UNIVERSIDADE DE SÃO PAULO
INSTITUTO DE FÍSICA DE SÃO CARLOS

Ana Carolina Ferreira Luchesi

Utilizando o aprendizado de máquina para análise de órbitas caóticas

São Carlos

2022

Ana Carolina Ferreira Luchesi

Utilizando o aprendizado de máquina para análise de órbitas caóticas

Dissertação apresentada ao Programa de Pós-Graduação em Física do Instituto de Física de São Carlos da Universidade de São Paulo, para obtenção do título de Mestre em Ciências.

Área de concentração: Física Aplicada
Opção: Física Computacional

Orientador: Prof. Dr. Odemir Martinez Bruno

Versão corrigida
(versão original disponível na Unidade que aloja o Programa)

São Carlos
2022

Ferreira Luchesi, Ana Carolina
Utilizando o aprendizado de máquina para análise de órbitas caóticas / Ana Carolina Ferreira Luchesi; orientador Odemir Martinez Bruno - versão corrigida -- São Carlos, 2022.
79 p.

Dissertação (Mestrado - Programa de Pós-Graduação em Física Aplicada Computacional) -- Instituto de Física de São Carlos, Universidade de São Paulo, 2022.

1. Teoria do caos. 2. Órbitas caóticas. 3. Mapa logístico. 4. Aprendizado de máquina. 5. Redes neurais. I. Bruno, Odemir Martinez, orient. II. Título.

À minha família.

AGRADECIMENTOS

Agradeço primeiramente à minha família. Aos meus pais, Regina e Luis, por todo amor, suporte e incentivo que me deram a vida toda. À minha irmã, Renata, por ser a minha maior parceira. À minha avó Abadia, por sempre me oferecer tanto carinho e acolhimento. E não menos importante, ao meu grande amor peludinho, Boo. Não me lembro da minha vida antes da sua chegada, você trouxe à tona partes de mim que até eu desconhecia. Agradeço todos os dias por você simplesmente existir e por ser meu mais fiel companheiro.

Ao meu orientador Prof. Dr. Odemir Martinez Bruno, por ter me orientado e me apresentado a um universo fantástico que até então eu desconhecia dos sistemas caóticos e das redes neurais.

Aos meus amigos, pelo amor, apoio e pelos momentos valiosos compartilhados. Há muitos nomes que eu gostaria de colocar aqui, mas tentarei ser breve e colocar apenas alguns que foram muito presentes desde o início da minha jornada no IFSC: Iago, Murilo, Tapia, Rosanna, Leon, Bethania, Manivela, Espeto e Uirá. Reforço a minha imensa gratidão a todos os meus amigos, não sei o que seria de mim sem vocês! Também acrescento um pedido de desculpas pela ausência nos tempos de pandemia.

Aos meus amigos e colegas da pós, pela companhia e auxílio nessa etapa: Bruno, Willian e Leonardo.

Aos professores e funcionários do IFSC que me auxiliaram nessa trajetória.

Ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) pelo apoio financeiro através da concessão da bolsa de mestrado.

"Stories about the creation of machines having human qualities have long been a fascinating province in the realm of science fiction. Yet we are about to witness the birth of such a machine – a machine capable of perceiving, recognizing and identifying its surroundings without any human training or control."

Frank Rosenblatt

RESUMO

LUCHESI, A.C.F. **Utilizando o aprendizado de máquina para análise de órbitas caóticas**. 2022. 79p. Dissertação (Mestrado em Ciências) - Instituto de Física de São Carlos, Universidade de São Paulo, São Carlos, 2022.

O comportamento caótico pode ser observado nos mais diversos sistemas, incluindo órbitas planetárias, clima, mercado de ações, entre outros. Por conseguinte, investigar meios de prever o futuro de sistemas caóticos podem nos ajudar a compreender muitos dos sistemas que nos cercam. O mapa logístico, originalmente proposto como um modelo para descrever um crescimento populacional, apresenta sensibilidade a condições iniciais, o que resulta em uma imprevisibilidade no futuro desse sistema. Ele é um dos exemplos mais famosos de comportamento caótico emergindo de um sistema dinâmico simples. A escolha por estudar as órbitas caóticas do mapa logístico se deu pela sua simplicidade e grande grau de complexidade. As redes neurais são a categoria mais popular de aprendizado de máquina e alcançaram resultados estado da arte para diversas tarefas distintas. As redes neurais recorrentes são capazes de recordar entradas anteriores, sendo, portanto, as mais adequadas para lidar com dados sequenciais. Nesse trabalho, dois tipos de redes recorrentes foram utilizadas para investigar como preveem o futuro de órbitas caóticas do mapa logístico: *Long Short-Term Memory* (LSTM) e *Echo State Network* (ESN). Os resultados obtidos mostram que as ESNs são capazes de prever essas órbitas com maior acurácia que LSTMs e confirmam que são uma ferramenta promissora para desafiar a imprevisibilidade do caos.

Palavras-chave: Teoria do caos. Órbitas caóticas. Mapa logístico. Aprendizado de máquina. Redes neurais.

ABSTRACT

LUCHESI, A.C.F. **Using machine learning to analyze chaotic orbits**. 2022. 79p. Dissertation. (Master in Science) - Instituto de Física de São Carlos, Universidade de São Paulo, São Carlos, 2022.

Chaotic behaviour can be observed in the most diverse systems, including planetary orbits, weather, stock market and so on. Therefore, investigating means to predict the future of chaotic systems can help us comprehend many systems that surround us. The Logistic Map, originally proposed as a model to describe a population growth, presents sensitivity to initial conditions, which leads to the unpredictability of the future of this system. It is one of the most famous examples of chaotic behaviour emerging from a simple dynamical system. In this study, the chaotic orbits of the logistic map were chosen because of both their simplicity and wide range of complexity. Neural networks are the current most popular approach to machine learning and have achieved state-of-the art results in many different tasks. Recurrent Neural Networks are able to recollect previous inputs, they are therefore more suitable to handle sequential data. In this dissertation, two types of recurrent neural networks were used to investigate how they predict future terms of chaotic orbits from the Logistic Map: Long Short-Term Memory (LSTM) and Echo State Networks (ESN). The results obtained show that ESNs are able to predict these orbits with higher accuracy than LSTMs and confirm them as promising tool to challenge the unpredictability of chaos.

Keywords: Chaos theory. Chaotic orbits. Logistic map. Machine learning. Neural networks.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ESN	*Echo State Networks*
LSTM	*Long Short-Term Memory*
LE	Expoente de Lyapunov (*Lyapunov Exponent*)
MAE	Erro Médio Absoluto (*Mean Absolut Error*)
MLP	Perceptron Multi Camadas (*Multi Layered Perceptron*)
MSE	Erro Médio Quadrático (*Mean Squared Error*)
RMSE	Raiz do Erro Médio Quadrático (*Root Mean Squared Error*)
RNN	Redes Neurais Recorrentes (*Recurrent Neural Networks*)
SVM	Máquina de Vetores de Suporte (*Support Vector Machine*)
SGD	Gradiente Descendente Estocástico (*Stochastic Gradient Descent*

SUMÁRIO

1 INTRODUÇÃO

O século XX é marcado por grandes quebras de paradigmas. Prevalecia até então uma visão de que os princípios da mecânica newtoniana eram absolutos, e que dadas as informações suficientes, tudo seria possível de ser previsto. Também acreditava-se que era possível analisar o todo o quebrando em partes. Assim como a relatividade geral e a mecânica quântica, o estudo da complexidade e teoria do caos mostraram os grandes erros e limitações do paradigma anterior. Na segunda metade do século, na tentativa de prever as condições meteorológicas a partir de um conjunto de equações diferenciais, Edward Lorenz se deparou com o fato de que alterações que julgava insignificantes nas condições iniciais geravam previsões muito distintas, o que tornaria a previsão a longo prazo impossível. (1) Embora tal evento já houvesse sido notado por outros antes de Lorenz — como Henri Poincaré (2) e James Clerk Maxwell (3) — os recursos trazidos pela computação contribuíram para que os estudos na área de sistemas dinâmicos não lineares pudessem ser aprofundados e expandidos. Descobriu-se que sistemas determinísticos podiam apresentar um comportamento que foi denominado caótico.

O estudo do caos é multidisciplinar, uma vez que ele pode ser observado nas mais diversas áreas, como economia, geologia, meteorologia, medicina, entre outros. Ao longo da segunda metade do século XX, diversos cientistas se dedicaram a estudar sistemas caóticos. Notou-se que esses sistemas, embora imprevisíveis, possuíam padrões de comportamento. Lorenz, por exemplo, notou que, embora o valor de uma previsão nunca era exatamente repetido, ele não assumia qualquer posição no espaço de fase, pelo contrário: ele parecia sempre estar em espaço bem delimitado, o que mais tarde seria chamado de atrator estranho. Feigenbaum foi um dos maiores contribuintes no estudo do comportamento caótico. Mais notoriamente, ele foi responsável por notar que padrões nas bifurcações de mapas unidimensionais. (4) Mandelbrot, por sua vez, teve como maior foco o estudo de fractais, que são figuras geométricas que apresentam autosimilaridade —padrões de que se repetem indefinidamente. (5)

Em 1976, Robert May publicou um artigo discorrendo sobre a grande complexidade que equações simples poderiam exibir. (6) Dentre os exemplos citados no artigo, destaca-se o mapa logístico, uma equação de recorrência usada na ecologia para descrever a evolução de uma população. (7) Para certos parâmetros, o mapa logístico apresenta comportamento caótico. Nesse trabalho, May defende o uso do mapa logístico como exemplo didático para compreender os mecanismos do comportamento caótico, por se tratar de um sistema simples com um rico comportamento.

O aprendizado de máquinas, especialmente com redes neurais, tem apresentado excelentes resultados para resolver problemas de extração e reconhecimento de padrões.

As redes neurais recorrentes possuem conexões internas que permitem que elas guardem informações de entradas anteriores, o que as torna aptas para processar dados sequenciais. Elas foram inventadas na década de 80, mas foi a partir da invenção das redes *Long Short-Term Memory* (LSTM) na década seguinte (8), uma arquitetura mais sofisticada, que essas redes conseguiram ser usados com sucesso para tarefas mais complexas. Um paradigma mais recente em redes neurais é a computação de reservatório, um termo guarda chuva que abrange arquiteturas como a *Echo State Network* (ESN), que é um tipo de redes recorrentes em que somente os pesos de saída são treinados. Essa categoria de redes neurais tem se mostrado particularmente eficiente para analisar sistemas não lineares. (9) Nesse contexto, o objetivo desse trabalho é usar redes neurais para estudar as órbitas caóticas do mapa logístico, a fim de analisar o quão longe podemos prever corretamente a evolução desse sistema.

O Capítulo 2 apresenta uma introdução aos conceitos de sistemas dinâmicos, sistemas caóticos e do mapa logístico. Já o Capítulo 3 descreve os princípios por trás das redes *Long Short Term Memory* (8) e *Echo State Networks*. (10) O Capítulo 4 descreve como o conjunto de dados para o treinamento das redes foram gerados, assim como descreve como as redes foram treinadas, enquanto, no Capítulo 5, são apresentados os resultados obtidos, assim como discussão a respeito deles e comparações entre as acurácias obtidas. Por fim, o Capítulo 6 apresenta as considerações finais do trabalho.

2 TEORIA DO CAOS

2.1 Histórico

O nascimento do caos é geralmente atribuído a descoberta do efeito borboleta por Lorenz. (1) Embora ele não tenha sido o primeiro a notar a existência de sensibilidade às condições iniciais, foi somente a partir da invenção dos computadores de alta velocidade nos anos 50 que esse novo campo da ciência pode ser explorado de forma devida (11), uma vez que poderia se analisar soluções dos sistemas dinâmicos na sua forma mais "pura", descartando a possibilidade de algum comportamento poder ser atribuído a efeitos experimentais desconhecidos. (12)

Em Principia, Newton propõe o problema dos N corpos, uma vez que a solução analítica para a órbita de dois corpos não era suficiente para lidar com o sistema solar. Em 1989, o rei sueco Oscar II ofereceu um prêmio para aquele que conseguisse resolver o problema ou oferecesse uma contribuição relevante. O vencedor, o matemático francês Henri Poincaré, não conseguiu solucionar o problema, mas desenvolveu diversas novas ferramentas e propôs uma nova perspectiva qualitativa de se analisar o problema, que seria essencial para outros no futuro criarem o estudo de sistema dinâmicos. Outrossim, Poincaré foi capaz de vislumbrar o que seria chamado de caos no futuro, ao notar que quaisquer mudanças nas condições iniciais, mesmo que muito pequenas, causavam resultados muito distintos. (2) Ainda na primeira metade do século seguinte, houveram algumas contribuições importantes a respeito de sistemas dinâmicos. Dentre essas, destacam-se os trabalhos de Birkhoff (13), Cartwrigth e Littlewood (14) e Kolmogorov. (15, 16)

Em 1955, Edward Lorenz, ao refazer uma simulação meteorológica com condições iniciais ligeiramente diferentes, notou que a previsão obtida era drasticamente diferente da anterior. Em 1963, publicou um artigo discorrendo sobre essa descoberta. Nele, ele usa o bater de asas de uma borboleta como metáfora para uma pequena mudança nas condições iniciais e questiona se esse bater de asas era capaz de alterar drasticamente o tempo de uma região. Em outras instâncias ele também fez o uso dessa comparação, como por exemplo, uma palestra que deu com título *"Does the flap of a butterfly's wings in Brazil set off a tornado in Texas?"*. Tal feito também teve grande impacto cultural, que usou o conceito de efeito borboleta — na maioria das vezes sem rigor — para criação de obras literárias e audiovisuais. Ademais, ao plotar as órbitas no espaço de fase, Lorenz notou que elas ficavam confinadas em certos intervalos, como ilustrado na Figura 1, mostrando que apesar da aparente aleatoriedade, o comportamento desse sistema apresentava alguma espécie de ordem.

A partir da década de 70, diversos outros artigos foram publicados discorrendo

sobre o comportamento caótico. Ruelle e Takens estudaram o comportamento caótico em turbulência de fluídos (17), além de cunharem o termo atrator estranho. Mais tarde Takens desenvolveu o teorema de Takens, que permite reconstruir um sistema dinâmico a partir de uma séries temporal. (18) Outros trabalhados importantes a respeito de caos no comportamento de fluídos foram publicados por Gollub (19), Libchaber (20) e Swinney. (21) Em 1976, Robert May publicou um dos artigos de maior impacto dentro da área (6), estudando as propriedades que o mapa logístico, uma simples equação de recorrência, apresentava. Feigenbaum, por sua vez, explorou padrões dentro do caos. (4) O matemático Benoît Mandelbrot foi um dos pioneiros no uso da computação gráfica para ilustrar e explorar fractais. (5)

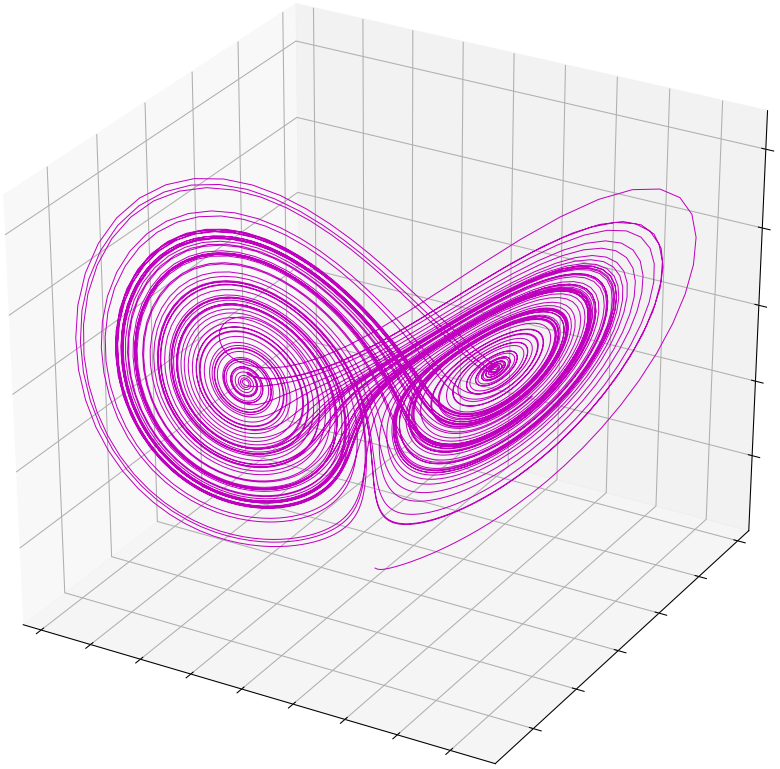

Figura 1 – Atrator de Lorenz

Fonte: Elaborada pela autora.

2.2 Sistemas Dinâmicos Não Lineares

Um sistema dinâmico pode ser descrito como uma função ou um conjunto de funções determinísticas que descreve a evolução de um sistema no tempo. (12) Seja S um espaço de fase, T o tempo e A uma regra de evolução, temos

$$A : S \times T \to S \tag{2.1}$$

Se o tempo de um sistema dinâmico é discreto, ele é descrito pelo mapa 2.2

$$x_{t+1} = f(x_t), \tag{2.2}$$

Definimos uma órbita como o conjunto de estados $\mathcal{O} = \{x_0, x_1, ...x_t\}$ obtidos pela regra de evolução do sistema dinâmico com a condição inicial x_0.

2.2.1 Expoente de Lyapunov

O expoente de Lyapunov é a taxa com que a trajetória de pontos muito próximos no espaço de fase divergem ou convergem. (22) Ele recebe esse nome devido ao matemático Aleksandr Lyapunov, cujas contribuições tiveram grande impacto no desenvolvimento da teoria da estabilidade de sistemas dinâmicos. (23) Consideremos λ o expoente de Lyapunov, e x_0 e $x_0' = x_0 + \varepsilon$ dois pontos, sendo ε positivo e infinitesimalmente pequeno. Após t iterações, a distância entre esses pontos é dada por:

$$\varepsilon(t) = |f^t(x_0 + \varepsilon) - f^t(x_0)| \approx \varepsilon e^{\lambda(x_0)t}. \tag{2.3}$$

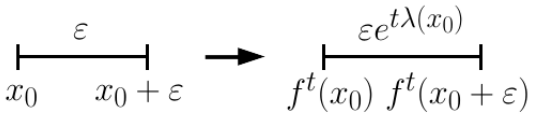

Figura 2 – Evolução da distância entre dois pontos x_0 e $x_0 + \varepsilon$ após um intervalo t.

Fonte: Elaborada pela autora.

Para um problema n-dimensional, seria necessário considerar um vetor e não apenas uma distância ε. Nesse caso exite um espectro de expoentes de Lyapunov $\{\lambda_1, \lambda_2, ..., \lambda_n\}$(24), mas apenas o maior deles, chamado de expoente de Lyapunov máximo, costuma ser considerado para análise da caoticidade da órbita.

O expoente de Lyapunov de uma órbita para sistemas unidimensionais é dado por:

$$\lambda(x_0) = \lim_{t\to\infty} \lim_{\varepsilon\to 0} \frac{1}{t} \ln \left| \frac{f^t(x_0 + \varepsilon) - f^t(x_0)}{\varepsilon} \right| = \lim_{t\to\infty} \frac{1}{t} \ln \left| \frac{df^t(x_0)}{dx_0} \right|. \tag{2.4}$$

Especificamente para mapas unidimensionais, o expoente de Lyapunov é dado por:

$$\lambda(x_0) = \lim_{t\to\infty} \frac{1}{t} \sum_{t=0}^{t-1} \ln |f^t(x_i)|. \tag{2.5}$$

Se a distância entre dois pontos próximos após t iterações é dado pela Equação 2.3, então um expoente de Lyapunov negativo indica que esses pontos se aproximam exponencialmente, positivo indica que eles se afastam e um expoente de Lyapunov igual a zero, que a distância entre eles permanece constante. Por conseguinte, é possível classificar o comportamento de uma órbita de acordo com o seu expoente de Lyapunov da seguinte forma:

- Se $\lambda < 0$, trata-se de um regime periódico.
- Se $\lambda = 0$, trata-se de um regime quase-periódico.
- Se $\lambda > 0$, trata-se de um regime caótico.

2.3 Mapa Logístico

A equação logística, também conhecida como modelo de Verhulst-Pearl, foi inicialmente introduzida em um contexto ecológico para descrever um crescimento populacional. O modelo foi proposto em 1838 por Pierre-François Verhulst (25), como uma versão modificada do crescimento exponencial de Malthus (26) impondo um valor máximo a população devido a limitação de recursos disponíveis para a sobrevivência desta. O nome equação logística veio alguns anos mais tarde (27), assim como uma correção dela. (7) A equação foi redescoberta de forma independente anos mais tarde, em 1920, por Raymond Pearl e Lowell Reed. (28)

O mapa logístico é uma equação análoga com tempo discreto e consiste na equação de recorrência 2.6.

$$x_n = x_{n-1} R(1 - x_{n-1}). \tag{2.6}$$

O mapa logístico foi popularizado no contexto de estudos de sistemas caóticos após Robert May publicar um artigo (6) muito influente em 1976.

2.3.1 Valores do parâmetro R

As Figuras 3 a 7 mostram as 100 primeiras iterações do mapa logístico no espaço de fase para alguns valores do parâmetro de dois valores iniciais bastante próximos: $x_0 = 0,1$ e $x'_0 = 0,100001$.

Para $R = 2,8$, nota-se que após algumas iterações, x mantém-se em um só valor — trata-se de um atrator simples. Para $R = 3,2$, após transiente, x alterna entre dois valores, enquanto para $R = 3,5$, quatro valores, ambos são casos de atratores periódicos, com periodicidade 2 e 4 respectivamente. Já para $R = 3,8$ e $R = 4$, observa-se, além da falta

de periodicidade, que as órbitas geradas por valores iniciais muito próximos divergiram. Nesse caso, estamos lidando com uma órbita caótica.

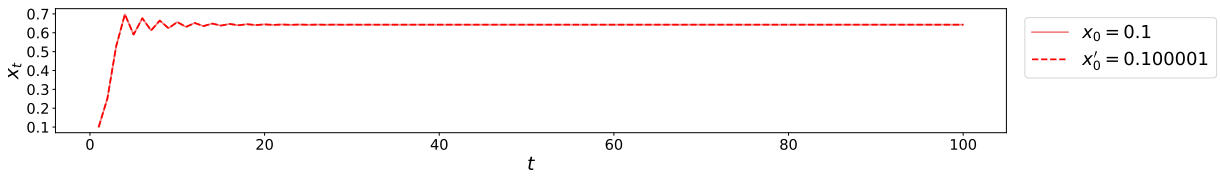

Figura 3 – Órbitas de dois pontos próximos para $R = 2,8$

Fonte: Elaborada pela autora.

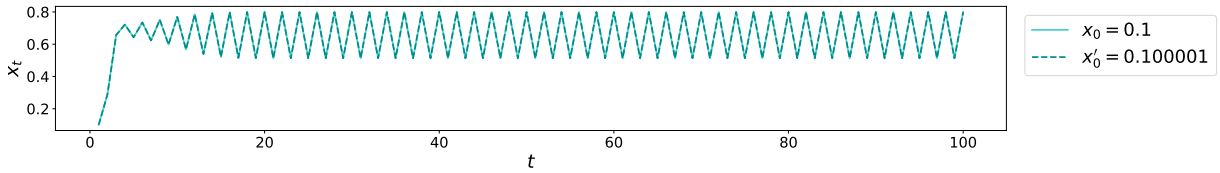

Figura 4 – Órbitas de dois pontos próximos para $R = 3,2$

Fonte: Elaborada pela autora.

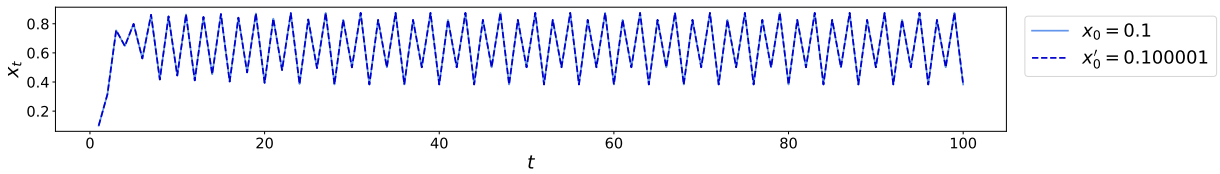

Figura 5 – Órbitas de dois pontos próximos para $R = 3,5$

Fonte: Elaborada pela autora.

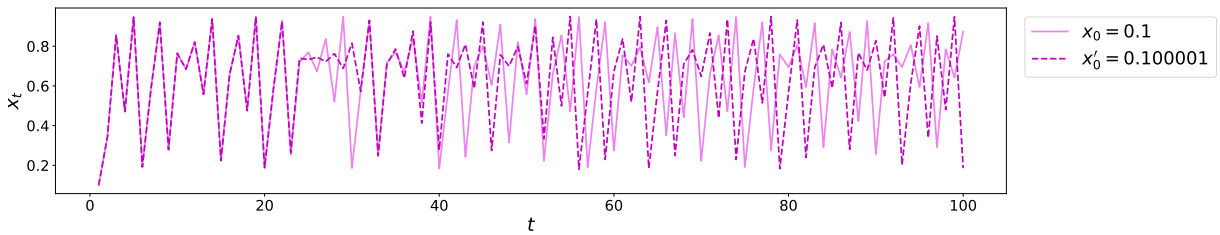

Figura 6 – Órbitas de dois pontos próximos para $R = 3,8$

Fonte: Elaborada pela autora.

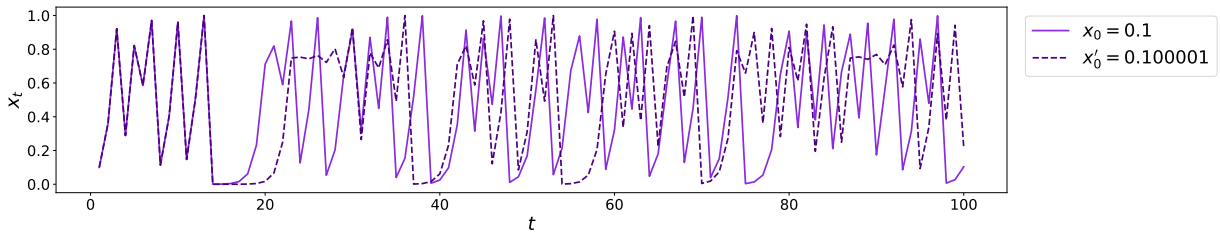

Figura 7 – Órbitas de dois pontos próximos para $R = 4$

Fonte: Elaborada pela autora.

2.3.2 Bifurcação e rota para o caos

Uma forma de ilustrar com clareza como são as órbitas do mapa logístico para diferentes valores do parâmetro R é através do diagrama de bifurcação. A Figura 8 consiste nesse diagrama. Ela mostra os valores de x_t de órbitas do mapa logístico (ignorando os valores de transiente) geradas para diferentes valores do parâmetro R. A partir desse diagrama é possível ver com clareza o comportamento das órbitas. Para $R < 3$, as órbitas convergem para um só ponto. Para $R > 3$, as órbitas passam a convergir para uma oscilação de período 2. A partir de aproximadamente $R = 3,45$, há uma estabilização em uma oscilação com período 4 e assim sucessivamente.

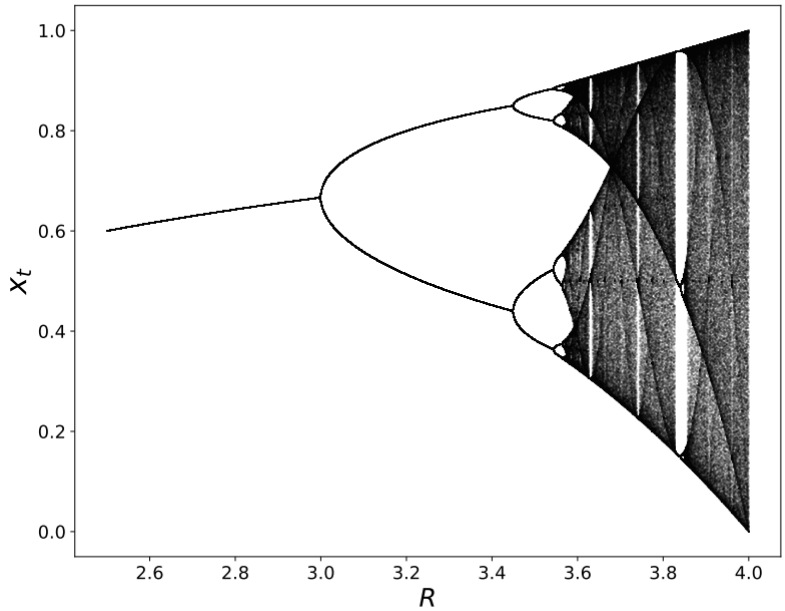

Figura 8 – Diagrama de bifurcação do mapa logístico

Fonte: Elaborada pela autora.

É notável que após a primeira bifurcação surgir, cada vez um incremento menor

em R é necessário para o surgimento das próximas. O físico Mitchell Jay Feigenbaum, interessado em buscar padrões no caos, notou que a razão do intervalo entre duas bifurcações sucessivas convergia para um valor, denominado constante de Feigenbaum. (4) Se a_n são os valores do parâmetro R em que surgem as bifurcações, a constante de Feigenbaum é dada pelo limite:

$$\delta = \lim_{n\to\infty} \frac{a_{n-1} - a_{n-2}}{a_n - a_{n-1}} = 4,669201... \tag{2.7}$$

Mais tarde, foi descoberto que essas características não eram exclusivas para o mapa logístico e eram universais para mapas unidimensionais com um único máximo quadrático. (29–31)

Conforme R aumenta, a distância entre duas bifurcações consecutivas diminui. Ela tende a zero quando $R \to 3,56995$, e o regime caótico começa. É possível notar que após esse valor ainda existem zonas de estabilidade, em que as órbitas voltam a ser periódicas.

2.3.3 Expoente de Lyapunov

Para determinar o expoente de Lyapnunov para o caso específico do mapa logístico, a Equação 2.4 fica:

$$\lambda(x_0) = \lim_{t\to\infty} \frac{1}{t} \sum_{i=0}^{t-1} \ln |R - 2Rx_i|. \tag{2.8}$$

A Figura 9 mostra os valores dos expoentes de Lyapunov de órbitas do mapa logístico para diferentes valores do parâmetro R. É fácil verificar que os valores são negativos até por volta de $3,56995$, que é onde o regime caótico começa. Também é possível constatar que existem algumas regiões em que o expoente de Lyapunov volta a ser negativo, que são as zonas de estabilidade mencionadas anteriormente.

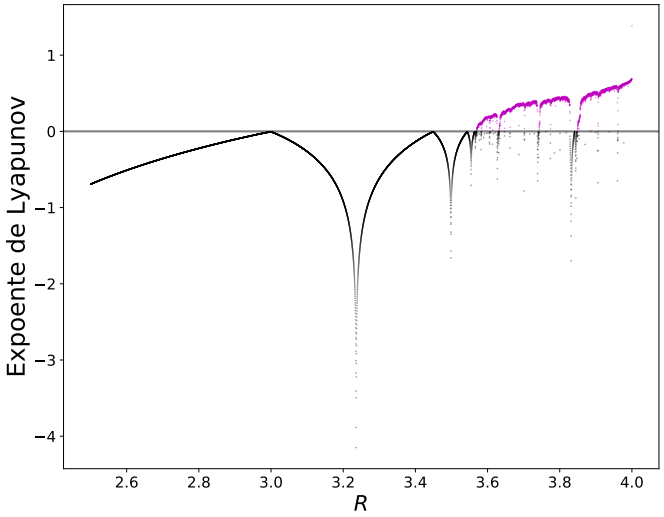

Figura 9 – Expoente de Lyapunov

Fonte: Elaborada pela autora.

3 APRENDIZADO DE MÁQUINAS

O aprendizado de máquinas é uma área de estudos que surgiu por volta de 1950 e foi construída ao longo dessa década graças aos trabalhos de diversos pesquisadores como Donald Hebb (32), Warren McCulloch, Walter Pitts (33), Arthur Samuel (34), Alan Turing (35), Frank Rosenblatt (36), entre outros. Embora muitos pesquisadores considerem o aprendizado de máquina uma área dentro de inteligência artificial, outros preferem classificá-las como áreas separadas, mas próximas e que se sobrepõem em diversos momentos. Independente da definição exata, é inegável que essas áreas caminham juntas. A definição mais popular do conceito de aprendizado de máquinas foi dada por Arthur Samuel, que também foi quem cunhou o termo, como sendo "o campo de estudo que dá aos computadores a habilidade de aprender sem serem explicitamente programados". De forma geral, pode-se dividir o processo de aprendizagem em três tipos:

- supervisionado
- não supervisionado
- por reforço

O aprendizado supervisionado consiste em apresentar a uma máquina um conjunto de dados de entrada juntamente com a saída esperada. Com esses dados, a máquina ajusta seus parâmetros de forma a tentar inferir uma regra de generalização para aquela determinada tarefa, como ilustrado no diagrama da Figura 10. No aprendizado não supervisionado, por sua vez, a máquina não tem uma referência explícita sobre as entradas que recebe, no entanto, ela consegue associar ou agrupar entradas semelhantes, podendo assim, ao receber novas entradas, determinar a qual grupo de entradas ela apresenta maior similaridade. Já o aprendizado por reforço se dá pela obtenção de pontuações ou penalizações como resultado da realização de tarefas. Nesse sentido, o algoritmo faz com que o agente explore opções de forma a tentar maximizar a pontuação total obtida e é um tipo de aprendizagem bastante utilizado na robótica. No entanto, vale ressaltar que existem aprendizados híbridos, como o semi-supervisionado, além de outras variações, como o aprendizado por transferência ou online.

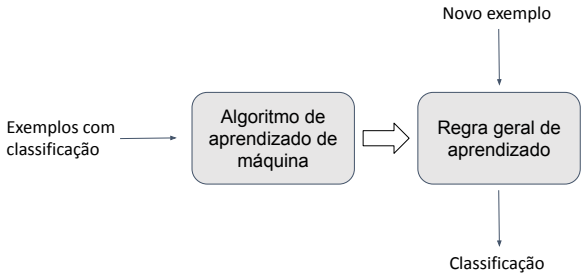

Figura 10 – Diagrama geral do aprendizado supervisionado.

Fonte: Elaborada pela autora.

Dentre os métodos de aprendizados de máquinas, além das redes neurais, podemos citar máquina de vetores de suporte (SVM) (37), árvore de decisão, k-médias (38), k-vizinhos (39) e agrupamento hierárquico. (40) As redes neurais são hoje o método de aprendizado de máquina em maior destaque, especialmente quando se trata de tarefas com maior complexidade. Assim como os outros algoritmos de aprendizado de máquina, as redes neurais podem aprender de forma supervisionada, não supervisionada, por reforço, ou por outras variações. Diversas tarefas realizadas por outros algoritmos de aprendizado de máquina podem ser também realizados pelas redes neurais, muitas vezes até de forma mais sofisticada. Um exemplo seria a classificação de imagens, que, com as redes convolucionais, basta usar aquelas como entrada e treinar a rede, enquanto se quiséssemos usar algoritmos mais tradicionais de aprendizado de máquina, como SVM, seria necessário antes usar extratores de características adequados nas imagens para então classificar vetores de características obtidos por eles. Atualmente, as redes neurais conseguem realizar as mais diversas atividades como, por exemplo, reconhecimento de fala (41), criação de vídeos falsos (42), jogar Go (43) ou até mesmo dirigir carros. (44)

3.1 Breve histórico e conceitos básicos de redes neurais

Inspirados em neurônios biológicos, em 1943, o neuroanatomista Warren McCulloch e o matemático Walter Pitts elaboraram um modelo matemático de um neurônio artificial. (33) Analogamente aos dendritos, o neurônio artificial recebe informações de entrada e após processá-las, dispara um resultado de saída (análogo ao axônio). Os valores de entrada são multiplicados por um peso, o que biologicamente pode ser comparado com a espessura de um neurônio. No que seria análogo ao corpo celular, as entradas são multiplicadas por seus respectivos pesos e a soma total destes é comparada a um limiar. Nesse modelo proposto, a saída é binária: para valores maiores que o limiar, ela é 1, e para valores menores, 0. Ou seja, o neurônio pode ou não disparar um sinal. As Figuras 11 e 12 mostram uma esquematização de um neurônio biológico e o neurônio de McCulloch e Pitts,

respectivamente, enquanto a Equação 3.1 diz respeito a como a informação é processada por este.

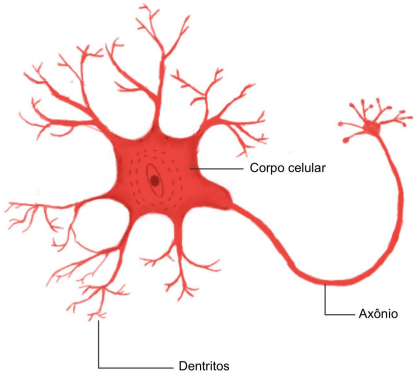

Figura 11 – Esquema de um neurônio.

Fonte: Elaborada pela autora.

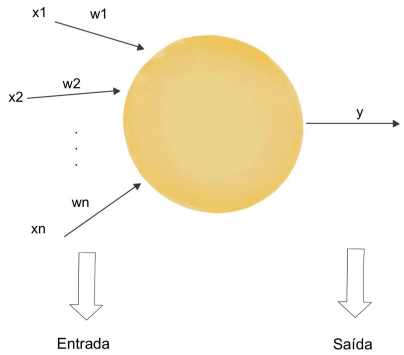

Figura 12 – Modelo do neurônio de McCulloch e Pitts

Fonte: Elaborada pela autora.

$$t_{mp} = \sum_{i=1}^{N} w_i x_i \qquad y_{mp} = \begin{cases} 1, & \text{se } t_{mp} > \mu \\ 0, & \text{se } t_{mp} < \mu \end{cases} \tag{3.1}$$

Em 1949, Donald Hebb propõe o conceito de aprendizado Hebbiano (32), em que define um postulado para descrever como as conexões entre neurônios pode ser alterada com a estimulação destes.

Regra de Hebb: *"quando um axônio da célula A esta próximo o suficiente para excitar a célula B e repetidamente ou persistentemente segue fazendo com que a célula*

dispare, algum processo de crescimento ou alteração metabólica ocorre em uma ou ambas as células, de forma que aumente a eficácia de A, como uma das células capazes de fazer com que B dispare."

Matematicamente, o peso w_{ij} que conecta um neurônio x_i a um neurônio y_j segundo a regra de Hebb é dado pela Equação 3.2

$$w_{ij} = x_i y_j \tag{3.2}$$

Em 1958, Frank Rosenblatt inventou o Perceptron (36), usando como unidades básicas o neurônio artificial de Mcculoch e Pitts. Um esquema geral do perceptron é ilustrado na Figura 13 A forma mais geral de descrever matematicamente como a informação é processada em uma unidade neural é dada pela Equação 3.3. A adição de um peso de viés b adiciona um grau de liberdade, e a função de ativação (ou função de transferência) σ pode assumir outras formas além da degrau descrita inicialmente no modelo de Mcculoch e Pitts. Algumas das funções de ativação mais populares são apresentadas na Figura 14.

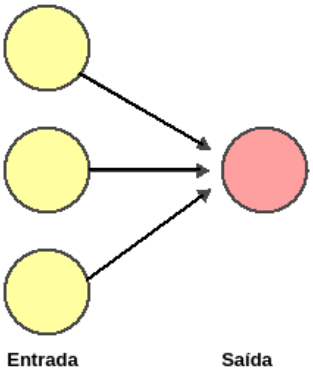

Figura 13 – Esquema de um Perceptron.

Fonte: Elaborada pela autora.

$$t = \sum_{i=1}^{N} w_i x_i + b. \qquad y = \sigma(t). \tag{3.3}$$

O algoritmo para atualização dos pesos do perceptron foi inspirado na regra de aprendizado de Hebb, com a adição da taxa de aprendizagem η, de valor positivo menor do que 1, que define o tamanho do passo de alteração do valor dos pesos w. A Equação 3.4 define a regra proposta para atualizar os pesos do perceptron.

$$w_{t+1} = w_t + \eta x_i y_i. \tag{3.4}$$

Pouco mais tarde, em 1960, Widrow buscou refinar a ideia do Perceptron com a rede Adaline. (45) Foi proposto um novo algoritmo superior para treinar essa rede denominado LMS, regra delta ou ainda, regra de Widrow-Hoff. Ele se baseia no método dos mínimos quadrados para atualizar os pesos e, para um peso w_i, é dado pela Equação 3.5.

$$w_{t+1} = w_t + \eta E x_i. \tag{3.5}$$

onde E é a diferença ente o valor obtido pela rede e o valor desejado. Além das limitações de recursos computacionais, em 1969, após e Minsky e Papert (46) apontarem grandes limitações que o o Perceptron em si possuía, como a incapacidade de resolver qualquer problema que não fosse linearmente separável. Dessa forma, a ideia acabou abandonada pela maior parte da comunidade científica por anos.

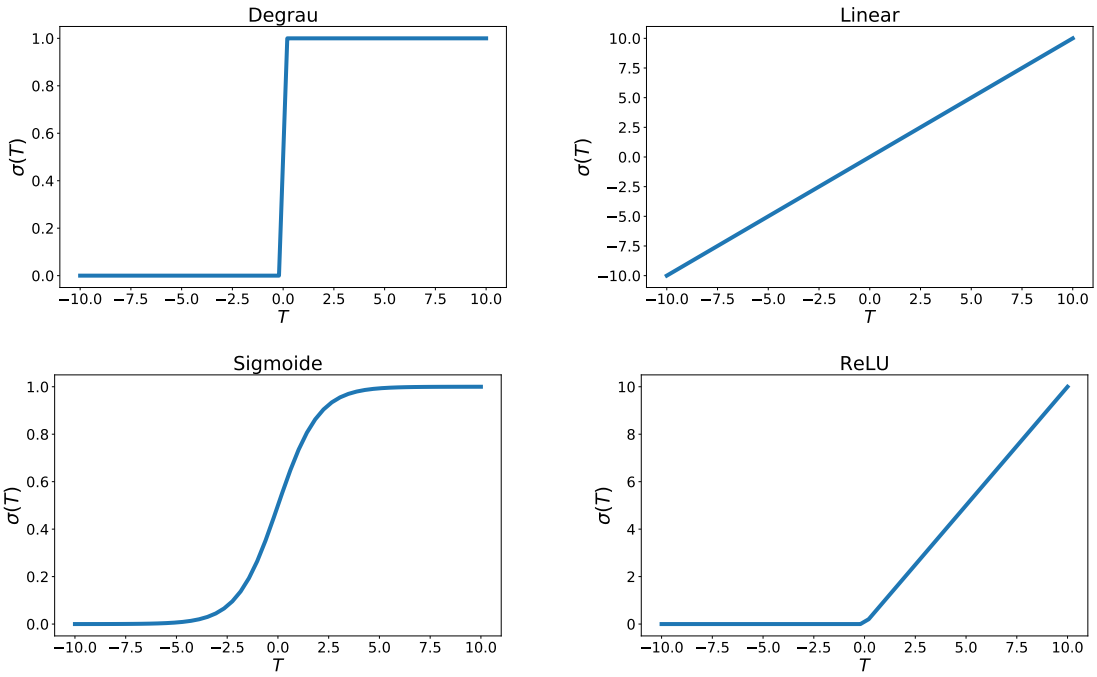

Figura 14 – Funções de ativação.

Fonte: Elaborada pela autora.

Em 1986, o trabalho de Rumelhart (47) popularizou um novo algoritmo: o *back-propagation*, embora ele já tivesse sido descoberto de forma independente por outros pesquisadores. (48) O algoritmo do *backpropagation* fornece uma método para treinar uma rede *feedfoward* com mais do que uma camada, tornando possível acrescentar mais camadas ao Perceptron, dando origem, assim, ao Multi Layered Perceptron (MLP) ou Perceptron Multi Camadas, o que resolveria os problemas e limitações apontados por Misnky e Papert a respeito do Perceptron. O *backpropagation* foi responsável pelo ressurgimento da área

das redes neurais, e o MLP é uma das arquiteturas mais usadas até os dias atuais. Um esquema de sua estrutura básica é ilustrado na Figura 15.

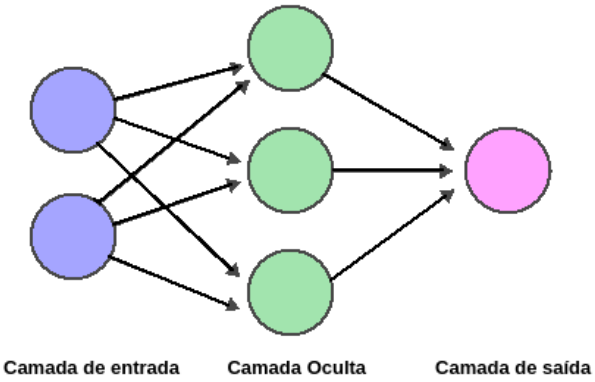

Figura 15 – Esquema de um MLP com uma camada oculta.

Fonte: Elaborada pela autora.

O trabalho de Rumelhart também inspirou a criação de redes recorrentes simples na década de 80, embora já existissem outros tipos de redes que também possuem recorrência, como as de Hopfield. (49) As redes neurais recorrentes (RNN) são uma categoria de rede que possuem em sua arquitetura conexões de *feedback* entre seus neurônios, simulando assim, um efeito análogo à memória.

O interesse nas redes neurais foi aos poucos restaurado após a popularização do *backpropagation* e as novas arquiteturas de redes que surgiram devido a esse algoritmo, além de um avanço no poder computacional. Em 2012, o interesse da comunidade científica por redes neurais se intensificou graças aos resultados extraordinários alcançado por redes convolucionais (CNN) (50) de camadas profundas na classificação de imagens. (51)

As *Echo State Networks* (ESN) foram inventadas por Jaeger (10) e consistem em um tipo de rede recorrente de reservatório. *Reservoir Computing* é um termo guarda chuva utilizado para se referir a uma classe de redes neurais que possuem um grande reservatório de neurônios esparsamente conectados. Apesar de terem uma arquitetura recorrente, seu funcionamento é distinto das outras redes recorrentes mencionadas, uma vez que só o treinamento dos pesos que ligam o reservatório com a saída é necessário. Fazem parte dessa categoria as *Echo State Networks* (9) e as *Liquid State Machines* (52), ambas desenvolvidas de forma independente e simultânea. Essa categoria de rede tem se mostrado particularmente eficiente para lidar com sistemas dinâmicos não lineares.

3.1.1 *Backpropagation*

A premissa do *backpropagation* consiste em verificar o quão próximo o resultado obtido por uma rede y está do resultado esperado Y e, posteriormente, diminuir a distância

entre esses vetores de Y e y. Para isso, seleciona-se uma função de perda J. Podemos defini-la como o erro médio quadrático como na Equação 3.6, mas essa não é a única possibilidade. Para cada par de vetores y e Y usados para treinar a rede, se j, podemos definir a função de perda como na Equação 3.6. Dessa forma, o objetivo do *backpropagation* é buscar os valores dos pesos que minimizem essa função de perda. Para fazê-lo, usamos os otimizadores, sendo o mais popular o gradiente descendente. (53, 54)

$$J = \frac{1}{2}\sum_{j}(Y - y)^2. \tag{3.6}$$

É sabido que o gradiente de uma função aponta na direção do máximo crescimento desta. Da mesma forma, o negativo do gradiente de uma função aponta na direção da maior taxa de decrescimento dela. Por conseguinte, para encontrar o mínimo de uma função deve-se caminhar na direção oposta ao gradiente. Para podermos usar o gradiente descendente com objetivo de minimizar a função de perda J, é necessário calcular as derivadas parciais dela em relação aos pesos w_{ij} (peso que liga o neurônio x_i ao neurônio y_j). Aplicando a regra da cadeia, temos:

$$\frac{\partial J}{\partial w_{ij}} = \frac{\partial J}{\partial y_j}\frac{\partial y_j}{\partial w_{ij}} = \underbrace{\frac{\partial J}{\partial Y_j}}\underbrace{\frac{\partial Y_j}{\partial t_j}}\underbrace{\frac{\partial t_j}{\partial w_{ij}}}. \tag{3.7}$$

Calcularemos cada um desses três fatores separadamente. Usando a definição de t (Equação 3.3) para calcular o último fator, temos:

$$\frac{\partial t_j}{\partial w_{ij}} = \frac{\partial(\sum_{k=1}^{N} w_{kj}x_{kj} + b)}{\partial w_{ij}} = \frac{\partial w_{ij}x_i}{\partial w_{ij}} = x_i. \tag{3.8}$$

Pela própria definição de y (Equação 3.3), sabemos que a derivada parcial de y em relação a t é a própria derivada da função de ativação usada. Caso a função de ativação utilizada seja a sigmoide, o segundo fator fica da seguinte forma:

$$\frac{\partial y_j}{\partial t_j} = \frac{\partial \sigma(t_j)}{\partial t_j} = \sigma(t_j)(1 - \sigma(t_j)) = y_j(1 - y_j). \tag{3.9}$$

Para encontrar a derivada de J em função de y, primeiro é preciso definir em que camada o neurônio está. Se ele está na camada de saída, então ele é calculado como em 3.10, se ele está nas camadas ocultas, então ele é calculado como em 3.12.

$$\frac{\partial J}{\partial y_j} = -(Y_j - y_j). \tag{3.10}$$

Dessa forma, no caso do neurônio estar na camada de saída, a Equação 3.7 se torna:

$$\frac{\partial J}{\partial w_{ij}} = -\underbrace{(Y_j - y_j)y_j(1-y_j)}_{\delta_j} x_i. \tag{3.11}$$

Se o neurônio não está na camada de saída, então a derivada parcial da função de perda em relação a y_j se torna:

$$\frac{\partial J}{\partial y_j} = \sum_k \frac{\partial J}{\partial t_k}\frac{\partial t_k}{\partial y_j} = \sum_k \frac{\partial J}{\partial t_k}\frac{\partial(\sum_j w_{kj}y_j)}{\partial y_j} = \sum_k \frac{\partial J}{\partial t_j} w_{kj}. \tag{3.12}$$

Logo, a derivada parcial da função de perda em relação aos pesos, caso eles não se liguem aos neurônios da camada de saída é dada pela Equação 3.13.

$$\frac{\partial J}{\partial w_{ij}} = (\sum_k \frac{\partial J}{\partial t_j} w_{kj}) y_j(1-y_j)x_i. \tag{3.13}$$

3.1.2 Otimizadores

Conhecer o gradiente fornece a direção em que se deve andar de forma a minimizar a função de perda, no entanto, não se sabe o quanto. Para isso define-se uma taxa de aprendizagem η pequena. O gradiente descendente atualiza os pesos conforme a Equação 3.14.

$$w_{t+1} = w_t - \eta \nabla_{w_t} J. \tag{3.14}$$

O gradiente descendente passa por todos os dados de treinamento antes de atualizar os pesos, o que pode ser um processo demorado. O gradiente descendente estocástico (SGD) (55) realiza o mesmo processo que o gradiente descendente, mas levando em consideração apenas uma entrada de treinamento por vez para atualizar os pesos.

Um dos maiores desafios dos métodos de gradiente descendente é a dificuldade na escolha da taxa de aprendizagem. Caso ela seja grande, a atualização de pesos pode passar pelo mínimo sem notá-lo, mas caso ela seja pequena, o processo se torna muito lento. Além disso, o algoritmo pode encontrar mínimos locais ao invés do mínimo global. Para contornar esse desafio, foi proposto um método otimizado do gradiente descendente, adicionando o termo *momentum*. A adição dessa constante permite um aumento da velocidade de aprendizado e promove estabilização quando adequado. A atualização de pesos então ocorre conforme as Equações 3.15 e 3.16.

$$\nu_{t+1} = \nu_t + \alpha \nabla_{w_t} J, \tag{3.15}$$

$$w_{t+1} = w_t - \eta\nu. \tag{3.16}$$

Outros modelos de otimizadores baseados em gradiente descendente foram propostos, como, por exemplo, o AdaGrad, RMSprop e Adam. O otimizador AdaGrad (56) propõe que a taxa de aprendizado mude para cada peso e conforme as iterações, o que dá maior grau de liberdade para o algoritmo buscar o mínimo da função de perda. Ele faz isso mantendo a soma do quadrado dos gradientes em um termo G e, na atualização dos pesos, usando uma taxa de aprendizagem atualizada que é obtida dividindo o valor inicial da taxa de aprendizagem pela raiz quadrada de G. A atualização de pesos é feita conforme as Equações 3.17 a 3.19. O termo ϵ é uma constante de valor muito pequeno, usada apenas para que o denominador nunca seja nulo.

$$G_{t+1} = \sum_{k=1}^{t} \nabla_{w_t} J^2, \tag{3.17}$$

$$\eta'_{t+1} = \frac{\eta}{\sqrt{G_{t+1}} + \epsilon}, \tag{3.18}$$

$$w_{t+1} = w_t - \eta'_{t+1} \nabla_{w_t} J. \tag{3.19}$$

O AdaGrad apresenta um problema quando o termo G se torna grande demais, tornando a taxa de aprendizado η' pequena e fazendo com o que o treinamento fique demasiadamente lento. Já o otimizador RMSProp (57) consegue contornar essa dificuldade acrescentando um fator $(1 - \beta)$ no cálculo de G que não permite que a soma do quadrado dos gradientes cresça indefinidamente. A atualização de pesos desse otimizador é dada pelas Equações 3.20 a 3.22.

$$G_{t+1} = \beta G_t + (1 - \beta) \nabla_{w_t} J^2, \tag{3.20}$$

$$\eta'_{t+1} = \frac{\eta}{\sqrt{G_{t+1}} + \epsilon}, \tag{3.21}$$

$$w_{t+1} = w_t - \eta'_{t+1} \nabla_{w_t} J. \tag{3.22}$$

O otimizador Adam (58), por sua vez, foi proposto como uma combinação das vantagens apresentadas pelos otimizadores AdaGrad e RMSProp com a adição de um termo análogo ao *momentum*. A atualização de pesos é feita conforme a Equações 3.23 a 3.26.

$$M_{t+1} = \gamma_1 M_t + (1 - \gamma_1) \nabla_{w_t} J, \tag{3.23}$$

$$G_{t+1} = \gamma_2 G_t + (1 - \gamma_2)\nabla^2_{w_t} J, \tag{3.24}$$

$$\eta'(t+1) = \frac{\eta}{\sqrt{G_{t+1} + \epsilon}}, \tag{3.25}$$

$$w_{t+1} = w_t - \frac{\eta}{\sqrt{G_{t,ii} + \epsilon}} M_{t+1}. \tag{3.26}$$

Os vetores M e G são denominados momento primário e secundário e são inicializados como nulos, o que faria com que primeiro passo da atualização dos pesos fosse muito grande. Para contornar esse problema, é necessária a inclusão das correções a seguir:

$$M'_{t+1} = \frac{M_{t+1}}{1 - \gamma_1^t}, \tag{3.27}$$

$$G'_{t+1} = \frac{G_{t+1}}{1 - \gamma_2^t}. \tag{3.28}$$

3.2 Redes Recorrentes Simples

Ao contrário das redes *feedforward*, como o MLP, em que a informação é propagada da primeira à última camada, as redes recorrentes possuem loops internos, guardando as informações de entradas anteriores em seu estado oculto. Dessa forma, essas redes são as mais indicadas para processar dados sequenciais, como textos, vídeos, séries temporais, DNA, áudios, entre outros. Essas redes foram capazes de realizar atividades como aprender representação de frases (59), gerar textos (60), e fazer reconhecimento de fala. (61) A Figura 16 mostra um esquema geral de uma rede recorrente com uma camada.

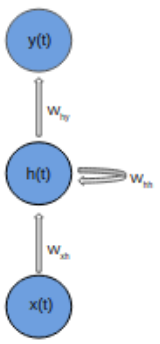

Figura 16 – Esquema de uma RNN simples.

Fonte: Elaborada pela autora.

Seja $x(t)$ a função de entrada, $y(t)$ a função de saída e $h(t)$ a função do estado oculto, onde t se refere a t-ésima iteração. Se W_{xh} é a matriz de pesos que conecta a função de entrada com a do estado oculto, W_{hy} os pesos que conectam a função de estado oculto com a de saída e W_{hh} é a matriz de pesos de estado oculto ou matriz de transição, o estado oculto $h(t)$ e a saída $y(t)$ são calculados da seguinte forma:

$$h(t) = \sigma_h(x(t)W_{xh} + h(t-1)W_{hh}), \tag{3.29}$$

e

$$y(t) = \sigma_y(h(t)W_{hy}), \tag{3.30}$$

onde σ_h se trata da função de ativação da camada oculta e σ_y, da camada de saída.

O modelo descrito de rede recorrente possui um defeito intrínseco chamado de problema da dissipação e explosão do gradiente. (62) Ele consiste na dificuldade da rede para capturar dependências de longo prazo devido ao fato de seu gradiente poder aumentar ou diminuir exponencialmente com respeito ao número de iterações. As redes do tipo Long Short-Term Memory surgiram com o intuito de resolver esse problema. (63)

3.3 Long Short-Term Memory

A rede LSTM é uma adaptação das rede recorrentes simples, projetada para lidar de forma mais adequada com o problemas de dissipação e explosão do gradiente. Para isso, a arquitetura inclui novas estruturas denominadas portões (ou *gates*). Existem três tipos de portões: os de entrada, de saída e de esquecimento. (64) Com eles, a rede consegue selecionar quais informações são relevantes de serem mantidas e quais devem ser esquecidas. As LSTMs são responsáveis pelos avanços mais notáveis dentre as redes recorrentes e são umas das redes mais utilizadas no contexto do deep learning. As LSTMs foram capazes de traduzir textos (65), gerar textos em letra cursiva (66), reconhecer emoções em vozes (67), compor música (68) e, juntamente com as redes convolucionais, de criar legendas para imagens (69) e vídeos. (70) A Figura 17 ilustra o esquema de uma célula de LSTM.

Precisamos definir as matrizes de peso dessa célula. A matriz que conecta a função de entrada com o portão de entrada será denominada W_{xi}, a que conecta a função de entrada com o portão de esquecimento, W_{xf}, a que conecta a entrada com o portão de saída, W_{xo} e a que conecta a entrada com o estado da célula, W_{xc}. A matriz de pesos W_{hi} liga o estado oculto com o portão de entrada, W_{hf} liga o estado oculto com a portão de esquecimento, W_{hc} liga a função de estado oculto com o estado da célula e W_{ho} liga o estado oculto com o portão de saída.

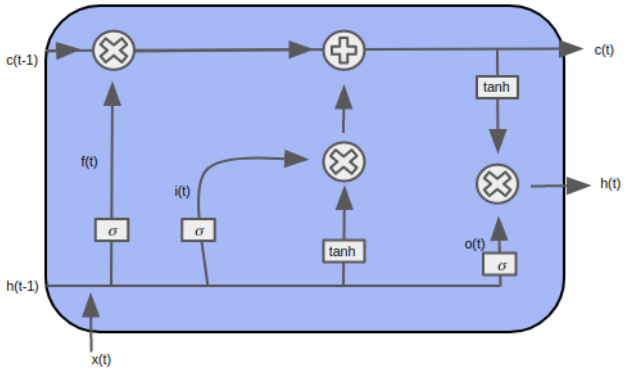

Figura 17 – Esquema de uma célula da LSTM.

Fonte: Elaborada pela autora.

Sejam b_i, b_o, b_f e b_c os vetores de *bias*. Se o estado de uma célula de uma LSTM é denominado $c(t)$, a entrada da mesma, $x(t)$, o vetor de estado oculto, $h(t)$ e os *gates* de entrada, saída e esquecimento são denominados $i(t)$, $o(t)$ e $f(t)$, respectivamente, as Equações 3.31 3.32 e 3.33 definem como as funções dos portões de uma célula da LSTM são calculados.

$$i(t) = \sigma_g(x(t)W_{xi} + h(t-1)W_{hi} + b_x). \tag{3.31}$$

$$f(t) = \sigma_g(x(t)W_{xf} + h(t-1)W_{hf} + b_f). \tag{3.32}$$

$$o(t) = \sigma_g(x(t)W_{xo} + h(t-1)W_{ho} + b_o). \tag{3.33}$$

A função do estado da célula é dada pela Equação 3.34.

$$c(t) = f(t)c(t-1) + i(t)\sigma_c(x(t)W_{xc} + h(t-1)W_{hc} + b_c), \tag{3.34}$$

onde σ_g se refere a uma função de ativação sigmoidal e σ_c, a uma função de ativação tangente hiperbólica. O estado oculto é dado por:

$$h(t) = f(t)\sigma_g(c(t)). \tag{3.35}$$

3.4 Echo State Networks

A computação de reservatório (71) engloba redes que possuem um reservatório que exibe comportamento dinâmico não linear. O reservatório consiste em um conjunto de

neurônios de tamanho fixo N esparsamente conectados. Por apresentarem conexões de feedback, essas redes são também consideradas redes recorrentes, embora o seu treinamento se dê de forma distinta.

As *Echo State Networks* (ESNs) fazem parte da categoria computação de reservatório. Elas foram capazes de prever a série temporal de Mackey-Glass com acurácia 2400 vezes maior aos métodos anteriores (9), e a de Kuramoto-Sivashinsky por 8 vezes o seu tempo de Lyapunov. (72) Um esquema básico de uma rede *Echo State* é ilustrado na Figura 18.

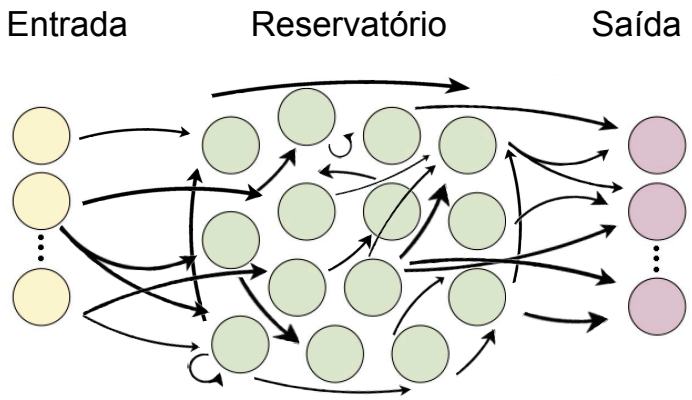

Figura 18 – Esquema de uma *Echo State Network*.

Fonte: Elaborada pela autora.

Nota-se que os neurônios da camada de entrada estão conectados ao reservatório pelos pesos W_{in}, os neurônios do reservatório estão conectados à saída pelos pesos W_{out} e os neurônios do reservatório são conectados entre si pela matriz de pesos W. É possível que haja conexões entre pesos de entrada e saída ou entre os pesos de saída de volta ao reservatório. As *Echo State Networks* não só possuem uma arquitetura simples, como o treinamento também. Ao contrário das redes tradicionais treinadas com *backpropagation*, nem todos os pesos de uma ESN precisam ser modificados ao longo do seu treinamento. W_{in} é inicializada aleatoriamente e não sofre alterações ao longo do treinamento da rede, apenas seus pesos de saída W_{out}. No entanto, para que isso seja possível, a matriz de pesos do reservatório W deve ser criada de forma a possuir características específicas, apresentando comportamento dinâmico não linear e alta dimensionalidade. Dessa forma, a entrada é mapeada para um espaço de alta dimensão, sendo posteriormente o resultado lido para a camada de saída por um processo mais simples — no caso, regressão linear.

Para que o reservatório exiba esse comportamento, é necessário que ele possua a propriedade *Echo State*. Essa propriedade garante que os sinais de entrada desaparecerão assintoticamente. Inicialmente, foi proposto — devido a observações empíricas — que para possuir a propriedade *Echo State*, a matriz W deveria ser normalizada de forma que seu

maior autovalor, denominado raio espectral, fosse menor que uma unidade. No entanto, uma definição mais geral para garantir que um reservatório possui essa propriedade é dada pelo teorema abaixo. (73)

Teorema: 1 *Uma ESN tem a propriedade Echo State se a matriz de pesos do seu reservatório W é Schur diagonalmente estável.*

Definição 1: *Uma matriz $W \in R^{n \times n}$ é dita ser Schur diagonalmente estável se e somente se existir uma matriz P diagonal definida positiva tal que*

$$W^T P W - P$$

é definida negativa.

O estado do reservatório de uma *Echo State Network* é dado pele Equação 3.36, onde $u(n)$ é a entrada e G a função de ativação.

$$x(n) = G(W x(n-1) + W_{\text{in}} u(n)). \tag{3.36}$$

Sendo c um vetor de correção, a saída é dada por:

$$y(n) = W_{\text{out}} x(n) + c. \tag{3.37}$$

Por conseguinte, para calcular os pesos de saída é necessário encontrar os valores que minimizam o erro quadrático médio do vetor de saída. A Equação 3.38 mostra uma solução conhecida como regressão com regularização de Tikhonov, onde I é a matriz identidade e β se refere a um coeficiente de regularização.

$$W_{\text{out}} = Y X^T (X X^T + \beta I)^{-1}. \tag{3.38}$$

4 EXPERIMENTOS

4.1 Órbitas Caóticas

Foram selecionadas cinco órbitas caóticas do mapa logístico para serem analisadas: todas foram geradas usando $X_0 = 0,1$ e, para cada uma delas, os valores do parâmetro R indicados na Tabela 1 foram utilizados. Na obtenção dos termos da órbita, é importante ressaltar que precisamos descartar os valores de transiente, que não são claros quando estamos lidando com atratores estranhos, mas uma estimativa é de que deve-se eliminar três vezes a precisão utilizada. (74) Seja p um valor maior do que o número de 3 vezes a precisão utilizada, então só consideraremos os valores a partir dele.

$$X_0, X_1, X_2, X_3, X_4, X_5, X_6, X_7, X_8, \ldots, \underbrace{X_p, X_{p+1}X_{p+2}, X_{p+3}, X_{p+4}, X_{p+5}, X_{p+6}\ldots}_{\text{órbita caótica}}$$

Tabela 1 – Órbitas caóticas do mapa logístico selecionadas para análise.

Órbita	R
A	3,8
B	3,9
C	3,997
D	3,9998
E	4

Fonte: Elaborada pela autora.

4.2 Previsões de um termo com redes Long Short-Term Memory

4.2.1 Conjunto de dados

O primeiro objetivo é treinar redes LSTM para que prevejam um único termo futuro x_n, alimentando-as com uma sequência de dez termos consecutivos indo de x_{-9} a x_0, como esquematizado na Figura 19. Para cada uma das cinco órbitas caóticas escolhidas, foram criados *datasets* de diferentes tamanhos d para que seja possível verificar se o aumento do tamanho do conjunto de treinamento melhora a acurácia da previsão. Dentro de um conjunto de dados, todas as sequências pertencem a uma mesma órbita do mapa logístico, ou seja, foram geradas com o mesmo parâmetro R e valor inicial. Foram testados três conjuntos de dados de treinamento com tamanhos d 100,1000 e 5000. Com o objetivo

de prever um único termo futuro x_6, x_8 ou x_{10}, conjuntos de dados cuja entrada é uma sequência que vai de x_{-9} a x_0 foram gerados.

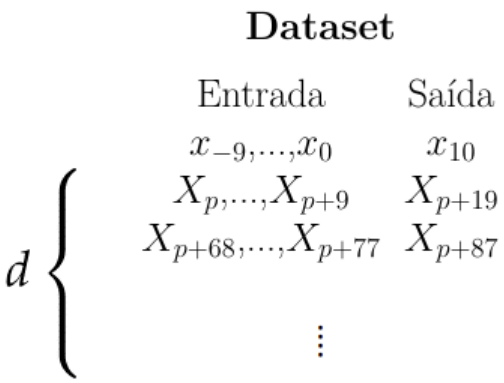

Figura 19 – Estrutura geral do *dataset* usado para treinar LSTMs para prever um único termo futuro.

Fonte: Elaborada pela autora.

A Figura 20 ilustra como é feita a obtenção dessas sequências para a criação de um conjunto de dados. Primeiro, escolhe-se um valor de p de forma a garantir que o transiente não está sendo incluído. Depois, são selecionados uma primeira sequência de entrada (de tamanho 10) e a saída (x_n, que nesse exemplo específico é x_{10}). A próximas sequências selecionadas estão espaçadas de forma que não exista sobreposições das entradas do *dataset*. Ademais, as sequências selecionadas são espaçadas por um intervalo de tamanho variável e aleatório.

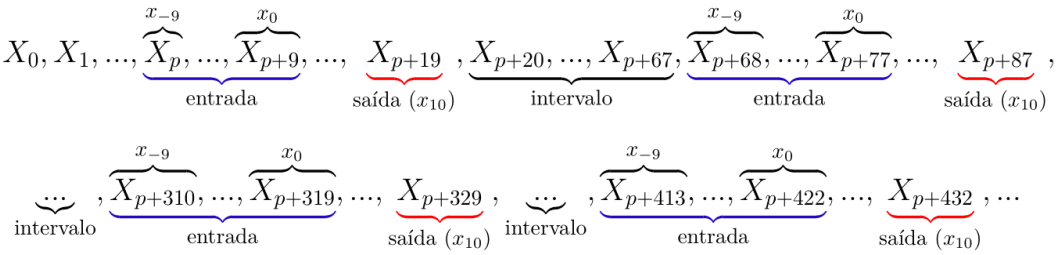

Figura 20 – Seleção de entradas e saídas de uma órbita para compor o *dataset*.

Fonte: Elaborada pela autora.

4.2.2 Parâmetros da rede e treinamento

As LSTMs treinadas possuem uma única camada oculta, e o número de neurônios nela foi variado de 5 a 1000, com o objetivo de analisar se esse aumento poderia promover

uma melhor previsão. Para treiná-las, a função de otimização utilizada foi a Adam, a função de ativação dos neurônios foi ReLU e a função de perda, *Mean Squared Error*. Em todos os casos, o treinamento foi feito com 70% do *dataset*.

4.3 Previsões de sequência com redes *Long Short-Term Memory*

4.3.1 Conjunto de dados

Para estudar a relação entre o número de dados usados para treinar a rede e acurácia, as redes LSTM foram treinadas com três diferentes tamanhos do conjunto de dados (d): 100, 1000 e 5000. Para analisar a relação entre o número de neurônios e acurácia, LSTMs com número de neurônios variando de 5 a 900 na camada oculta foram treinadas.

O conjunto de daods usado nesses testes são similares aos descritos na Seção 4.2, mas ao invés da saída ser apenas o termo x_n, ela consiste na sequência que vai de x_1 a x_n, sendo n no máximo dez. A estrutura do conjunto de dados usado está esquematizado na Figura 21. Similarmente, foram usados *datasets* com diferentes números de entradas d. A Figura 22 deixa claro como é feita a seleção de sequências para compor o conjunto de dados usados para treinar essas redes. O mesmo cuidado com espaçamento aleatório entre as sequências foi tomado assim como descrito anteriormente.

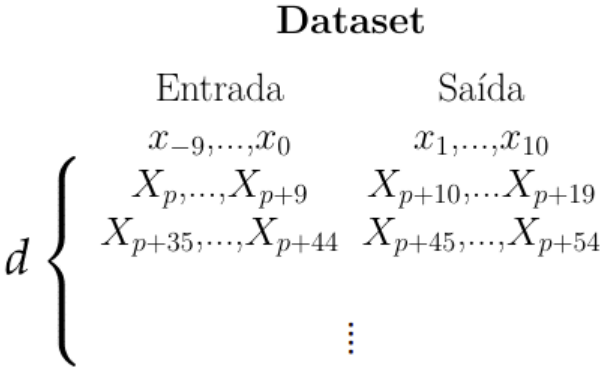

Figura 21 – Estrutura geral do *dataset* usado para treinar LSTMs para prever uma sequência futura.

Fonte: Elaborada pela autora.

4.3.2 Parâmetros da rede e treinamento

Assim como anteriormente, os treinamentos foram feitos com função de ativação ReLU, função de perda MSE e a função de otimização Adam.

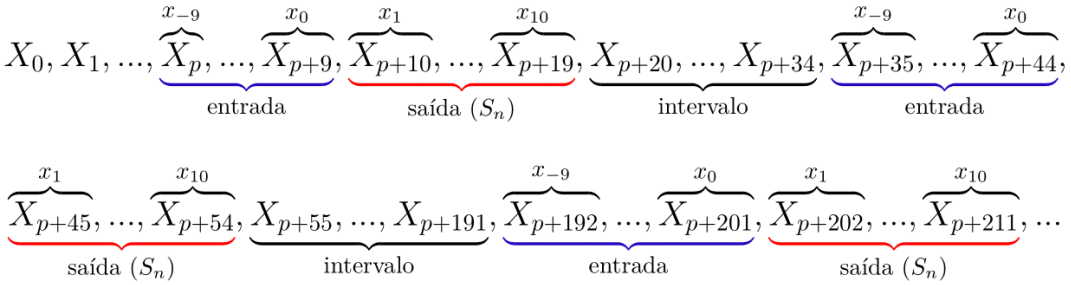

Figura 22 – Seleção de entradas e saídas de uma órbita para compor o *dataset*.

Fonte: Elaborada pela autora.

4.4 Previsões de sequência com *Echo State Networks*

4.4.1 Conjunto de dados

Os dados usados para treinar as redes *Echo State* consistem em uma sequência de entrada que vai de x_{-l} até x_0, e sua saída esperada é uma sequência S_n (vai de x_1 a x_{10}), conforme ilustrado na Figura 23. Assim como ilustrado na Figura 22, houve a cautela de selecionar sequências variadas, espaçadas por um intervalo de tamanho aleatório.

Figura 23 – Esquematização do treinamento das *Echo State Networks*.

Fonte: Elaborada pela autora.

4.4.2 Parâmetros da rede e treinamento

Os valores de l analisados foram 10, 20, 70, 100, 200, 500 e 800. Para cada entrada, alguns parâmetros da rede Echo State foram variados com o objetivo de obter o menor RMSE de uma sequência prevista S_n. O número de neurônios do reservatório foi variado até 2000, enquanto raio espectral foi variado de 0,001 a 0,99.

4.5 Comparação

Para fins comparativos, as acurácias obtidas para cada órbita caótica, a Tabela 2 indica os valores do expoente de Lyapunov das cinco órbitas analisadas calculados conforme descrito na Subseção 2.3.3.

Tabela 2 – Expoente de Lyapunov para algumas órbitas caóticas do mapa logístico

Órbita	Expoente de Layapunov
A	0,419782
B	0,483996
C	0,642937
D	0,664025
E	0,667523

Fonte: Elaborada pela autora.

Para sabermos quando a previsão de um termo futuro x_n de uma órbita caótica feita por uma rede passa ser igual ou pior a um chute, calculamos os erros médio absoluto, médio quadrático e raiz do erro médio quadrático considerando que valor da previsão como uma média de 100000 termos daquela órbita. Os valores estão na Tabela 3.

Tabela 3 – Erros obtidos quando o valor médio é usado como predição

Órbita	MAE	MSE	RMSE
A	0,211	0,0609	0,2467
B	0,2695	0,0895	0,2992
C	0,3096	0,119	0,345
D	0,316	0,1234	0,3512
E	0,3183	0,1253	0,3536

Fonte: Elaborada pela autora.

5 RESULTADOS E DISCUSSÕES

5.1 Previsões com *Long Short-Term Memory*

5.1.1 Influência do número de neurônios e tamanho do *dataset* na acurácia

Com o objetivo de prever um único termo futuro x_n, redes LSTM foram treinadas como descrito na Seção 4.2. A Figura 24 mostra o erro quadrático médio obtido por LSTMs em função do número de neurônios N_o na camada oculta para previsões de x_6, x_8 ou x_{10}.

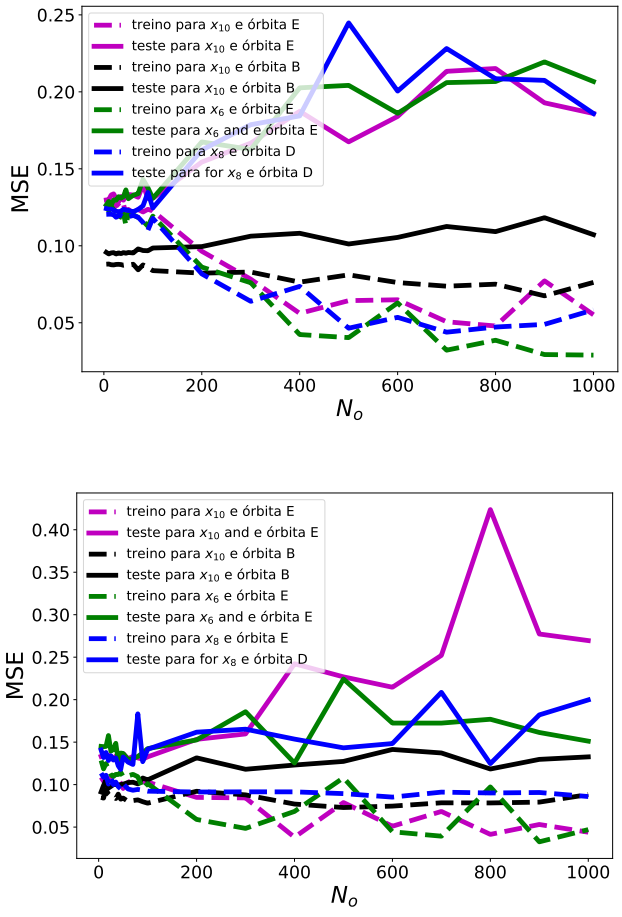

Figura 24 – Erro médio quadrático de treinamento e teste de x_6, x_8 e x_{10} para *datasets* de tamanhos $d = 1000$ e $d = 100$, respectivamente.

Fonte: Elaborada pela autora.

É possível constatar que, de modo geral, nos casos ilustrados na Figura 24, o aumento do número de neurônios na camada oculta não resultou em uma melhora na previsão dos termos futuros. Observou-se o fenômeno de *overfitting*, em que a rede cria uma regra muito específica para os dados de teste, não conseguindo inferir as regras gerais

que regem o conjunto de dados inteiro, uma vez que as redes conseguiram diminuir o erro dos dados de treinamento, no entanto, isso não foi refletido nos de teste, mostrando que a rede não foi capaz de aprender a regra de generalização pretendida. Outrossim, os erros são comparáveis aos obtidos chutando uma média, mostrados na Tabela 3. Dessa forma, a predição dos termos x_6, x_8 e x_{10} foi insatisfatória.

Para os próximos testes, o objetivo foi prever toda uma sequência de termos futuros S_n (que vai de x_1 a x_n), ao invés de um único termo. As entradas consistem em sequências com 10 valores (x_{-9} a x_0) e a saídas com n valores (x_1 a x_n), sendo n variado de 2 a 10. A Seção 4.3 contém mais detalhes sobre o treinamento dessas redes.

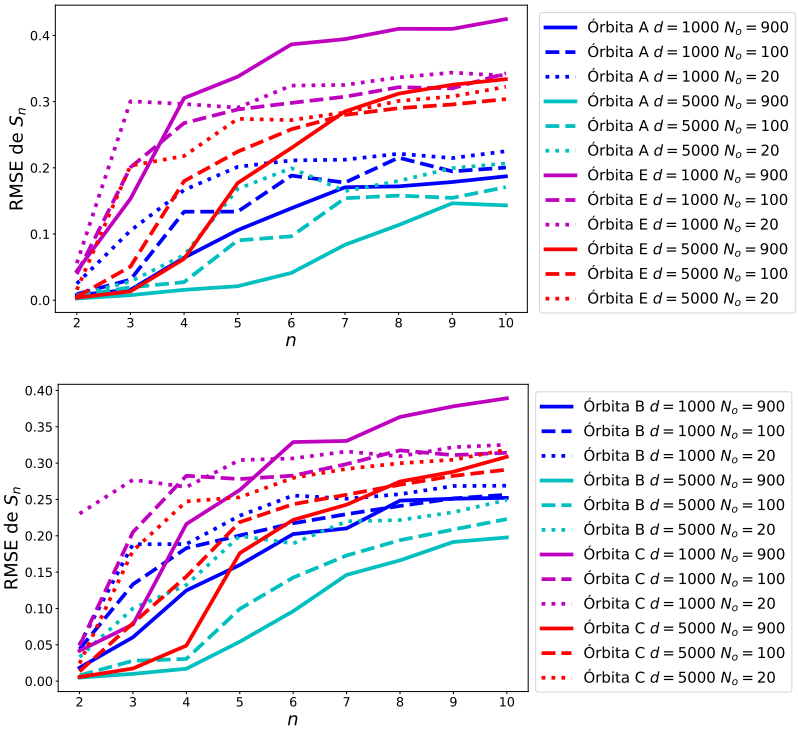

Figura 25 – RMSE de uma sequência prevista S_n.

Fonte: Elaborada pela autora.

Para estudar a relação entre o número de dados usados para treinar a rede e acurácia, as redes LSTM foram treinadas com três diferentes tamanhos do conjunto de dados d: 100, 1000 e 5000. O gráfico da Figura 25 mostra como a raiz do erro médio quadrático da previsão de uma sequência varia de acordo com o tamanho da sequência prevista S_n para diferentes tamanhos de *dataset* (d) e para diferentes números de neurônios na camada oculta (N_o). As curvas pontilhadas consistem em erros referentes a redes treinadas com $N_o = 20$ neurônios, já as curvas tracejada são referentes às treinadas com $N_o = 100$ neurônios enquanto as contínuas, $N_o = 900$. Nesses gráficos, as curvas verdes e azuis representam erros de sequências de órbitas com valor do expoente de Lyapunov menor, A e B, enquanto as roxas e vermelhas, de órbitas com expoente de Lyapunov

maior, C e E, sendo as curvas verde e roxa referentes a resultados obtidos com o dataset de tamanho $d = 1000$ e as curvas azul e vermelha, $d = 5000$.

Para as órbitas A e B, um aumento no número de neurônios resulta em uma diminuição no erro, enquanto para as órbitas C, D e E, o mesmo não ocorre. Entretanto, uma investigação um pouco mais detalhada se faz necessária, uma vez que os erros apresentados são da sequência inteira, e não de cada termo previsto especificamente. A Figura 26 mostra o RMSE de cada termo previsto x_i de uma sequência S_{10} de diferentes órbitas para três diferentes números de neurônios na camada oculta (20, 100 e 900). Nota-se que, para as órbitas C, D e E, o aumento do número de neurônios implica em um RMSE menor para os termos futuros mais próximos, mas em um RMSE maior para termos futuros mais distantes, resultando em um aumento do RMSE da sequência inteira ao aumentar o número de neurônios.

Ao tentar prever temos futuros das sequências, constatou-se que aumentar o tamanho do *dataset* gera uma melhoria na acurácia. Aumentar o números de neurônios na camada oculta da LSTM pode melhorar a acurácia da previsão. No caso das órbitas C, D e E, essa melhoria é limitada a algumas iterações — até x_5 no máximo para $d = 5000$ e até x_3 para $d = 1000$.

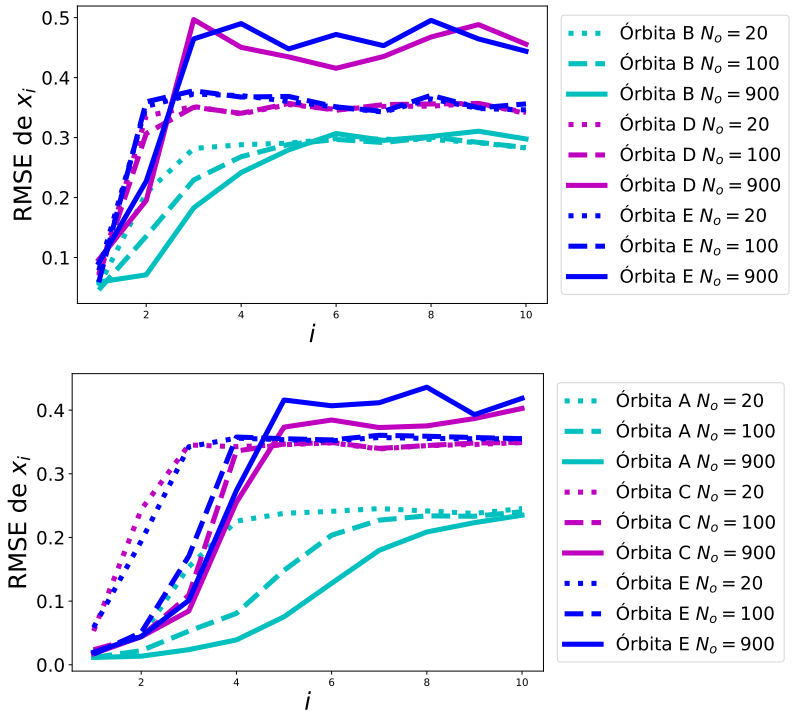

Figura 26 – RMSE de x_i obtido por uma LSTM tentando prever S_{10} para *datasets* de tamanho $d = 1000$ e $d = 5000$, respectivamente.

Fonte: Elaborada pela autora.

5.1.2 Influência da órbita na previsão

Com o intuito de comparar a acurácia obtida para sequências pertencentes a diferentes órbitas, as Figuras 27 e 28 ilustram o média dos erros absolutos de cada termo x_i obtida na tentativa de prever uma sequência S_n. A Figura 27 mostra os erros de cada órbita para a previsão das sequências de tamanho $n = 10$, $n = 9$ e $n = 8$, respectivamente, enquanto a Figura 28, $n = 7$, $n = 6$ e $n = 5$. Em todos os casos, os resultados se referem a redes treinadas com 900 neurônios na camada oculta e com conjunto de dados com 5000 vetores de entrada.

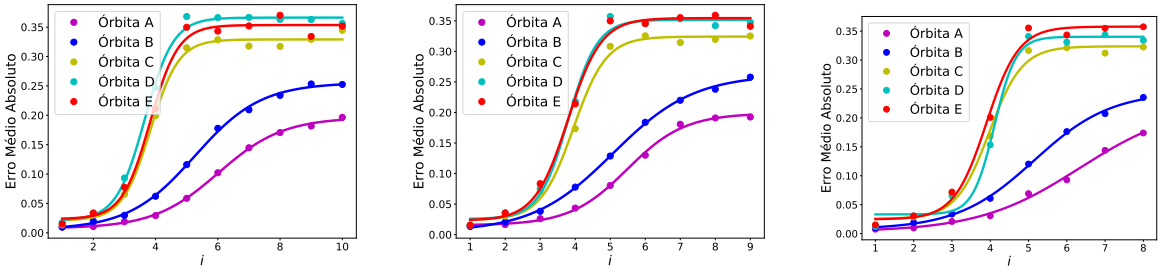

Figura 27 – MAE de x_i obtido por LSTMs ao prever S_{10}, S_9 E S_8, respectivamente.

Fonte: Elaborada pela autora.

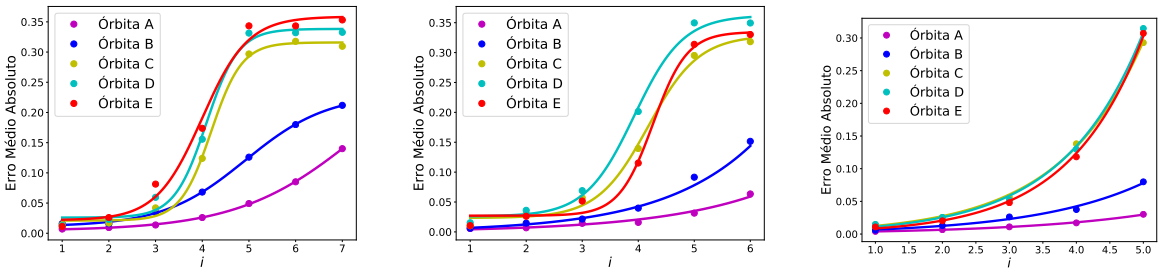

Figura 28 – MAE de x_i obtido por LSTMs ao prever S_7, S_6 e S_5, respectivamente.

Fonte: Elaborada pela autora.

Em todos os casos, podemos notar que os erros médios absolutos das previsões de sequências da órbita A são menores do que os da órbita B, que são menores do que os da órbita C, sendo esses menores do que os erros médios absolutos das sequências da órbita D. Contudo, nem sempre os erros da órbita E são maiores do que os da órbita D.

Também é possível notar que, no caso das órbitas A e B, quando a órbita prevista vai até x_6, a curva é melhor aproximada por uma exponencial. A partir desse ponto o erro não aumenta na mesma taxa, o que faz com que a curva passe a ganhar uma forma de sigmoide, e passe a ser aproximada melhor por uma função logística. Já para as órbitas C,

D e E, nota-se uma estagnação no erro a partir de x_5, ou seja, a partir da previsão de S_6 a curva passa ser melhor aproximada pela função logística.

5.1.3 Exemplos de previsões

Para exemplificar o quão bem uma LSTM é capaz de prever o futuro do mapa logístico e ilustrar como o parâmetro R e o tamanho de uma sequência prevista afetam a acurácia obtida por essas redes, alguns exemplos de previsões de sequências de tamanhos distintos foram escolhidos aleatoriamente de cada órbita. As Figuras 29, 30, 31, 32 e 33 mostram algumas sequências para as órbitas A, B, C, D e E, respectivamente. Assim como na seção anterior, os erros são aqueles obtidos por LSTMs treinadas com 900 neurônios na camada oculta e com *dataset* de tamanho 5000.

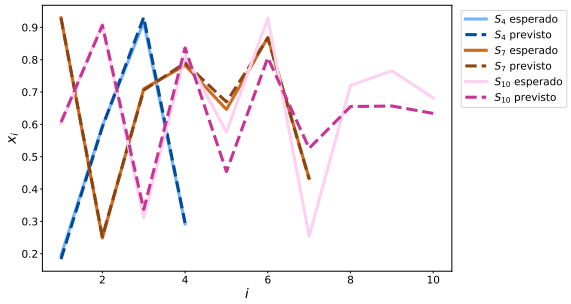

Figura 29 – Valores esperados e previstos para algumas sequências da órbita A.

Fonte: Elaborada pela autora.

Tabela 4 – Valores esperados e previstos para algumas sequências da órbita A.

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}
S_4 esperado	0,1941	0,5944	0,9161	0,2920						
previsto	0,1832	0,5922	0,9309	0,2951						
S_7 esperado	0,9296	0,2486	0,7098	0,7828	0,6461	0,8689	0,4330			
previsto	0,9287	0,2529	0,7056	0,7902	0,6695	0,8671	0,4305			
S_{10}esperado	0,6023	0,9102	0,3106	0,8136	0,5762	0,9279	0,2541	0,7202	0,7657	0,6817
previsto	0,6068	0,9062	0,3369	0,8366	0,4536	0,8062	0,5260	0,6550	0,6568	0,6339

Fonte: Elaborada pela autora.

No caso da órbita A, na previsão de uma sequência S_4, a rede conseguiu prever a primeira casa decimal de x_1 e x_3 e segunda de x_2 e x_4. Similarmente, a LSTM foi capaz de prever corretamente a primeira casa decimal para todos os termos, e no caso de x_1 e x_3, até a segunda. Já na previsão de S_{10}, a rede só consegue prever corretamente e primeira cada até x_4. A partir de x_5, o erro absoluto é maior do que $0,1$.

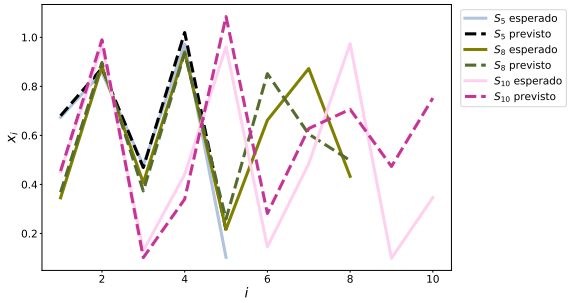

Figura 30 – Valores esperados e previstos para algumas sequências da órbita B.

Fonte: Elaborada pela autora.

Tabela 5 – Valores esperados e previstos para algumas sequências da órbita B.

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}
S_5 esperado	0,6737	0,8573	0,4771	0,9730	0,1026					
previsto	0,6802	0,8742	0,4688	1,0196	0,2135					
S_8 esperado	0,3454	0,8818	0,4066	0,9410	0,2166	0,6619	0,8728	0,4330		
previsto	0,3681	0,8975	0,3731	0,9356	0,2591	0,8520	0,6051	0,4977		
S_{10}esperado	0,4507	0,9655	0,1298	0,4406	0,9612	0,1454	0,4845	0,9741	0,0985	0,3464
previsto	0,4547	0,9902	0,1015	0,3398	1,0846	0,2813	0,6281	0,7063	0,4732	0,7522

Fonte: Elaborada pela autora.

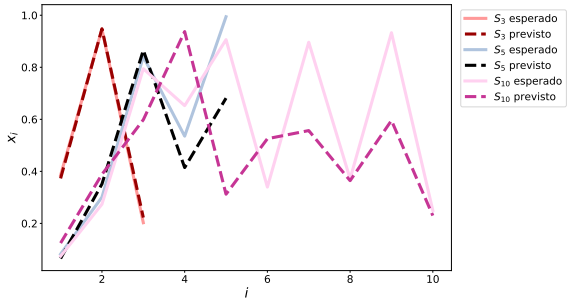

Figura 31 – Valores esperados e previstos para algumas sequências da órbita C.

Fonte: Elaborada pela autora.

Já para uma sequência com cinco termos da órbita B, a LSTM foi capaz de prever corretamente a primeira casa decimal até x_3. Apesar de não prever essa casa em x_4, o erro absoluto entre o previsto e o esperado é de apenas $0,0466$, já x_5 o erro passa ser $0,1109$. Para uma sequência de tamanho 8 ainda da órbita B, o erro da previsão está na segunda casa decimal até x_5, com exceção para x_3, mas erro nesse caso é apenas 0,0335. A sequência prevista S_{10}, por sua vez, teve apenas x_1 com as duas primeira casas decimais corretas, x_2 e x_3 com a primeira casa decimal correta. De forma geral, o erro cresceu conforme n.

Tabela 6 – Valores esperados e previstos para algumas sequências da órbita C.

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}
S_3 esperado	0,3851	0,9464	0,2026							
previsto	0,3729	0,9482	0,2225							
S_5 esperado	0,0820	0,3008	0,8407	0,5353	0,9943					
previsto	0,0651	0,3513	0,8656	0,4145	0,6814					
S_{10}esperado	0,0739	0,2737	0,7945	0,6525	0,9063	0,3396	0,8964	0,3713	0,9331	0,2496
previsto	0,1244	0,3884	0,5983	0,9374	0,3122	0,5255	0,5568	0,3642	0,5956	0,2300

Fonte: Elaborada pela autora.

Para sequências da órbita C, a LSTM conseguiu prever a primeira casa decimal de x_1 e x_3 e a segunda de x_2, no caso de S_2. Para S_5, até x_3 a rede conseguiu prever corretamente ao menos a primeira casa decimal, enquanto, no caso da sequência S_{10} não foi capaz de prever nenhuma casa. Notou-se um aumento considerável no erro absoluto à medida que n aumentou.

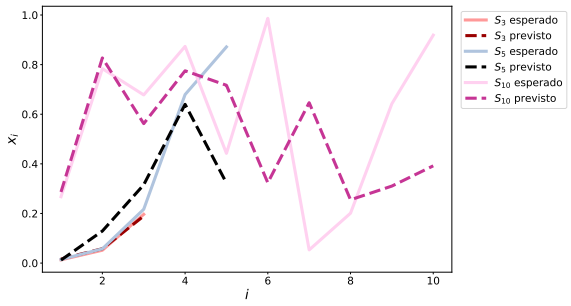

Figura 32 – Valores esperados e previstos para algumas sequências da órbita D.

Fonte: Elaborada pela autora.

Tabela 7 – Valores esperados e previstos para algumas sequências da órbita D.

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}
S_3 esperado	0,0131	0,0518	0,1964							
previsto	0,0166	0,0579	0,1897							
S_5 esperado	0,0146	0,0575	0,2169	0,6793	0,8714					
previsto	0,0120	0,1292	0,3156	0,6400	0,3228					
S_{10}esperado	0,2675	0,7838	0,6778	0,8735	0,4421	0,9865	0,0531	0,2010	0,6425	0,9188
previsto	0,2867	0,8271	0,5621	0,7751	0,7168	0,3238	0,6466	0,2552	0,3104	0,3920

Fonte: Elaborada pela autora.

No caso de um sequência de tamanho 3 da órbita D, a LSTM foi capaz de prever corretamente as duas primeiras casas decimais de x_1 e x_2 e a primeira de x_3. Já no caso de uma sequência S_5, ela só conseguiu prever as duas primeiras casas de x_1, enquanto para S_{10}, somente foi capaz de obter um erro menor do que 0,05 para x_1 e x_2 — para x_8 também, mas por sorte.

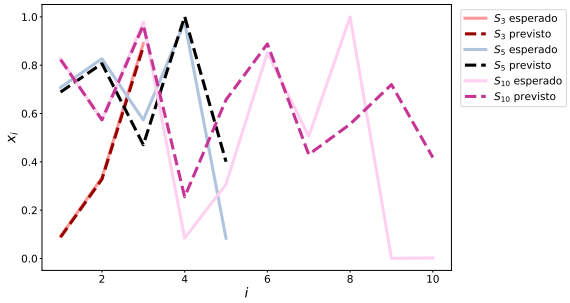

Figura 33 – Valores esperados e previstos para algumas sequências da órbita E.

Fonte: Elaborada pela autora.

Tabela 8 – Valores esperados e previstos para algumas sequências da órbita E.

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}
S_3 esperado	0,0921	0,3344	0,8903							
previsto	0,0882	0,3280	0,8764							
S_5 esperado	0,7082	0,8267	0,5732	0,9786	0,0838					
previsto	0,6887	0,8068	0,4706	1,0013	0,4005					
S_{10}esperado	0,8267	0,5732	0,9786	0,0839	0,3073	0,8515	0,5059	0,9999	0,0006	0,0022
previsto	0,8232	0,5736	0,9651	0,2550	0,6569	0,8883	0,4305	0,5560	0,7198	0,4179

Fonte: Elaborada pela autora.

No caso da órbita E, foi possível prever corretamente a primeira casa de todos os termos de S_3, com o erro absoluto sendo $0,0039$ para x_1, $0,0064$ para x_2 e $0,0139$ para x_3. A rede conseguiu prever a primeira casa de x_2 de S_5 e conseguiu prever x_1 e x_4 com erro absoluto menor do que $0,02$. No caso da sequência S_{10}, foi possível prever corretamente até a segunda casa decimal de x_1 e x_2, até a primeira casa de x_3 e x_6. A a partir de x_8, os erros absolutos das previsões foram maiores do que $0,4$.

5.1.4 Influência do tamanho da sequência prevista na acurácia

Ao prevermos o futuro apenas até x_3, tal previsão, em média, tem erro absoluto na terceira casa decimal. Ao tentarmos prever sequências maiores que vão até x_6, notamos que, em média, as redes conseguem obter um erro na terceira casa apenas na previsão de x_1 e x_2. Para os demais termos, em média, o erro já aparece na segunda casa decimal. Aumentando a sequência prevista até x_8 não muda a casa decimal que encontramos os erros anteriores, mas as previsões para os termos x_7 e x_8, em média, tem o erro já na primeira casa decimal, sendo ele menor do que 0,2. No entanto, ao aumentarmos a sequência de saída para 10 termos, os erros absolutos médios na previsão x_1 a x_5 vai para a segunda casa decimal enquanto os erros de x_6 a x_{10} vai para a primeira.

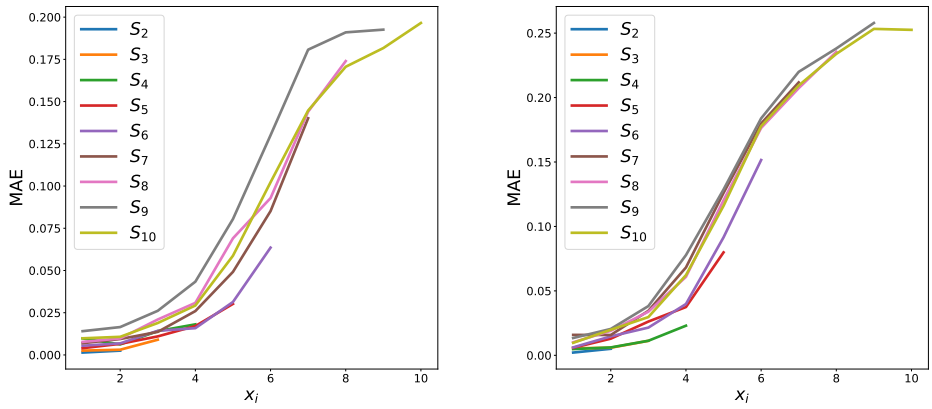

Figura 34 – Erro Médio Absoluto obtido por LSTMs tentando prever x_i para diferentes tamanhos de sequências de saída, para as órbitas A e B.

Fonte: Elaborada pela autora.

Tabela 9 – Erro Médio Absoluto obtido por LSTMs tentando prever x_i para diferentes sequências S_n da órbita A.

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}
S_2	0,0014	0,0026								
S_3	0,0026	0,0031	0,0090							
S_4	0,0098	0,0062	0,0141	0,0181						
S_5	0,0040	0,0066	0,0110	0,0171	0,0302					
S_6	0,0055	0,0069	0,0143	0,0159	0,0314	0,0635				
S_7	0,0070	0,0094	0,0137	0,0260	0,0492	0,0852	0,1401			
S_8	0,0077	0,0098	0,0210	0,0308	0,0691	0,0929	0,1440	0,1740		
S_9	0,0141	0,0165	0,0261	0,0434	0,0805	0,1301	0,1807	0,1910	0,1926	
S_{10}	0,0098	0,0107	0,0189	0,0293	0,0587	0,1022	0,1447	0,1706	0,1817	0,1965

Fonte: Elaborada pela autora.

Tabela 10 – Erro Médio Absoluto obtido por LSTMs tentando prever x_i para diferentes sequências S_n da órbita B.

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}
S_2	0.0021	0.0051								
S_3	0,0055	0,0059	0,0112							
S_4	0,0048	0,0061	0,0113	0,0230						
S_5	0,0059	0,0129	0,0262	0,0375	0,0799					
S_6	0,0062	0,0147	0,0215	0,0398	0,0915	0,1515				
S_7	0,0159	0,0158	0,0345	0,0682	0,1260	0,1799	0,2118			
S_8	0,0101	0,0190	0,0338	0,0609	0,1203	0,1764	0,2073	0,2355		
S_9	0,0134	0,0205	0,0382	0,0778	0,1287	0,1839	0,2200	0,2381	0,2579	
S_{10}	0,0097	0,0198	0,0297	0,0622	0,1160	0,1778	0,2094	0,2339	0,2533	0,2525

Fonte: Elaborada pela autora.

Para a previsão da órbita B, um padrão similar é observado, mas a previsão de x_3, seja ela feita com uma sequência de saída S_3 ou S_{10} tem o erro na terceira casa. Só

vemos o erro na terceira casa até x_2 ao prever sequências que vão até x_4, e em x_1 ao prever sequências que vão até x_5. Prevendo S_7 a S_{10}, em média, os erros de x_1 a x_4 estão na segunda casa decimal, enquanto de x_5 a x_{10} estão já na primeira, alcançando no máximo 0,2579.

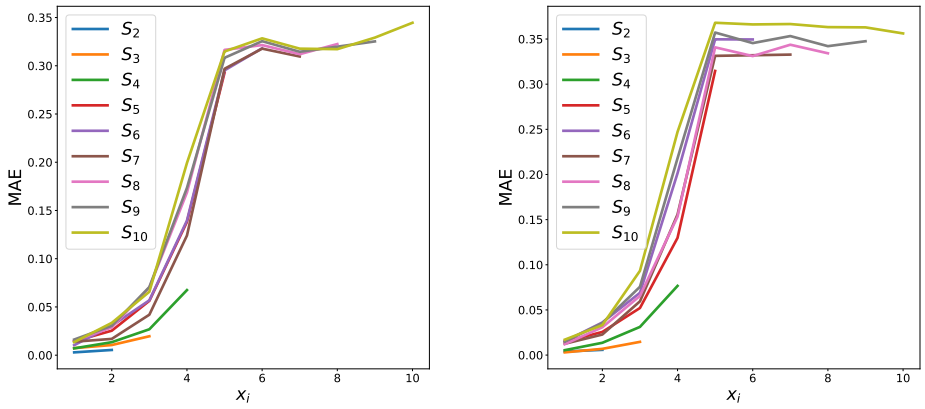

Figura 35 – Erro Médio Absoluto obtido por LSTMs tentando prever x_i para diferentes tamanhos de sequências de saída, para as órbitas C e D.

Fonte: Elaborada pela autora.

Tabela 11 – Erro Médio Absoluto obtido por LSTMs tentando prever x_i para diferentes sequências S_n da órbita C.

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}
S_2	0,0029	0,0054								
S_3	0,0073	0,0105	0,0196							
S_4	0,0071	0,0135	0,0269	0,0675						
S_5	0,0144	0,0254	0,0563	0,1381	0,2928					
S_6	0,0104	0,0305	0,0569	0,1395	0,2952	0,3182				
S_7	0,0140	0,0169	0,0420	0,1240	0,2969	0,3178	0,3096			
S_8	0,0150	0,0288	0,0701	0,1686	0,3165	0,3214	0,3120	0,3225		
S_9	0,0162	0,0304	0,0706	0,1734	0,3083	0,3255	0,3145	0,3198	0,3252	
S_{10}	0,0135	0,0335	0,0658	0,1992	0,3148	0,3284	0,3177	0,3172	0,3292	0,3446

Fonte: Elaborada pela autora.

Para as órbitas C, D e E, só encontramos o erro na terceira casa decimal prevendo x_2 em S_1 e S_2, e x_1 prevendo sequências que podem ir até x_3. O erro médio absoluto na segunda casa decimal é obtido na previsão até apenas x_3. Ao prever sequências maiores que S_4, o erro dos termos x_4 a x_{10}, em média, já aparece na primeira casa. Ademais, os erros médios absolutos na previsão de termos a partir de x_5 giram em torno de 0,32. No caso da previsão da órbita C, começam em 0,29 e vão até 0,35, enquanto para a órbita D vão de 0,31 a 0,36 e para a órbita E, de 0,30 a 0,37. Tais valores são comparáveis — ou até maiores — do que os na Tabela 3. Ou seja, a no caso dessas órbitas, a tentativa de prever

termos a partir de x_5 não tem uma precisão melhor do que um chute. No caso das órbitas A e B, os erros das previsões nunca ultrapassaram aqueles que poderiam ser obtidos por chute, mas ficaram bastante próximos a partir de x_7.

Tabela 12 – Erro Médio Absoluto obtido por LSTMs tentando prever x_i para diferentes sequências S_n da órbita D.

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}
S_2	0,0040	0,0058								
S_3	0,0030	0,0069	0,0146							
S_4	0,0054	0,0136	0,0312	0,0766						
S_5	0,0148	0,0256	0,0521	0,1298	0,3146					
S_6	0,0153	0,0361	0,0690	0,2014	0,3497	0,3495				
S_7	0,0125	0,0228	0,0595	0,1557	0,3314	0,3320	0,3327			
S_8	0,0121	0,0308	0,0655	0,1534	0,3408	0,3311	0,3437	0,3342		
S_9	0,0149	0,0355	0,0756	0,2179	0,3572	0,3455	0,3533	0,3421	0,3476	
S_{10}	0,0171	0,0332	0,0933	0,2472	0,3680	0,3662	0,3666	0,3632	0,3629	0,3562

Fonte: Elaborada pela autora.

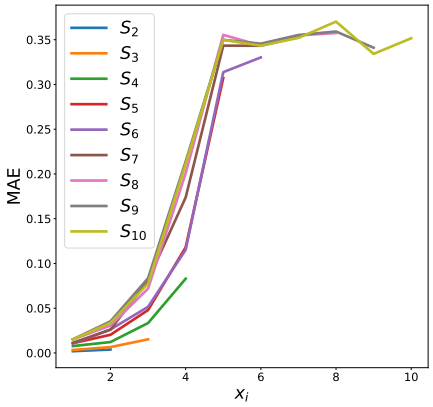

Figura 36 – Erro Médio Absoluto obtido por LSTMs tentando prever x_i para diferentes tamanhos de sequências de saída, para a órbita E.

Fonte: Elaborada pela autora.

Tabela 13 – Erro Médio Absoluto obtido por LSTMs tentando prever x_i para diferentes sequências S_n da órbita E.

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}
S_2	0,0020	0,0038								
S_3	0,0033	0,0066	0,0152							
S_4	0,0077	0,0122	0,0334	0,0829						
S_5	0,0109	0,0203	0,0479	0,1181	0,3072					
S_6	0,0105	0,0262	0,0516	0,1152	0,3138	0,3302				
S_7	0,0113	0,0260	0,0814	0,1738	0,3434	0,3433	0,3532			
S_8	0,0150	0,0306	0,0719	0,2009	0,3555	0,3434	0,3548	0,3575		
S_9	0,0152	0,0354	0,0835	0,2140	0,3499	0,3456	0,3553	0,3592	0,3411	
$S102$	0,0151	0,0336	0,0775	0,2111	0,3498	0,3433	0,3520	0,3702	0,3342	0,3516

Fonte: Elaborada pela autora.

É notável que há uma tendência, em todas as órbitas analisadas, que ao aumentarmos o tamanho n de uma sequência prevista, a acurácia com que a rede neural consegue prever os primeiros termos — como x_1 e x_2 — decai. Por exemplo, no caso da órbita A, o erro médio absoluto da previsão de x_1 quando tentamos prever uma sequência S_2 é 0,0014 e quando tentamos prever uma sequência S_4, x_1 aumenta para 0,0098 e na previsão de S_9 passa a ser 0,0141. Os gráficos ilustrados na Figura 37 relacionam os erros absolutos médios de cada valor x_i previstos para sequências de diferentes tamanhos S_n. No caso de x_1 até x_7, as inclinações são sempre positivas, indicando o que já havíamos notado — o aumento no erro da previsão de x_i conforme se tenta prever uma sequência maior. Ademais, a inclinação da retas aumenta conforme i, indicando que quanto mais longe o termo previsto x_i, mais ele sofre pioras em sua previsão com o aumento da sequência S_n.

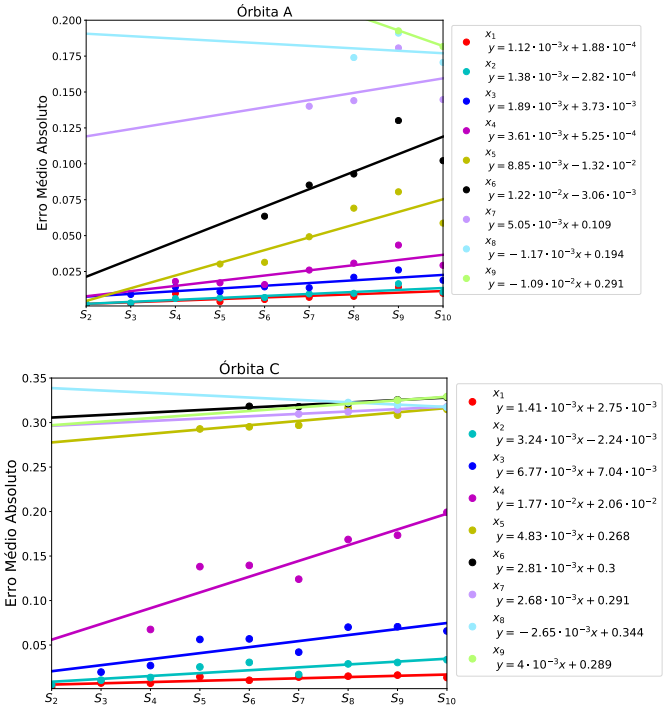

Figura 37 – Erro Médio Absoluto obtido por uma LSTM tentando prever x_i para diferentes tamanhos de sequência de saída.

Fonte: Elaborada pela autora.

5.2 Previsões com *Echo State Networks*

Os resultados dessa seção se referem àqueles obtidos pelas redes treinadas com os *datasets* e conforme as descrições contidas na Seção 4.4. O objetivo é analisar como a rede prevê uma sequência S_n sendo alimentada com sequências de diferente tamanho l, que começam em x_{-l} e vão até x_0.

5.2.1 Influência do tamanho da sequência de entrada

As Figuras 38, 39 e 40 mostram os erros da previsão de sequências S_n, com n indo até 10. Constata-se que, em ambas as órbitas A e B, o aumento do erro em função de n é melhor ajustado linearmente, enquanto o erros das previsões para as órbitas C, D e E são melhores aproximados exponencialmente.

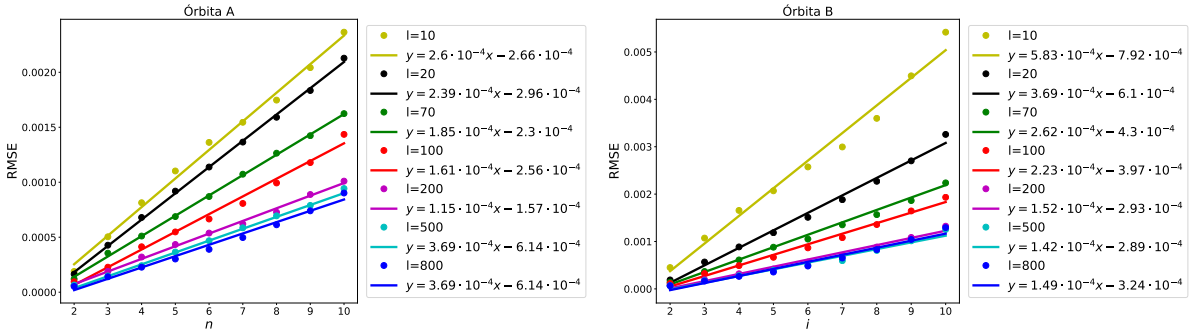

Figura 38 – RMSE de uma ESN tentando prever S_n para diferentes tamanhos de sequência de entrada das órbitas A e B.

Fonte: Elaborada pela autora.

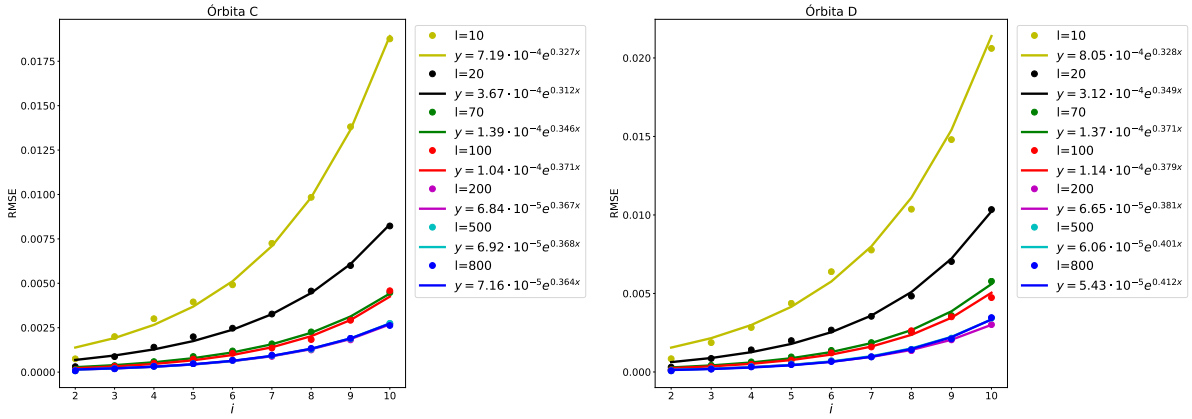

Figura 39 – RMSE da previsão de S_n para diferentes tamanhos de sequência de entrada das órbitas C e D obtidos por ESNs.

Fonte: Elaborada pela autora.

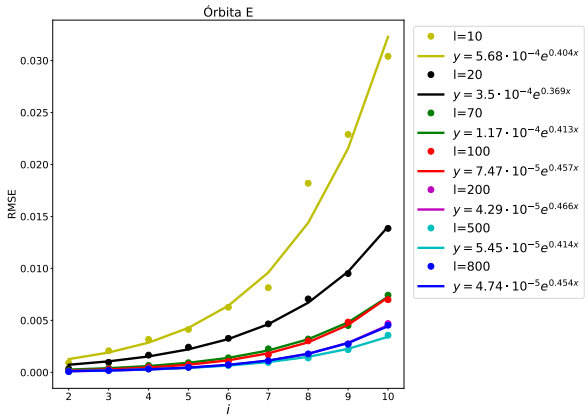

Figura 40 – RMSE da previsão de S_n para diferentes tamanhos de sequência de entrada da órbita E obtidos por ESNs.

Fonte: Elaborada pela autora.

Para todo domínio analisado, aumentar o tamanho da sequência de entrada até $l = 200$ resultou em uma acurácia melhor nas previsões. No entanto, nota-se uma estagnação para valores maiores.

No caso das previsões de sequências pertencentes à órbita A, o aumento do tamanho da entrada de $l = 10$ para $l = 70$ resultou em um RMSE 45% menor para S_2 e 30% menor para S_{10}. Já aumentar de $l = 10$ para $l = 100$ gera um um erro pelo menos 42% menor, enquanto aumentar para $l = 200$, 57%. É notável que para todos os casos analisados, quanto maior a sequência S_n prevista, menor é a porcentagem com que o erro diminui aumentando-se a entrada l. Para a órbita B, o aumento do tamanho da entrada de 10 para 70, 100 e 200 gerou uma diminuição do RMSE das sequências previstas de 56%, 63% e 80%, respectivamente. Os erros das previsões para a órbita C sofreram uma diminuição média de, no mínimo, 75% quando o tamanho da entrada foi aumentado de 10 para 70, 76% quando aumentado para 100 e 85%, para 200. Sequências da órbita D tiveram seus RMSE reduzidos em 73%, quando o tamanho da entrada l foi elevado de 10 para 70, em 76% quando de 10 para 100 86% quando para 200. Já para a órbita E, o aumento do tamanho da entrada de 10 para 70 e 100 resultou em uma diminuição média do RMSE das sequências previstas S_{10} em 77% em ambos os casos em 85% para 200.

5.2.2 Influência do tamanho da sequência prevista na acurácia

Ao prever termos futuros até x_{10} da órbita A, para $l = 200$, o erro aparece, em média, na quarta casa decimal. Para $l = 100$, o erro na quarta casa aparece ao prever sequências que vão até x_8, e os erros médios das previsão dos termos de S_9 e S_{10} estão majoritariamente na terceira casa decimal, enquanto para $l = 70$, a mudança do erro ir da quarta para a terceira casa acontece em S_8, e, para $l = 10$, em S_5.

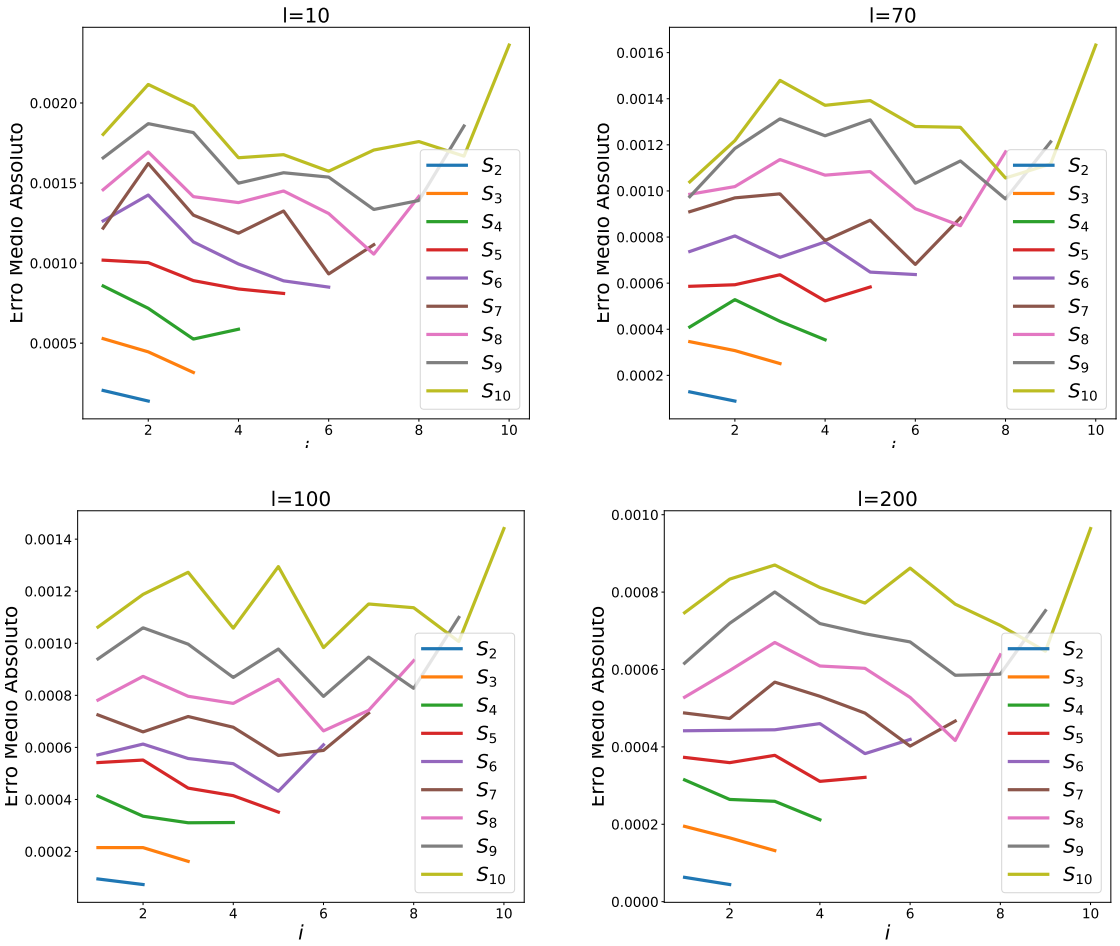

Figura 41 – Erro Médio Absoluto de x_i previsto para diferentes tamanhos de sequência S_n da órbita A.

Fonte: Elaborada pela autora.

No caso da órbita B, usando uma sequência de entrada de tamanho 10, a média dos erros absolutos dos termos futuros previstos estão na ordem de 10^{-3}, enquanto usando uma sequência de tamanho 70 como entrada, a média dos erros absolutos estão nessa ordem ao tentarmos prever sequências de tamanho 7 até 10. Para previsões de sequências menores, o erro médio absoluto fica na quarta casa decimal.

Na previsão do futuro das órbitas C, D e E, notamos um comportamento similar: para $l = 10$, a média do erro absoluto de cada termo consegue ficar na terceira casa decimal até S_7. Para prever sequências maiores, a maioria dos erros já aparece na segunda casa. Para $l = 70$, o erros aparecem na terceira casa em todos os casos para a órbita C. Nas outras órbitas, a única exceção está na previsão do termo x_{10}, cujo erro absoluto aparece, em média, na segunda casa decimal. Para $l = 100$ e $l = 200$, os erros das previsões estão na terceira casa decimal.

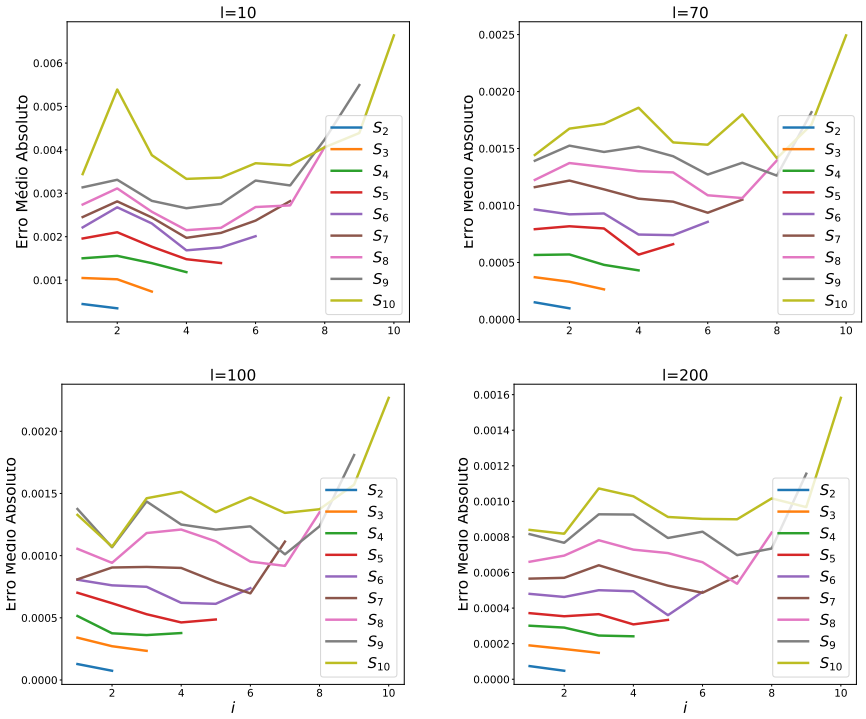

Figura 42 – Erro Médio Absoluto de x_i previsto para diferentes tamanhos de sequência S_n da órbita B.

Fonte: Elaborada pela autora.

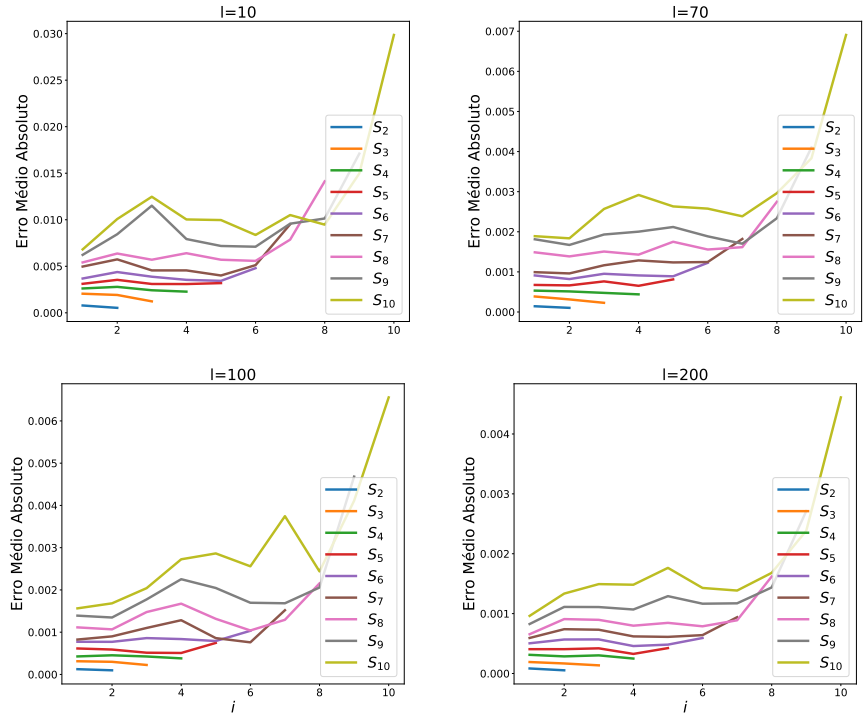

Figura 43 – Erro Médio Absoluto de x_i previsto para diferentes tamanhos de sequência S_n da órbita C.

Fonte: Elaborada pela autora.

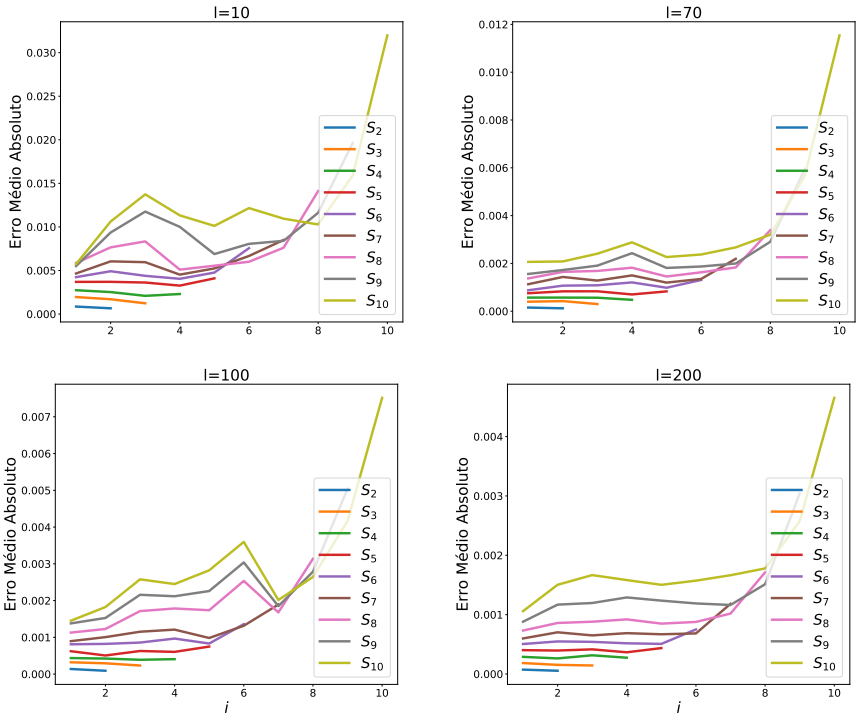

Figura 44 – Erro Médio Absoluto de x_i previsto para diferentes tamanhos de sequência S_n da órbita D.

Fonte: Elaborada pela autora.

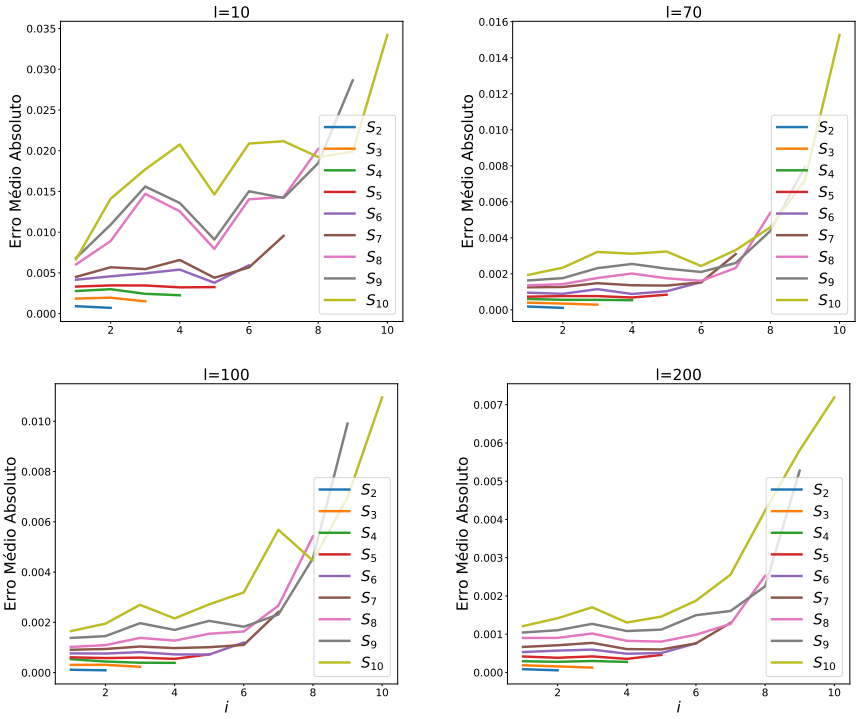

Figura 45 – Erro Médio Absoluto de x_i previsto para diferentes tamanhos de sequência S_n da órbita E.

Fonte: Elaborada pela autora.

5.2.3 Previsão para futuro mais distante

Os gráficos dessa seção mostram os erros dos termos x_i pertencentes a sequências S_n previstas pelas ESNs, com o tamanho da sequência n podendo ir até 40. Os resultados apresentados nessa seção foram todos obtidos usando uma entrada de tamanho $l = 200$.

Já estudamos com detalhes em seções anteriores os erros obtidos para a sequência S_{10}, eles são apresentados novamente nos gráficos como base de comparação para sequências maiores. Os erros médios absolutos da previsão para as sequências das órbitas A e B são apresentados no gráfico da Figura 46. Em previsões de sequências da órbita A, e erro médio absoluto dos termos previstos x_i de uma sequência S_{10} são menores do que 10^{-3} — variam de 0,00075 até 0,00097 —, enquanto quando a sequência prevista é S_{20}, o erros médio absoluto estão na terceira casa decimal até x_{17} e na segunda casa decimal de x_{18} a x_{20}. Para uma sequência prevista com 30 termos, o erro médio absoluto está na terceira casa até x_4 e na segunda casa decimal até x_{19}. Já para a previsão de uma sequência com 40 termos, o erro médio absoluto encontra-se na terceira casa decimal até x_3 na segunda de x_4 até x_{18}.

No caso de previsões para sequências da órbita B, nota-se que o erro médio absoluto quando a previsão é de uma sequência de 10 termos encontra-se na quarta decimal para quase todos os termos x_i, só com exceção de x_{10}, que está na terceira. Quando a previsão é de S_{20}, os erros médios absolutos estão terceira casa decimal até x_{15}. Para previsões de sequência com 30 termos, erro está na terceira casa decimal só até x_3 e na segunda até x_{17}, enquanto na de 40 termos, erro na terceira casa é só até x_2, e na terceira x_{19}.

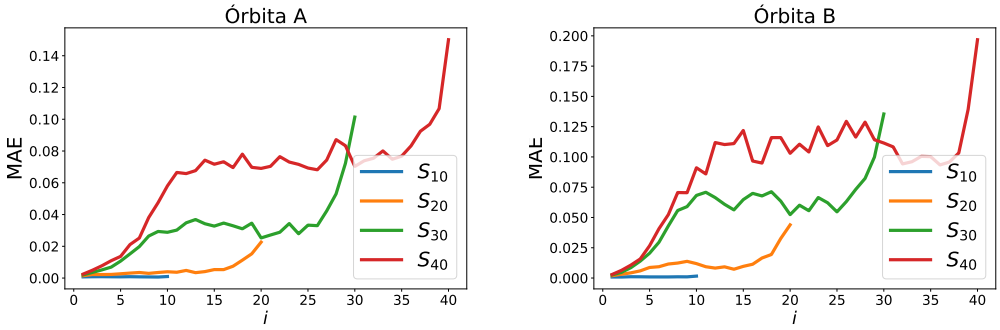

Figura 46 – Erro Médio Absoluto das ESNs tentando prever sequências S_i das órbitas A e B para diferentes tamanhos de sequência de saída i.

Fonte: Elaborada pela autora.

A Figura 47 apresenta os erros na previsão para as órbitas C, D e E. Para essas sequências, os erros médios absolutos dos termos x_i de S_{10} estão na terceira decimal (ao invés da quarta como no caso das órbitas A e B). No caso de S_{20}, o erro é encontrado na segunda casa decimal até por volta de x_{18}. Já no caso da previsão da sequência S_{30},

só até x_7 o erro médio está na segunda casa. Quando se trata da previsão de S_{40}, o erro médio absoluto encontra-se na segunda casa decimal até x_7 (para a órbita C) e x_8 (para as órbitas D e E).

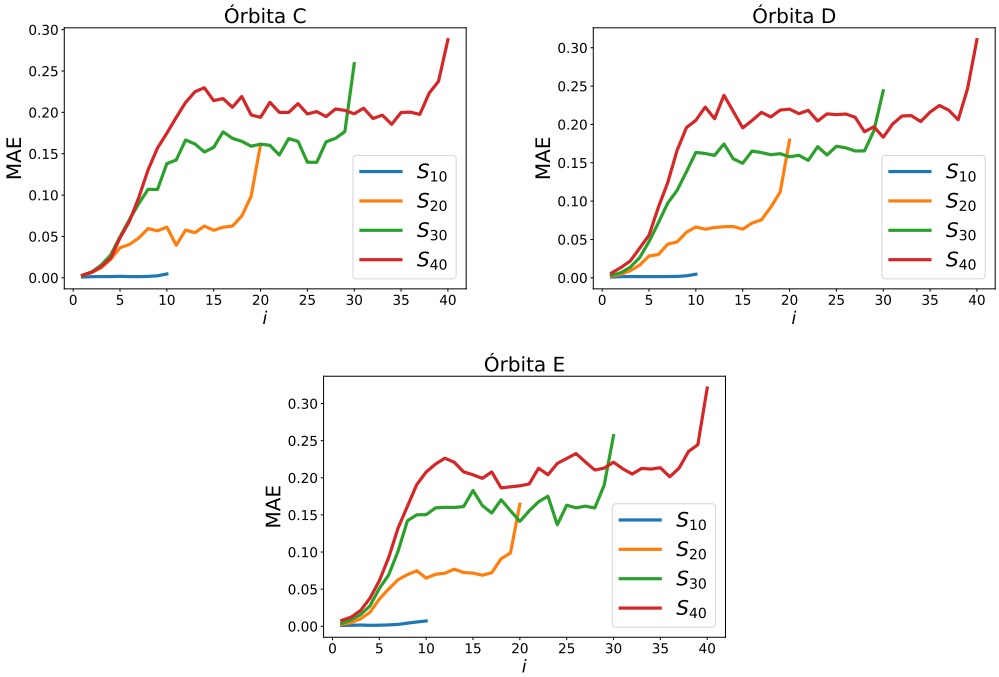

Figura 47 – Erro Médio Absoluto das ESNs tentando prever sequências S_i das órbitas A e B para diferentes tamanhos de sequência de saída i.

Fonte: Elaborada pela autora.

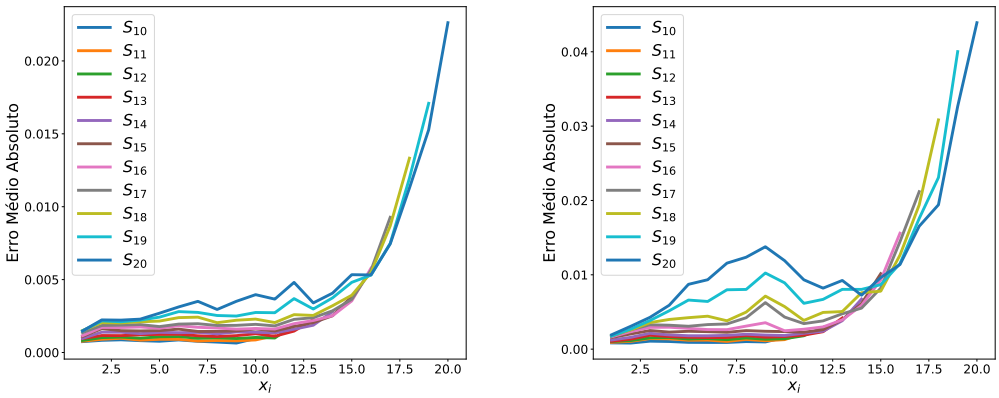

Figura 48 – Erro Médio Absoluto das ESNs tentando prever sequências S_i das órbitas A e B para diferentes tamanhos de sequência de saída i.

Fonte: Elaborada pela autora.

A Figura 48 apresenta com mais detalhes os erros médios absolutos de cada termo

na previsão de sequência S_n com n variando de 10 a 20. Podemos notar uma inclinação maior nas curvas, ressaltando que nesse intervalo, se concentra uma piora significativa das acurácias. Após S_{20}, parece ocorrer uma certa estabilização, mas depois os erros voltam a subir, como já visto nas Figuras 46 e 47.

5.3 Comparação

Previsões feitas por redes LSTM e ESN para algumas sequências pertencentes às órbitas A e D estão ilustradas na Figura 49 e Tabela 14, assim como os valores que eram esperados.

Nos exemplos apresentados, podemos notar que as previsões obtidas para todos os 10 termos pela ESN para a órbita A, em geral, diferem do esperado apenas na terceira casa decimal. A previsão feita pela LSTM conseguiu prever apenas x_1 até a segunda casa decimal, os termos x_2, x_3 e x_4 ela foi capaz de prever corretamente só a primeira casa decimal. Já no caso das sequência pertencentes à órbita D, a previsão feita pela ESN conseguiu prever até a segunda casa decimal, enquanto a previsão feita pela LSTM só foi capaz de prever a primeira casa decimal para x_1. Para termos futuros, a diferença entre previsto e esperado aumenta conforme i aumenta.

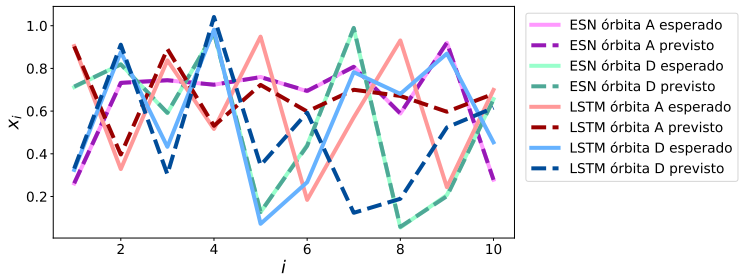

Figura 49 – Comparação dos valores esperados e previstos por uma ESN e LSTM ao tentar prever sequências de tamanho 10.

Fonte: Elaborada pela autora.

Com o intuito de deixar mais clara a comparação entre os erros obtidos por essas duas redes neurais na previsão do mapa logístico, o gráfico da Figura 50 contém o erros médios absolutos de x_i previstos por LSTMs para diferentes tamanhos de sequências S_n (com o tamanho da sequência prevista n variando de 2 a 10), além dos erros obtidos pelas ESNs tentando prever sequências de tamanho 10 (S_{10}). Os resultados das LSTMs se referem àqueles obtidos treinando-as com 900 neurônios na camada oculta e dataset de tamanho 5000. A única curva referente aos erros da ESN se refere aos piores resultados apresentados por essa arquitetura: foram obtidos ao tentar prever uma sequência S_{10} usando uma sequência de entrada de tamanho $l = 10$. Embora os erros possam ser comparáveis para

Tabela 14 – Comparação de valores esperados e previstos para previsões feitas por ESN e LSTM.

	ESN órbita A		ESN órbita D		LSTM órbita A		LSTM órbita D	
	esperado	previsto	esperado	previsto	esperado	previsto	esperado	previsto
x_1	0,260907	0,261800	0,712080	0,714924	0,904495	0,902882	0,324444	0,330923
x_2	0,732771	0,732192	0,820047	0,818652	0,328259	0,396665	0,876677	0,911317
x_3	0,744107	0,744674	0,590249	0,592078	0,837920	0,890956	0,432436	0,303969
x_4	0,723565	0,723701	0,967372	0,966116	0,516080	0,530993	0,981691	1,041047
x_5	0,760070	0,759052	0,126248	0,127287	0,949018	0,722892	0,071891	0,345973
x_6	0,692981	0,694777	0,441216	0,437050	0,183856	0,598015	0,266878	0,592949
x_7	0,808481	0,807332	0,986129	0,989873	0,570202	0,699986	0,782578	0,123534
x_8	0,588389	0,589164	0,054713	0,057866	0,931272	0,671007	0,680565	0,189136
x_9	0,920312	0,920276	0,206868	0,201496	0,243216	0,596997	0,869542	0,523701
x_{10}	0,278683	0,278169	0,656263	0,654871	0,699436	0,680863	0,453732	0,615463

Fonte: Elaborada pela autora.

previsões de termos de um futuro próximo (x_3), para futuros mais distantes, é evidente a superioridade na acurácia das *Echo State Networks*. No caso da órbita A, apenas na previsão de x_i de S_2 a LSTM consegue uma acurácia menor do que ESN, já no caso da órbita de expoente de Lyapunov maior como a D, o erro obtido pelas LSTM é menor que o obtido pela ESN até x_3 (de S_3).

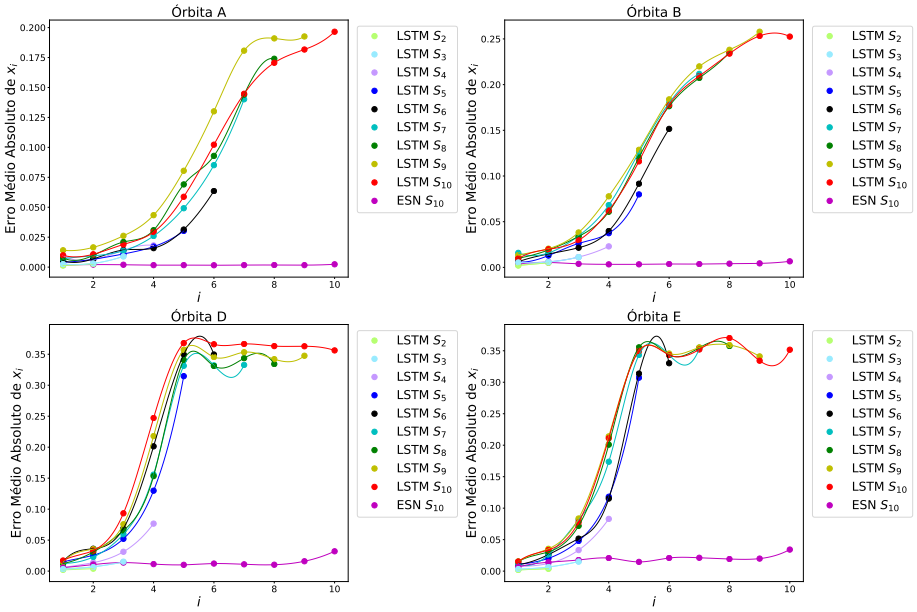

Figura 50 – Comparação dos erros médios absolutos de x_i obtidos.

Fonte: Elaborada pela autora.

6 CONCLUSÃO

Nesta dissertação foi apresentada uma introdução aos conceitos de sistemas caóticos e redes neurais. O mapa logístico foi escolhido para análise por se tratar de um dos exemplos mais famosos de sistemas caóticos gerado a partir de uma regra simples. Neste trabalho, também foram apresentadas previsões de futuras iterações de órbitas caóticas do mapa logístico realizadas por LSTMs e ESNs, ambas redes neurais recorrentes.

No caso das LSTMs, o aumento do tamanho do *dataset* melhorou a acurácia da previsão, enquanto o aumento do número de neurônios na camada oculta diminuiu o erro de previsões de termos futuros mais próximos (até x_4). Para previsões dos termos além de x_4, o erro também diminuiu no caso de órbitas com expoente de Lyapunov menor (A e B), mas aumentou para os termos mais distantes no caso de órbitas com LE maior (C, D e E). No caso das *Echo States Networks*, o aumento da sequência de entrada l gerou diminuição do erro da sequência prevista, sendo que por volta de $l = 200$ essa melhora sofre uma estagnação, pelo menos para os parâmetros usados nos testes.

As redes *Long Short-Term Memory* (LSTM) só conseguiram prever com acurácia superior as da *Echo State Networks* (ESN) termos futuros muito próximos, x_3 no máximo. Em relação ao termo x_{10}, para sequências da órbita A, a rede LSTM obteve um erro 38 vezes maior do que ESN, e, para a órbita E, 10 vezes maior. Por conseguinte, as ESNs apresentam resultados promissores para a previsão de sistemas caóticos, podendo ser investigadas mais alterações em seus parâmetros com intuito de melhorar ainda mais as previsões.

Nesse sentido, os resultados obtidos neste trabalho reforçam as *Echo State Networks* como ferramenta de aprendizado de máquina interessante no estudo de sistemas caóticos, pois confrontam sua absoluta imprevisibilidade. Dessa forma, essas redes poderiam ser usadas no estudo de outros sistemas caóticas e, de forma mais específica ao que foi apresentado neste trabalho, poderiam ser utilizadas na quebra de criptografia baseada em geradores pseudoaleatórios baseados no mapa logístico.

REFERÊNCIAS

1 LORENZ, E. N. Deterministic nonperiodic flow. **Journal of the Atmospheric Sciences**, v. 20, n. 2, p. 130–141, 1963.

2 BARROW-GREEN, J. **Poincaré and the three body problem**. Providence: American Mathematical Society, 1997.

3 HUNT, B. R.; YORKE, J. A. Maxwell on chaos. **Nonlinear Science Today**, v. 3, n. 1, p. 1–4, 1993.

4 FEIGENBAUM, M. J. Quantitative universality for a class of nonlinear transformations. **Journal of Statistical Physics**, v. 19, n. 1, p. 25–52, 1978.

5 MANDELBROT, B. B. **The fractal geometry of nature**. New York: WH Freeman, 1982. v. 1.

6 MAY, R. M. Simple mathematical models with very complicated dynamics. **Nature**, v. 261, n. 5560, p. 459–467, 1976.

7 VERHULST, P. Deuxième mémoire sur la loi d'accroissement de la population. **Mémoires de l'Académie Royale des Sciences, des Lettres et des Beaux-Arts de Belgique**, v. 20, p. 1–32, 1847.

8 HOCHREITER, S.; SCHMIDHUBER, J. **LSTM can solve hard long time lag problems**. 1997. Disponível em: https://proceedings.neurips.cc/paper/1996/file/a4d2f0d23dcc84ce983ff9157f8b7f88-Paper.pdf. Acesso em: 17 mar. 2022.

9 JAEGER, H.; HAAS, H. Harnessing nonlinearity: predicting chaotic systems and saving energy in wireless communication. **Science**, v. 304, n. 5667, p. 78–80, 2004.

10 JAEGER, H. **The "echo state" approach to analysing and training recurrent neural networks-with an erratum note**. Disponível em: https://www.ai.rug.nl/minds/uploads/EchoStatesTechRep.pdf. Acesso em: 10 mar. 2022.

11 MITCHELL, M. **Complexity:** a guided tour. Oxford: Oxford University Press, 2009.

12 OTT, E. **Chaos in dynamical systems**. Cambridge: Cambridge University Press, 2002.

13 BIRKHOFF, G. D. On the periodic motions of dynamical systems. **Acta Mathematica**, v. 50, p. 359–379, 1927. DOI: 10.1007/BF02421325.

14 CARTWRIGHT, M. L.; LITTLEWOOD, J. E. On non-linear differential equations of the second order: I. the equation y¨- k (1-y2) y˙+ y= bλk cos (λl+ α), k large. **Journal of the London Mathematical Society**, v. 1, n. 3, p. 180–189, 1945.

15 KOLMOGOROV, A. N. On conservation of conditionally periodic motions for a small change in hamilton's function. **Doklady Akademii Nauk SSSR**, v. 98, p. 527–530, 1954.

16 KOLMOGOROV, A. N. A new metric invariant of transient dynamical systems and automorphisms in lebesgue spaces. **Doklady Akademii Nauk SSSR**, v. 119, n. 5, p. 861–864, 1958.

17 RUELLE, D.; TAKENS, F. On the nature of turbulence. **Communications in Mathematical Physics**, v. 20, p. 167–192, 1971. DOI: 10.1007/BF01646553.

18 TAKENS, F. Detecting strange attractors in turbulence. *In*: RAND, D.; YOUNG, L.-S. (ed.). **Dynamical systems and turbulence, Warwick 1980**. Berlin: Springer, 1981. p. 366–381. (Lecture notes in mathematics, v. 898).

19 GOLLUB, J. P.; BENSON, S. Chaotic response to periodic perturbation of a convecting fluid. **Physical Review Letters**, v. 41, n. 14, p. 948, 1978.

20 LIBCHABER, A. Convection and turbulence in liquid helium i. **Physica B+ C**, v. 109, p. 1583–1589, 1982.

21 GOLLUB, J. P.; SWINNEY, H. L. Onset of turbulence in a rotating fluid. **Physical Review Letters**, v. 35, n. 14, p. 927, 1975.

22 BENETTIN, G. *et al.* Lyapunov characteristic exponents for smooth dynamical systems and for hamiltonian systems; a method for computing all of them. part 1: theory. **Meccanica**, v. 15, n. 1, p. 9–20, 1980.

23 LYAPUNOV, A. M. The general problem of the stability of motion. **International Journal of Control**, v. 55, n. 3, p. 531–534, 1992.

24 WOLF, A. *et al.* Determining lyapunov exponents from a time series. **Physica D:** nonlinear phenomena, v. 16, n. 3, p. 285–317, 1985.

25 VEHRULST, P.-F. Notice sur la loi que la population poursuit dans son accroissement. **Correspondance Mathématique et Physique**, v. 10, p. 113–121, 1838.

26 MALTHUS, T. R. **An essay on the principle of population.** London: J. Johnson, 1872.

27 VEHRULST, P.-F. Recherches mathématiques sur la loi d'accroissement de la population. **Nouveaux Mémoires de l'Académie Royale des Sciences et Belles-Lettres de Bruxelles**, v. 18, p. 14–54, 1845.

28 PEARL, R.; REED, L. J. On the rate of growth of the population of the united states since 1790 and its mathematical representation. **Proceedings of the National Academy of Sciences**, v. 6, n. 6, p. 275–288, 1920.

29 III, O. E. L. A computer-assisted proof of the feigenbaum conjectures. **Bulletin of the American Mathematical Society**, v. 6, n. 3, p. 427–434, 1982.

30 ECKMANN, J.-P.; WITTWER, P. A complete proof of the feigenbaum conjectures. **Journal of Statistical Physics**, v. 46, n. 3, p. 455–475, 1987.

31 LYUBICH, M. Feigenbaum-coullet-tresser universality milnor's hairiness conjecture. **Annals of Mathematics**, v. 149, n. 2, p. 319–420, 1999.

32 HEBB, D. O. **The organization of behavior:** a neuropsychological theory. New York: John Wiley & Sons, 1949.

33 MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. **Bulletin of Mathematical Biophysics**, v. 5, n. 4, p. 115–133, 1943.

34 SAMUEL, A. L. Some studies in machine learning using the game of checkers. ii—recent progress. **IBM Journal of Research and Development**, v. 11, n. 6, p. 601–617, 1967.

35 TURING, A. M. Computing machinery and intelligence. **Mind**, v. 59, n. 236, p. 433–460, 1950.

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

37 CORTES, C.; VAPNIK, V. Support-vector networks. **Machine Learning**, v. 20, n. 3, p. 273–297, 1995.

38 HARTIGAN, J. A. **Clustering algorithms**. New York: John Wiley & Sons, 1975.

39 COVER, T.; HART, P. Nearest neighbor pattern classification. **IEEE Transactions on Information Theory**, v. 13, n. 1, p. 21–27, 1967.

40 JOHNSON, S. C. Hierarchical clustering schemes. **Psychometrika**, v. 32, n. 3, p. 241–254, 1967.

41 HINTON, G. *et al.* Deep neural networks for acoustic modeling in speech recognition: the shared views of four research groups. **IEEE Signal Processing Magazine**, v. 29, n. 6, p. 82–97, 2012.

42 WESTERLUND, M. The emergence of deepfake technology: a review. **Technology Innovation Management Review**, v. 9, n. 11, p. 40–53, 2019. DOI: 10.22215/timreview/1282.

43 SILVER, D. *et al.* Mastering the game of go with deep neural networks and tree search. **Nature**, v. 529, n. 7587, p. 484–489.

44 TIAN, Y. *et al.* **Deeptest:** automated testing of deep-neural-network-driven autonomous cars. 2018. Disponível em: http://dl.acm.org/doi/pdf/10.1145/3180155.3180220. Acesso em: 07 mar. 2022.

45 WIDROW, B. **Adaptive adaline neuron using chemical memistors**. 1960. Disponível em: https://www-isl.stanford.edu/~widrow/papers/t1960anadaptive.pdf. Acesso em: 10 mar. 2022.

46 MARVIN, M.; SEYMOUR, A. P. **Perceptrons**. Massaschutts: MIT Press, 1969.

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

48 WERBOS, P. **Beyond regression:** new tools for prediction and analysis in the behavioral sciences. 1974. Thesis (Doctor) - Harvard University, Massachusetts, 1974.

49 HOPFIELD, J. J. Neural networks and physical systems with emergent collective computational abilities. **Proceedings of the National Academy of Sciences**, v. 79, n. 8, p. 2554–2558, 1982.

50 LECUN, Y. *et al.* Backpropagation applied to handwritten zip code recognition. **Neural Computation**, v. 1, n. 4, p. 541–551, 1989.

51 KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. **Imagenet classification with deep convolutional neural networks**. 2012. Disponível em: https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf. Acesso em: 07 mar. 2022.

52 MAASS, W.; MARKRAM, H. On the computational power of circuits of spiking neurons. **Journal of Computer and System Sciences**, v. 69, n. 4, p. 593–616, 2004.

53 CAUCHY, A. Méthode générale pour la résolution des systemes d'équations simultanées. **Comptes Rendus de l'Académie des Sciences**, v. 25, p. 536–538, 1847.

54 COURANT, R. Variational methods for the solution of problems of equilibrium and vibrations. **Bulletin of the American Mathematical Society**, v. 49, p. 1–23, 1943.

55 QIAN, N. On the momentum term in gradient descent learning algorithms. **Neural Networks**, v. 12, n. 1, p. 145–151, 1999.

56 DUCHI, J.; HAZAN, E.; SINGER, Y. Adaptive subgradient methods for online learning and stochastic optimization. **Journal of Machine Learning Research**, v. 12, n. 7, p. 2121–2159, 2011.

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

58 KINGMA, D. P.; BA, J. **Adam:** a method for stochastic optimization. 2014. Disponível em: https://arxiv.org/pdf/1412.6980.pdf. Acesso em: 18 mar. 2022.

59 CHO, K. *et al.* **Learning phrase representations using RNN encoder-decoder for statistical machine translation**. 2014. Disponível em: https://arxiv.org/pdf/1406.1078.pdf. Acesso em: 18 mar. 2022.

60 SUTSKEVER, I.; MARTENS, J.; HINTON, G. E. **Generating text with recurrent neural networks**. 2011. Disponível em: https://icml.cc/2011/papers/524_icmlpaper.pdf. Acesso em: 18 mar. 2022.

61 GRAVES, A.; MOHAMED, A.-R.; HINTON, G. Speech recognition with deep recurrent neural networks. *In:* IEEE CONFERENCE ON ACOUSTICS, SPEECH AND SIGNAL PROCESSING, 2013, Vancouver. **Procedings** [...] Vancouver: IEEE, 2013. DOI: 10.1109/ICASSP.2013.6638947.

62 BENGIO, Y.; SIMARD, P.; FRASCONI, P. Learning long-term dependencies with gradient descent is difficult. **IEEE Transactions on Neural Networks**, IEEE, v. 5, n. 2, p. 157–166, 1994.

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

64 GERS, F. A.; SCHMIDHUBER, J.; CUMMINS, F. Learning to forget: continual prediction with lstm. **IET**, 1999. DOI: 10.1049/cp:19991218.

65 SUTSKEVER, I.; VINYALS, O.; LE, Q. V. **Sequence to sequence learning with neural networks**. 2014. Disponível em: https://icml.cc/2011/papers/524_icmlpaper.pdf. Acesso em: 22 mar. 2022.

66 GRAVES, A. **Generating sequences with recurrent neural networks**. 2013. Disponível em: https://arxiv.org/pdf/1308.0850.pdf. Acesso em: 22 mar. 2022.

67 WOLLMER, M. *et al.* **Abandoning emotion classes** - towards continuous emotion recognition with modelling of long-range dependencies. 2008. Disponível em: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.446.9820&rep=rep1&type=pdf. Acesso em: 12 mar. 2022.

68 ECK, D.; SCHMIDHUBER, J. **A first look at music composition using lstm recurrent neural networks**. 2002. Disponível em: http://www.iro.unmontreal.ca/~eckdoug/blues/IDSIA-07-02.pdf. Acesso em: 03 mar. 2022.

69 XU, K. *et al.* **Show, attend and tell:** neural image caption generation with visual attention. 2015. Disponível em: http://proceedings.mlr.press/v37/xuc15.pdf. Acesso em: 03 mar. 2022.

70 DONAHUE, J. *et al.* Long-term recurrent convolutional networks for visual recognition and description. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 39, n. 4, p. 677–691, 2017.

71 LUKOŠEVIČIUS, M.; JAEGER, H. Reservoir computing approaches to recurrent neural network training. **Computer Science Review**, v. 3, n. 3, p. 127–149, 2009.

72 PATHAK, J. *et al.* Model-free prediction of large spatiotemporally chaotic systems from data: A reservoir computing approach. **Physical Review Letters**, v. 120, n. 2, p. 024102, 2018.

73 YILDIZ, I. B.; JAEGER, H.; KIEBEL, S. J. Re-visiting the echo state property. **Neural Networks**, v. 35, p. 1–9, 2012. DOI:10.1016/j.neunet.2012.07.005.

74 MACHICAO, J.; BRUNO, O. M. Improving the pseudo-randomness properties of chaotic maps using deep-zoom. **Chaos:** an interdisciplinary journal of nonlinear science, v. 27, n. 5, p. 053116, 2017.