
Série	RMSE Teste [kW]
MAC000017	0,310
MAC000024	0,934
MAC000247	0,688
MAC000315	0,217
MAC000520	0,150
MAC000836	0,569
MAC001267	0,951
MAC001563	0,542
MAC002749	0,619
MAC004583	0,258
MAC004748	0,333
MAC004914	0,345
MAC004940	0,367
MAC005269	0,286

Figura 38– Resultados de teste para a série MAC000024



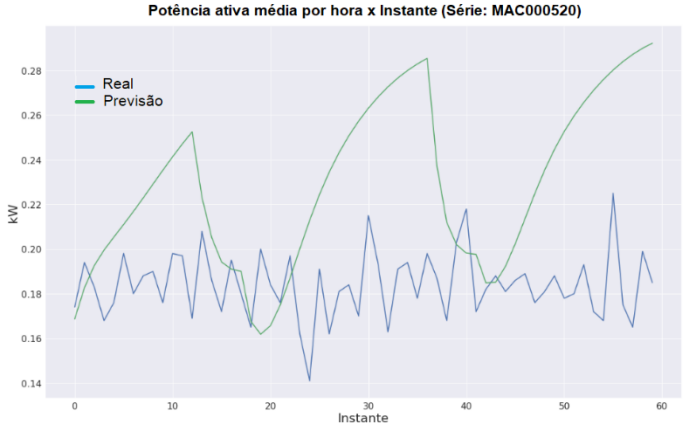
Fonte: Própria

Figura 39– Resultados de teste para a série MAC000315



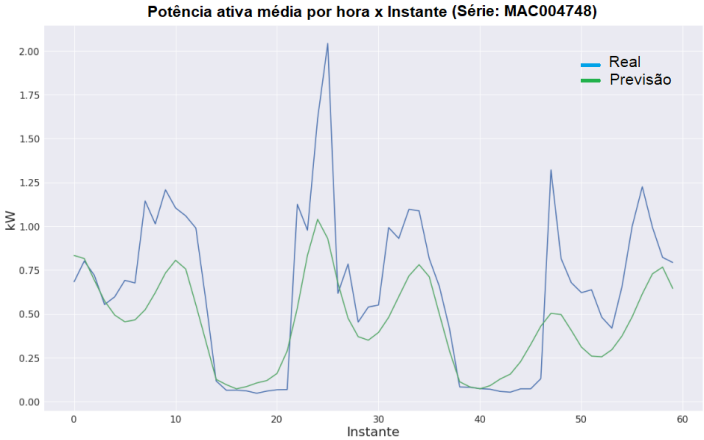
Fonte: Própria

Figura 40– Resultados de teste para a série MAC000520



Fonte: Própria

Figura 41– Resultados de teste para a série MAC004748



Fonte: Própria

5 DISCUSSÃO

Os resultados apresentados nesse trabalho sugerem a viabilidade das redes neurais LSTM na solução do problema de previsão de carga. Os resultados seguem o sentido da literatura disponível no campo, que aponta o grande potencial dos algoritmos de aprendizado profundo nas mais diversas áreas da engenharia.

Sob um ponto de vista mais amplo, os resultados sugerem que a aplicação de modelos de aprendizado profundo em problemas de previsão de séries temporais pode ser viável. A aplicação de modelos capazes de realizar previsões para séries temporais com eficiência é bastante promissora, haja vista que tal capacidade tem aplicação em diversos campos, envolvendo basicamente qualquer tipo de sinal.

Sob o ponto de vista da engenharia elétrica, os modelos de previsão de séries temporais possuem amplo espaço de aplicação, sendo essenciais para iniciativas que envolvam redes elétricas inteligentes. As redes inteligentes vêm ganhando cada vez mais espaço na comunidade científica, dadas as condições climáticas e os desafios globais ligados à energia.

Em suma, os resultados apresentados nesse trabalho reforçam a necessidade de se desenvolver mão-de-obra capacitada na aplicação de modelos de aprendizado de máquina, tendo em vista o grande potencial que representam. Nesse sentido, o projeto busca deixar uma contribuição que sirva como caminho inicial para outros estudantes que tenham interesse em se aprofundar no assunto.

6 CONCLUSÃO

O desenvolvimento desse trabalho pautou-se inicialmente na pesquisa sobre referências na literatura que indicassem o estado da arte no problema de previsão de carga. Seguindo esse caminho, foi possível notar que o setor energético, assim como todos os outros setores da economia, tende a se aproximar dos modelos de aprendizado profundo. O grande impacto do sucesso dos modelos de aprendizado profundo na atualidade impulsiona a pesquisa nesse sentido, de modo que cada vez mais surgirão novas aplicações desses modelos dentro do contexto da engenharia elétrica.

Os resultados obtidos sugerem a viabilidade dos modelos de aprendizado profundo na solução de problemas característicos do setor elétrico, e abrem caminho para que novos modelos sejam desenvolvidos. A grande flexibilidade dos modelos de aprendizado profundo permite que diferentes redes sejam construídas e adaptadas para a solução de problemas específicos.

Pode se afirmar finalmente que a experiência adquirida com o projeto foi extremamente positiva. Não só pelos bons resultados obtidos, mas também pelo longo caminho de aprendizado percorrido, que permitiu desenvolver habilidades extremamente úteis no que diz respeito à aplicação prática.

REFERÊNCIAS

FERREIRA, H. C.; LAMPE, L.; NEWBURY, J.; SWART, T. G. **Power Line Communications. Theory and Applications for Narrowband and Broadband Communications over Power Lines**. 2th ed. Ed. John Wiley and Sons, 2010

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. MIT Press, 2016. Disponível em: < https://www.deeplearningbook.org/>. Acesso em: 17 de nov. de 2018.

HEBRAIL, G.; BERARD, A. **Individual household electric power consumption Data Set**. UCI machine learning repository, 2013

HEINEMANN, G.T.; NORDMIAN, D. A.; PLANT, E. C. **The Relationship Between Summer Weather and Summer Loads – A Regression Analysis**. IEEE Transactions on Power Apparatus and Systems, 1966, PAS-85, 1144-1154

HERNANDEZ, L.; BALADRON, C.; AGUIAR, J. M.; CARRO, B.;SANCHEZ-ESGUEVILLAS, A. J.; LLORET, J.;MASSANA, J. **A Survey on Electric Power Demand Forecasting: Future Trends in Smart Grids, Microgrids and Smart Buidings**. IEEE Communications Surveys Tutorials, 2014, 16, 1460-1495

HOCHREITER, S.; SCHMIDHUBER, J. **Long short-term memory**. Neural computation 9 (8): 1735 – 1780, 1997

HONG, T.; FAN, S. **Probabilistic electric load forecasting: A tutorial review**. International Journal of Forecasting, 2016, 32, 914-938

KIM, T.; CHO, S. **Predicting the Household Power Consumption Using CNN-LSTM Hybrid Networks**. Intelligent Data Engineering and Automated Learning – IDEAL, 2018

MARINO, D. L.; AMARASINGHE, K.; MANIC, M. **Building energy load forecasting using Deep Neural Networks**. IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society, 2016, 7046-7051

MCCULLOCH, W.; PITTS, W. **A logical calculus of the ideas imanente in nervous activity**. Bulletin of Mathematical Biophysics, 115-133, 1943

MCMENAMIN, J. C.; MONFORTE, F. A. **Short-term load forecasting with neural networks**. The Energy Journal, vol. 19, no. 4, pp. 43-61, 1998

MINSKY, M.; PAPERT, S. **An Introduction to Computational Geometry**. M.I.T. Press, Cambridge, MA, 1969

MITCHELL, T. M. **Machine Learning**. McGraw-Hill, New York, 1997

RITCHIE, H.; ROSER, M. **Energy Production & Changing Energy Sources**. Publicado online em OurWorldInData.org. Fonte: 'https://ourworldindata.org/energy-production-and-changing-energy-sources', 2018

SRIVASTAVA, N.; HINTON, G.; KRIZHEVSKY, A; SUTSKEVER, I; SALAKHUTDINOV, R. **Dropout: A Simple Way to Prevent Neural Networks from Overfitting**. Journal of Machine Learning Research, University of Toronto, 2014

SUTSKEVER, I.; VINYALS, O.; LE, Q. V. **Sequence to Sequence Learning with Neural Networks**. NIPS 2014, 2014

WANG, Y.; CHEN, Q.; HONG, T.; KANG, C. **Review of Smart Meter Data Analytics: Applications, Methodologies and Challenges**. IEEE Transactions on Smart Grid, 2018, 1-1

WILLE-HAUSSMANN, B.; ERGE, T.; WITTNER, C. **Decentralized optimisation of cogeneration in virtual power plant**. Solar Energy, vol. 84, no. 4, pp. 604-611, 2010