Universidade de Brasília

Instituto de Física - IF-UnB

O Algoritmo New Generation Reservoir Computing e Suas Aplicações a Sistemas Dinâmicos

Victor Ribeiro Borges

Brasília-DF
2024

Victor Ribeiro Borges

O Algoritmo New Generation Reservoir Computing e Suas Aplicações a Sistemas Dinâmicos

Trabalho de Conclusão de Curso apresentado à Universidade de Brasília como parte das exigências para obtenção do título de Bacharel em Física

Orientador:
Tarcísio Marciano da Rocha Filho

Brasília-DF
2024

1 Introdução

Nesta seção, serão dadas noções gerais sobre as teorias neste trabalho empregadas.

1.1 Redes Neurais

Uma rede neural é um modelo de aprendizado de máquina inspirado no cérebro humano. É composta por camadas de "neurônios"conectados entre si. Cada neurônio recebe entradas, recebe uma função de ativação e passa a saída para a próxima camada. O treinamento da rede ajusta os pesos dessas camadas para minimizar o erro entre as previsões da rede e os resultados reais. Há vários modelos de redes artificiais, cada um mais adequado para certos processos e tarefas.

Em geral, há dois grandes tipos de redes neurais. As Redes Neurais Feedforward, as FNNs (*feed-forward neural networks*) são redes em que as informações apenas viajam do input em direção ao output. Ou seja, as informações não são reprocessadas pela rede. Portanto não são utilizadas relações de recorrência, ciclos ou loops dentro do algoritmo. É importante frisar que as FNNs são adequadas para tarefas onde as entradas e saídas são independentes entre si. Portanto não é necessário que haja uma memória entre os dados adjacentes.

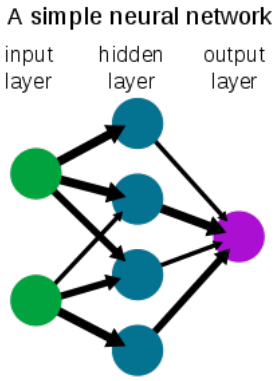


Figura 1: Um exemplo básico de uma FNN

O outro grande tipo são as redes neurais recorrentes, as RNNs (*recurrent neural networks*), que vão na linha oposta das FNNs. São redes em que as informações serão reprocessadas e retroalimentadas através da rede. Ou seja, as saídas de alguns neurônios são usadas como entradas em passos seguintes, em forma de relações de recorrência. Isto permite que a rede mantenha uma "memória"do que foi processado anteriormente. Portanto, as RNNs são adequadas para tarefas onde as entradas e saídas estão relacionadas entre si ao longo de uma sequência, o que exige a retenção de informações passadas. Por exemplo, em textos, onde as palavras adjacentes têm correlação e o contexto importa. Sistemas dinâmicos têm leis de formação, e portanto contêm essa correlação entre passos adjacentes.

Séries temporais, em sua grande maioria, mesmo não tendo leis de formação por motivos óbvios (meteorologia, ativos financeiros, etc), também contêm cor-

relações subliminares. E não apenas entre passos consecutivos, mas contêm sazonalidades e até mesmo aleatoriedades (preços oscilando por motivos externos, ou internos ao próprio mercado de ações). Em suma, quando é necessária essa memória entre passos adjacentes dentro do contexto dos dados de entrada e os dados de saída, é mais adequada o uso de RNNs.

Os dados de entrada impactam na escolha da arquitetura da rede. Mas dados podem ser reprocessados utilizando de meios diversos, tal que os mesmos dados podem treinar tanto FNNs como RNNs, a depender de seus formatos finais antes do processo de treino da rede. Como mostrado em [6], é utilizada a princípio uma série temporal tradicional dos valores dos contratos futuros da IBOVESPA ao longo de dias, mas depois os valores de entrada são, além de abreviados, descorrelacionados entre si utilizando de uma análise do componente principal (seção 5.2). Em suma, a estrutura de dados inicialmente mais adequada para um treino de uma RNN acaba, depois de um processamento intermediário, resultando em um conjunto de dados (de entrada) mais adequado para uma FNN.

Esta análise de regra também se alia ao tamanho do conjunto de dados. Por via de regra, quando o conjunto de dados é muito grande, é recomendado um processamento intermediário dos dados de entrada, isto é, tratamento estatístico, identificação de correlações entre coordenadas, etc. A rede nem sempre consegue identificar por si só essas correlações, fazendo com que o treinamento seja insuficiente para uma replicação do clima do sistema. Em suma, a rede neural é capaz até certo ponto de usar o input e manipulá-lo para virar um output verossímil. A estrutura e o tamanho dos dados também influi na rede.

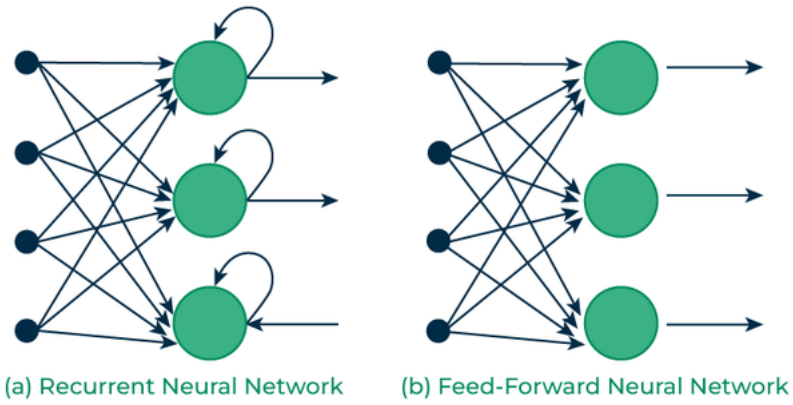


Figura 2: Um exemplo básico de uma RNN comparado a uma FNN. Vemos que a camada oculta reprocessa as informações já processadas. Retirado de [4]

O principal problema do treinamento de uma RNN é o de dissipação de gradiente (*vanishing gradient*). Acontece quando os gradientes calculados ao longo dos loops se tornam muito pequenos. Isso faz com que os pesos das camadas iniciais mudem muito lentamente ou quase não mudem, dificultando o aprendizado da rede. Então a rede tem dificuldade em capturar dependências de longo prazo, pois a influência das entradas iniciais da sequência diminui drasticamente. Na prática, os outputs não são mais condizentes com os inputs,

já que o output estagna em torno de um ponto.

O outro problema semelhante é o de explosão de gradiente (*exploding gradient*), que é o oposto. O gradiente vai a infinito, fazendo com que o output exploda ao infinito descontroladamente.

Esses dois problemas podem ser causados pela arquitetura da rede inadequado para o tipo de dado a ser processado, e o processamento dos dados em si. Funções de ativação inadequadas podem fazer com que o problema seja irremediável, mesmo com a escolha adequada de metaparâmetros. A solução passa por fazer renormalizações ao longo da rede, clipping dos gradientes, aplicar conceitos da arquitetura *long short-term memory*, dentre outros recursos.

1.2 Reservoir Computing

O Reservoir Computing, ou RC, é um tipo de rede neural recorrente, que mapeia o input para um reservatório, que é uma camada de neurônios conectados de forma aleatória. Este reservatório recria a dinâmica do input, que uma vez criada, essa dinâmica só necessita de dados iniciais para então formar uma previsão. A idéia é mapear o input para um vetor de dimensão maior, o reservatório.

O reservatório é representado pela dinâmica abaixo:

$$\boldsymbol{r}_{i+1} = (1-\gamma)\boldsymbol{r}_i + \gamma f(\boldsymbol{A}\boldsymbol{r}_i + \boldsymbol{W}\boldsymbol{X}_i + \boldsymbol{b}) \tag{1}$$

Desta equação é importante notar γ, que é a taxa de decaimento dos nodos, ou seja, o quanto demora para que os nodos mais antigos se percam no tempo. Nota-se também a função de ativação f, que ativa as conexões e dá o dinamismo ao treinamento. $\boldsymbol{r}_i = [r_{1,i}, r_{2,i}, ..., r_{N,i}]^T$ é o reservatório, e suas componentes são os n-ésimos estados dos nodos no i-ésimo instante de tempo. Ou seja, depende de uma função de ativação, e há vários parâmetros a serem ajustados. Observe que para cada instante de tempo no treinamento, há N nodos diferentes.

A matriz de saída, ou matriz de pesos é gerada desta forma:

$$\boldsymbol{W}_{out} = \boldsymbol{Y}_d \boldsymbol{O}_{total}^T (\boldsymbol{O}_{total}\boldsymbol{O}_{total}^T + \alpha \boldsymbol{I})^{-1} \tag{2}$$

Onde Y_d é o output desejado, $\boldsymbol{I}$ é a matriz identidade, α é a constante de regularização (*ridge parameter*) $\boldsymbol{O}_{total}$ é o vetor de características. No caso do RC, ao usar uma função de ativação não-linear, pode-se então usar o vetor de característica linear, ou seja, apenas as componentes do reservatório:

$$\boldsymbol{O}_{total,i} = \boldsymbol{O}_{lin,i} = \boldsymbol{r}_i \tag{3}$$

Em suma, mapeia-se o input ao reservatório. Depois, treina-se a matriz de saída. E depois disso, se faz uma transformação linear usando da matriz de saída:

$$\boldsymbol{Y}_{i+1} = \boldsymbol{W}_{out}\boldsymbol{O}_{total,i+1} \tag{4}$$

1.3 Next Generation Reservoir Computing

O NG-RC, ou *Next Generation Reservoir Computing*, é uma versão otimizada do *Reservoir Computing*. O NG-RC usa da autoregressão de vetores não-lineares, que diminui consideravelmente o número de parâmetros ajustáveis. Deve-se

frisar que os vetores são não-lineares por causa que agora não há mais a função de ativação do RC tradicional. Os resultados expressados por ambas arquiteturas mostram uma equivalência entre o RC com a autoregressão de vetores não-lineares, e uma equivalência desta com o NG-RC.

O funcionamento desta rede é semelhante ao do RC tradicional. Primeiramente, precisa-se ter os inputs, que são chamados de séries temporais (*time series*). Essas séries são coordenadas em função do tempo:

$$\boldsymbol{r}(t) = [r_1(t), r_2(t), ..., r_n(t)]^T \tag{5}$$

Pode-se (no caso deste algoritmo é recomendado) normalizar essas séries, isto é:

$$r_i(t) \to \frac{r_i(t)}{\max(|r_i(t)|)} \tag{6}$$

Isto é computacionalmente recomendado, já que números muito grandes em magnitude podem criar distorções numéricas nas iterações, ou seja, explosão dos gradientes. Claro que a normalização não resolve este problema por si só.

Depois disso, se ajustam os vários parâmetros de entrada do algoritmo. Por exemplo, o parâmetro de regularização α (*ridge parameter*) que pode ser interpretado como um multiplicador de Lagrange de um certo vínculo da série temporal.

Depois, são determinadas as dimensões dos vetores de entrada d e saída (em geral essas dimensões são iguais), o passo inicial na série n_{ini}, o número de passos k usados no vetor de característica (*feature vector*), o número de passos que será usado no aprendizado da máquina n_{learn}, e os graus dos monômios usados no vetor de característica. Pode-se ajustar também s, que é basicamente o passo (*step*) entre cada dado da série temporal.

A determinação destes parâmetros é feita usando da tentativa e erro, exigindo experimentação ou heurísticas para otimizar sua configuração. Leva-se em conta o comportamento do forecast em relação aos dados reais, a razoabilidade numérica das séries e o cálculo do erro. O erro será definido como o RMSE, mais conhecido como raiz do erro quadrático médio:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{\boldsymbol{r}}_i - \boldsymbol{r}_i)^2} \tag{7}$$

Onde $\hat{\boldsymbol{r}}_i$ é o *forecast* feito pelo algoritmo NGRC, e $\boldsymbol{r}_i$ são os dados reais com os quais se compara o *forecast*. Considerando uma mudança de parâmetros do algoritmo que mantenha o número de passos de *forecast* n constante, pode-se comparar o RMSE de vários *forecasts* diferentes, tal que o de menor magnitude seja preferível aos demais.

É importante notar que o RMSE não é o único valor de referência, já que a ideia é replicar o clima dos dados de entrada. Há casos em que o RMSE de um treinamento é menor, e o forecast gerado por este treinamento é de pior qualidade que algum treinamento com RMSE maior.

Para simplicidade da notação, definimos:

$$\boldsymbol{O}_i = \boldsymbol{O}_{lin,i} \tag{8}$$

Este é o vetor de característica linear, do qual se constrói todo o resto dos vetores não-lineares.

O vetor de característica no instante de tempo i, agora não-linear (*features vector*) será dado por:

$$\boldsymbol{O}_{total,i} = \bigoplus_j \boldsymbol{O}_i^j \tag{9}$$

Onde $\boldsymbol{O}_i^j$ são os vetores de característica no instante de tempo i, de grau j:

$$\boldsymbol{O}^j = \lceil \bigotimes_j \rceil \boldsymbol{O}_t = \boldsymbol{O}_t \lceil \otimes \rceil \boldsymbol{O}_t \lceil \otimes \rceil \boldsymbol{O}_t \lceil \otimes \rceil ... \lceil \otimes \rceil \boldsymbol{O}_t \text{ (produto de } j \text{ vetores lineares)} \tag{10}$$

Onde $\lceil\ \rceil$ denota apenas os monômios únicos do produto tensorial.

A título de exemplo, se escolhermos $j = [1]$, temos:

$$\boldsymbol{O}_{total,i} = \boldsymbol{O}_i \tag{11}$$

Determinados os metaparâmetros, o algoritmo começa a rodar, iniciando por compor o vetor de característica. Ele na prática é uma matriz onde o número de linhas é determinado pelo número de monômios únicos derivados dos graus desejados e o número de colunas é o número de passos usados para o aprendizado. Por exemplo, usando os graus 0, 1 e 2, $k = 2$, número de passos pro aprendizado $n_{learn} = 500$, temos uma matriz 28 x 500.

Após a composição de $\boldsymbol{O}_{total}$ será preparada a matriz $\boldsymbol{Y}_d$, que é a matriz usada na fase de aprendizado, composta das diferenças entre passos consecutivos.

Agora deve-se computar a matriz $\boldsymbol{W}_{out}$ que é a matriz peso de saída que será usada no *forecast*. Ela é calculada como:

$$\boldsymbol{W}_{out} = \boldsymbol{Y}_d \boldsymbol{O}_{total}^T (\boldsymbol{O}_{total} \boldsymbol{O}_{total}^T + \alpha \boldsymbol{I})^{-1} \tag{12}$$

Onde $\boldsymbol{I}$ é a matriz identidade. Após isso, há a fase de *forecasting*, onde é usada a equação:

$$\boldsymbol{X}_{i+1} = \boldsymbol{X}_i + \boldsymbol{W}_{out} \boldsymbol{O}_{total,i} \tag{13}$$

Onde $\boldsymbol{O}_{total,i}$ é o vetor de característica apenas no instante de tempo i. Então há essa relação de recorrência que gera a previsão para cada instante de tempo. Em teoria, pode-se levar essa relação de recorrência até onde for desejado. Na prática apenas é fixado um número de passos onde se cessa o forecast.

1.4 Expoente de Lyapunov

Dado um sistema dinâmico, os expoentes de Lyapunov caracterizam como que duas trajetórias se separam, dado uma separação dos seus pontos iniciais.

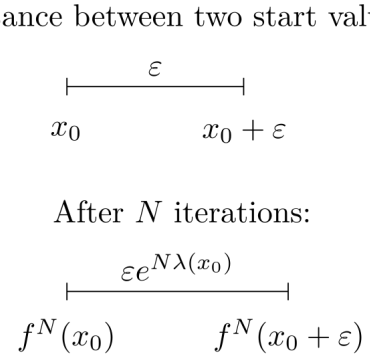


Figura 3: Separação inicial e separação após várias iterações na série temporal

O expoente de Lyapunov λ é definido como o expoente que satisfaz a relação abaixo:

$$|\delta \boldsymbol{r}(t)| \approx e^{\lambda t} |\delta \boldsymbol{r}(0)| \tag{14}$$

Onde $\delta \boldsymbol{r}(t)$ é a separação das duas trajetórias em função do tempo. Fazendo $\epsilon = |\delta \boldsymbol{r}(0)|$ se tem:

$$\lambda(t) = \frac{1}{t} \ln \frac{|\boldsymbol{\delta}(t)|}{\epsilon} \tag{15}$$

Ou seja, este expoente depende do tempo. Entretanto, se for observado o gráfico desta expressão ao longo do tempo para qualquer sistema, se vê que depois de muito tempo o gráfico se estabiliza em torno de um número. Então é comum o limite:

$$\lambda_{max} = \lim_{t \to \infty} \lim_{\epsilon \to 0} \frac{1}{t} \ln \frac{|\boldsymbol{\delta}(t)|}{\epsilon} \tag{16}$$

λ_{max} é o chamado **Expoente de Lyapunov Maximal**. Então na prática utiliza-se um ϵ muito menor em módulo que a magnitude da separação usual $|\boldsymbol{\delta}(t)|$. Também se fixa uma coordenada para se fazer a separação inicial, já que se percebe que este limite resulta no mesmo número ao fazer a separação inicial em qualquer coordenada.

De forma discretizada, pode ser feito de várias formas, mas uma forma é fazer uma função auxiliar $l(t_i)$:

$$l(t_i) = \frac{\log(|\delta(t_i)|)}{2\Delta t} \tag{17}$$

Onde Δt é o intervalo de tempo fixo entre duas iterações sucessivas. Esta função auxiliar mostra o comportamento da separação ao longo do tempo.

Esta função não é uma função simplesmente afim. Mas em suma, o expoente de Lyapunov maximal seria a **inclinação média** desta função. Considerando que há várias destas funções considerando pontos iniciais diferentes e suas separações iniciais respectivas, temos J funções auxiliares:

$$\lambda_{max,j} = \frac{d}{dt} l_j(t_i) \tag{18}$$

Onde j denota a j-ésima função auxiliar.

Isto é, se faz um ajuste linear que mais se aproxime da função ao longo prazo, e a inclinação deste ajuste se aproxima do expoente desejado. Na seção seguinte será mostrado na prática como se utiliza desta função.

Adaptando este cálculo ao algoritmo NG-RC, é buscada uma média de vários expoentes de Lyapunov calculados ao longo de várias trajetórias e pontos iniciais distintos. Primeiramente se fixa um ponto inicial na série temporal e é feito um forecast a partir deste. Após isso, é feita uma separação muito pequena no ponto inicial anterior, e se faz o forecast deste novo ponto. E então se gera a função auxiliar, da qual se extrai $\lambda_{max,j}$. Então para cada ponto inicial da série, calcula-se um par de forecasts atrelados. Depois de m expoentes calculados, se espera uma convergência das médias destes expoentes calculados ao valor real de λ_{max}:

$$\lambda_{max} = \frac{1}{m} \sum_{j=1}^{m} \lambda_{max,j} \tag{19}$$

1.5 Algoritmo de Cálculo do Expoente de Lyapunov Maximal

Primeiramente, se faz o treinamento da rede neural usando do algoritmo NG-RC. Se usa uma série temporal de dados reais para treinar a rede. Depois de treinada, tem-se a matriz de pesos $\boldsymbol{W}_{out}$ que constrói uma dinâmica treinada do sistema dado pontos iniciais. E por fim, se pode usar pontos iniciais com uma separação ϵ da trajetória inicial, para isto comparar as separações na dinâmica dos dois sistemas e por fim fazer o cálculo de λ. Usa-se (17) para estimar $\lambda_{max,j}$.

Após isso, encontrar a inclinação de $f_i(t_i)$:

$$\lambda_{max,j} = \frac{d}{dt} f_j(t_i) \tag{20}$$

E depois, repetir o procedimento para vários intervalos de tempo dentro da mesma série temporal, e tirar a média dos expoentes de Lyapunov de cada treinamento diferente:

$$\lambda_{max} = \frac{1}{n} \sum_{j=1}^{n} \lambda_{max,j} \tag{21}$$

Abaixo uma figura de um exemplo do plot desta função, usando como exemplo o sistema de Lorenz. É importante ressaltar que foi cortado o início e o fim desta série, já que no início desta, as trajetórias mantêm uma separação persistentemente constante. E há no final da série uma "saturação", isto é, a separação se mantém constante após uma evolução, na média, afim (linear). Também é importante ressaltar que se busca a inclinação desta curva apenas após a separação inicial evoluir para valores maiores.

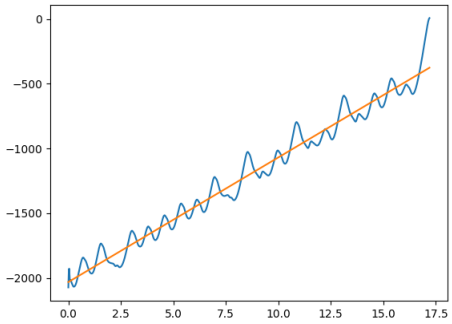


Figura 4: Plot de $l(t)$

Se vê que a curva oscila bastante, mas há esta inclinação média. Os resultados são melhores quando se calcula a inclinação usando apenas os vales da função. Isso será demonstrado na seção referente aos resultados do cálculo deste expoente.

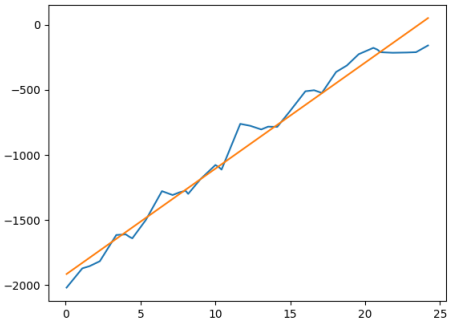


Figura 5: Plot de $l(t)$ usando apenas os vales da função.

2 Aplicações e Resultados

Nesta seção, serão trazidos alguns sistemas e serão feitos *forecasts* usando o algoritmo NGRC. Será dado, tanto quanto possível, os conjuntos dos parâmetros empregados na confecção do *learning* e do *forecast*, e os ajustes empregados para que o resultado fosse mais aceitável. Também é importante ressaltar que se usa a tentativa e erro no ajuste destes parâmetros, o que pode dar ao leitor uma sensação de aleatoriedade. Também se usa o RMSE como descrito na equação (7). Também em alguns sistemas será feito o cálculo do expoente de Lyapunov maximal utilizando dos *forecasts* trazidos pelo algoritmo NGRC.

2.1 Sistema de Lorenz

O sistema de Lorenz é um conjunto de três equações diferenciais:

$$\frac{dx}{dt} = \sigma(y - x) \tag{22}$$

$$\frac{dy}{dt} = x(\rho - z) - y \tag{23}$$

$$\frac{dz}{dt} = xy - \beta z \tag{24}$$

x, y e z são as coordenadas cartesianas usuais, e σ, ρ e β são parâmetros. Dependendo da escolha destes parâmetros, a solução será caótica, o que gera o chamado atrator de Lorenz.

Aqui se usa uma série temporal gerada utilizando os parâmetros padrão[2]: $\sigma = 10$, $\beta = 8/3$, $\rho = 28$, com pontos iniciais $(x, y, z) = (0.1, 0, 0)$

Cortando os primeiros 1000 passos da série para pular a fase de transiente, tem-se aqui um forecast de 1000 passos, usando os graus do algoritmo 1, 2 e 3, e $\alpha = 10^{-6}$

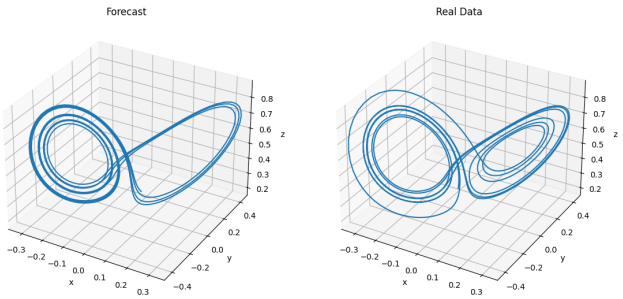
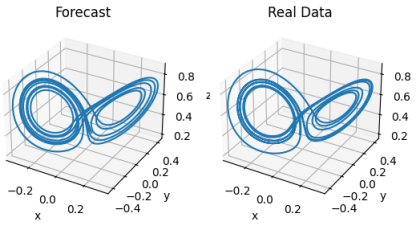
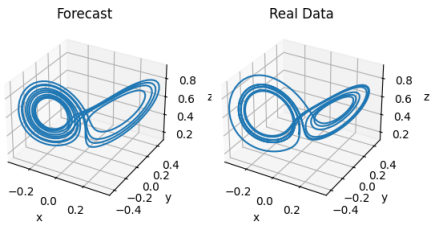


Figura 6: usando k=2, s=1. Erro=0.13677



(a) usando k=3, s=2. Erro=0.113



(b) usando k=5, s=3. Erro=0.16168

Aqui é comparado o forecast feito pelo NG-RC com os dados reais feito por integração numérica usual, usando do sistema de equações diferenciais. Aqui é demostrado que o algoritmo consegue fazer boas replicações do sistema.

Para o cálculo de λ_{max} usando do algoritmo NGRC, será usado 1000 passos para o treinamento, 3000 passos para o forecast, $k = 3$, $s = 2$, mantendo os graus utilizados anteriormente e o mesmo α. Usando do método mencionado na introdução, temos a estimativa:

$$\lambda_{max,comp} = 0.843 \tag{25}$$

Além da média, temos a mediana em 0.860. Foram usadas 68 amostras. Segundo [1], utilizando de outros métodos numéricos para o cálculo deste expoente, se tem $\lambda_{max} = 0.904$.

2.2 Sistema de Rössler

O sistema de Rössler é um conjunto de três equações diferenciais:

$$\frac{dx}{dt} = -y - z \tag{26}$$

$$\frac{dy}{dt} = x + ay \tag{27}$$

$$\frac{dz}{dt} = b + z(x - c) \tag{28}$$

Aqui se usa uma série temporal gerada utilizados originalmente: $a = b = 0.2$, $c = 5.7$, com pontos iniciais $(x, y, z) = (0.1, 0, 0)$

Abaixo temos aqui um forecast de 3000 passos, usando os graus do algoritmo 1, 3 e 4, e $\alpha = 10^{-3}$

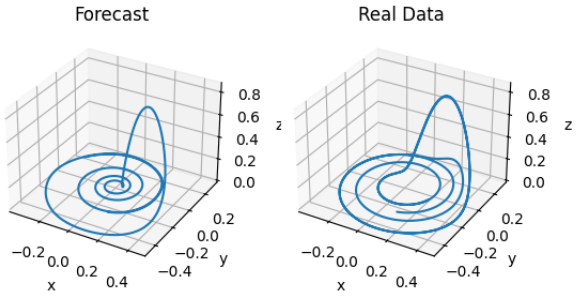


Figura 8: usando k=2, s=1. Erro=0.0224

Para o cálculo de λ_{max} usando do algoritmo NGRC, será usado 1000 passos para o treinamento, 3000 passos para o forecast, $k = 2$, $s = 1$, graus 1, 3 e 4, e $\alpha = 10^{-3}$. Usando do método mencionado na introdução, temos a estimativa:

$$\lambda_{max,comp} = 0.101 \tag{29}$$

Além disso, temos a mediana em 0.097. Foram usadas 62 amostras. Segundo [1], o valor mais aceito é $\lambda_{max} = 0.071$.

2.3 População Mundial

Aqui se utilizam os dados de população mundial segundo o site *Our World in Data*[5], entre os anos de 1950 e 2001 para alimentar a rede neural. Será feita

uma tentativa de fazer uma previsão da população entre 2002 e 2021 e comparar com os dados reais, e depois fazer uma previsão para os anos após 2021.

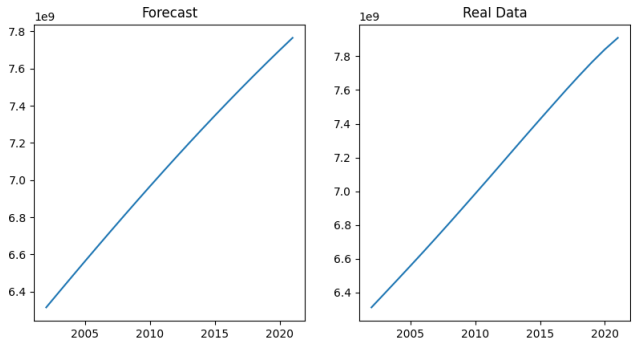


Figura 9: usando k=1, s=1, graus 1 e 3. Erro=0,00942

A progressão da população é quase afim, sendo que se tem um achatamento no final da série quase imperceptível. Aqui se tem uma série de facílimo aprendizado que um simples ajuste de curva daria conta.

Agora, considerando que o algoritmo aprendeu o ritmo da série temporal, será feito um extrapolamento para 100 anos após o fim da série temporal.

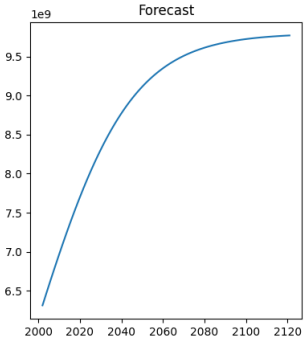


Figura 10: Forecast usando do mesmo W_{out} da série acima.

Aqui se vê o achatamento mais acentuado previsto pela rede. Se usado, ao invés dos graus 1 e 3, os graus 1 e 2, tem-se um erro menor e o achatamento diminui.

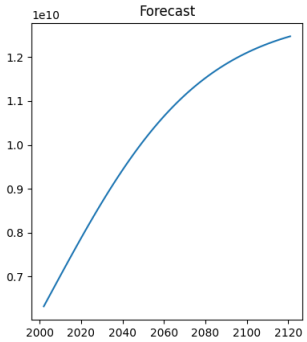


Figura 11: Forecast usando dos graus 1 e 2. Erro=0.00336

Dado o erro menor, é mais plausível a estimativa de que, daqui 100 anos, teremos em torno de 12,5 bilhões de pessoas no mundo.

2.4 Previsões Meteorológicas

Aqui, usando a série histórica de dados meteorológicos da capital do Brasil, Brasília, entre 2018 e 2022 busca-se fazer previsões meteorológicas a partir destes dados.[3]

Os dados, que são medidos de hora em hora, abrangem temperatura, velocidade e direção do vento, radiação solar, precipitação, dentre outros. Inicialmente será usado vários desses dados para uma previsão, mas após falhar em produzir um *forecast* minimamente realista, o foco está agora na temperatura.

Primeiramente se utiliza a temperatura média por dia. Tentando vários parâmetros para otimização dos resultados, não foi conseguido algum resultado significativo. Abaixo os dados desta abordagem, plotando apenas a temperatura mínima:

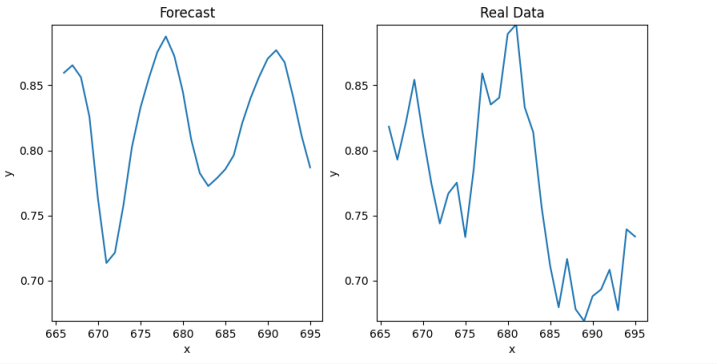


Figura 12: k=10, s=2, $\alpha = 5e - 5$, 365 passos de learning, 30 de forecast, graus 0, 1 e 2 e constante=0.1

O resultado acima foi o mais significativo usando a temperatura média. Como pode-se ver, não replica a dinâmica dos dados reais no mesmo intervalo. Foi tentado vários pontos iniciais da série, mas nenhuma obteve êxito.

Além disso, o forecast é bem limitado, já que ao usar mais passos de forecast, se vê que a curva estagna em algo semelhante a uma exponencial negativa.

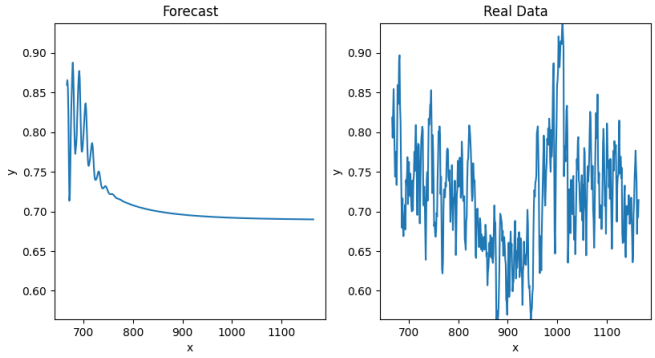


Figura 13: Mesmo modelo da figura logo acima, com 500 passos de forecast

Após isso, se tenta utilizar a temperatura mínima e máxima de cada dia, e utilizar uma terceira coluna que simbolizasse, ao menos de forma grosseira, a estação do ano. 1 para os primeiros 90 dias do ano, 2 para os seguintes 90 dias e assim por diante. Esta terceira coluna tem a intenção de ensinar para a rede uma correlação com a temperatura. Esta coluna será apelidada de "coluna de estação"

Mas os resultados novamente não fazem jus aos dados. Abaixo os gráficos.

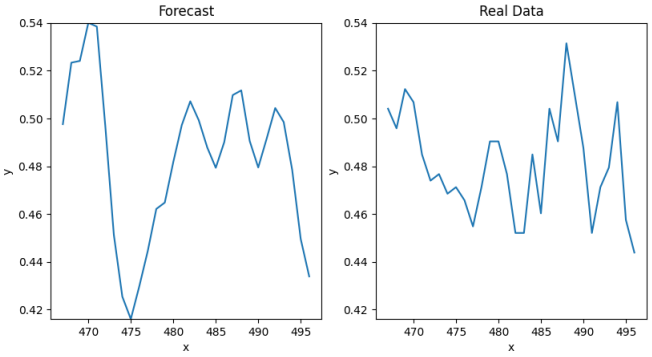


Figura 14: k=10, s=2, $\alpha = 5e-3$, 365 passos de learning, 30 de forecast, graus 0, 1 e 2 e constante=0.1, com a coluna de estação

Os gráficos acima estão em escala. Observa-se que o *forecast* está com amplitude maior. E ao aumentar o forecast para 60, o gráfico do *forecast* explode ao infinito.

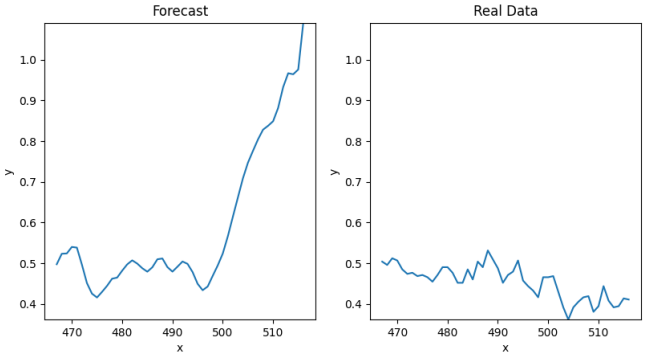


Figura 15: k=10, s=2, $\alpha = 5e-3$, 365 passos de *learning*, 50 de *forecast*, graus 0, 1 e 2 e constante=0.1, com a coluna de estação

O *forecast* acima mostra que a dinâmica se mantém semelhante aos dados reais por um tempo limitado. Após isso, explode ao infinito. Vale ressaltar que foram tentadas várias combinações de parâmetros, mas sem sucesso. Então a chamada coluna de estação não foi bem-sucedida ao ajudar na replicação dos dados meteorológicos.

3 Discussão

Neste trabalho exposto, é mostrada uma certa eficácia de replicação do algoritmo NG-RC em certos sistemas dinâmicos, se corretamente empregados os parâmetros, e em certos sistemas temporais aplicados, se os parâmetros forem suficientes para criar o ambiente de replicação.

No tocante ao sistema de Lorenz, o treinamento foi bem-sucedido. No sistema de Rössler, houve uma certa dificuldade ao encontrar parâmetros adequados. Mas estas são séries com menor grau de dificuldade de treinamento, já que não necessita de uma análise estatística mais complexa. Apenas uma normalização simples dos dados é suficiente. Além disso, há dados históricos e uma grande bibliografia sobre estes sistemas dinâmicos. Então já se conhecem os sistemas, além de como estes devem evoluir no tempo. A inclusão destes sistemas neste trabalho visa o cálculo do expoente de Lyapunov maximal utilizando da rede neural NG-RC. Os valores encontrados dos expoentes não são satisfatórios, já que foi utilizado muito recurso computacional nestes cálculos. Em teoria, o uso de muitos forecasts utilizando de apenas um treino reduziria o custo computacional e aumentaria a precisão do cálculo. Na prática, a variação dos pontos iniciais mantendo o mesmo treino gerou uma variância significativa entre o conjunto dos expoentes calculados. O algoritmo de cálculo do expoente parece ter um viés a depender do sistema analisado. Uma ideia de correção seria fazer treinamentos diferentes para cada forecast, ou então fazer uso de alguma arquitetura de rede neural distinta da utilizada neste trabalho.

Sobre o forecast da população mundial, há estimativas totalmente diferentes para o número de habitantes para daqui 20 anos, e as estimativas diferem muito mais se o intervalo analisado é maior. Um dos consensos sobre este assunto é que a curva de população está de fato achatando, por vários motivos socioculturais

e políticos. Esta curva ainda não achatou atualmente de forma tão acentuada quanto irá no futuro, mais ou menos daqui 50 anos.

Sobre as previsões meteorológicas, foi-se tentado alguns métodos de processamento dos dados de entrada, mas nenhum com sucesso no longo prazo. O processo intermediário dos dados de entradas não foram suficientes para que um forecast adequado fosse gerado.

Aqui neste sistema é mais evidente o problema de dissipação de gradientes, no qual se inviabiliza o forecast. Poderia ser tentado algo como em [6] usando de análise do componente principal, já que a natureza dos dados é semelhante ao utilizado neste trabalho, além de que neste artigo há certo sucesso no forecast. Em suma, a estrutura dos dados não foi adequada o suficiente à rede neural utilizada.

4 Código no Github

O código utilizado por este trabalho estará disponível aqui, no Github. Este é um repositório onde ficará armazenado o código. Tanto o NG-RC quanto o algoritmo usado no cálculo do expoente de Lyapunov maximal estarão neste repositório.

5 Agradecimentos

Agradeço ao Professor Tarcísio Marciano pelo seu tempo dispendido em meu auxílio, tanto na confecção deste Trabalho de Conclusão de Curso, quanto na mentoria em meu Projeto de Iniciação Científica. Também agradeço ao Professor Annibal e ao Professor Ismael que compuseram minha banca do PTCC. Também os agradeço pela inspiração teórica que resultou nestes dois trabalhos.

Referências

[1] Yu I Bogdanov e N A Bogdanova. "The study of Lorenz and Rössler strange attractors by means of quantum theory". Em: *Laser Physics* 25.3 (fev. de 2015), p. 035203. DOI: 10.1088/1054-660X/25/3/035203. URL: https://dx.doi.org/10.1088/1054-660X/25/3/035203.

[2] Daniel J. Gauthier et al. "Next generation reservoir computing". Em: *Nature Communications* 12.1 (set. de 2021), p. 5564. ISSN: 2041-1723. DOI: 10.1038/s41467-021-25801-2. URL: https://doi.org/10.1038/s41467-021-25801-2.

[3] *Instituto Nacional de Meteorologia (INMET). Available at:* 2024. URL: https://portal.inmet.gov.br/dadoshistoricos.

[4] *Introduction to Recurrent Neural Network*. URL: https://www.geeksforgeeks.org/introduction-to-recurrent-neural-network/.

[5] Hannah Ritchie et al. "Population Growth". Em: *Our World in Data* (2023). https://ourworldindata.org/population-growth.

[6] Tareísio M. Rocha Filho e Paulo M.M. Rocha. “Evidence of inefficiency of the Brazilian stock market: The IBOVESPA future contracts”. Em: *Physica A: Statistical Mechanics and its Applications* 543 (2020), p. 123200. ISSN: 0378-4371. DOI: https://doi.org/10.1016/j.physa.2019.123200. URL: https://www.sciencedirect.com/science/article/pii/S0378437119317996.