Projeto de Graduação

DEPARTAMENTO DE ENGENHARIA ELÉTRICA

02 de Julho de 2019

PREVISÃO DE DEMANDA POR MÉTODO DE INTELIGÊNCIA ARTIFICIAL

Rafael Cardoso de Assumpção

PUC RIO

Projeto de Graduação

DEE DEPARTAMENTO DE ENGENHARIA ELÉTRICA

PREVISÃO DE DEMANDA POR MÉTODO DE INTELIGÊNCIA ARTIFICIAL

Aluno: Rafael Cardoso de Assumpção

Orientador: Delberis Araujo Lima

Trabalho apresentado com requisito parcial à conclusão do curso de Engenharia Elétrica na Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, Brasil.

Agradecimentos

Agradeço especialmente aos meus pais, irmãos e esposa pelo apoio, suporte e incentivo fundamentais durante essa jornada.

Ao meu orientador e ao meu coorientador, professores Delberis Araujo Lima e Leonardo Alfredo Forero Mendoza, respectivamente, que foram incansáveis em suas orientações e pesquisas.

Sou grato aos meus colegas e amigos da PUC-Rio, com os quais compartilhei inesquecíveis experiências acadêmicas e de vida.

Agradeço à PUC-Rio, pela oportunidade e por me proporcionar o melhor ambiente educacional, e sobretudo pelo aprendizado. Estendo esse agradecimento ao corpo docente e discente, com quem tanto aprendi durante esse período.

Por fim, agradeço a Deus que me deu força e me permitiu realizar esse sonho.

Resumo

Este Projeto de Graduação em Engenharia Elétrica consiste na previsão de uma série temporal de carga de um consumidor especial, por meio de técnicas de Inteligência Artificial.

Diversos métodos podem ser utilizados para prever o consumo de energia elétrica, entretanto, redes neurais recorrentes, em especial as redes Long Short-Term Memory, vêm se destacando na previsão de séries temporais devido a sua capacidade de armazenamento de memória a longo prazo. Este trabalho tem como objetivo geral o desenvolvimento de um programa computacional que emprega redes Long Short-Term Memory para a previsão de demanda, utilizando métricas de distância para sua avaliação.

Palavras-chave: Série temporal; Previsão; Redes Neurais; LSTM

DEPARTAMENTO
DE ENGENHARIA
ELÉTRICA

ENERGY DEMAND FORECASTING BY ARTIFICIAL INTELLIGENCE TECHNIQUE

Abstract

This Electrical Engineering Undergraduate Project consists in the time series forecasting of the load of a special consumer, by Artificial Intelligence techniques.

Several methods are available for predicting of the consumption of electrical energy. However, recurrent neural networks, especially Long Short-Term Memory networks, have been standing out in time series forecasting due to their long term memory capacity. The objective of this work is the development of a computational program that applies Long Short-Term Memory networks to predict demand, with the use of distance metrics for result evaluation.

Keywords: Time series; Prediction; Neural Networks; LSTM

Sumário

Lista de Figuras

1. Introdução

a. Considerações iniciais

Diversos métodos podem ser utilizados para prever o consumo de energia elétrica, entretanto, as redes neurais recorrentes, em especial as com *Long Short-Term Memory* (LSTM), vêm se destacando na previsão de séries temporais devido a sua capacidade de armazenamento de memória a longo prazo. A rede neural LSTM utilizada neste trabalho realiza a previsão de carga da Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio), com base em seus registros históricos mensais de demanda.

b. Motivação

A energia elétrica é um insumo fundamental e estratégico para qualquer tipo de indústria ou empresa. Ser capaz de prever a demanda medida, permite fazer melhores escolhas para na hora de contratar ou renovar contratos de energia elétrica e com isso reduzir custos que podem ser vitais para o futuro da indústria ou empresa.

c. Objetivos

Este trabalho tem como objetivo geral o desenvolvimento de um programa computacional, empregando redeis neurais com *Long Short-Term Memory* (LSTM), para a previsão de demanda e utilizado das métricas de distância (previsto/real) para avaliá-la.

d. Organização do trabalho

Este trabalho está organizado em 6 (seis) capítulos, incluindo a introdução.

No Capítulo 1 apresenta a introdução ao tema do trabalho, dissertando sobre a importância, contendo a motivação para o trabalho, os objetivos e a organização do trabalho.

No Capítulo 2 apresenta uma breve fundamentação teórica do Setor Elétrico Brasileiro, de séries temporais, do algoritmo de aprendizagem de máquina, de redes neurais artificias, de redes neurais recorrentes e do modelo de redes neurais LSTM.

No Capítulo 3 apresenta a metodologia utilizada.

No Capítulo 4 apresenta os resultados obtidos.

No Capítulo 5 mostra as principais conclusões relacionadas ao tema de estudo, assim como as considerações finais sobre os resultados obtidos.

No Capítulo 6 apresenta o Apêndice com o código completo do programa computacional desenvolvido.

2. Fundamentação Teórica

Neste capítulo são apresentados os conceitos utilizados ao longo deste trabalho.

a. Setor Elétrico Brasileiro

Energia elétrica é um insumo essencial à sociedade, indispensável ao desenvolvimento socioeconômico das nações. O custo com energia elétrica no processo de produção pode ser fator de relevância para alcançar resultados econômicos mais satisfatórios. Muitas empresas possuem a energia elétrica entres seus principais custos de produção e por isso torna-se necessário entender como funciona o Setor Elétrico Brasileiro visando encontrar a melhor opção para contratação de energia.

No Brasil, a principal fonte de geração é a hidrelétrica (água corrente dos rios), seguida das termelétricas (gás natural, carvão mineral, combustíveis fósseis, biomassa e nuclear).O mercado de energia do Setor Elétrico Brasileiro é composto pelas geradoras que produzem a energia, pelas transmissoras que transportam do ponto de geração até os centros consumidores, de onde as distribuidoras a levam até a casa dos cidadãos. Há ainda as comercializadoras, empresas autorizadas a comprar e vender energia para os consumidores livres (geralmente consumidores que precisam de maior quantidade de energia).

1. Conceitos

i. Consumo de energia elétrica

Quantidade de potência elétrica (kW) consumida em um intervalo de tempo, expresso em quilowatt-hora (kWh). No caso de um equipamento elétrico o valor é obtido através do produto da potência do equipamento pelo seu período de utilização e, em uma instalação residencial, comercial ou industrial, através da soma do produto da demanda medida pelo período de integração.

ii. Demanda

Média das potências elétricas ativas ou reativas, expressa em quilowatts (kW), solicitadas ao sistema elétrico pela parcela da carga instalada em operação na unidade consumidora, durante um intervalo de tempo especificado.

iii. Demanda medida

Maior demanda de potência ativa, expressa em quilowatts (kW), verificada por medição, integralizada no intervalo de 15 (quinze) minutos durante o período de faturamento.

iv. Demanda contratada

Demanda de potência ativa a ser obrigatoriamente e continuamente disponibilizada pela concessionária, no ponto de entrega, conforme valor e período de vigência no contrato de fornecimento e que deverá ser integralmente paga, seja ou não utilizada durante o período de faturamento, expressa em quilowatts (kW).

v. Demanda de ultrapassagem

Parcela da demanda medida que excede o valor da demanda contratada, expressa em quilowatts (kW). Quando ocorre demanda de ultrapassagem o consumidor fica sujeito a pagar multas, que normalmente possuem valores elevados, dependendo de quanto foi o valor ultrapassado com relação ao contratado. Calcula-se multiplicando a tarifa de ultrapassagem pelo valor da demanda medida que supera a Demanda Contratada.

2. O mercado brasileiro de energia elétrica

Este mercado vem sendo marcado por profundas transformações em suas estruturas organizacionais e produtivas nas últimas décadas. Atualmente existem dois ambientes de negociação:

- Ambiente de Contratação Regulada (ACR): estão presentes os agentes geradores, as distribuidoras e os consumidores ditos "cativos". A contratação de energia é regulada, evitando assim, a formação de preços abusivos por parte das concessionárias de distribuição. Os consumidores cativos pagam uma tarifa de energia fixa, pré-estabelecida e regulada pela Agência Nacional de Energia Elétrica (ANEEL) no início de cada ano, variável por distribuidora e independentemente do seu consumo.

- Ambiente de Contratação Livre (ACL): é o mercado em que os consumidores podem escolher seu fornecedor de energia, negociando livremente um conjunto de variáveis como prazo contratual, preços, variação do preço ao longo do tempo e serviços associados à comercialização. A comercialização de energia elétrica é realizada entre geradores, importadores de energia, comercializadores e consumidores livres.

b. Séries Temporais

Uma série temporal é um conjunto de observações ordenadas no tempo, não necessariamente igualmente espaçadas, que apresentam dependência entre instantes de tempo (MORETTIN; TOLOI, 2006). Uma grande quantidade de fenômenos pode ser enquadrada nesta categoria, tais como: índices diários de ações, casos semanais de uma determinada doença, temperaturas máximas e mínimas diárias em uma cidade, resultado de um eletrocardiograma, acidentes ocorridos em uma determinada avenida durante um mês, etc.

Segundo o modelo clássico, uma série temporal pode ser composta de 4 (quatro) padrões (nem sempre uma série temporal irá apresentar todos os padrões): tendência, variações cíclicas ou ciclos, variações sazonais ou sazonalidade e variações irregulares.

A tendência indica o comportamento de longo prazo de uma série, isto é, se ela cresce, decresce ou permanece estável, e qual a velocidade destas mudanças. Nos casos mais comuns trabalha-se com tendência constante, linear ou quadrática. Por exemplo, pode ser causada pelo crescimento demográfico, ou mudança gradual de hábitos de consumo, ou qualquer outro aspecto que afete a variável de interesse no longo prazo.

As variações cíclicas ou ciclos são caracterizadas pelas flutuações nos valores da variável nas séries, de forma suave e repetida, ao longo da componente de tendência. Por exemplo, podem ser resultado de variações da atividade econômica ou ciclos meteorológicos.

As variações sazonais ou sazonalidade em uma série, correspondem às flutuações nos valores da variável que sempre ocorrem em um determinado período do ano, do mês, da semana ou do dia. A diferença essencial entre as componentes sazonal e cíclica é que a primeira possui movimentos facilmente previsíveis, ocorrendo em intervalos regulares de tempo, enquanto movimentos cíclicos tendem a ser irregulares.

As variações irregulares ou aleatórios são caracterizadas pelas flutuações inexplicáveis, resultado de fatos fortuitos e inesperados. Por exemplo, podem ser resultados de catástrofes naturais, atentados terroristas, decisões intempestivas de governos, etc.

Em geral ao estudarmos uma série temporal estamos interessados em:

- Análise e modelagem da série temporal - descrever a série, verificar suas características mais relevantes e suas possíveis relações com outras séries;
- Previsão na série temporal - a partir de valores históricos da série procura-se estimar previsões de curto prazo. O número de instantes à frente para o qual é feita a previsão é chamado de horizonte de previsão;
- Descrever seu comportamento - descrever o comportamento da série, utilizando gráficos, verificando a existência de tendências, ciclos, variações sazonais, pontos influentes entre outros;
- Procurar a periodicidade relevante nos dados.

c. Aprendizado de Máquina

A Inteligência Artificial é um dos campos mais recentes em ciências e engenharia. O trabalho começou logo após a Segunda Guerra Mundial, e o próprio nome foi cunhado em 1956. Atualmente, a Inteligência Artificial abrange uma enorme variedade de subcampos, do geral (aprendizagem e percepção) até tarefas específicas, como jogos de xadrez, demonstração de teoremas matemáticos, criação de poesia, direção de um carro em estrada movimentada e diagnóstico de doenças. A Inteligência Artificial é relevante para qualquer tarefa intelectual; é verdadeiramente um campo universal (Norvig; Russel, 2013).

Aprendizado de Máquina, do inglês *Machine Learning*, é um campo da Ciência da Computação que estuda algoritmos para que o software aprenda autonomamente sem uma descrição explícita de cada regra de operação, baseados na teoria de álgebra, probabilidades, estatística e otimização. Estes algoritmos devem tomar decisões inteligentes baseados em comportamentos que não foram diretamente programados, mas sim aprendidos ou adaptados por meio de algum conhecimento extraído dos dados.

É uma área que trabalha com otimizações matemáticas com o objetivo de reconhecimento e detecção de padrões. Seu objetivo é extrair características latentes dos dados que permitem uma classificação imediata de cada dado de entrada em uma classe particular. Existem diversas classes de problemas de Aprendizado de Máquina, sendo as mais comuns:

- Classificação - onde ao se entregar um determinado dado para o algoritmo de Aprendizado de Máquina, ele deve ser capaz de dizer à qual ou quais das já conhecidas classes do problema esse novo dado pertence;

- Clusterização - onde o objetivo é agrupar os dados em classes de acordo com suas semelhanças, sem conhecer previamente as classes existentes;

- Regressão - que é semelhante à classificação, mas os valores a serem previstos são contínuos e não discretos como na classificação.

Os principais algoritmos de Aprendizado de Máquina são:

- Aprendizado supervisionado – constroem um modelo matemático de um conjunto de dados que contém tanto as entradas quanto as saídas desejadas (Norvig; Russel, 2013). Os dados são conhecidos como dados de treinamento. Por meio de uma otimização de uma função de custo, os algoritmos de aprendizado supervisionado modelam uma função que pode ser usada para prever os dados de saída. Um algoritmo que melhora a precisão, ao longo do tempo, de seus dados saídas, é dito ter aprendido a executar essa tarefa. Os algoritmos de aprendizado supervisionado envolvem problemas de classificação e regressão;

- Aprendizado não supervisionado - estuda métodos onde não se tem um conjunto de dados de saída, apenas um conjunto de dados de entrada. Os algoritmos, portanto, aprendem com dados de teste que não foram rotulados, classificados ou categorizados. Um dos principais métodos utilizado no aprendizado não supervisionado é a análise de cluster;

- Aprendizagem por reforço - é um método de programação de agentes através do oferecimento de recompensas e punições, sem a necessidade de especificar como uma tarefa deve ser realizada. O agente que deve aprender como se comportar em um ambiente dinâmico através de interações do tipo "tentativa e erro", de modo a maximizar a recompensa.

O Aprendizado de Máquina envolve a criação de um modelo, que é treinado, utilizando alguns dados de treinamento, e utilizado para fazer previsões. Vários tipos de modelos foram usados e pesquisados para sistemas de Aprendizado de Máquina:

- Algoritmos genéticos;

- Árvores de Decisões;

- Redes Bayesianas;

- Redes Neurais Artificiais.

Neste trabalho utilizamos as Redes Neurais Artificiais, por isso iremos focar em detalhar o seu funcionamento.

1. Redes Neurais Artificiais

O trabalho em Redes Neurais Artificiais (RNA ou ANN do inglês *Artificial Neural Networks*) tem sido motivado desde o começo pelo reconhecimento de que o cérebro humano é um computador altamente complexo, não-linear e paralelo, que processa informações de uma forma inteiramente diferente do computador digital convencional (Haykin, 2001).

As redes neurais artificiais tiveram seu primeiro modelo formalizado em 1943, pelo neurofisiologista Warren McCulloch, do Massachusetts Institute of Technology (MIT), e pelo matemático Walter Pitts, da Universidade de Illinois, com o objetivo de desenvolver um modelo computacional análogo a neurônios biológicos e circuitos eletrônicos. O trabalho consistia em um modelo de resistores variáveis e amplificadores representando conexões sinápticas de um neurônio biológico

As Redes Neurais Artificiais (RNA) são estruturas matemáticas que tentam imitar o modo básico de funcionamento do cérebro humano. Os blocos de construção das RNA, como sua contrapartida biológica, modelam o comportamento de alto nível dos neurônios biológicos, no sentido de que negligenciam aspectos biológicos desnecessários e retêm apenas sua fundamental função matemática subjacente, que é uma combinação linear ponderada de suas entradas sujeitas a uma função de ativação, ou seja, uma função que gera um valor de decisão dependendo de suas entradas (Fonseca, 2016).

As RNA são formadas por neurônios interligados entre si simulando sinapses artificiais estruturadas, baseado em modelos biológicos. Desta forma, as RNA possuem características como capacidade de aprendizagem através de treinamento, habilidade de generalização e tolerância a falhas (Haykin, 2001).

Um neurônio é uma unidade de processamento de informação fundamental em uma RNA. A Figura 1 mostra o modelo de um neurônio, onde podemos identificar 3 elementos básicos do modelo neuronal:

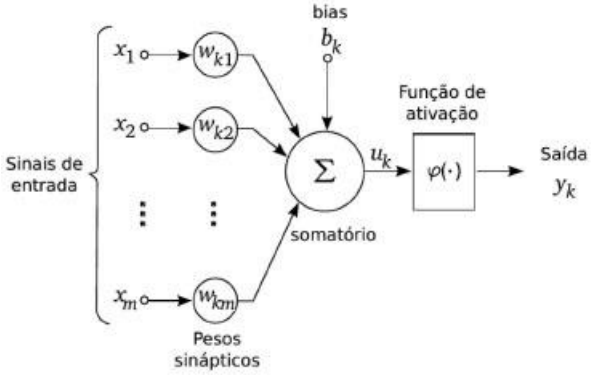

Figura 1 – Modelo de um neurônio (Haykin, 2001)

- Um conjunto de sinapses, cada uma caracterizada por um peso próprio. Por exemplo, um sinal x_j na entrada *j* é multiplicada pelo peso w_{kj}, onde *k* se refere ao neurônio em questão;

- Um somador para realizar a soma ponderada dos sinais de entrada, realizada pelo combinador linear Σ;

- Uma função de ativação para restringir a amplitude da saída de um neurônio. Ela limita o intervalo permissível de amplitude do sinal de saída a um valor finito. O intervalo de saída de um neurônio assumirá sempre valores reais entre 0 (zero) e 1 (um) ou alternativamente valores reais entre -1 (menos um) e 1 (um);

O bias b_k pode aumentar ou diminuir a entrada da função de ativação.

Podemos descrever o neurônio k conforme as Equações 1 e 2. Onde $x_1, x_2, \dots, x_m$ são os sinais de entrada, $w_{k1}, w_{k2}, \dots, w_{km}$ são os pesos sinápticos do neurônio *k*, b_k é o bias, u_k é a saída do combinador linear, $\varphi(.)$ é a função de ativação e y_k é o sinal de saída do neurônio.

$$u_k = \sum_{j=1}^{m} w_{kj} x_j \qquad (1)$$

$$y_k = \varphi(u_k + b_k) \qquad (2)$$

Os 3 (três) tipos básicos de função de ativação são:

- Função de Limiar: está descrita conforme a Equação 3.

$$\varphi(v) = \begin{cases} 1, & se\ v \geq 0 \\ 0, & se\ v < 0 \end{cases} \qquad (3)$$

- Função linear por partes: está descrita conforme a Equação 4. Pode ser vista como uma aproximação de um amplificador não-linear, se a região linear de operação não entrar em saturação, observamos um combinador linear.

$$\varphi(v) = \begin{cases} 1, & se\ v \geq \frac{1}{2} \\ v, & se\ \frac{1}{2} > v > -\frac{1}{2} \\ 0, & se\ v \leq -\frac{1}{2} \end{cases} \quad (4)$$

- Função sigmoide: é definida como uma função estritamente crescente com os comportamentos linear e não-linear balanceados. A função logística e a função tangente hiperbólica, descritas conforme as Equações 5 e 6 respectivamente, são bons exemplos de função sigmoide.

$$\varphi(v) = \frac{1}{1+e^{-\beta v}} \quad (5)$$

$$\varphi(v) = \tanh(v) \quad (6)$$

Onde β é o parâmetro de inclinação da função sigmoide. Variando este parâmetro obtemos funções sigmoides com diferentes inclinações.

Uma RNA pode ser dividida em três camadas básicas:

- A camada de entrada é responsável por receber os sinais de entrada x_m;
- As camadas ocultas ou intermediárias são compostas pelos neurônios artificiais responsáveis pelo processamento interno da rede;
- A camada de saída é composta pelos neurônios artificiais responsáveis por gerar o sinal y_k.

Os possíveis arranjos de neurônios são chamados de arquitetura da RNA. Podemos identificar 3 (três) classes principais de arquitetura de rede:

- Redes neurais alimentadas adiante com camada única;
- Redes neurais alimentadas diretamente com múltiplas camadas;
- Redes neurais recorrentes.

O grande diferencial de uma rede neural em relação a outros modelos matemáticos é a sua capacidade de aprender com o passado. Uma rede neural é capaz de, ao longo de um processo de aprendizado, realizar tal feito por meio do ajuste dos seus pesos sinápticos até obter uma resposta satisfatória.

Existem várias técnicas matemáticas voltadas para a previsão de séries temporais, como por exemplo: regressões lineares, regressões logísticas e o método dos mínimos quadrados. Estes métodos, contudo, não se comportam bem quando a série apresenta algum ruído, ou não alcançam desempenho satisfatório quando o número de variáveis de entrada é muito grande. As redes neurais artificiais surgem como uma alternativa para este tipo de problema, pois além de contornarem as limitações apresentadas, conseguem modelar problemas não lineares e identificar características de sazonalidade, tendência e periodicidade, comumente presentes em séries temporais (Dametto, 2018).

Neste trabalho utilizamos as redes neurais recorrentes, por isso iremos focar em detalhar o seu funcionamento.

i. Rede Neural Recorrente

As redes neurais recorrentes (RNN do inglês *Recurrent Neural Networks*) são estruturas de processamento capazes de representar uma grande variedade de comportamentos dinâmicos não lineares. As redes neurais recorrentes são redes que possuem uma ou mais conexões de realimentação. A realimentação em um sistema dinâmico ocorre sempre que a saída de um elemento do sistema influencia em parte a entrada aplicada àquele elemento particular, originando assim um ou mais de um caminho fechado para transmissão de sinais em torno do sistema (Haykin, 2001).

A relevância dessa estrutura é a possibilidade de reter informações sobre as sequencias dos dados. Antes, cada dado de entrada só contribuía para o treinamento da rede, mas as informações sobre a correlação entre cada dado de entrada e o dado que o antecede não influenciava a etapa de treinamento.

A conexão de realimentação do neurônio para ele mesmo atua como um tipo de elemento de memória, que para uma tomada de decisão atual, leva em conta a história das decisões tomadas anteriormente e, portanto, os dados anteriores.

As redes recorrentes produzem modelos dinâmicos, modelos que mudam ao longo do tempo, de formas que produzem classificações precisas dependentes do contexto dos exemplos que estão expostos.

A Figura 2 apresenta o diagrama de uma rede neural recorrente simples, proposta por Elman. As redes de Elman utilizam realimentação de cada um dos neurônios da camada escondida para todos os neurônios da camada escondida, isto é, cada neurônio escondido recebe o vetor de estados "anterior" completo $a(k-1)$. Apesar do exemplo exibido conter apenas uma entrada e uma saída, essas redes são genéricas, podendo ter várias entradas e saídas (Castro; Barros; Lazo; Tupac; Miranda, 1997).

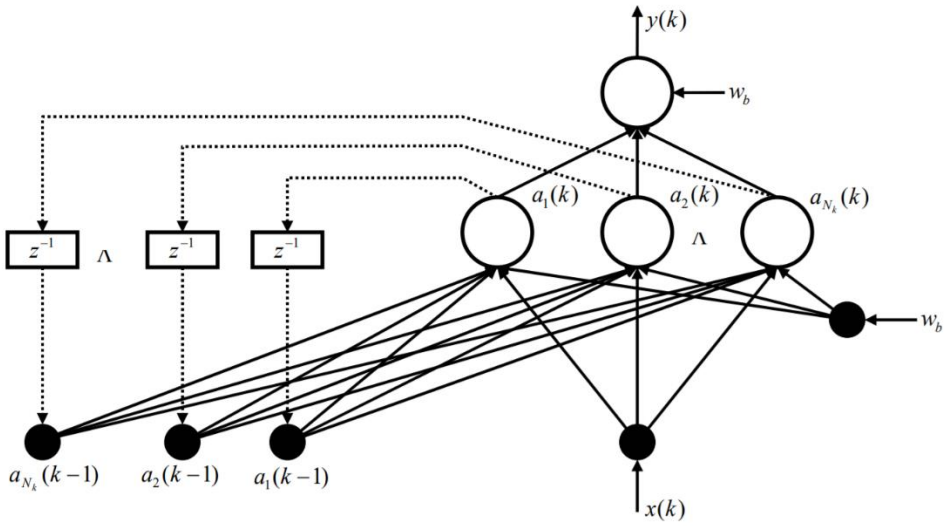

Figura 2 - Diagrama rede neural recorrente de Elman (Castro; Barros; Lazo; Tupac; Miranda, 1997)

As redes neurais recorrentes possuem muitas variações, a seguir temos algumas delas:

- Hopfield;
- Long Short-Term Memory (LSTM);
- Recurrent Multi-Layer Perceptron (RMLP);
- Totalmente Recorrente.

As redes neurais recorrentes podem apresentar desafios no processo de treinamento. Ainda que na teoria sejam capazes de lidar com dependências de longo termo (longas sequências), na prática isso muitas vezes não é possível, pois em métodos de treinamento baseados em gradientes a informação de erro desaparece.

Diferentes lembranças de curto prazo devem ser recontadas em momentos diferentes, a fim de atribuir o significado certo à entrada atual. Algumas dessas memórias terão sido forjadas recentemente e outras memórias terão forjado muitos passos antes de serem necessários. A rede recorrente que efetivamente associa memórias e entrada remota no tempo é chamada de Long Short-Term Memory (LSTM), a qual veremos em seguida.

ii. Rede Neural Long Short-Term Memory

As redes neurais com células Long Short-Term Memory (LSTM) são uma variação das redes neurais recorrentes, proposta pelos pesquisadores alemães Sepp Hochreiter e Juergen Schmidhuber, em 1997, como uma solução para o problema do gradiente de desaparecimento.

Uma rede neural LSTM possui as mesmas propriedades que uma rede neural recorrente padrão, contudo é capaz de armazenar informações por longos períodos de tempo ao processar uma sequência.

Uma unidade comum de LSTM é composta de uma célula, uma porta de entrada, uma porta de saída e uma porta de esquecimento. A célula lembra valores em intervalos de tempo arbitrários e as três portas regulam o fluxo de informações para dentro e para fora da célula.

Os nós de memória de uma LSTM são chamados de células. Células possuem propriedades mais complexas que suas equivalentes nas redes neurais recorrentes tradicionais. Estas são capazes de carregar informações até o final do processamento de uma sequência bem como selecionar informações que devem ser "esquecidas" a partir de um certo ponto. A Figura 3 representa o funcionamento de uma célula LSTM, onde no instante t, C_t representa o estado de célula, h_t a saída da célula, x_t a amostra da sequência, f_t a porta de esquecimento, i_t a porta de entrada o_t a porta de saída. Todos esses valores acabam sendo concatenados, multiplicados ou somados.

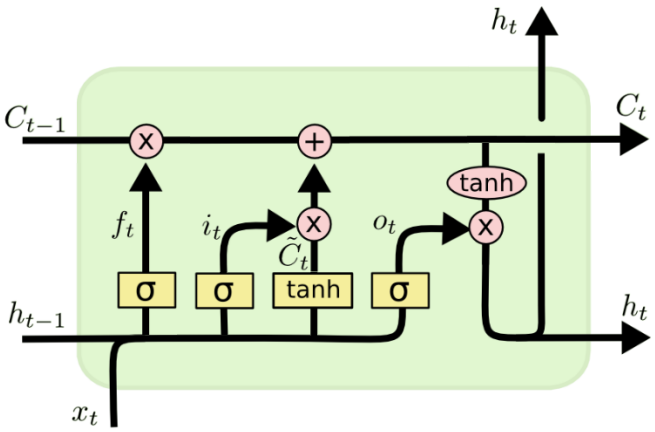

Figura 3 - Funcionamento de uma célula LSTM (adaptado de Olah, 2015)

A seguir iremos detalhar o funcionamento da célula LSTM e suas portas:

- Porta de esquecimento - seu papel é controlar o conteúdo da célula de memória, seja para configurá-los ou redefini-los. Esta decisão é feita por uma camada sigmoide, que tem como entrada o valor de x_t e de h_{t-1}, saída da célula anterior, e emite como resposta um número entre 0 e 1 para o estado de célula C_{t-1}. Onde 0 representa que o estado máximo para remover o dado e 1 representa o estado máximo para manter o dado. Pode ser representado pela Equação 7.

$$f_t = \sigma\left(W_f . [h_{t-1}, x_t] + b_f\right) \quad (7)$$

- Porta de entrada – seu papel é decidir que valores vamos atualizar, esta decisão é feita por uma camada sigmoide que posteriormente é combinada com $\tilde{C}_t$, que é um vetor de novos valores candidatos de uma camada $tanh(.)$ que podem ser adicionados ao estado. Após esta combinação, o estado de célula antigo, C_{t-1}, é atualizado para o novo estado de célula C_t. A etapa da porta de esquecimento já decidiu anteriormente se o estado de célula seria modificado ou não. Pode ser representado pelas Equações 8, 9 e 10.

$$i_t = \sigma(W_i . [h_{t-1}, x_t] + b_i) \quad (8)$$

$$\tilde{C}_t = tanh(W_C . [h_{t-1}, x_t] + b_C) \quad (9)$$

$$C_t = f_t \times C_{t-1} + i_t \times \tilde{C}_t \quad (10)$$

- Porta de saída – tem um funcionamento similar ao da porta de entrada onde uma camada sigmoide que decide que valores serão lançados. O novo estado de célula C_t passa por uma função de ativação, representada na figura pela função $tanh(.)$ por ser uma função comumente utilizada, e sua saída é multiplicada pela saída da sigmoide, de modo que apenas produzimos as partes que decidimos. Pode ser representado pelas Equações 11 e 12.

$$o_t = \sigma(W_o . [h_{t-1}, x_t] + b_o) \quad (11)$$

$$h_t = o_t \times tanh(C_t) \quad (12)$$

As redes neurais LSTM são, hoje em dia, um dos algoritmos de última geração em aprendizado profundo (do inglês *Deep Learning*), e suas desempenho tem se mostrado superior ao de outros tipos de RNN que são geralmente usados para resolver o mesmo conjunto de problemas onde o LSTM é empregado, prever e produzir decisões de classificação a partir de dados de séries temporais. A seguir temos algumas das principais aplicações das redes neurais LSTM.

- Aplicações Biomédicas;
- Análise de Sentimentos;
- Análise e Composição Musical;
- Reconhecimento de voz, escrita, rostos e objetos;
- Tradução.

d. Métricas de Desempenho

Nesta seção estão apresentadas as funções mais comuns para avaliar o desempenho de modelos.

1. Erro Médio Absoluto

O erro médio absoluto, MAE (do inglês *Mean Absolute Error*), é definido como o desvio médio entre os valores reais e previstos. É dado pela Equação 13, onde n é o número de predições geradas sendo analisadas, $real_t$ é o valor real da série no instante t e $previsto_t$ é o valor previsto da série no instante t.

$$MAE = \frac{1}{n}\sum_{t=1}^{n}|real_t - previsto_t| \quad (13)$$

2. Erro Quadrático Médio

O erro quadrático médio, MSE (do inglês *Mean Squared Error*), provavelmente é a métrica mais utilizada para calcular o desempenho de modelos de previsão. É definido como sendo a soma entre a variância e do quadrado das diferenças obtidas entre os valores reais e previstos. Podemos simplificar dizendo que o MSE é a média dos quadrados dos erros, conforme é dado pela Equação 14, onde n é o número de predições geradas sendo analisadas, $real_t$ é o valor real da série no instante t e $previsto_t$ é o valor previsto da série no instante t.

$$MSE = \frac{1}{n}\sum_{t=1}^{n}(real_t - previsto_t)^2 \quad (14)$$

3. Raiz Quadrada do Erro Quadrático Médio

A raiz quadrada do erro quadrático médio, RMSE (do inglês *Root-Mean-Squared Error*), é definido como sendo a raiz quadrada do erro quadrático médio. O quanto mais próximo de zero for o erro, melhor são os resultados obtidos. O RMSE é utilizado quando grandes erros são indesejáveis (o que é o caso da previsão da demanda), pois ele considera um peso maior para os maiores erros. O RMSE é dado pela Equação 15, onde n é o número de predições geradas sendo analisadas, $real_t$ é o valor real da série no instante t e $previsto_t$ é o valor previsto da série no instante t.

$$RMSE = \sqrt{\frac{1}{n}\sum_{t=1}^{n}(real_t - previsto_t)^2} \quad (15)$$

4. Raiz Quadrada do Erro Quadrático Médio Normalizado

A raiz quadrada do erro quadrático médio normalizado, NRMSE (do inglês *Normalized Root-Mean-Squared Error*), é definido como sendo a relação da raiz quadrada do erro quadrático médio ao intervalo dos dados reais. O NRMSE é dado pela Equação 16, onde n é o número de predições geradas sendo analisadas, $real_t$ é o valor real da série no instante t, $previsto_t$ é o valor previsto da série no instante t, $real_{máx}$ é o maior valor dos dados reais e $real_{min}$ é o menor valor dos dados reais.

$$NRMSE = \frac{\sqrt{\frac{1}{n}\sum_{t=1}^{n}(real_t - previsto_t)^2}}{real_{máx} - real_{min}} \quad (16)$$

5. Coeficiente de Determinação

O coeficiente de determinação, R^2, é definido como a fração da variância total nos valores reais que pode ser explicada pelo modelo. O R^2 varia entre 0 e 1, indicando o quanto o modelo consegue explicar os valores reais. Quanto maior o valor de R^2, maior a concordância entre e os valores encontrados no modelo e os valores reais. O R^2 é dado pela Equação 18, onde n é o número de predições geradas sendo analisadas, $real_t$ é o valor real da série no instante t, $previsto_t$ é o valor previsto da série no instante t e $\overline{real}$ é a média dos valores reais analisados.

$$\overline{real} = \frac{1}{n}\sum_{t=1}^{n} real_t \quad (17)$$

$$R^2 = \frac{\sum_{t=1}^{n}(previsto_t - \overline{real})^2}{\sum_{t=1}^{n}(real_t - \overline{real})^2} \quad (18)$$

3. Metodologia

Todo o trabalho foi realizado utilizando a linguagem Python [Rossum, 1995]. Python foi criada no início dos anos 90 por Guido van Rossum, no Instituto Nacional de Pesquisa para Matemática e Ciência da Computação da Holanda (CWI). É uma linguagem de script interpretada, interativa e orientada a objetos [Python, 2019].

Para carregar os dados e criação dos datasets foi utilizada a biblioteca Pandas [McKinney 2010], para tratar as matrizes foi utilizada a biblioteca Numpy [Walt, Colbert e Varoquaux 2011], para separar os dados em dados de validação, treinamento e teste foi utilizada a biblioteca Scikit Learn [Pedregosa et al. 2011], para criar as camadas dos modelos foi utilizada a biblioteca Keras [Chollet et al. 2015] e para geração e visualização dos gráficos foi utilizada a biblioteca Matplotlib [Hunter 2007].

Neste trabalho, foi utilizado como base os registros históricos mensais de demanda nos períodos de ponta e fora de ponta, entre os anos de 2002 e 2015, da Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio). Os dados são baseados em medições feitas a cada 15 minutos.

a. Normalização e Tratamento de Dados

Para realizar as previsões, um conjunto de valores são usados como entradas para os modelos de previsão. Os dados foram, então, normalizados usando a Equação X, onde Z é o valor final normalizado, x_i é o valor original a ser normalizado, $min(x)$ é o valor mínimo do conjunto de dados a serem normalizados e $max(x)$ é o valor máximo do conjunto de dados a serem normalizados. Todos os valores das séries temporais são então transformados para a escala entre 0 e 1.

$$Z = \frac{x_i - min(x)}{max(x) - min(x)}$$

Antes do modelo receber os dados, foram aplicadas técnicas que visam melhorar o desempenho do sistema, removendo informações desnecessárias. Essas técnicas são úteis pois diminuem a necessidade de obtenção exaustiva de dados, contribuindo para o objetivo de generalização do modelo.

Os dados referentes ao ano de 2015 foram separados para serem utilizados como teste do modelo. Isto foi feito com o propósito de avaliar a aptidão de generalização do modelo, apresentando dados para treinamento isolando os dados de teste.

b. Método de previsão

O método de previsão foi dividido em etapas. Primeiro foram realizados os tratamentos dos dados das séries, o treinamento das redes neurais e por último ocorre a previsão dos dados de teste. As previsões são, por fim, avaliadas seguindo as métricas de desempenho.

Determinar o modelo neural é uma tarefa fundamental em qualquer aplicação. Diversas métricas são descritas na literatura que oferecem uma indicação das melhores estimativas dos parâmetros da rede, entretanto, uma metodologia sistemática para obtenção dos valores ótimos não é disponível. Não é possível construir um algoritmo de predição que seja capaz de prever qualquer tipo de série temporal. O que se pode é encontrar o modelo mais adequado para certos tipos de dados.

Tendo em vista o exposto, as configurações referentes ao número de neurônios, quantidade de dados para treinamento e validação, número de épocas e funções de ativação em cada camada, foram selecionados através de testes.

Para realização do treinamento, os dados foram separados em dados de treinamento e dados de validação. Foram utilizados os dados de 2002 até 2014 para treinamento da rede, para isto os dados foram separados em *mini batches* (foram separados em lotes de 10 dados sequenciais) e embaralhados, para que cada *mini batch* possa ser uma representação melhor do conjunto inteiro, e depois foram separados aleatoriamente, sendo 80% dos dados para treinamento e 20% para validação.

Para avaliação do número de épocas a serem submetidas para treinamento das redes foram feitos testes e adotou-se 300 (trezentas) épocas como padrão.

Para treinar o modelo foram utilizados 256 neurônios, foi utilizado 256 como batch-size (define o número de amostras que serão propagadas através da rede a cada iteração), a métrica de avaliação escolhida

para treinar o modelo foi MSE, a função de ativação escolhida foi linear. A rede foi treinada utilizando o otimizador Adam, que é um algoritmo para otimizar funções estocásticas, baseado em estimativas de baixa ordem.

Foi configurado *early stopping*, mecanismo que para o treinamento quando uma variável monitorada tiver parado de melhorar, para evitar que ocorresse *overfitting*. *Overfitting* é um dos principais problemas em redes com múltiplas camadas e da aprendizagem de máquina em geral. O *overfitting* se dá quando a rede começa a ficar extremamente especializada no conjunto de dados de entrada no treinamento e tenta replicá-los sempre. Dessa forma, a rede funciona muito bem para dados similares aos utilizados no treinamento, mas não tão bem para entradas com comportamentos diferentes dos já conhecidos.

4. Resultados

O modelo de previsão criado, foi utilizado para a previsão dos valores de demanda de energia elétrica do ano de 2015, e com isso, foi possível comparar com os dados reais do ano de 2015.

Após o treinamento e validação do modelo, foi possível obter em ambos os valores de perda e mensuração do erro através da métrica do MSE. A demonstração gráfica dos valores de perda pode ser observada através da Figura 4. Já os valores do erro MSE podem ser observados na Figura 5. Ressaltando que em ambos os gráficos se encontram destacados os respectivos valores de treino e validação.

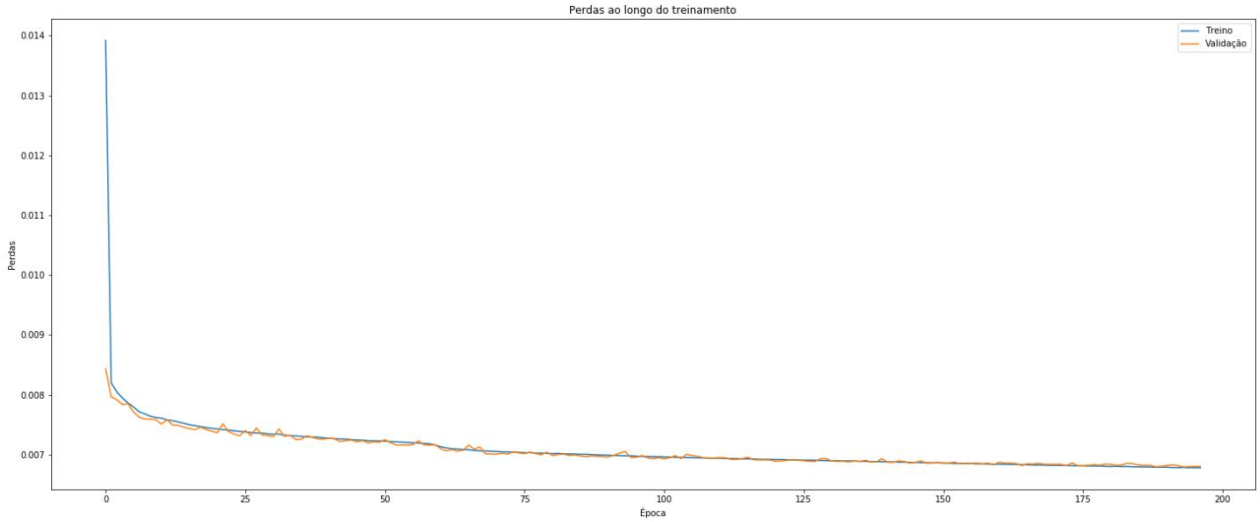

Figura 4 – Gráfico das perdas ao longo do tempo

Analisando o gráfico, podemos verificar que perdas foram diminuindo o que significa que o modelo estava aprendendo melhor e evoluindo ao longo das épocas. Isso também foi observado no gráfico da Figura 5, confirmando que durante o treinamento, o modelo foi melhorando sua precisão ao longo do tempo.

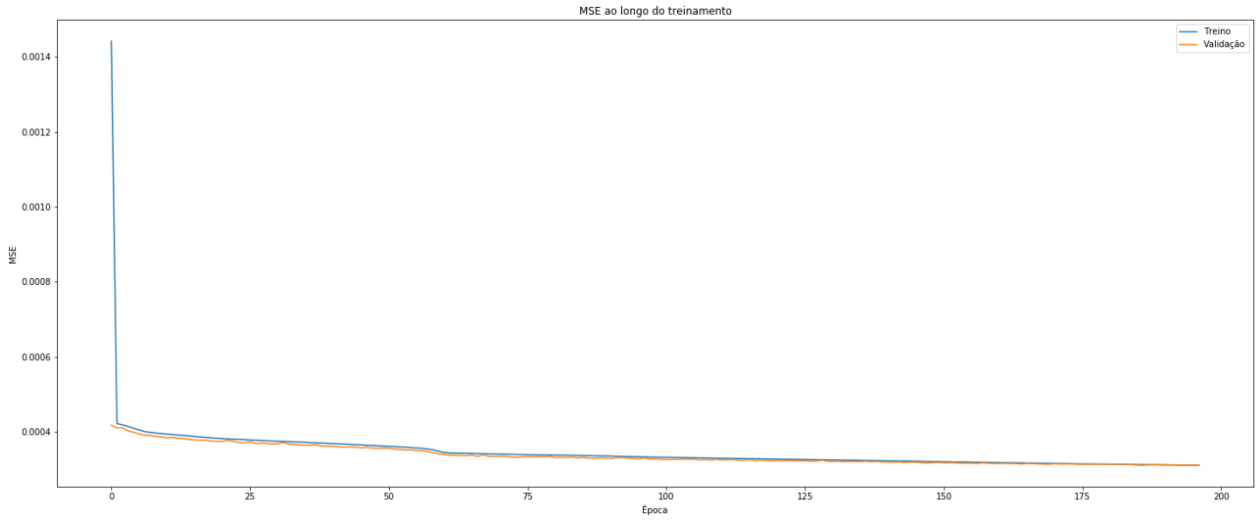

Figura 5 – Gráfico do erro MSE ao longo do tempo

Por possuir muitos valores, ficaria de difícil entendimento se todos os dados de 2015, dados reais e previstos, fossem exibidos no gráfico. Sendo assim, a Figura 6 apresenta o gráfico comparando somente os valores previstos e reais do mês de dezembro de 2015.

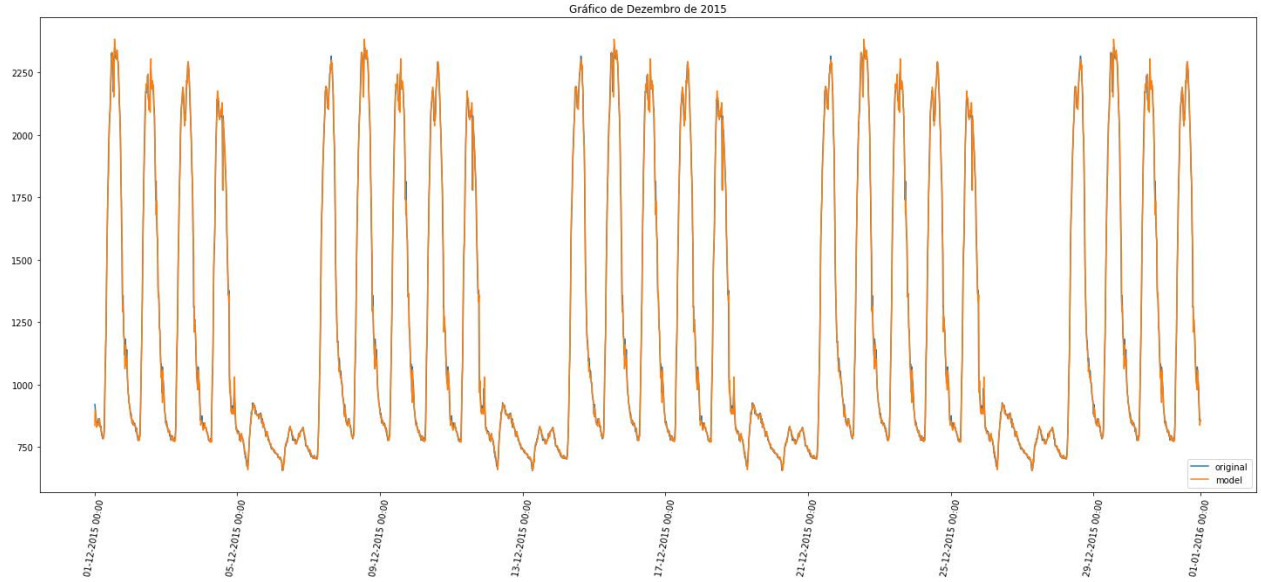

Figura 6 – Gráfico comparando os valores previstos e reais do mês de Dezembro de 2015

Podemos ver que os dados previstos estão próximos aos dados reais. Indicando que o modelo teve sucesso para prever os dados do ano de 2015.

A Figura 7 apresenta um gráfico comparando o os valores previstos e reais do dia 31 de dezembro de 2015, para que se possa visualizar melhor o desempenho do modelo.

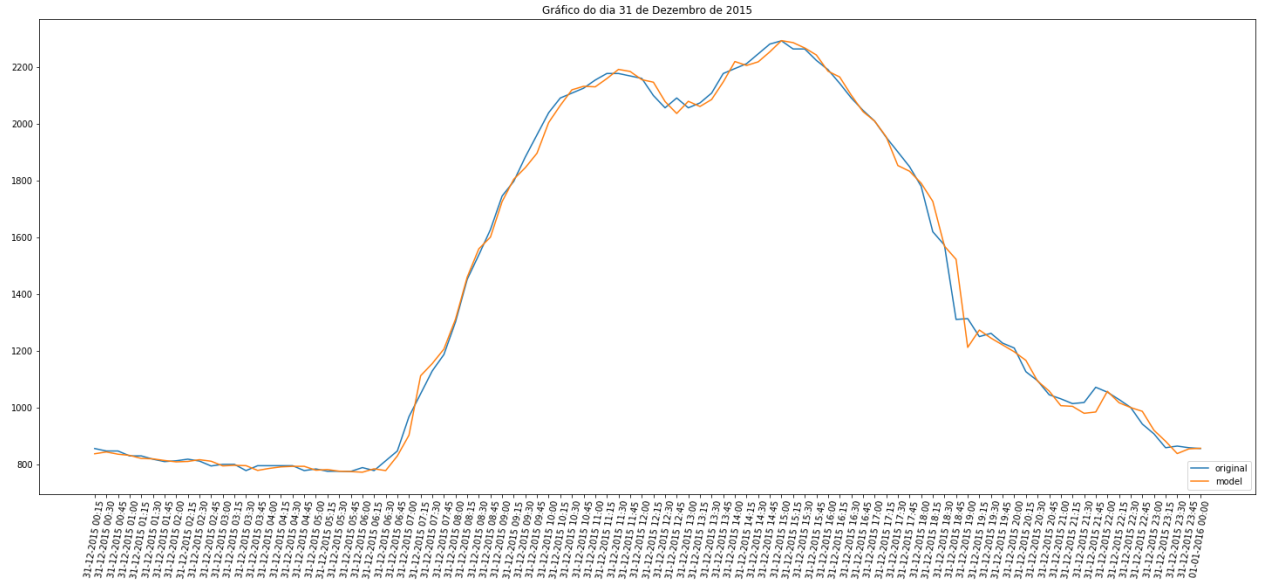

Figura 7 – Gráfico comparando os valores previstos e reais do dia 31 de Dezembro de 2015

Os erros encontrados comparando os dados reais com os dados previstos pelo modelo foram:

- MAE: 17.624369284724185;
- MSE: 844.6244537780782;
- RMSE: 29.062423398231576;
- NRMSE: 0.7249294935952002%;
- R2: 0.9983372036407177.

Analisando os erros encontrados, podemos confirmar que o modelo obteve sucesso e conseguiu prever dados próximos aos dados reais. Isso pode ser visualizado, por exemplo, analisando o coeficiente R^2, onde quanto mais próximo de 1 for o valor de R^2, maior a concordância entre e os valores encontrados na previsão e os valores reais.

5. Conclusão e trabalhos futuros

Conforme o objetivo principal desse trabalho de desenvolver um modelo de redes neurais LSTM e, após obtidos os resultados de estimação e predição dos modelos e avaliar os resultados, podemos concluir que o modelo apresentou resultados muito próximos dos reais na métrica adotada para avaliação de precisão. Com isso podemos afirmar que o modelo de redes neurais LSTM se mostrou eficaz, na métrica de precisão e configurações de rede escolhidas, para prever os dados deste trabalho.

Com isso, sugere-se para próximos trabalhos, avaliar outras métricas de precisão, assim como, utilizar outras funções de ativação e configurações de rede.

Referências

ANEEL. *Condições Gerais de Fornecimento de Energia Elétrica*, Disponível em: <http://www.aneel.gov.br/documents/656835/14876406/2016_ResolucaoNormativa4142010.pdf>, 2016. Acesso em: 20 de abril de 2019.

BISHOP, C. M. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. 1 ed. Springer: New York, 2006.

CASTRO, R. S.; BARROS, M.; LAZO, J. G. L.; TUPAC, Y.; MIRANDA, C. V. C. *Previsão de dados de alta freqüência para carga elétrica usando uma combinação entre redes neurais e holt-winters com dois ciclos*, 1997 [https://doi.org/10.13140/2.1.3646.6246].

CHOLLET, F. et al. *Keras*. <https://keras.io>, 2015.

DAMETTO, R. C. *Estudo da aplicação de redes neurais artificiais para predição de séries temporais financeiras*. Unesp. Faculdade de Engenharia, Bauru, 2018.

ELMAN, J. L. *Finding structure in time*. Cognitive Science, vol. 14, p. 179-211, 1990.

FONSECA, J. P. C. *FPGA implementation of a LSTM Neural Network*. FEUP, 2016.

HAYKIN, S. S. *Redes Neurais: princípios e práticas*. 2 ed. Porto Alegre: Bookman, 2001.

HUNTER, J. D. Matplotlib: A 2d graphics environment. *Computing In Science & Engineering*, IEEE COMPUTER SOC, v. 9, n. 3, p. 90–95, 2007.

HOCHREITER, S.; SCHMIDHUBER, J. *Long Short-Term Memory, Neural Computation*, v.9 n.8, p.1735-1780, 1997 [https://doi.org/10.1162/neco.1997.9.8.1735].

MCKINNEY, W. Data structures for statistical computing in python. In: WALT, S. van der; MILLMAN, J. (Ed.). *Proceedings of the 9th Python in Science Conference*. [S.l.: s.n.], 2010. p. 51 – 56.

MITCHELL, T. M. *Machine Learning*. McGraw Hill, 1997.

MORETTIN, P. A.; BUSSAB, W. O. *Estatística Básica*. 6 ed. São Paulo: Saraiva, 2010.

MORETTIN, P. A.; TOLOI, C. M. C. *Análise de séries temporais*. 2 ed. São Paulo: Egard Blucher, 2006.

NORVIG, P.; RUSSEL, S. *Inteligência artificial*. 3 ed. São Paulo: Elsevier, 2013.

OLAH, C. *Understanding lstm networks*. Disponível em: <http://colah.github.io/posts/2015-08-

Understanding-LSTMs/>, 2015. Acesso em: 18 de abril de 2019.

PEDREGOSA, F. et al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, v. 12, p. 2825–2830, 2011.

PROCEL. Manual de Tarifação da Energia Elétrica, Disponível em: <http://www.mme.gov.br/documents/10584/1985241/Manual de Tarif En El - Procel_EPP - Agosto-2011.pdf>, 2011. Acesso em: 20 de abril de 2019.

PYTHON. *Python*. Disponível em: <http://www.python.org/>. Acesso em: 05 de junho de 2019.

ROSSUM, G. *Python tutorial, Technical Report CS-R9526*, Centrum voor Wiskunde en Informatica (CWI), Amsterdam, 1995.

WALT, S. van der; COLBERT, S. C.; VAROQUAUX, G. The numpy array: A structure for efficient numerical computation. *Computing in Science Engineering*, v. 13, n. 2, p. 22–30, March 2011. ISSN 1521-9615.

Apêndice

a. Código completo em Python do Modelo

```
import pandas as pd
import numpy as np
from tqdm import tqdm
from sklearn.preprocessing import MinMaxScaler
import keras
from keras.models import Sequential
from keras.layers import Dense, LSTM
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error
from math import sqrt
from matplotlib import pyplot
from keras.callbacks import EarlyStopping
from datetime import datetime, timedelta
import datetime as dt
import matplotlib.dates as mdates

filepath = r'dados_cardoso.csv'

csv_data = pd.read_csv(filepath, delimiter=',', header=None).values

for column in range(0, csv_data.shape[1], 5):
    if column == 0:
        original_data = csv_data[:, column+3:column+4]
    else:
        original_data = np.concatenate([original_data, csv_data[:, column+3:co-
lumn+4]], axis=0)

original_data_max = max(original_data)[0]
original_data_min= min(original_data)[0]

original_data_last_year = original_data[-35040:,]
original_data = original_data[:-35040,]

scaler = MinMaxScaler().fit(original_data)

original_data = scaler.transform(original_data)
original_data_last_year = scaler.transform(original_data_last_year)

# Parâmetros variáveis:
window = 1
batch_size = 10
split = 0.8

data = np.zeros([0, original_data.shape[1] * (window + 1)])
data_last_year = np.zeros([0, original_data_last_year.shape[1] * (window + 1)])

for i in tqdm(range(window, original_data.shape[0])):
    columns = np.zeros([1, 0])
    for j in range(i - window, i + 1):
        columns = np.concatenate([columns, original_data[j, :].reshape(1, -1)], axis=1)
    data = np.concatenate([data, columns], axis=0)

for i in tqdm(range(window, original_data_last_year.shape[0])):
    columns = np.zeros([1, 0])
    for j in range(i - window, i + 1):
        columns = np.concatenate([columns, original_data_last_year[j, :].reshape(1, -
1)], axis=1)
    data_last_year = np.concatenate([data_last_year, columns], axis=0)

```

```
data_lstm = np.zeros([0, batch_size, data.shape[1]])

for i in tqdm(range(0, data.shape[0] - batch_size)):
    temp = data[i:i + batch_size, :]
    data_lstm = np.concatenate([data_lstm, temp.reshape(1, temp.shape[0], temp.shape[1])])

p = np.random.permutation(data_lstm.shape[0])
data_lstm = data_lstm[p]

split_1 = round(split * data_lstm.shape[0])

train, val = data_lstm[:split_1, :, :], data_lstm[split_1:, :, :]

train_x, train_y = train[:, :, :-1], train[:, :, -1:]
val_x, val_y = val[:, :, :-1], val[:, :, -1:]

data_lstm_last_year = data_last_year.reshape(1, data_last_year.shape[0], data_last_year.shape[1])
test_x, test_y = data_lstm_last_year[:, :, :-1], data_lstm_last_year[:, :, -1:]

# Parâmetros variáveis:
units = 256
epochs = 300
batch_size = 256

early_stopping_callback = EarlyStopping(monitor='val_mean_squared_error', patience=10, mode='min')

model = Sequential()
model.add(LSTM(units, input_shape=(None, train_x.shape[2]), return_sequences=True))
model.add(Dense(train_y.shape[2],activation='linear'))
model.compile(loss='mae', metrics=['mse'], optimizer='adam')
print(model.summary())

history = model.fit(train_x, train_y, validation_data=(val_x, val_y), epochs=epochs, batch_size=batch_size, verbose=2, shuffle=True, callbacks=[early_stopping_callback])

fig, ax = pyplot.subplots(figsize=(25, 10))
pyplot.plot(history.history['mean_squared_error'])
pyplot.plot(history.history['val_mean_squared_error'])
pyplot.title('MSE ao longo do treinamento')
pyplot.ylabel('MSE')
pyplot.xlabel('Época')
pyplot.legend(['Treino', 'Validação'], loc='best')
pyplot.savefig(r'C:\TCC\img\fig01.png', bbox_inches='tight')
pyplot.clf

fig, ax = pyplot.subplots(figsize=(25, 10))
pyplot.plot(history.history['loss'])
pyplot.plot(history.history['val_loss'])
pyplot.title('Perdas ao longo do treinamento')
pyplot.ylabel('Perdas')
pyplot.xlabel('Época')
pyplot.legend(['Treino', 'Validação'], loc='best')
pyplot.savefig(r'C:\TCC\img\fig02.png', bbox_inches='tight')

# Trazendo os dados para o intervalo original
y_original = test_y.reshape(test_y.shape[0] * test_y.shape[1], test_y.shape[2])  # Voltando para 2D

```

```
113. y_model = model.predict(test_x)  # Previsão da rede
114. y_model = y_model.reshape(y_model.shape[0] * y_model.shape[1], y_model.shape[2])  # Vol-
     tando para 2D
115.
116. y_original = scaler.inverse_transform(y_original)
117. y_model = scaler.inverse_transform(y_model)
118.
119. # Calculando erro
120. mae = list()
121. for i in range(y_original.shape[1]):
122.     mae.append(mean_absolute_error(y_original[:, i], y_model[:, i]))
123.     print('MAE: {}'.format(mae[i]))
124.
125. mse = list()
126. for i in range(y_original.shape[1]):
127.     mse.append(mean_squared_error(y_original[:, i], y_model[:, i]))
128.     print('MSE: {}'.format(mse[i]))
129.
130. rmse = list()
131. for i in range(y_original.shape[1]):
132.     rmse.append(sqrt(mean_squared_error(y_original[:, i], y_model[:, i])))
133.     print('RMSE: {}'.format(rmse[i]))
134.
135. nrmse = list()
136. for i in range(y_original.shape[1]):
137.     nrmse.append(
138.         (sqrt(mean_squared_error(y_original[:, i], y_model[:, i]))) * 100 / (origi-
     nal_data_max - original_data_min))
139.     print('NRMSE: {}%'.format(nrmse[i]))
140.
141. r2 = list()
142. for i in range(y_original.shape[1]):
143.     r2.append(r2_score(y_original[:, i], y_model[:, i]))
144.     print('R2: {}'.format(r2[i]))
145.
146. def datetime_range(start, end, delta):
147.     current = start
148.     while current < end:
149.         yield current
150.         current += delta
151.
152. dates = [dt.strftime('%d-%m-%Y %H:%M') for dt in
153.         datetime_range(datetime(2015, 12, 31, 0, 15), datetime(2016, 1, 1, 0,15),
154.                        timedelta(minutes=15))]
155.
156. fig, ax = pyplot.subplots(figsize=(25, 10))
157. ax.plot(dates, y_original[-96:, 0], label='original')
158. ax.plot(dates, y_model[-96:, 0], label='model')
159. ax.legend(loc='lower right')
160. pyplot.title('%s' % 'Gráfico do dia 31 de Dezembro de 2015')
161. pyplot.xticks(rotation=80)
162. pyplot.savefig(r'C:\TCC\img\fig03.png', bbox_inches='tight')
163.
164. dates = [dt for dt in
165.         datetime_range(datetime(2015, 12, 1, 0, 15), datetime(2016, 1, 1, 0,15),
166.                        timedelta(minutes=15))]
167.
168. fig, ax = pyplot.subplots(figsize=(25, 10))
169. ax.plot(dates, y_original[-2976:, 0], label='original')
170. ax.plot(dates, y_model[-2976:, 0], label='model')
```

```
171.  ax.legend(loc='lower right')
172.  myFmt = mdates.DateFormatter('%d-%m-%Y %H:%M')
173.  ax.xaxis.set_major_formatter(myFmt)
174.  pyplot.title('%s' % 'Gráfico de Dezembro de 2015')
175.  pyplot.xticks(rotation=80)
176.  pyplot.savefig(r'C:\TCC\img\fig04.png', bbox_inches='tight')
```