UNIVERSIDADE FEDERAL DA GRANDE DOURADOS

MATHEUS DE MATTOS PEREIRA

APRENDIZADO PROFUNDO: REDES LSTM

DOURADOS

2017

MATHEUS DE MATTOS PEREIRA

APRENDIZADO PROFUNDO: REDES LSTM

Trabalho de Conclusão de Curso de graduação apresentado a Faculdade Ciências Exatas e Tecnologia, como requisito para obtenção do título de Bacharel em Sistemas de Informação.

Orientador: Prof. Ms Anderson Bessa da Costa

DOURADOS

2017

MATHEUS DE MATTOS PEREIRA

APRENDIZADO PROFUNDO: REDES LSTM

Trabalho de Conclusão de Curso aprovado como requisito para obtenção do título de Bacharel em Sistemas de Informação na Universidade Federal da Grande Dourados, pela comissão formada por:

_____

Orientador: Prof. Ms Anderson Bessa da Costa

FACET - UFGD

_____

Prof. Dr. Joinvile Batista Junior

FACET – UFGD

_____

Profª. Drª. Valguima Victória Viana Aguiar Odakura

FACET – UFGD

Dourados, 24 de março de 2017.

RESUMO

Desde a última década vem ocorrendo um avanço muito rápido na pesquisa de redes neurais. Devido ao sucesso de novas técnicas, grandes resultados em competições de diversas áreas, e seu crescente uso na indústria, essa área ganhou um nome próprio, que vêm se popularizando cada vez mais nos últimos anos: aprendizado profundo (*deep learning)*. Este trabalho busca oferecer uma breve introdução ao aprendizado profundo, explorando suas técnicas e explicando o que de fato torna uma rede neural profunda. Após essa breve introdução, será discutido mais detalhadamente uma das técnicas responsáveis pelo grande sucesso do aprendizado profundo recentemente, as redes *Long Short Term Memory* (LSTM), e como essa técnica pode ser aplicada para a predição de séries temporais. A partir dos resultados obtidos de uma rede LSTM treinada para a predição de séries temporais em quatro conjuntos de dados, a rede LSTM se saiu melhor em três desses conjuntos testados em comparação com outros métodos, o que mostra sua eficácia contra outras técnicas já consolidadas para esse tipo de problemas, como redes neurais artificiais e ARIMA.

Palavras-chave: Aprendizado profundo. Redes neurais. Aprendizado de máquina. Séries temporais.

ABSTRACT

Over the last decade, there has been a very rapid advance in neural networks research. Due to the success of new techniques, big results in competitions in several fields, and its increasing use in industry, this field gained its own name, which has become a buzzword over the past years: deep learning. This work seeks to offer a brief introduction on deep learning, exploring its techniques and explaining what constitutes a deep neural network. After this brief introduction, it will be discussed in more detail one of the techniques responsible for the recent success of deep learning, the Long Short Term Memory (LSTM) networks, and how this technique can be applied to time series prediction. From the results obtained training a LSTM network for time series prediction using four datasets, the LSTM network performed better in three of these sets compared to other methods, which shows their effectiveness against other already consolidated techniques for this kind of problem, such as artificial neural networks and ARIMA.

Keywords: Deep learning. Neural networks. Machine learning. Time series.

SUMÁRIO

1 INTRODUÇÃO

O campo de aprendizado de máquina busca resolver questões em como construir programas de computadores que automaticamente melhoram com experiência (MITCHELL, 1997). É um campo em constante evolução, explorando novas técnicas para solucionar diversos problemas computacionais existentes, tais como: reconhecimento de imagens (SIMONYAN; ZISSERMAN, 2014), reconhecimento de voz (HINTON et al., 2012) e classificação de imagens (ZHONG; LIU; LIU, 2011). Uma das abordagens recentes que vem recebendo grande atenção por parte dos pesquisadores e da indústria nos últimos anos é chamada de aprendizado profundo (*deep learning*, em inglês).

Aprendizado profundo se refere a uma classe de técnicas de aprendizado de máquina, desenvolvidas em grande escala desde 2006, onde vários estágios de processamento de informação não-linear são usados para classificação de padrões e aprendizado de características (DENG et al., 2013). Aprendizado profundo vem de forma sistemática superando muitas das melhores técnicas para diversos problemas.

A área de visão computacional é uma das áreas mais beneficiadas pelo aprendizado profundo. Sistemas de visão computacional que usam aprendizado profundo chegaram a vencer humanos em muitas tarefas de reconhecimento de imagens (HE, 2015). Outra recente aplicação do aprendizado profundo ocorreu na área de tradução de línguas na Google. O *Google Translate*, serviço de tradução online da *Google*, recentemente mudou sua técnica de tradução para uma abordagem baseada no aprendizado profundo, mais especificamente, com uma técnica conhecida como LSTM. De acordo com a Google, o novo algoritmo reduziu os erros em 60% em comparação ao sistema anterior (WU et al., 2016).

Originalmente, o conceito de aprendizado profundo surgiu a partir da pesquisa em redes neurais artificiais (YU; DENG, 2011). O exemplo mais simples de um modelo de aprendizado profundo é conhecido como *feedforward deep network* ou perceptron com múltiplas camadas (*Multilayer Perceptron* - MLP).

O objetivo deste trabalho consiste em realizar um estudo sobre o aprendizado profundo, buscando seu histórico, motivação e o contexto em que foi criado. Utilizando destes conceitos, estudaremos ferramentas existentes que possibilitam aplicar em um problema real, fazendo uma breve comparação a outras técnicas e modelos já existentes.

2 REVISÃO BIBLIOGRÁFICA

Redes Neurais Artificiais (RNA), comumente referenciada apenas como "redes neurais", foram motivadas desde a sua concepção pelo reconhecimento de que o cérebro humano computa de uma forma totalmente diferente da forma como é realizada a computação por computadores digitais (HAYKIN, 1994).

O primeiro modelo computacional de redes neurais foi desenvolvido por Rosenblatt (1958). O perceptron desenvolvido por Rosenblatt consiste de uma rede neural com uma única camada e usa o modelo McCulloch–Pitts de um neurônio, conforme apresentado na Figura 1.

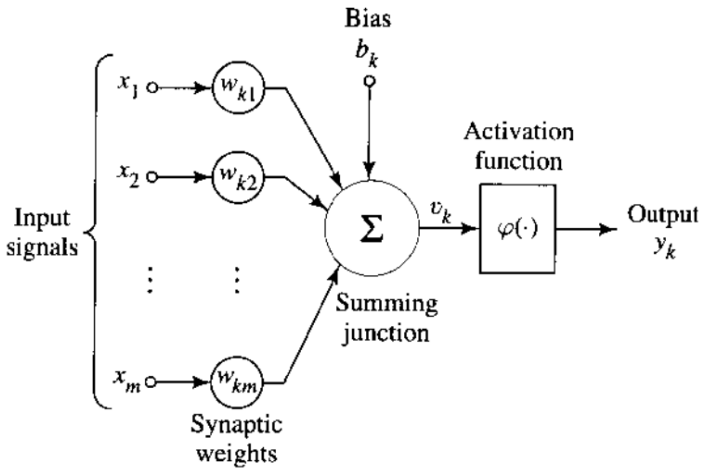


Figura 1: Modelo de neurônio não linear. Fonte: HAYKIN, 1994.

Este modelo possui um conjunto de sinais (entradas) $x_1...x_m$, sendo associada cada entrada a um neurônio $k_1...k_m$, com cada conexão entre a entrada e um neurônio sendo representada por um peso w. Todos os neurônios de entrada (juntamente com seu peso) se unem em um somatório; a operação que estamos descrevendo aqui é um combinador linear. A função de ativação é utilizada para limitar a amplitude da saída de um neurônio e gerar uma saída que tipicamente varia entre -1 e 1 ou 0 e 1 dependendo da função de ativação usada.

Apesar de ser uma grande inovação na época, o perceptron criado por Rosenblatt tinha

uma importante limitação: o modelo era capaz de resolver apenas modelos linearmente separáveis. Modelos linearmente separáveis são modelos que podem ser separados por uma reta (em um sistema de coordenadas 2D) ou por um plano (sistema multidimensional). A Figura 2 apresenta um exemplo de um problema linearmente separável e de um problema não linearmente separável.

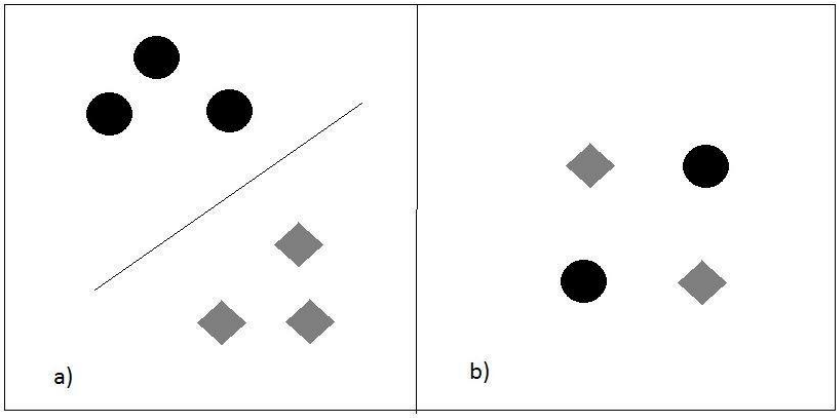


Figura 2: a) Modelo linearmente separável. b) Modelo não linearmente separável (XOR). Fonte: Elaborada pelo autor.

A primeira crítica do perceptron criado por Rosenblatt foi apresentada por Minsky e Selfridge (1961). Minsky e Selfridge afirmaram que o modelo não era capaz de resolver um problema simples não linearmente separável, como a porta lógica XOR, logo, era impensável utilizar o perceptron para fazer abstrações gerais (Haykin, 1994).

Uma crítica ainda mais forte foi escrita no livro Perceptrons, por Minsky e Papert (1969). Neste livro, Minsky e Papert provam a partir de funções matemáticas que o perceptron de Rosenblatt é incapaz de fazer qualquer generalização global com base nos exemplos aprendidos.

Esse livro foi responsável por trazer dúvidas sobre o poder computacional não apenas do perceptron, mas de redes neurais em geral até a metade dos anos 80 (Haykin, 1994), causando uma diminuição na popularidade das redes neurais na comunidade acadêmica até então.

Um dos trabalhos que mais contribuíram com a volta da popularização das redes neurais na década de 80 foi publicado por Rumelhart (1986), utilizando um algoritmo supervisionado conhecido como retropropagação, ou *backpropagation* em inglês, que demonstrou a importância de representações internas em redes neurais, conhecidas como

camadas escondidas. *Backpropagation* é um método para treinar redes neurais artificiais e usado em conjunção com métodos de otimização, tal como o gradiente descendente. Com esta técnica é possível treinar redes que são capazes de generalizar qualquer tipo de função, sendo possível resolver problemas não lineares. A figura 3 contém um exemplo de uma rede neural com uma camada escondida.

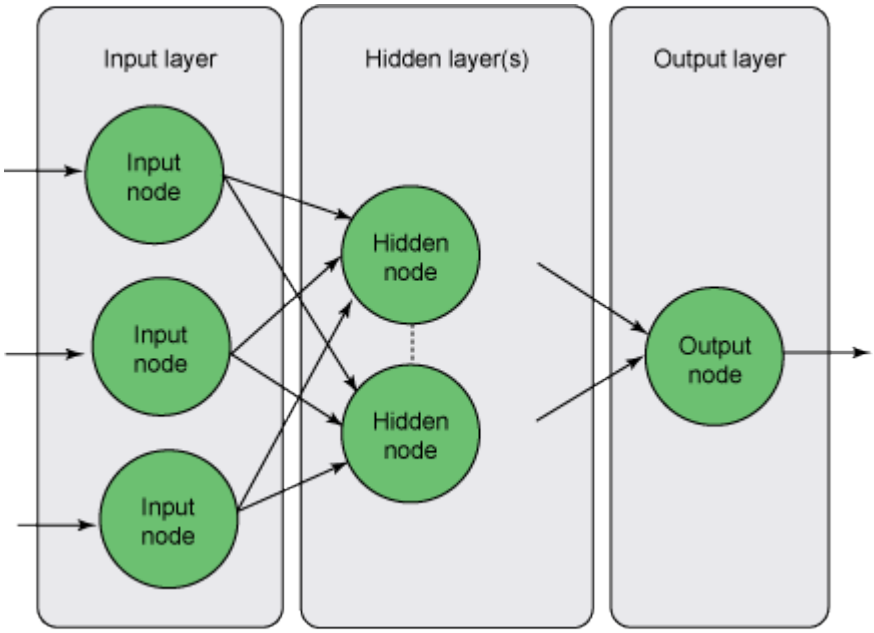


Figura 3: Uma rede neural com uma camada de entrada com três entradas, uma camada intermediária com dois neurônios escondidos e uma camada de saída com apenas uma saída.

A ideia principal deste algoritmo consiste em treinar uma rede neural com os dados de entrada e saída de um conjunto de dados, e calcular o erro (a diferença entre o resultado esperado e o resultado gerado pela rede neural a partir de pesos gerados aleatoriamente no início) para cada iteração. Assim que esse erro é encontrado, os pesos da rede neural são ajustados com o objetivo de diminuir este erro, e um novo erro é calculado para a próxima iteração. O algoritmo converge quando os ajustes dos pesos na rede não afetam mais o erro geral, estabilizando assim o erro da rede.

Até o final da década de 80, a maior parte das aplicações de redes neurais focaram em redes neurais com poucas camadas escondidas. Camadas adicionais não ofereciam melhores resultados (YU; DENG, 2011). Isto acontecia devido a um problema conhecido como *vanishing gradient problem*.

Para exemplificar esse problema, durante o treino da rede neural com o algoritmo *backpropagation*, cada peso da rede é atualizado de acordo com a mudança nos pesos em relação ao erro em cada iteração do treinamento, esse valor é conhecimento como a gradiente. Como o *backpropagation* faz o cálculo dessas gradientes usando a regra da cadeia, a multiplicação dessas gradientes, que geralmente são números menores que 1, se torna um número cada vez menor, fazendo com que as primeiras camadas da rede tenham uma gradiente muito pequena, o que consequentemente implica que pouco está sendo aprendido por essas camadas.

Embora não fosse possível o treinamento correto de redes profundas com o *backpropagation* nessa época, existia uma intuição de que redes profundas fossem capazes de aprender e representar modelos mais complexos. Existe uma analogia com projeto de circuitos, onde circuitos profundos tornam o processo de projetar mais fácil. Assim, intrinsecamente é possível afirmar que circuitos profundos são mais poderosos que circuitos rasos (NIELSEN, 2015).

3 APRENDIZADO PROFUNDO

O aprendizado profundo se refere a uma classe de técnicas de aprendizado de máquina. Essa área pode ser considerada como uma intersecção entre as áreas de redes neurais, modelagem gráfica, otimização, reconhecimento de padrões e processamento de sinais. Alguns dos importantes motivos para a popularidade do aprendizado profundo atualmente são o aumento drástico de processamento nos computadores com a utilização de placas de vídeo (*Graphical Processing Units* - GPUs), um custo menor de *hardware* e os recentes avanços na pesquisa de aprendizado de máquina e processamento de informações (DENG et al., 2013).

Esse drástico aumento na velocidade do treinamento com o uso de GPUs se deve ao fato do treinamento de redes neurais profundas usarem diversas operações matemáticas em grandes matrizes para o treinamento da rede, o que leva bastante tempo para se computar em um processador comum, mas que são computadas rapidamente por GPUs.

3.1 CONCEITO

A Figura 4 apresenta o contexto do aprendizado profundo com relação às áreas de Inteligência Artificial e Aprendizado de Máquina. Como apresentado, a área de Aprendizado Profundo está contida dentro da área de Aprendizado de Máquina. Esta área por sua vez dentro da área de Inteligência Artificial.

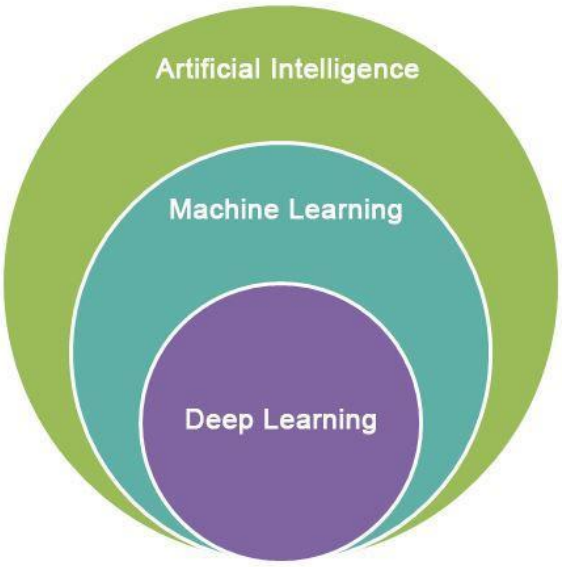


Figura 4: O aprendizado profundo (*deep learning)* está inserido na área de aprendizado de máquina (*machine learning)* e na área geral de inteligência artificial (*artificial intelligence)*. Fonte[1]: https://www.codeproject.com.

De uma maneira geral, aprendizado profundo é um nome mais fácil de se entender do que uma Rede Neural Artificial, e que se popularizou bastante recentemente. O termo "profundo" se refere a profundidade da rede, ou seja, a sua quantidade de camadas. Uma rede neural artificial geralmente contém apenas uma ou duas camadas escondidas, enquanto uma rede profunda pode contar com um número muito maior de camadas, dependendo de sua arquitetura e aplicação.

A primeira camada de uma rede neural é conhecida como a camada de entrada. Cada nó nessa camada recebe um valor de entrada, e então passa sua saída como a entrada para cada nó na seguinte camada. Não existem conexões entre nós da mesma camada, e a última camada produz a saída.

As camadas intermediárias são conhecidas como "camadas escondidas". Na Figura 5 é possível visualizar e a forma como se organizam.

[1] Disponível em: <https://www.codeproject.com/Articles/1170474/Learning-Machine-Learning-Part-Application>. Acesso em mar. 2017.

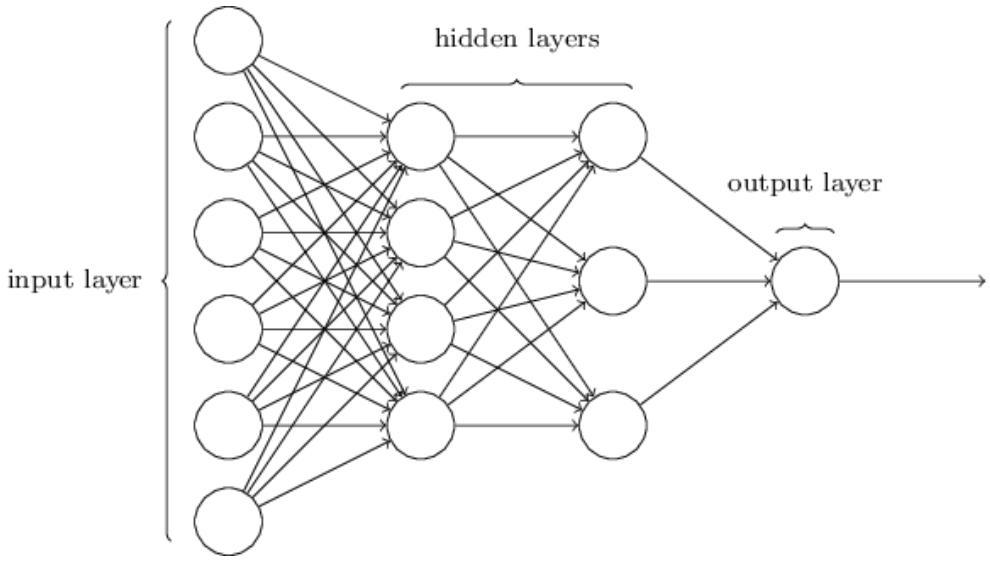


Figura 5: Uma rede neural com duas camadas escondidas (*hidden layers)*. A primeira camada (*input layer)* se conecta com as camadas escondidas, que por sua vez produz um resultado na camada de saída (*output layer*). Fonte: NIELSEN, 2015.

Por meio de um conjunto de técnicas de aprendizado profundo foi possível superar a limitação no número de camadas de uma rede, ajudando a resolver o *vanishing gradient problem* e a treinar redes com mais camadas intermediárias. Algumas das técnicas que contribuíram para esse avanço são abordadas a seguir.

3.1.2 ReLUs

ReLU (*Rectified Linear Unit)* é uma função de ativação não linear criada por Hinton e Nair (2010) que segue a seguinte equação:

$$f(x) = \text{Max}(0,x)$$

onde Max (0,x) é uma função que retorna 0 caso x < 0, e x caso> 0.

Essa função é bastante usada pelo fato de ser calculada rapidamente ao contrário de outras funções de ativação não lineares mais convencionais como a função sigmóide, representada pela equação: $\frac{1}{1+e^{-x}}$ e que retorna um valor entre 0 e 1, ou a função tangente, representada por □□□*h* (□)e que retorna um valor entre -1 a 1. Pelo fato de ser tão simples de se computar uma ReLU, o tempo de treinamento é reduzido, o que facilita o treinamento

de redes com várias camadas.

3.2 Dropout

O *Dropout* (SRIVASTAVA et al., 2014) é uma técnica usada para aumentar a velocidade de treinamento de redes com várias camadas utilizando um método para aleatoriamente excluir neurônios e suas conexões da rede neural durante o treinamento. Essa técnica previne um fenômeno conhecido como *overfitting*, onde a rede neural treinada aprende um problema específico muito bem, mas não consegue generalizar esse aprendizado para outros problemas. A figura 6 contém um exemplo de como funciona o *Dropout*.

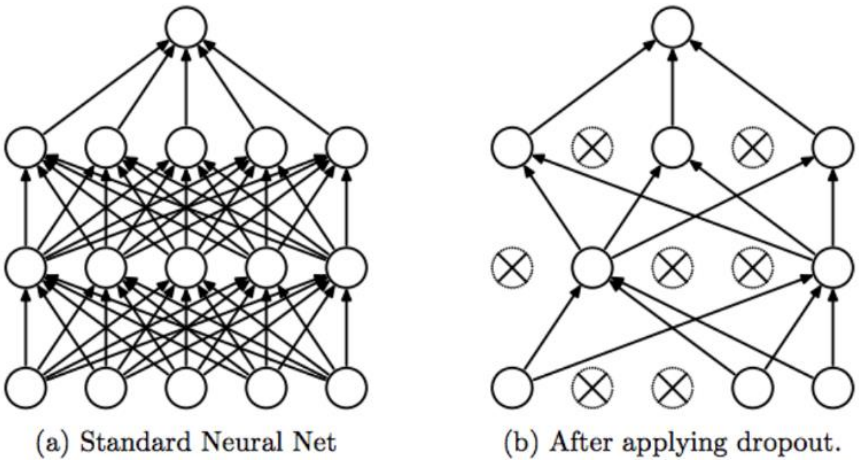


Figura 6: a) Uma rede neural comum com duas camadas escondidas. b) uma rede neural com duas camadas escondidas após o *dropout*. Fonte: SRIVASTAVA et al, 2014.

Com uma rede mais profunda é possível aprender e representar modelos mais complexos. Neste trabalho abordaremos um tipo específico de rede profunda: as redes *Long Short-Term Memory* (LSTM), que correspondem a um tipo especial de uma rede neural recorrente.

3.2 REDES NEURAIS RECORRENTES

Redes Neurais Recorrentes (*Recurrent Neural Networks* - RNNs, em inglês) são uma classe de redes neurais artificiais onde conexões entre as unidades formam um ciclo direcionado. São modelos de redes neurais bastante populares e que vem mostrando bastante potencial em diversas áreas de aprendizado de máquina como processamento de linguagem natural, legenda automática de imagens, tradução e reconhecimento de voz (BRITZ, 2015).

A principal ideia por trás das RNNs é de fazer o uso de informações sequenciais. Em uma rede neural tradicional se assume que todas as entradas e saídas são independentes uma das outras, entretanto, para muitas tarefas essa não é a melhor abordagem. Por exemplo, para fazer a predição da próxima palavra em uma sentença é necessário saber as palavras que antecedem a nova palavra. RNNs são ditas *recorrentes* pois realizam a mesma tarefa para cada elemento de uma sentença, com a saída sendo dependente dos resultados anteriores, a Figura 7 ilustra uma comparação entre uma rede *feedforward* e uma rede recorrente.

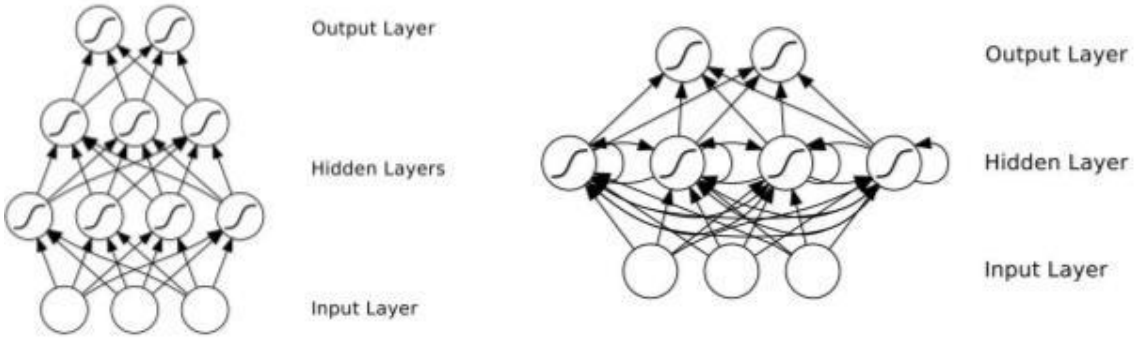


Figura 7: Comparação da arquitetura de uma rede *feedforward* e uma rede recorrente. As camadas escondidas também trocam informações na mesma camada, diferentemente de uma rede *feedforward*. Fonte: Sapunov, Grigory. Disponível em: <https://www.slideshare.net/grigorysapunov/multidimensional-rnn>.

Uma maneira simples de se pensar sobre RNNs é que elas têm uma “memória” que captura as informações sobre o que foi calculado até o momento.

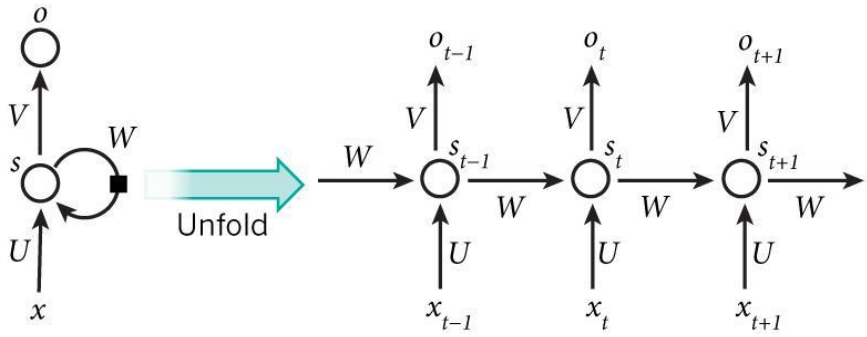


Figura 8: Uma rede neural recorrente e o processo do desdobramento (unfold) de seu tempo de computação. Fonte: BRITZ, 2015.

O diagrama na Figura 8 mostra uma RNN sendo desdobrada em uma rede completa. Esse processo de desdobramento significa apenas a sequência de toda a rede. Por exemplo, se uma sequência é uma sentença de 5 palavras, a rede seria desdobrada em uma rede neural com 5 camadas, com uma camada para cada palavra. Segue abaixo uma descrição dos símbolos da Figura 7.

X_t é a entrada em um passo de tempo t. Por exemplo, x_1 poderia corresponder a segunda palavra de uma sequência.

S_t é um estado escondido em um passo de tempo t. Essa é a "memória" da rede. s_t é calculado baseado no estado escondido anterior e na entrada no passo atual: $s_t = f(U_x + W_{St-1})$. A função f geralmente é uma função não linear como a função tanh ou ReLU.

S_{t-1} é usado para calcular o primeiro estado escondido, e é geralmente iniciado com 0.

O_t é a saída no passo t.

A parte mais importante desse diagrama da Figura 8 é o estado escondido S_t, que corresponde a memória da rede, e guarda informações sobre o que aconteceu em todos os passos anteriores. A saída no passo O_t é calculada a partir da memória atual do passo t.

O treinamento de uma RNN é similar ao treinamento de uma rede neural comum, ainda utilizando o algoritmo *backpropagation*, mas com uma pequena alteração para poder ser calculado o efeito de tempo/sequência na rede neural, fazendo com que esse novo algoritmo usado para redes recorrentes seja conhecido como *backpropagation through time* (BPTT).

Um problema bastante comum em RNNs é conhecido como “dependências de longo prazo”. Dependendo da tarefa que a RNN tenta executar, a RNN precisa buscar uma informação recente na rede para realizar essa tarefa. Porém, caso essa informação que a rede precise não seja tão recente, se torna difícil para a RNN aprendender a informação correta. Quanto mais “antiga” essa dependência, maior é a dificuldade da RNN aprender baseado nessa informação.

Uma abordagem que tenta resolver esse problema das dependências de longo pra são as chamadas redes *Long Short Term Memory*, popularmente conhecidas como LSTM.

3.2.1 Redes LSTM

Redes LSTM são um tipo especial de RNN, capazes de aprender dependências de longo prazo. Elas foram introduzidas inicialmente por Hochreiter e Schmidhuber (1997) e funcionam muito bem em uma grande variedade de problemas, sendo amplamente utilizadas atualmente.

Todas as RNNs têm a forma de uma cadeia de módulos que se repetem em uma rede neural. Em uma RNN padrão, esse módulo tem uma estrutura bem simples, como por exemplo uma camada com a função *tanh* conforme a Figura 9.

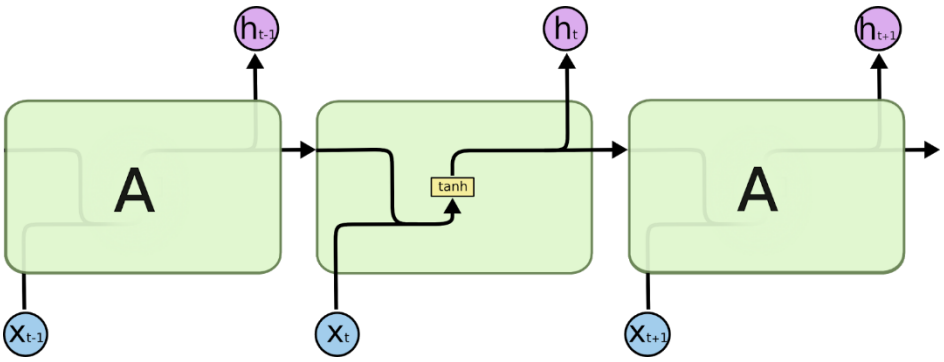


Figura 9: O módulo que se repete em uma RNN contém apenas uma única camada. Fonte: OLAH, 2015.

Redes LSTM também possuem essa estrutura de cadeia, mas o módulo que se repete possui uma estrutura diferente. Ao invés de ter apenas uma única camada de uma rede neural,

existem quatro, que interagem de uma maneira bastante específica. A Figura 8 apresenta visualmente este conceito.

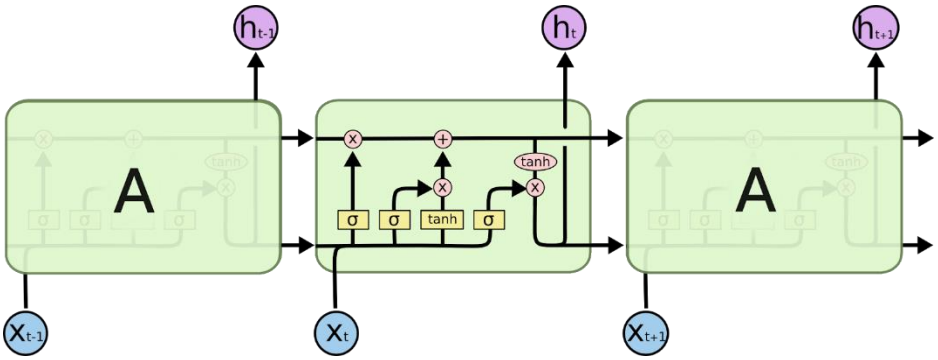


Figura 10: O módulo que repete em uma LSTM contém quatro camadas que se interagem. Fonte: OLAH, 2015.

Onde a Figura 11 apresenta a notação usada na imagem:

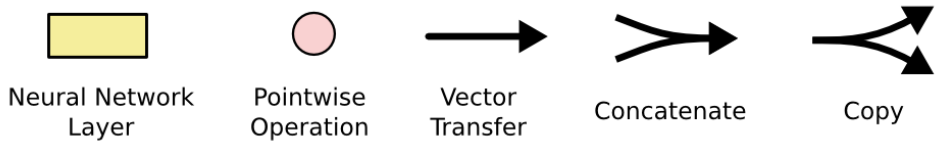


Figura 11: Notação utilizada no exemplo. Lê-se da esquerda para direita: Camada da rede neural, operação ponto a ponto, transferência de vetor, concatenar, copiar. Fonte: OLAH, 2015.

A principal ideia das redes LSTM é criar uma representação do estado da célula, que corresponde a linha horizontal na parte superior do diagrama (Figura 12). Esse estado da célula corre por toda a cadeia da célula, sofrendo apenas algumas interações lineares, fazendo com que a informação possa fluir sem muitas alterações.

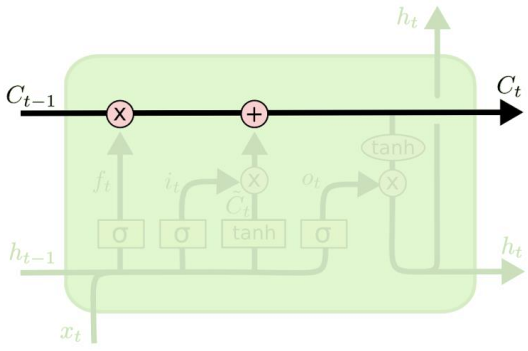


Figura 12: O estado da célula, representado por uma linha horizontal no topo do diagrama. Fonte: OLAH, 2015.

As redes LSTM também possuem a habilidade de remover ou adicionar informação para o estado da célula, sendo regulada por estruturas conhecidas como *gates*.

Os *gates* são uma maneira de deixar a informação fluir na rede neural. Eles são compostos por uma camada sigmóide de uma rede neural e uma multiplicação *pointwise*. A Figura 13 ilustra este conceito.

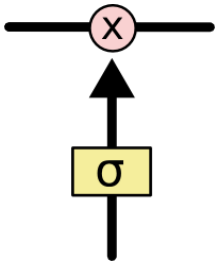


Figura 13: O exemplo de um *gate* com uma camada sigmóide de uma rede neural e uma operação de multiplicação *pointwise*. Fonte: OLAH, 2015.

A camada sigmóide tem como saída números variando entre zero e um, que descrevem o quanto de cada componente deve passar pelo *gate*. Quanto maior o valor, maior a informação que é passada por essa camada. Uma rede LSTM possui três *gates* para proteger e controlar o estado da célula.

4 MATERIAIS E MÉTODOS

Esta seção irá apresentar todos os materiais e métodos necessários para o trabalho. Na Seção 4.1 serão apresentadas as principais bibliotecas de aprendizado profundo que são utilizadas atualmente.

4.1 BIBLIOTECAS

Essa seção apresenta de maneira breve algumas das bibliotecas mais utilizadas pela comunidade acadêmica e indústria.

4.1.1 Caffe

Caffe[2] é uma biblioteca de aprendizado profundo desenvolvida pelo *Berkeley Vision and Learning Center* (BVLC) que foi criada pensando na velocidade e modularidade. Caffe oferece a cientistas e entusiastas uma coleção de exemplos de aprendizado profundo já implementados e diversos modelos de referência. (JIA et al., 2014)

4.1.2 Theano

Theano[3] é uma biblioteca de código aberto desenvolvida em Python pelo grupo de aprendizado de máquina da Universidade de Montreal. Assim como o Numpy, uma biblioteca que lida com matrizes em Python, a biblioteca Theano oferece suporte a operações sobre matrizes eficientemente e também oferece suporte a placas de vídeo, que podem realizar operações até 140 vezes mais rápido que uma CPU comum.

Apesar de sua alta eficiência para resolver expressões matemáticas envolvendo matrizes e personalização para otimizar o código, a biblioteca Theano é considerada difícil

[2] http://caffe.berkeleyvision.org

[3] http://deeplearning.net/software/theano/

para iniciantes e por isso existem outras bibliotecas que foram construídas usando o Theano como base, porém com uma interface mais amigável, como o *Keras* e *Lasagne*.

4.1.3 TensorFlow

TensorFlow[4] é uma biblioteca de código aberto usada para computação numérica, usando grafos de fluxo de data. O *TensowFlow* foi desenvolvido originalmente por pesquisadores e engenheiros trabalhando no projeto *Google Brain* dentro do grupo de Inteligência de Máquina do *Google*.

Assim como outras bibliotecas de aprendizado profundo, o *TensorFlow* é escrito como uma API em Python sobre um motor em C/C++ para a otimização de desempenho. Um dos benefícios do *TensorFlow* comparado a outras bibliotecas é o suporte à computação distribuída com suporte a múltiplas placas de vídeo.

4.1.4 Lasagne

Lasagne[5] é uma biblioteca leve usada para construir e treinar redes neurais, oferecendo uma interface simples e personalizável para se programar usando a biblioteca Theano como *backend*. O objetivo do *Lasagne* é a simplificar a construção de redes neurais, porém utilizando todo o poder que a biblioteca Theano oferece.

4.1.5 Keras

Keras[6] é uma biblioteca minimalista usada para construir redes neurais profundas que pode usar tanto a biblioteca *Theano* quanto o *TensorFlow* em seu *backend*. A maior motivação por trás do Keras é oferecer uma prototipação rápida para chegar aos resultados o mais rápido possível, por isso ela oferece uma interface de mais alto nível quando comparada

[4] https://www.tensorflow.org

[5] https://github.com/Lasagne/Lasagne

[6] https://keras.io

a outras bibliotecas de aprendizado profundo.

4.1.6 MXNet

MXNet[7] é uma biblioteca de aprendizado profundo para definir, treinar, e implantar redes neurais profundas em uma grande variedade de serviços, variando de infraestrutura na nuvem até dispositivos móveis. É uma biblioteca altamente escalável e com rápido treinamento, com suporte a múltiplas linguagens de programação.

Por ser altamente escalável, a biblioteca MXNet é bastante recomendada para treinar redes neurais profundas usando várias GPUs ou em sistemas na nuvem como o AWS da Amazon ou o Azure da Microsoft.

4.1.7 Torch

Torch[8] é uma biblioteca de aprendizado profundo na linguagem de programação Lua desenvolvido na Universidade de Nova Iorque. Torch vem sendo usado e aprimorado atualmente pelo laboratório de Inteligência Artificial do *Facebook*, pelo projeto *DeepMind* do Google, Twitter, entre outras empresas.

Um dos objetivos do Torch é oferecer bastante flexibilidade e velocidade para desenvolver algoritmos. Entre suas aplicações mais populares estão problemas de aprendizado supervisionado com Redes Neurais Convolucionais e aprendizado por reforço.

4.1.8 Deeplearning4j

Deeplearning4j[9] é uma biblioteca de código aberto focada em soluções para empresas

[7] https://github.com/dmlc/mxnet

[8] http://torch.ch

[9] https://deeplearning4j.org

desenvolvida para as linguagens Java e Scala. A decisão de criar uma biblioteca de aprendizado profundo para Java ocorre devido à popularidade de outras ferramentas utilizadas por grandes empresas também serem criadas em Java, como Hadoop[10] e Spark[11].

A Tabela 1 contém um rápido comparativo entre as tabelas citadas acima.

Tabela 1 – Comparativo das bibliotecas de Aprendizado Profundo disponíveis.

Nome	**Característica**
Caffe	Bastante popular, uma das primeiras bibliotecas a surgirem.
Theano	Rápida, porém de baixo nível.
TensorFlow	Bastante popular, desenvolvido pela empresa Google.
Lasagne	Leve, é uma interface para o treinamento de redes usando o Theano como *backend*.
Keras	Prototipação rápida, pode usar tanto o Theano quanto o TensorFlow como backend.
MXNet	Boa para usar várias GPUs ou treinar redes na nuvem.
Torch	Usado por empresas populares como Google, Facebook e Twitter. Desenvolvido em Lua.
Deeplearning4j	Biblioteca de aprendizado profundo para Java, tem como objetivo integrar os resultados a ferramentas como Hadoop e Spark.

Fonte: Elaborado pelo autor.

4.2 CONJUNTOS DE DADOS

Neste trabalho serão utilizadas quatro bases de dados para os procedimentos de tratamento e processamento das metodologias visando à predição de valores. As bases de dados fazem parte do DataMarket[12]. DataMarket é uma biblioteca virtual (repositório) de séries temporais, criada por Rob Hyndman, Professor de Estatística da Universidade de Monash, na Austrália. Em cada uma das subseções seguintes serão apresentadas uma breve descrição de cada conjunto de dados.

[10] http://hadoop.apache.org/
[11] http://spark.apache.org/
[12] https://datamarket.com

4.2.1 Quantitativo de Casos de Sarampo, New York, 1928-1972

Esta base de dados contém informações coletadas mensalmente entre o período de janeiro/1928 a junho/1972, com um total de 574 meses, registrando em números absolutos os casos de sarampo ocorridos na cidade de Nova York. A Figura 14 contém a série temporal gerada a partir desses dados.

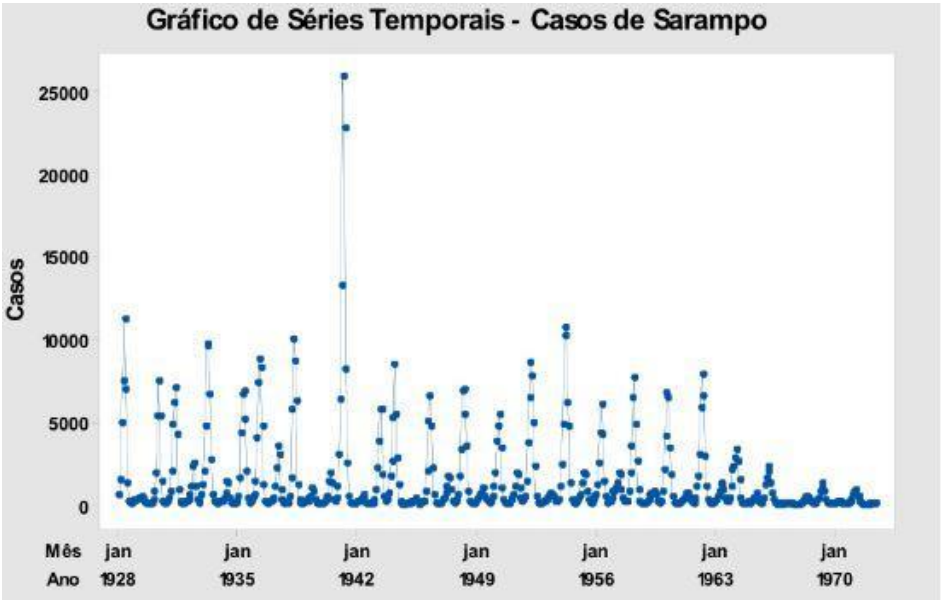


Figura 14: Casos de Sarampo, New York, (x: mês, y: casos absolutos). Fonte: Dados obtidos no site do DataMarket[13].

4.2.2 Produção de Carvão, EUA, 1920-1968

Esta base de dados contém os dados da produção anual de carvão nos Estados Unidos durante o período de 1920 a 1968, contendo um total de 49 amostras. A Figura 15 contém a série temporal gerada a partir desses dados.

[13] Disponível em: <https://datamarket.com/data/set/22z3/monthly-reported-number-of-cases-of-measles-new-york-city-1928-1972> Acesso em mar. 2017.

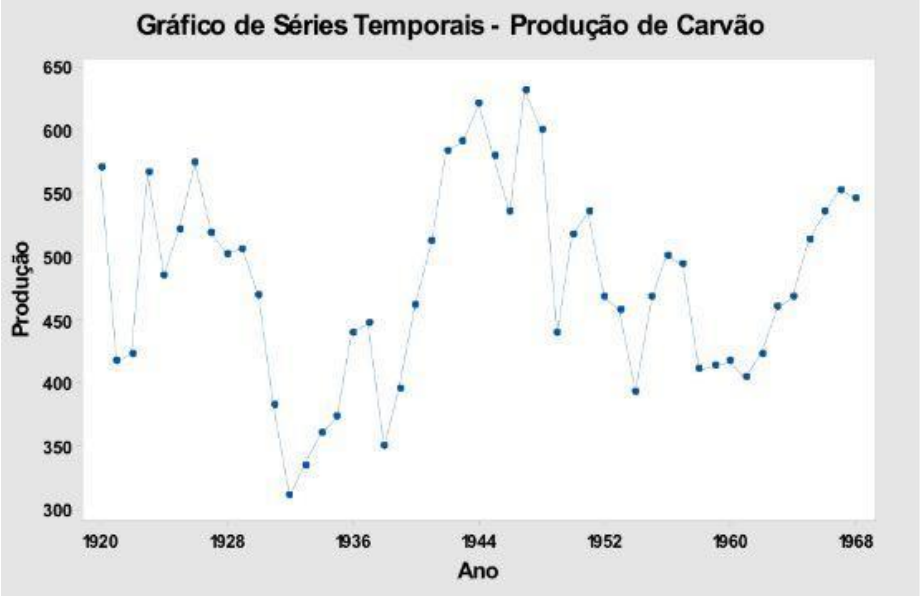
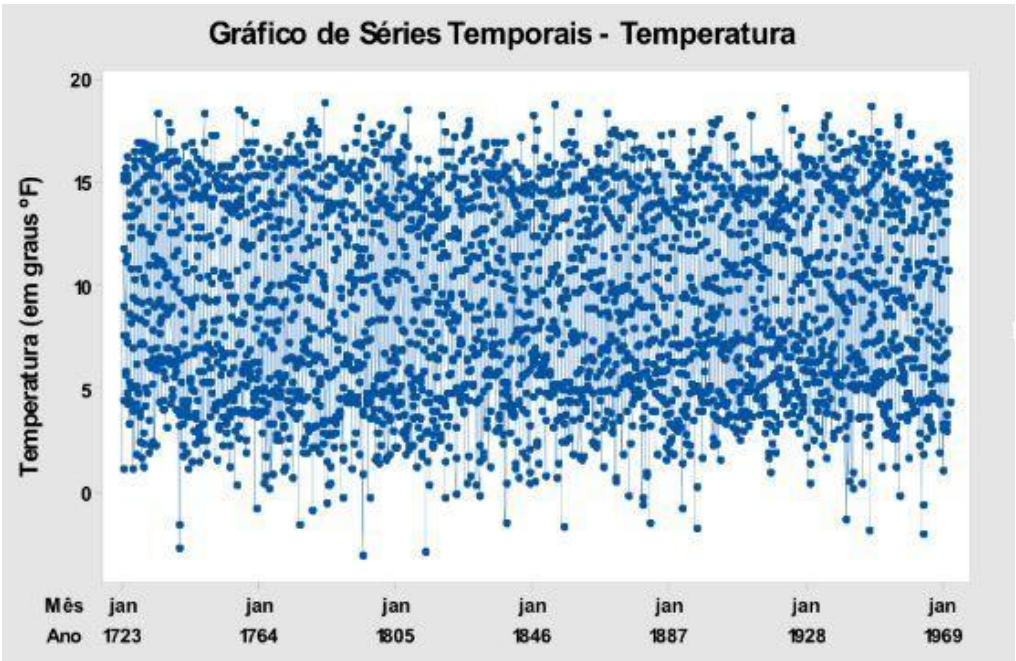


Figura 15: Produção de Carvão – EUA. (x: ano, y: milhões/ton). Fonte: Dados obtidos no site do DataMarket[14].

4.2.3 Temperatura mensal, Inglaterra, 1723-1970

Esta base de dados contém informações coletadas mensalmente entre o período de janeiro/1723 a dezembro/1970, referente à temperatura mensal na Inglaterra. A Figura 16 contém a série temporal gerada a partir desses dados.



[14] Disponível em: <https://datamarket.com/data/set/22yt/annual-bituminous-coal-production-usa-millions-of-net-tons-1920-1968> Acesso em mar. 2017.

Figura 16: Média Temperatura / Inglaterra. (x: mês, y: temperatura mensal em °F). Fonte: Dados obtidos no site do DataMarket[15].

4.2.4 Uso diário de Água, London, Ontario, Canadá, 1966-1988

Esta base de dados contém informações coletadas mensalmente referente ao consumo de água por habitante durante o período de janeiro/1966 a dezembro/1988 na cidade de London - Canadá. A base contém um total de 276 medições, sendo estas registradas em média mensal ml/hab. A Figura 17 contém a série temporal gerada a partir desses dados.

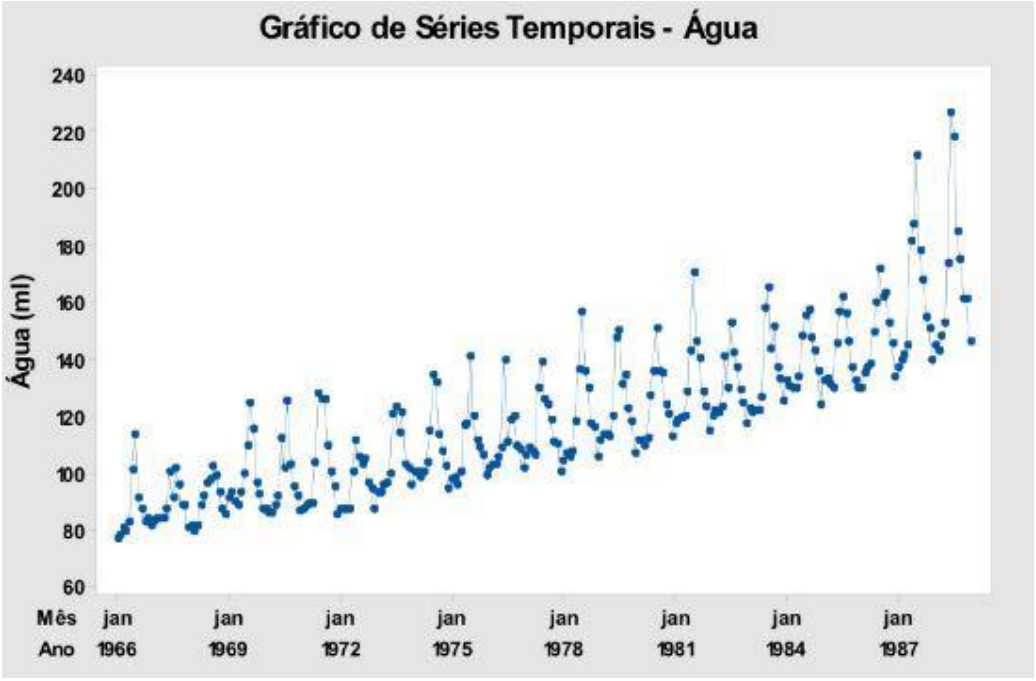


Figura 17: Consumo de água em London. (x: mês, y: ml/hab.). : Dados obtidos no site do DataMarket[16].

4.3 PYTHON

Python é uma linguagem de programação de propósito geral, frequentemente aplicada em funções de *script*. Ela é comumente definida como uma linguagem de *script* orientada a objetos – uma definição que combina suporte para Programação Orientada a Objeto com

[15] Disponível em: <https://datamarket.com/data/set/22vp/monthly-temperature-in-england-f-1723-1970> Acesso em mar. 2017.

[16] Disponível em: <https://datamarket.com/data/set/22qu/monthly-water-usage-mlday-london-ontario-1966-1988> Acesso em mar. 2017.

orientação global voltada para funções de *script*.

É uma linguagem de computação concisa e utilizada para os mais diversos domínios de aplicação. Entre suas vantagens está sua sintaxe compacta, suportar vários paradigmas de programação e sua manipulação de alto nível de tipos de dados. Sendo também uma linguagem interpretada, permite a construção e testes de programas para realização de tarefas não triviais rapidamente. Sendo a velocidade e desenvolvimento um fator importante, esta linguagem vem sendo muito bem aceita atualmente.

Uma vantagem de desenvolver em *Python* é o aumento de produtividade do programador, por ter menos digitação, menos depuração e por seus comandos serem executados imediatamente, sem a necessidade de compilação ou vinculação com outras ferramentas. Além disso, para utilizar um *script* desenvolvido no Linux, Windows ou Mac, seria necessário somente copiar o programa para a plataforma destino.

5 RESULTADOS EXPERIMENTAIS

Para a realização dos experimentos foi utilizada a biblioteca *Keras*, usando o *TensorFlow* como *backend*, por ser uma biblioteca que permite uma prototipação rápida e ser bastante simples de utilizar. A versão da linguagem *Python* utilizada nos testes é a 3.4.5. Os dados utilizados consistem de arquivos com a extensão .csv (*comma separated values)* com os dados de cada série temporal.

O desempenho da rede LSTM será avaliado de acordo com o valor do erro absoluto médio, ou *Mean Absolute Error* (MAE) em inglês. Na estatística, essa medida é usada para medir o quanto o resultado de uma predição se compara ao resultado real de um evento. A fórmula do erro médio absoluto é dada pela equação:

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n} |\, x_i - x \,|$$

Onde x_i corresponde ao valor da predição, x corresponde ao valor real que está no conjunto de dados e n corresponde ao número de amostras no conjunto de dados. A partir da equação acima é possível gerar o erro médio absoluto para saber como cada modelo de predição de comporta.

Para a realização do treinamento os conjuntos de dados foram divididos da seguinte forma: 80% para treinamento e 20% para teste. Os dados separados para treinamento serão utilizados para a rede LSTM aprender como cada conjunto de dados se comporta, enquanto os dados separados para testes serão utilizados para medir sua acurácia, utilizando o MAE, em dados que o modelo ainda não conhece.

A partir da Figura 18 até a Figura 21 são exibidos os gráficos gerados pela predição da rede LSTM em comparação com os dados reais do conjunto de dados. Os gráficos contêm os valores dos dados reais de cada conjunto de dados, os valores preditos nos dados separados para treinamento da rede LSTM (linha separada por traços) e os valores preditos nos dados separados para testes (linha separada por pontos).

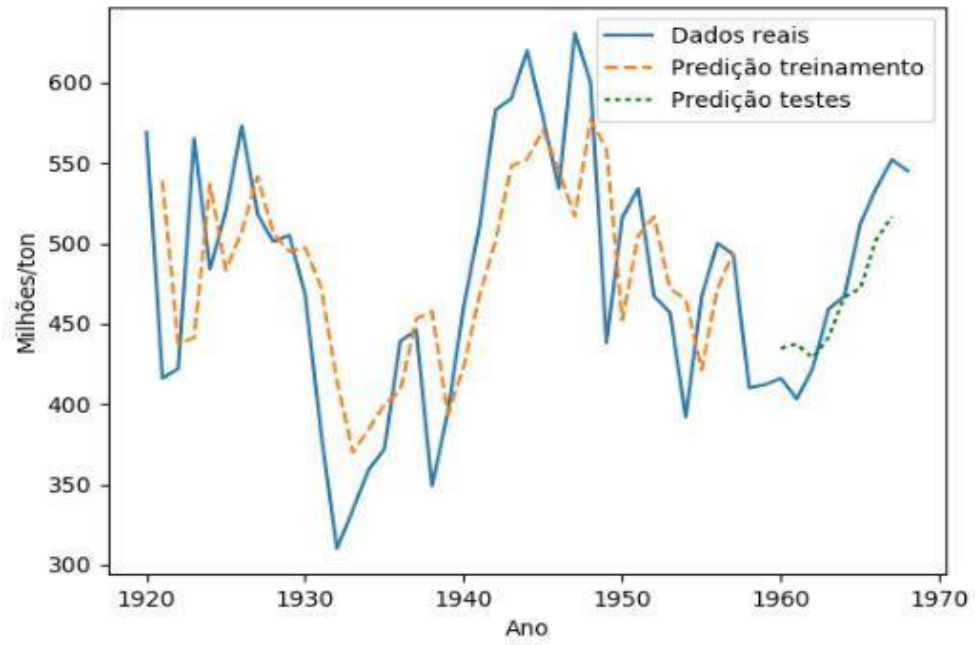


Figura 18: Produção de Carvão – EUA, (1920/1968). Resultados da predição da rede LSTM para os dados separados para treinamento e testes. (x: ano, y: milhões/ton). Fonte: Elaborado pelo autor

.

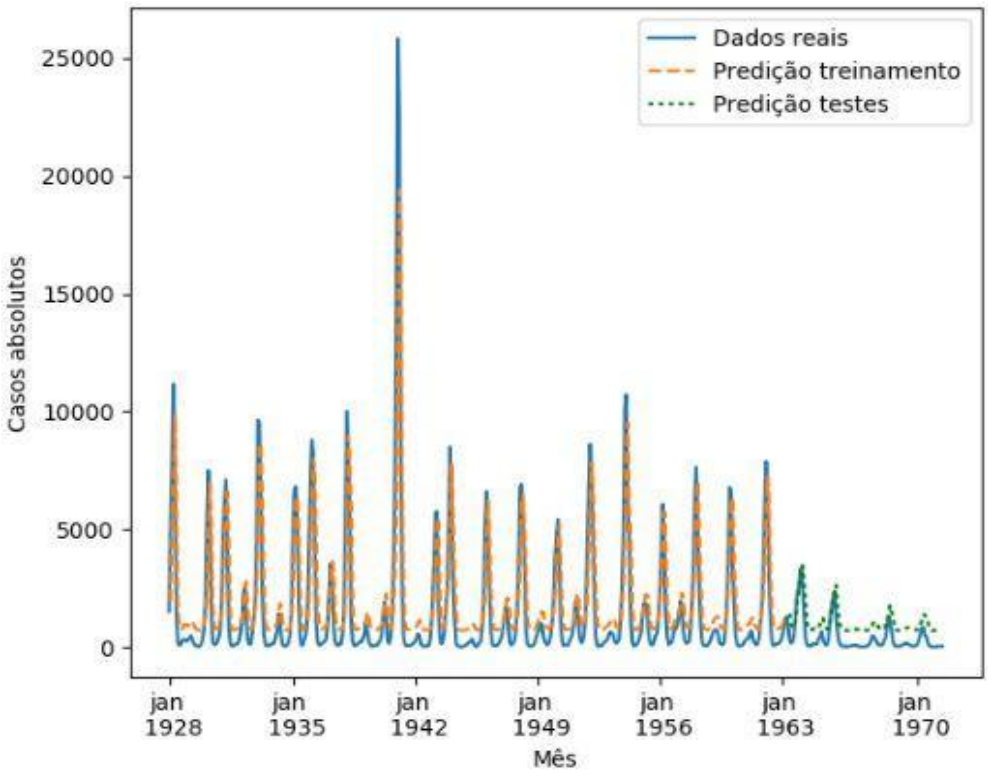


Figura 19: Casos de Sarampo, New York (1928/1972). Resultados da predição da rede LSTM para os dados separados para treinamento e testes. (x: mês, y: casos absolutos). Fonte: Elaborado pelo autor.

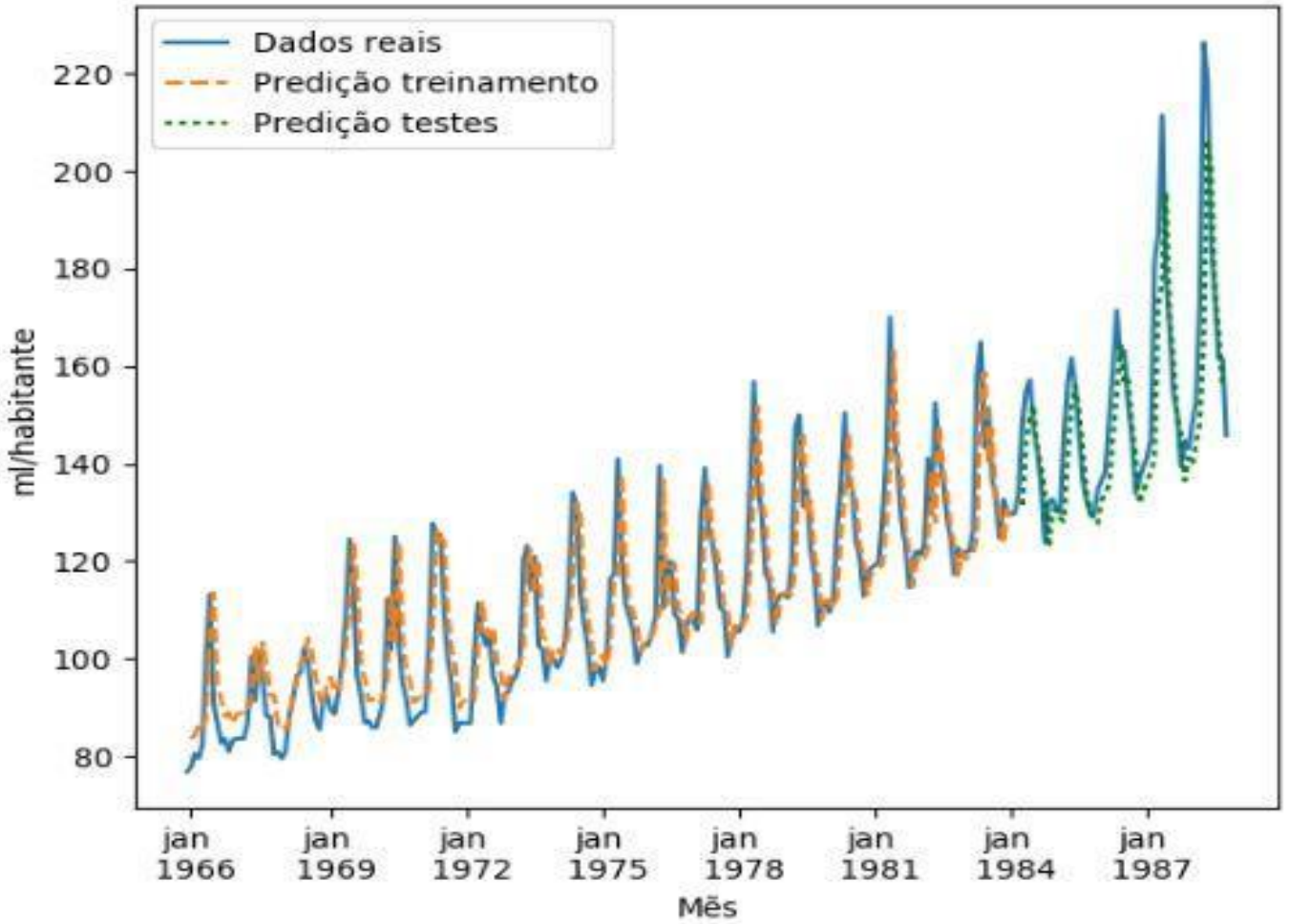


Figura 20: Consumo de água em London, Ontario (1966/1988). Resultados da predição da rede LSTM para os dados separados para treinamento e testes. (x: mês, y: ml/habitante.). Fonte: Elaborado pelo autor.

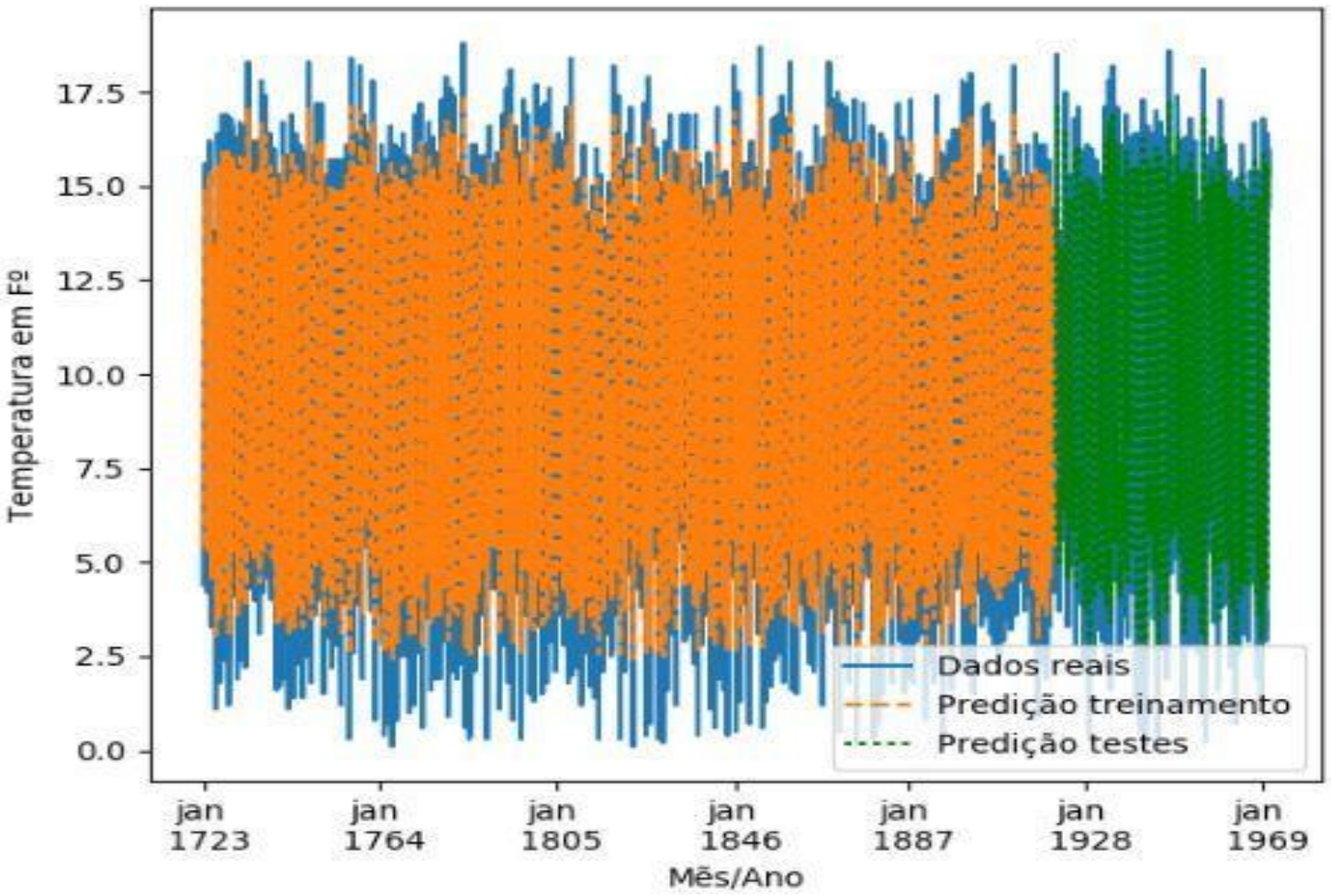


Figura 21: Temperatura mensal, Inglaterra (1723/1970). Resultados da predição da rede LSTM para os dados separados para treinamento e testes. (x: mês/ano, y: temperatura em ºF.). Fonte: Elaborado pelo autor.

A partir dos mesmos conjuntos de dados também foi calculado o erro médio absoluto para os modelos ARIMA e Redes Neurais Artificiais (RNA). Esses resultados foram gerados a partir do trabalho de Veríssimo (2016), onde foi construído um modelo ARIMA e uma RNA para testar a acurácia de cada modelo e avaliar qual modelo se comporta melhor nesses conjuntos de dados.

A partir dos resultados gerados será criada uma tabela com os resultados do MAE dos modelos ARIMA, RNA e LSTM, e uma comparação entre os resultados para analisar qual modelo de predição de séries temporais se saiu melhor nesses conjuntos de dados.

A Tabela 1 contém o valor previsto pela rede LSTM de cada dado do conjunto de testes, o valor real do conjunto de dados, e o erro absoluto gerado por esse modelo para o conjunto de dados da produção de carvão.

Tabela 1 - Resultado da predição dos valores no conjunto de testes da Produção de Carvão nos EUA (1928/1970) e o erro absoluto de cada predição.

Valor previsto LSTM	**Valor real**	**Erro absoluto**
434.63	416	18.63
437.32	403	34.32
428.61	422	6.61
441.36	459	17.64
473.50	467	6.5
471.83	512	40.17
502.06	534	31.94
516.57	552	35.43

Fonte: Elaborado pelo autor.

A Tabela 2 contém uma comparação do erro médio absoluto entre uma rede LSTM e os modelos ARIMA e RNA para todos os conjuntos de dados testados. Para gerar os resultados do MAE da Tabela 2 foi necessário gerar dois valores de MAE para a rede LSTM, um para o conjunto de dados normal e um para um conjunto de dados reduzido. Esse ajuste se deve ao fato do trabalho de Veríssimo (2016) ter usado apenas uma pequena parte dos conjuntos de dados, com a exceção do conjunto da produção de carvão, para fazer o treinamento e testes dos dados. A LSTM 1 corresponde ao mesmo conjunto de dados reduzido utilizado no trabalho de Veríssimo (2016), e a LSTM 2 corresponde ao conjunto de dados completo, obtido no site do DataMarket.

Tabela 2 – Resultado do MAE de cada modelo de predição de séries temporais para os quatro conjuntos de dados usados.

Base de dados	Metodologia	MAE	Melhor modelo
Casos de Sarampo	ARIMA	2117,82	LSTM
	LSTM 1	775,57	
	LSTM 2	**268,72**	
	Redes Neurais	2462,42	
Produção de Carvão	ARIMA	40,02	LSTM
	LSTM	**23,16**	
	Redes Neurais	71,97	
Temperatura	ARIMA	6,49	Redes Neurais
	LSTM 1	2,73	
	LSTM 2	2,24	
	Redes Neurais	**2,20**	
Consumo de Água	ARIMA	10,92	LSTM
	LSTM 1	**5,20**	
	LSTM 2	9,34	
	Redes Neurais	7.67	

Fonte: Elaborado pelo autor.

6 DISCUSSÃO

A partir dos resultados obtidos nos testes é possível afirmar que houve uma melhora significativa na previsão das séries temporais com a utilização de uma rede LSTM comparado com os resultados obtidos pelos modelos ARIMA e RNA. Conforme apurado nas tabelas de erros e gráficos na seção anterior, o modelo da rede LSTM conseguiu fazer uma previsão mais próxima do valor real em três dos quatro conjuntos de dados, perdendo apenas para uma RNA no conjunto de dados da média de temperatura na Inglaterra por uma pequena margem.

O resultado mais expressivo ocorreu na predição do conjunto de dados dos casos de sarampo em Nova York, onde a rede LSTM conseguiu diminuir em quase três vezes o valor do erro médio absoluto em comparação com o segundo melhor modelo para esse conjunto, e em quase oito vezes quando é utilizado o conjunto de dados com todos os dados. No conjunto de dados da produção de carvão a rede LSTM conseguiu um erro médio absoluto de 23,16 enquanto o segundo melhor modelo, a ARIMA, conseguiu um erro médio absoluto de 40,02, diminuindo o erro quase pela metade nesse conjunto de dados.

Ao longo do trabalho foram elaboradas algumas perguntas sobre o treinamento e o desempenho das redes LSTM. A partir dessas perguntas é possível responder algumas questões bastante relevantes com relação a séries temporais e redes profundas.

Como uma rede LSTM se compara em acurácia com o ARIMA e as Redes Neurais para predição de séries temporais?

A partir dos resultados obtidos através dos testes apresentados na seção anterior, é possível concluir que uma rede LSTM é uma ótima alternativa para a predição de séries temporais. Dos quatro conjuntos de dados testados, a rede LSTM obteve o menor erro médio absoluto (MAE) em três deles em comparação com a ARIMA e RNA, superando esses modelos por uma grande margem.

No único conjunto de dados onde a rede LSTM não obteve o melhor desempenho, o da média da temperatura mensal em Londres, a base de testes utilizada nos modelos da rede neural e ARIMA continham um número muito pequeno de entradas, o que aumenta bastante a

variação dos resultados. Ao treinar com uma base de treinamento e testes maior foi possível diminuir o erro absoluto médio da rede LSTM, melhorando o seu desempenho de maneira significativa.

Existe uma grande diferença no uso de recursos para a predição?

O uso de recursos aumenta conforme o tamanho do conjunto de dados aumenta. Também é possível configurar uma quantidade de epochs na rede LSTM que ajusta os pesos da rede a cada iteração, buscando minimizar seu erro. Porém, como a cada epoch adicionado é realizado uma nova iteração, é importante ficar atento a esse valor para que a rede não se torne muito lenta, e também para que não ocorra o overfitting, um fenômeno que ocorre quando o modelo de predição aprende muito bem os dados de treinamento, mas não se comporta tão bem para amostras do conjunto de teste.

Para conjuntos de dados pequenos, como a produção de carvão, foram utilizados 150 epochs no treinamento para obter os resultados apresentados, já para os outros conjuntos de dados foram utilizados apenas 30 epochs.

Como se compara a quantidade de parâmetros de uma rede LSTM com os modelos ARIMA e RNA?

Para a predição de séries temporais no modelo ARIMA é necessário definir o valor das métricas p e q que serão usados. Já para uma RNA é necessário definir a quantidade de nós da camada escondida, uma taxa de aprendizado, a quantidade de epochs que serão usados, a inicialização dos pesos da rede e sua função de ativação. Uma rede LSTM é muito parecida com uma RNA para se configurar, sendo necessário definir os mesmos parâmetros necessários para uma RNA, com a adição de também ser necessário definir a quantidade de camadas escondidas da rede.

Conjunto de dados que possuem uma quantidade maior de dados possuem resultados melhores?

Depende de cada caso. No conjunto de dados sobre a temperatura mensal em Londres

o erro médio absoluto diminuiu conforme mais dados foram introduzidos. Como esse conjunto de dados tinha a maior quantidade de dados dentre os quatro conjuntos de dados testados, o aumento na quantidade de dados de treinamento e testes ajudou o erro a se estabilizar, diminuindo sua variação a cada nova entrada de dados.

Entretanto, no conjunto de dados sobre o consumo de água, o erro médio absoluto aumentou conforme mais dados foram introduzidos. Isso ocorreu devido a um aumento acima do normal do consumo de água, algo que a rede LSTM não conseguiu prever tão bem, por isso o aumento no erro. Não foram realizados testes para saber como a ARIMA e uma rede neural comum se comportam nesse tipo de caso, mas como o aumento nos últimos pontos dos dados aumentam de uma maneira maior que a tendência, é bastante provável que ambos os modelos também teriam um aumento do erro absoluto médio.

7 CONCLUSÃO

Neste trabalho foi apresentado uma introdução ao aprendizado profundo, que consiste de uma classe de técnicas que vem se popularizando muito nos últimos anos devido ao seu ótimo desempenho em diversas áreas de pesquisa. Também foi realizada uma breve revisão bibliográfica que descreve os grandes avanços feitos na área, desde o Perceptron criado por Rosenblatt, até as redes LSTM, uma técnica bastante poderosa que surgiu a partir das Redes Neurais Recorrentes.

Nos experimentos realizados a partir de quatro conjunto de dados sobre séries temporais, a rede LSTM se saiu superior em três desses conjuntos em comparação com as metodologias ARIMA e Redes Neurais Artificiais. Dado a sua característica de armazenar em sua memória a sequência dos dados, as redes LSTM tiveram o melhor resultado nessa comparação.

Esses resultados iniciais demonstram o grande potencial das redes LSTM na tarefa de predição de séries temporais. Para trabalhos futuros, este trabalho pode ser usado como base para oferecer uma introdução no campo de aprendizado profundo e para a melhor compreensão das redes LSTM.

8 REFERÊNCIAS BIBLIOGRÁFICAS

BRITZ, Denny. Recurrent neural networks tutorial, Part 1 – Introduction to RNNs. Disponível em: <http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/>. Acesso em: 17 de mar. 2017.

DENG, L. et al. Deep Learning : Methods and Applications. p. 197–387, 2013.

HAYKIN, Simon. **Neural Networks: A Comprehensive Foundation.** New York: Mcmillan College Publishing Company Inc, 1994.

HE, K. et al. Delving Deep into Rectifiers : Surpassing Human-Level Performance on ImageNet Classification. CoRR, 2015.

HINTON, G. et al. Deep Neural Networks for Acoustic Modeling in Speech Recognition. **IEEE Signal Processing Magazine**, v. 29, n. 6, p. 82–97, 2012.

HINTON, G. E.; NAIR, V. Rectified Linear Units Improve Restricted Boltzmann Machines. **Proceedings of the 27th International Conference on Machine Learning (ICML-10)**, p. 807–814, 2010.

HOCHREITER, S.; SCHMIDHUBER, J. Long Short-Term Memory. **Neural Computation**, v. 9, n. 8, p. 1735–1780, 1997.

JIA, Y. et al. Caffe : Convolutional Architecture for Fast Feature Embedding ∗. CoRR, p. 4, 2014.

MINSKY, M. Steps toward Artificial Intelligence. **Proceedings of the IRE**, v. 49, n. 1, p. 8–30, 1961.

MINSKY, Marvin; PAPERT, Seymour. **Perceptrons.** Cambridge: Mit Press, 1969.

MITCHELL, Tom. **Machine Learning.** New York: McGraw-hill Education, 1997.

NIELSEN, Michael A. **Neural Networks and Deep Learning.** San Francisco: Determination Press, 2015.

OLAH, Christopher. Understanding LSTM Networks. Disponível em: < http://colah.github.io/posts/2015-08-Understanding-LSTMs/>. Acesso em: 17 de mar. 2017.

ROSENBLATT, F. The perceptron: a probabilistic model for information storage and organization in the brain. **Psychological review**, v. 65, n. 6, p. 386–408, 1958.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning representations by back-propagating errors. **Nature**, v. 323, n. 6088, p. 533–536, 1986.

SIMONYAN, K.; ZISSERMAN, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. **ImageNet Challenge**, p. 1–10, 2014.

SRIVASTAVA, N. et al. Dropout : A Simple Way to Prevent Neural Networks from Overfitting. v. 15, n. Journal of Machine Learning Research, p. 1929–1958, 2014.

VERÍSSIMO, J. L. **Séries temporais: Um estudo de previsão**. 2017. 66 f. Trabalho de Conclusão de Curso (Graduação) –Faculdade de Ciências Exatas e Tecnologia, Universidade Federal da Grande Dourados, Dourados, 2016.

WU, Y. et al. Google's Neural Machine Translation System : Bridging the Gap between Human and Machine Translation. p. 1–23, 2016.

YU, D.; DENG, L. Deep Learning and Its Applications to Signal and Information Processing [Exploratory DSP]. **Signal Processing Magazine, IEEE**, v. 28, n. 1, p. 145–154, 2011.

ZHONG, S.; LIU, Y.; LIU, Y. Bilinear deep learning for image classification. p. 343–352, 2011.