UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
CENTRO TECNOLÓGICO
DEPARTAMENTO DE ENGENHARIA ELÉTRICA
PROJETO DE GRADUAÇÃO

WILLIAN ABREU FERREIRA

AVALIAÇÃO DA SUBAMOSTRAGEM E DO RUÍDO NA PREDIÇÃO DE SÉRIES TEMPORAIS UTILIZANDO REDES NEURAIS

VITÓRIA – ES
OUTUBRO/2018

WILLIAN ABREU FERREIRA

AVALIAÇÃO DA SUBAMOSTRAGEM E DO RUÍDO NA PREDIÇÃO DE SÉRIES TEMPORAIS UTILIZANDO REDES NEURAIS

Parte manuscrita do Projeto de Graduação do aluno **Willian Abreu Ferreira**, apresentada ao Departamento de Engenharia Elétrica do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para aprovação na disciplina "ELE08552 – Projeto de Graduação 2".

DSc. Klaus Fabian Coco
Orientador

DSc. Patrick Marques Ciarelli
Comissão avaliadora

MSc. Vitor Faiçal Campana
Comissão avaliadora

VITÓRIA – ES
OUTUBRO/2018

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ANN	*Artificial Neural Networks* (Redes Neurais Artificiais)
CNS	*Central Nervous System* (Sistema Nervoso Central)
BC/UFES	Biblioteca Central / UFES
LCEE/UFES	Laboratório de Computação da Engenharia Elétrica / UFES
Ufes	Universidade Federal do Espírito Santo
SGD	*Stochastic Gradient Descent* (Gradiente Descendente Estocástico)
TanH	Tangente Hiperbólica
ReLU	*Rectified Linear Unit* (Ativação Linear Retificada)
ELU	*Exponential Linear Unit* (Unidade Linear Exponencial)
BPTT	*Back Propagation Through Time* (Propagação reversa através do tempo)
EBPTT	*Epochwise Back Propagation Throught Time*
ARCH	*Autoregressive Conditional Heteroskedasticity*
ARIMA	*Autoregressive Integrated Moving Average*
STAR	*Smooth Transition Autoregressive*
MAE	*Mean Absolute Error*
MSE	*Mean Squared Error*
RMSE	*Root Mean Squared Error*
MAPE	*Mean Absolute Percentage Error*
HP	Horizonte de Predição

LISTA DE SÍMBOLOS

x_t	Uma observação da série temporal
t_d	Intervalo de tempo definido e constante de uma série temporal
m_t	Função tendência
s_t	Função sazonalidade
Y_t	Função ruído
X_t	Modelo de decomposição clássico
w_{k_n}	*Synaptic weights*
t_s	Taxa de amostragem
φ	Função de ativação
b_k	Sinal de *Bias*
x_i	Amostras da série prevista
y_i	Amostras do conjunto de teste

RESUMO

Esse trabalho aborda conceitos de aprendizado de máquina e redes neurais para desenvolver uma interface didática que possibilita a avalição de dois efeitos indesejados em sistemas de aquisição de dados e predição, os efeitos da subamostragem e do ruído. Desenvolveu-se uma interface para que usuários interessados em estudar essa relação sejam capazes de chegar a conclusões sobre esses dois efeitos, sem a necessidade de terem um grande conhecimento sobre o assunto. Para tanto, são abordados conceitos de redes neurais, treinamento, aprendizado de máquina e de séries temporais, gerando ao final alguns exemplos práticos de como interpretar as saídas da interface através da análise do *Mean Squared Error*, comparação entre os dados reais e os dados de teste e gráficos auxiliares da funcionalidade NNTOOL do Matlab (Desempenho, Regressão Linear e Autocorrelação do Erro). Por fim, notou-se que, para o equacionamento matemático bem-comportado que foi utilizado nos experimentos, existiu uma suavização do efeito do ruído ao realizar a predição, mas que o sistema de predição não conseguiu, quando os dados estão subamostrados, predizer o comportamento original da série.

ABSTRACT

This work approaches the concepts of machine learning and neural networks to develop a didactic interface that allows the evaluation of two undesirable effects in data acquisition and forecasting systems, the effects of subsampling and noise. An interface has been developed in a way that users can be easily study those two effects, without much efforts and specific acknowledgment about neural networks, and despite that be able to reach conclusions about these two effects. In order to do it, the concepts of neural networks, machine learning and time series are approached, at the end of the work, are generated a few practical examples of how to interpret the interface outputs through Root Mean Squared Error analysis, comparison between real data and test data and auxiliary graphs from Matlab NNTOOL (Performance, Linear Regression and Error Autocorrelation). In the conclusion, it was noted that, for the well-behaved mathematical equation used as a case of study, there was a smoothing of the effect of the noise when performing the prediction of the data and the prediction system was not able, through subsampled data, to predict the original behavior of the series.

SUMÁRIO

1. APRESENTAÇÃO E OBJETO DE PESQUISA

1.1. Apresentação

As séries temporais são elementos muito comuns que podem ser encontradas em várias situações dentro dos campos da ciência e da engenharia, como nas finanças, negócios, indústrias, clima, saúde e muitos outros (GUTIERREZ, SESMERO e SANCHIS, 2016). Existem dois propósitos principais para se estudar as séries temporais: o primeiro consiste em entender ou modelar o processo estocástico relacionado a série temporal e o segundo consiste em predizer os valores futuros de uma série temporal baseado em seus valores passados (CRYER e CHAN, 2010).

As séries temporais vêm ganhando maior atenção recentemente com a tendência de aplicação de estratégias de análise de dados para grandes volumes de informação, os chamados sistemas de *big data*. Nesses sistemas, o uso de máquinas de inteligência artificial vem demonstrando grande habilidade em obter informações, especialmente no que diz respeito às análises para predição de séries temporais (*forecasting*).

Nesse contexto, o uso de máquinas baseadas em modelagem de sistemas biológicos, como as redes neurais artificiais, tem apresentado excelentes resultados (YAN, 2012). Entretanto, problemas resultantes da inadequação dos dados muitas vezes são deixados de lado, seja pelo desconhecimento de tal problema, ou pela inobservância das premissas básicas para a coleta dos dados.

Um desses problemas, que será abordado ao longo deste trabalho, é relacionado ao ruído. Geralmente esse efeito, que está presente em todos os sistemas existentes das mais variadas maneiras, acaba sendo desconsiderado quando o assunto é *forecasting*, e assim diminui a confiabilidade das análises realizadas (SANG, WANG e WU, 2009).

Outro problema a ser abordado, tratado como requisito imprescindível na aquisição de dados discretos, é a observância da taxa de amostragem com a qual o sistema de aquisição operou na obtenção dos dados. O teorema de Nyquist-Shannon específica valores mínimos para a taxa de amostragem de maneira a garantir a recomposição adequada do sinal desejado, mas o que é observado na maioria das situações é um descaso com esse rigor teórico primordial.

1.2. Objeto

O objeto de estudo consiste em Séries Temporais, com características controladas, para a aplicação do experimento proposto.

1.3. Escopo

Delimita-se o escopo deste trabalho na avaliação dos efeitos da taxa de amostragem e de ruído gaussiano branco na predição de séries temporais, utilizando redes neurais artificiais clássicas.

2. JUSTIFICATIVA

Tradicionalmente, a predição de séries temporais é realizada usando modelos estatísticos, entretanto durante os últimos anos as redes neurais artificiais (*Artificial Neural Networks*– ANN) vem apresentando um desempenho ligeiramente superior em grande parte dos trabalhos publicados (YAN, 2012). Comparado com os modelos baseados apenas em estatística, as ANN apresentam a particularidade de trabalhar com dados não lineares e ter maior aplicabilidade (YAN, 2012).

Apesar desse pensamento em relação as características das ANN, comum entre vários autores, e das vantagens que esses autores citam em relação as ANN, alguns estudos apontam um desempenho inferior desse método em relação aos métodos estatísticos convencionais e muito disso é atribuído à utilização e validação incorreta do método (YAN, 2012). De um estudo relacionado a 48 casos analisados, apenas 11 (22,9%) estavam corretamente implementados e validados (ADYA e COLLOPY, 1998), portanto, a investigação dos efeitos comumente ignorados na coleta de dados nos diversos processos industriais, como a existência de ruído e erros na taxa de amostragem, podem incrementar a discussão geral sobre o assunto.

3. OBJETIVOS

3.1. Objetivo geral

O objetivo principal de estudo deste trabalho é a avaliação dos efeitos de alguns tipos de ruídos e de erros de amostragem no processo de predição (*forecasting*) em séries temporais, utilizando redes neurais artificiais (*Artificial Neural Networks*).

3.2. Objetivos específicos

- Aplicar a teoria de *Artificial Neural Networks* para a predição de séries temporais em sinais controlados contendo ruídos e erros de amostragem;
- Avaliar se a subamostragem dos dados utilizados para treinamento da rede neural causa um efeito considerável na predição;
- Avaliar como a presença de ruído nos dados a serem utilizados para treinamento de séries temporais afetam a rede neural projetada;
- Desenvolver uma interface que possibilite de maneira simples os testes e possa ser utilizada para estudar esses efeitos sobre as series temporais.

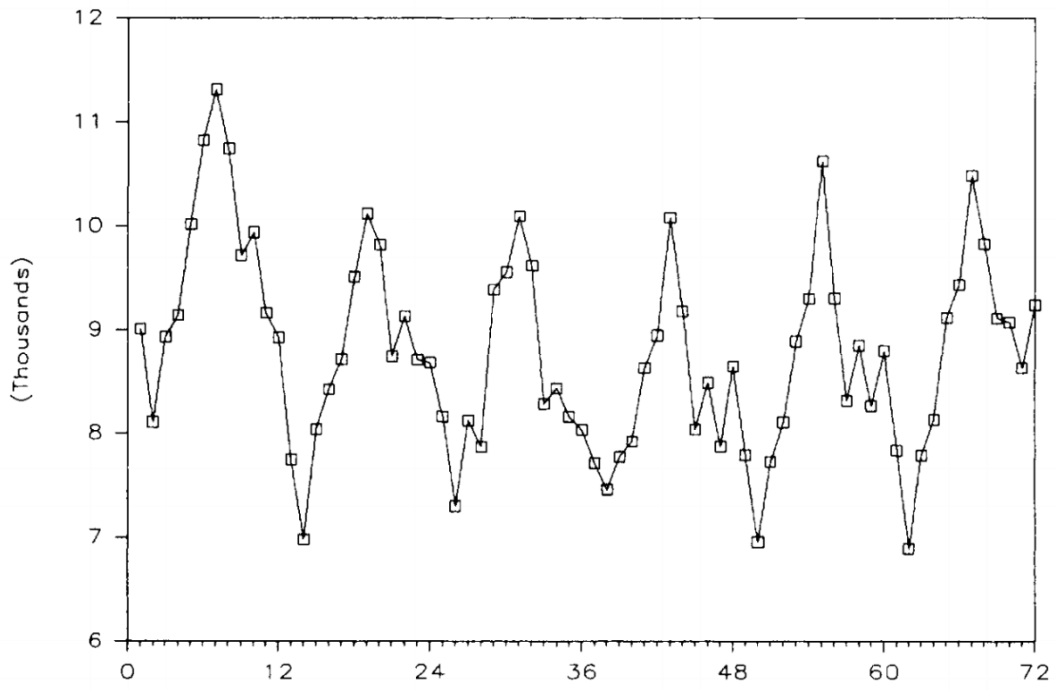
4. EMBASAMENTO TEÓRICO

Neste capítulo serão apresentados os fundamentos teóricos que deram base aos estudos desenvolvidos neste trabalho, serão abordados conceitos de series temporais, redes neurais, ruídos e subamostragem de sinais. Esses conceitos são abordados com o objetivo de desenvolver o conhecimento necessário para o autor atingir os objetivos do trabalho e auxiliar o leitor a entender as decisões tomadas ao longo do desenvolvimento e as interpretações apresentadas nos resultados e conclusões.

4.1. Séries temporais

Uma série temporal é um conjunto de observações x_t onde cada observação está associada a um tempo específico t (BROCKWELL e DAVIS, 2006). Existem séries temporais analógicas e discretas. Uma série temporal discreta é aquela onde as observações x_t são realizadas em um intervalo de tempo definido e constante t_d. Neste trabalho, os dados numéricos trabalhados seguirão o conceito de séries discretas. A Figura 1 apresenta o exemplo de uma série temporal discreta onde pode-se observar o número mensal de óbitos por causas acidentais nos Estados Unidos, entre os anos de 1973 e 1978.

Figura 1 - Óbitos por causas acidentais nos Estados Unidos entre 1973 e 1978.



Fonte: BROCKWELL E DAVIS, 2006.

As séries temporais podem ser representadas como um somatório de três funções fundamentais, de acordo com a Equação 1. Essa representação é denominada modelo de decomposição clássica de uma série temporal genérica (BROCKWELL e DAVIS, 2006).

$$X_t = m_t + s_t + Y_t \tag{1}$$

onde m_t é a função conhecida como tendência, s_t é a função conhecida como sazonalidade e Y_t é a função que representa a presença de ruído na composição da série.

Existem alguns métodos para o tratamento da Equação 1. Em geral, o objetivo é minimização ou eliminação da tendência, da sazonalidade e do ruído, pois o primeiro passo na análise de qualquer série temporal, quando se trata de um modelo estatístico, deve ser a adequação da série ao modelo utilizado (BROCKWELL e DAVIS, 2006).

4.2. Ruídos

Na prática, a utilização de séries temporais se torna uma tarefa difícil quando apresenta uma grande influência do ruído em seus valores. Espera-se que para obter um bom resultado nesse tipo de análise a influência desse efeito deve ser reduzido. Os dados provenientes de um processo sempre irão conter algum ruído devido a influência de muitos fatores complexos e aleatórios (SANG *et al.*, 2009).

Em processamento de sinais, o ruído branco é geralmente associado aos equipamentos do processo, os *hardwares*, e de erros no processo de medição. Esse tipo de ruído está presente em diferentes frequências e costuma conter pouca ou nenhuma informação proveniente do processo em que se deseja analisar (MONTILLET *et al*, 2013). O ruído colorido é uma junção de diversos ruídos de naturezas diferentes, como o ruído branco, o ruído de *flicker* (ruído rosa) e o ruído browniano (ruído marrom ou *random walk noise*). Essa diferença entre as cores dos ruídos está atrelada às faixas de frequência do espectro de potência dos sinais de ruído, sendo a cor uma analogia ao espectro de frequência da luz.

4.3. Redes Neurais Artificiais

Uma rede neural artificial é um processador paralelo distribuído robusto, composto de unidade menores de processamento chamados neurônios que apresentam a propriedade de guardar o conhecimento experimental, aprendendo-o e tornando-o disponível para uso (HAYKIN, 2008). As redes neurais vêm sendo utilizadas desde a década passada em sistemas de predição de séries temporais (WEIGEND e GERSHENFELD, 1994).

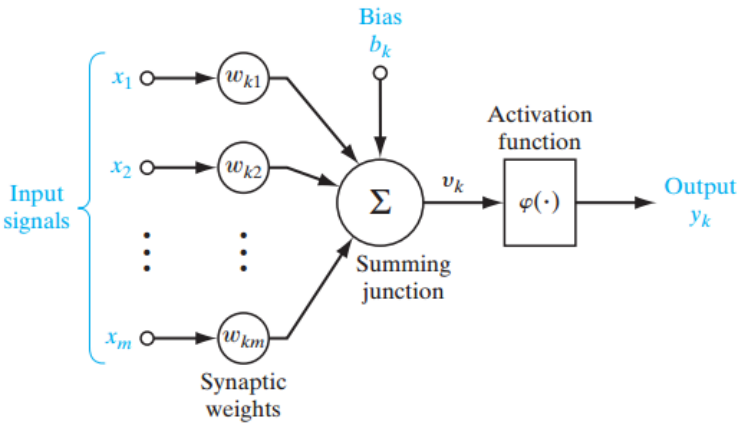
O objetivo principal de uma ANN é modelar, inspirado de forma simplória, a maneira com que o cérebro realiza as tarefas a ele atribuídas. O cérebro humano, por exemplo, tem um poder de processamento superior aos computadores que existem atualmente, apesar desses computadores apresentarem um grande poder de processamento. A apenas uma década atrás o pequeno cérebro de um morcego era capaz de realizar um processo complexo, como a eco localização, a uma taxa de sucesso superior aos radares e sonares produzidos pela engenharia (HAYKIN, 2008).

4.3.1. O Neurônio

Biologicamente, a unidade fundamental do cérebro é denominada neurônio. O neurônio tem a função de processar a informação decorrente das sinapses elétricas geradas pela atividade muscular e entregar a resposta adequada ao sistema nervoso central. Isso é realizado através do balanceamento entre as diferentes entradas das diversas sinapses existentes, onde os sinais mais importantes contribuem mais significativamente para o resultado. Analogicamente, a literatura apresenta um padrão de definição parecido para as ANN, onde a unidade fundamental de processamento de informações também é denominada neurônio (MOREIRA, 2013).

O neurônio de uma ANN é modelado de acordo com a Figura 2, onde observa-se os sinais de entrada (i*nput signals)* representando a entrada de valores referentes às sinapses cerebrais. Cada entrada apresenta um determinado peso sináptico w_{k_n} (*synaptic weights*) para a definição dos sinais mais importantes. Um bloco somador (s*umming juction)* é responsável por unir as diferentes sinapses com seus respectivos pesos. Essa união de elementos é denominada de neurônio artificial (HAYKIN, 2008).

Figura 2 – Modelo de um neurônio



Fonte: SIMON O. HAYKIN, 2008.

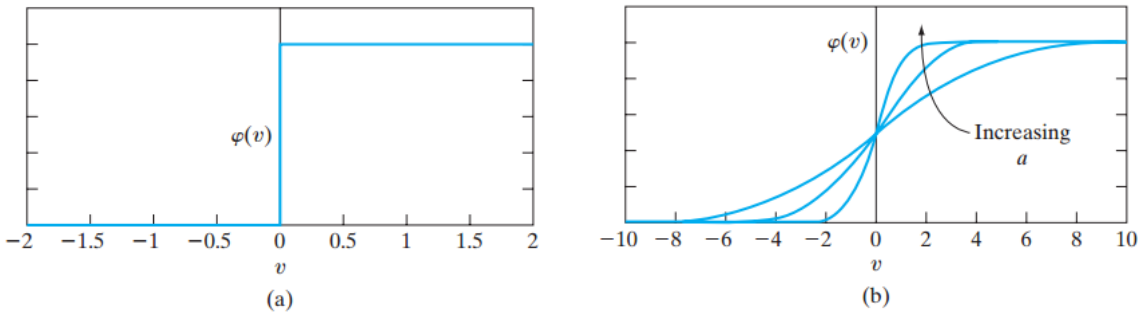
O modelo da Figura 2 apresenta ainda o sinal de *Bias*, que é responsável por sensibilizar mais ou menos a saída do neurônio. O sinal de *Bias* é importante pois em um neurônio, independentemente do valor dos pesos w_{k_n}, se os sinais de entrada apresentarem inicialmente o valor zero a saída sempre resultará em um valor nulo. Há também a presença de uma função de ativação (*activation function*), responsável por dizer como será a natureza da ativação do sinal de saída do neurônio. Matematicamente, pode-se modelar o neurônio conforme a Equação 2.

$$y_k = \varphi\left(\sum_{i=0}^{m} w_{ki}x_i + b_k\right) \tag{2}$$

onde φ representa a função de ativação, w_{ki} o peso sinaptico, x_i o valor da entrada e b_k o sinal de bias.

A função de ativação pode ser discreta, como ocorre com a função degrau unitário (*threshold*), ou suave, como ocorre na função *Sigmoid*. A Figura 3 apresenta o comportamento dessas duas funções de ativação.

Figura 3 – (a) Função *Threshold*. (b) Função *Sigmoid* de acordo com o parâmetro *a*



Fonte: SIMON O. HAYKIN, 2008.

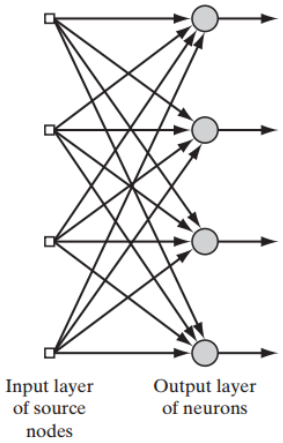
Para a ativação *sigmoid*, o valor de saída é limitado entre 0 e 1, recebendo valores próximos de 0 quando v_k apresenta valores negativos, e próximo de 1 quando v_k apresenta valores positivos. Esse comportamento é interessante quando o objeto de estudo é um problema de classificação de dados ou de *clustering*, onde a saída 0 pode ser interpretada como o resultado do reconhecimento da entrada como pertencendo ao Grupo A e a saída 1 como pertencente ao Grupo B.

Apesar de apresentar um comportamento parecido com a função *sigmoid*, uma das maiores desvantagens da função *threshold* é seu comportamento abrupto próximo à origem, gerando problemas computacionais no cálculo de derivadas em métodos como, por exemplo, o *Stochastic Gradient Descent* (SGD). Na prática, a função Tangente Hiperbólica (Tanh) pode ser utilizada para gerar um comportamento análogo à *sigmoid*, mas no caso desta função os valores limites variam entre -1 e 1. Existem ainda outras funções de ativação comumente utilizadas como a *Rectified Linear Unit* (ReLU) e a *Exponencial Linear Unit* (ELU), a escolha da função de ativação adequada depende da natureza do problema.

4.3.2. Arquiteturas de redes *feedforward*

Em uma rede neural os neurônios são organizados em forma de camadas. A forma mais simples de organização é a representada na Figura 4, onde existe uma camada de entrada (*input layer*) ligada diretamente à camada de saída (*output layer*). Esse modelo é conhecido como cama única (*Single-Layer)*.

Figura 4 – Arquitetura *Single-Layer*

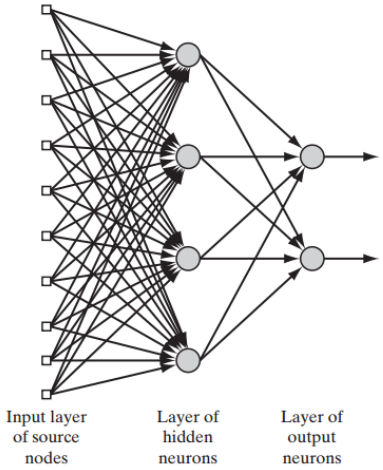


Fonte: SIMON O. HAYKIN, 2008.

Entretanto, o modelo da Figura 4, apesar de sua simplicidade e utilidade, não consegue obter uma grande interpretação estatística dos valores de entrada, como é possível com um modelo um pouco mais complexo: o modelo múltipla-camada (*multilayer)*. O modelo *multilayer* é mostrado na Figura 5 e apresenta a adição de uma camada denominada camada oculta (*hidden-layer*). Essa camada localizada entre a entrada e a saída do modelo aumenta a complexidade da rede simulando a existência de várias sinapses, assim como ocorre no cérebro humano, e incrementa a possibilidade de uma interpretação mais global aos dados da camada de entrada (CHURCHLAND e SEJNOWSKI, 1992).

Além disso, o modelo m*ultilayer* é dito totalmente conectado *(fully conected*), o que significa que todos os neurônios da camada anterior estão conectados com os neuronios da próxima camada. O modelo *multilayer* pode conter uma ou mais *hidden-layers*, sendo que o aumento do número de *hidden-layers* torna o sistema mais complexo. Porém, conforme o sistema se torna mais complexo é necessário um número maior de dados para ajustar os pesos de todos os neurônios da rede – ação conhecida como treinamento da rede neural – e conferir a ela a capacidade de generalização necessária. Como cada nó do sistema é uma unidade *perceptron*, esse modelo pode ser ainda denominado Arquitetura *multilayer perceptron*.

Figura 5 – Arquitetura *Multilayer* Perceptron

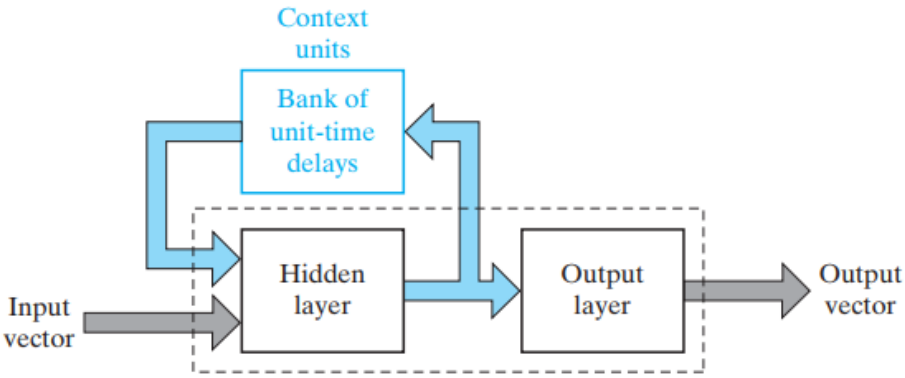


Fonte: SIMON O. HAYKIN, 2008.

4.4. redes neurais recorrentes

As arquiteturas de redes neurais podem apresentar diversas configurações, uma dessas configurações é denominada rede neural recorrente, ou *Recurrent Neural Netwok* (RNN). Essa rede implementa a arquitetura *multilayer perceptron*, herdando a sua capacidade de mapeamento não linear e possibilitando previsões em contextos mais complexos que não são abrangidos pela estatística convencional. Um modelo conceitual de uma RNN é representado na Figura 6, nele é possível notar que há como entradas da camada oculta um vetor de dados para treinamento e também uma realimentação de informações provenientes da rede neural.

Figura 6 – Esquemático de uma RNN

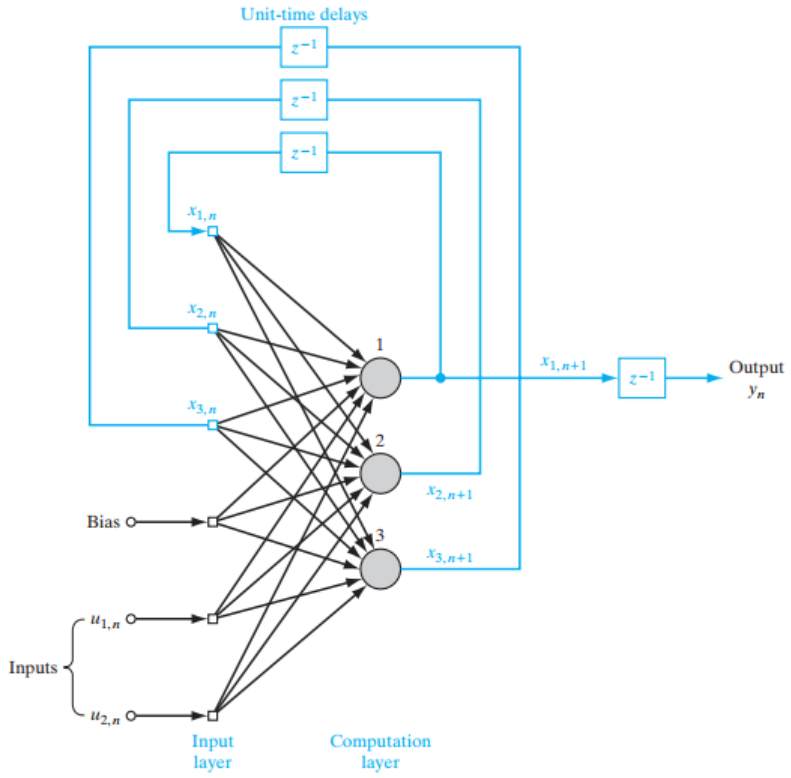


Fonte: SIMON O. HAYKIN, 2008.

A realimentação de informações da rede recorrente ocorre com um número determinado de eventos, esse número de eventos é denominado *delay* ou atraso. O *delay* é um dos hiperparâmetros de treinamento de redes neurais recorrentes.

Um olhar mais minucioso sobre as entradas de uma RNN é apresentado na Figura 7, nela pode-se observar a entrada de dados $u_{m,n}$ e a realimentação proveniente da saída da *multlayer perceptron* $x_{m,n}$ atuando como entradas defasadas no tempo, cujos atrasos são denotado pela transformada Z^{-1}.

Figura 7 – Conexões de uma RNN



Fonte: SIMON O. HAYKIN, 2008.

4.5. Treinamento e predição

As redes neurais estão inclusas no universo de estudo conhecido como aprendizado de máquina (*machine learning)* e trabalhar a questão do *forecasting* a luz desse ponto se mostra uma opção interessante na busca de um meio de representar e sumarizar o conhecimento humano. Esse pensamento é importante, pois no processo de treinamento e predição de uma rede neural é necessário ensinar a rede a habilidade de tomar decisões de acordo com a análise da série temporal histórica.

A ação de ensinar a rede neural é denominada treinamento que, tecnicamente, de acordo com o apresentado na seção 4.3.2, consiste na adaptação dos pesos dos neurônios das camadas do modelo neural. De um ponto de vista mais abrangente, o treinamento consiste em prover ao sistema um conjunto de informações que busquem abordar o maior número de possibilidades para a solução do problema em questão. Essa forma de ter o conhecimento como informação é denominada informação prévia e consiste na série temporal histórica de acontecimentos sobre os quais se quer aprender. Em geral, a informação prévia vem naturalmente de um processo em funcionamento e será utilizada para prever o comportamento futuro desse processo.

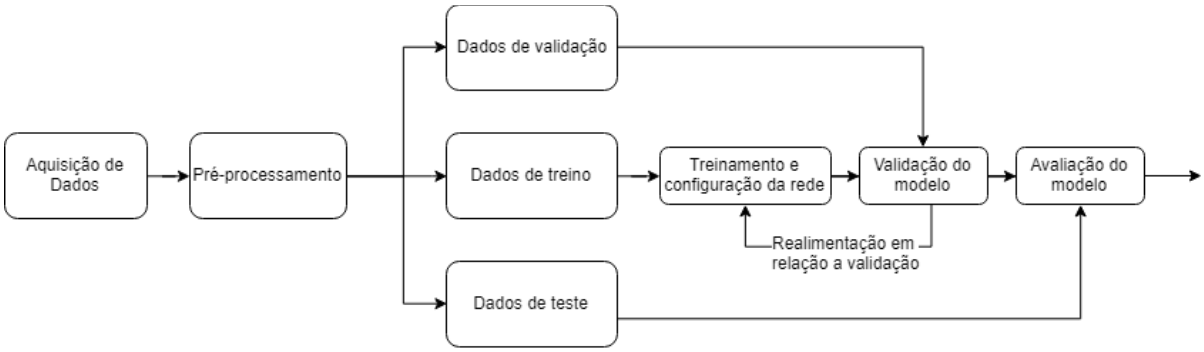
Também existem informações utilizadas para treinamento que são geradas objetivamente para a realização do treinamento, em geral é necessário assim quando o processo é novo ou não existe a serie história de dados. Esses dados são gerados abordando as diversas situações que podem ocorrer no processo, quando o problema em questão é bem-comportado, como o problema de estacionar um caminhão em uma área delimitada.

Após a coleta, os dados passam por uma fase de pré-processamento, onde são tratados para se adequarem a entrada da rede neural ou remover alguma característica indesejada. Essa é uma etapa primordial e grande parte da qualidade do resultado final depende de um bom pré-processamento. As informações prévias são então divididas em um conjunto designado para o treinamento da rede (*training data),* um conjunto para verificar se o modelo de treinamento está obtendo a resposta esperada (*validation data*) e um terceiro conjunto que é responsável por avaliar o modelo final do processo de treinamento (*test data*). A fase de teste busca conferir à rede a habilidades de generalização, verificando se a predição de alguns valores dentro da série histórica se aproxima do esperado. Quando a rede não consegue obter essa capacidade, significa

que a rede neural decorou o mapeamento realizado entre os dados de entrada e de saída. Esse problema é denominado *overfitting*, esse efeito pode ser observado na Figura 9.

O resultado do treinamento é uma rede neural treinada com a capacidade de prever os valores futuros de uma série de acordo com os valores anteriores. Essa divisão pode ser observada no esquemático apresentado na Figura 8.

Figura 8 – Divisão dos dados no processo de treinamento



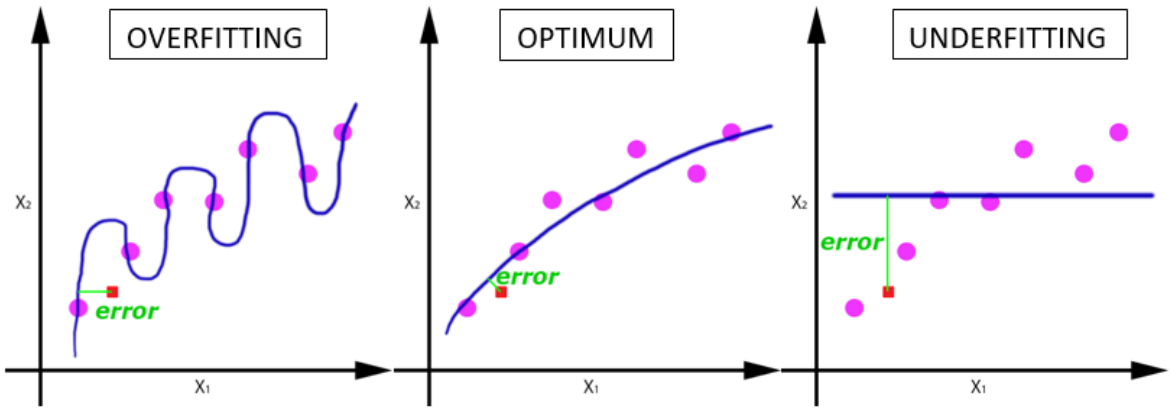
Fonte: Autoria própria

4.5.1. Métodos de treino

O treinamento de uma rede neural pode ser dividido em dois grupos. O primeiro grupo é denominado de *Epochwise Training*, nesse grupo, a rede recorrente usa um par de dados de entrada e saída e realiza o treinamento a partir de um estado inicial aleatório, prossegue com o treinamento até passar todos os dados de treinamento pela rede e uma nova época começa com outros dados de entrada e saída e um novo estado inicial (HAYKIN, 2008). Após um número definido de épocas, tem-se o final do treinamento.

Essa metodologia é utilizada, pois trabalha-se geralmente com uma quantidade de dados que não é suficiente para abordar todos os aspectos do problema real, então utiliza-se várias épocas de treinamento para realizar um ajuste fino dos pesos sinápticos e evitar o problema do *underfitting*. Os efeitos do *overfitting* e do *underfitting* podem que pode ser observado na Figura 9.

Figura 9 – Overfitting e Underfitting



Fonte: SAGAR SHARMA, 2017.

No segundo grupo tem-se o *Continuous Training*, onde a rede é treinada continuamente sem que ocorra o reinício do processo de treinamento e o controle de épocas. Essa situação é desejada quando tem-se treinamento em tempo real, a rede aprende enquanto realizada o processamento do sinal (HAYKIN, 2008).

4.5.2. *Back Propagation Through Time*

O algoritmo denominado *Back Propagation Through Time* (BPTT) é uma variação de um tradicional algoritmo, o *Back Propagation* (BP). O *Back Propagation Through Time* é uma ferramenta muito poderosa, com aplicações em reconhecimento de padrões, modelagem dinâmica, análise de sensibilidade e controle de sistema através do tempo, entre outras (WERBOS, 1990).

O BPTT pode ser utilizado nos dois métodos de treinamento citados na seção 4.5.1, o *Epochwise Training* e o *Continuos Training*. Quando esse algoritmo é aplicado ao *Epochwise Training*, passa a denominar-se *Epochwise Back Propagation Throught Time* (EBPTT) (HAYKIN, 2008).

O conjunto de dados inicial é particionado em um conjunto de dados independentes, então uma função de custo é atribuída para traçar a relação entre o valor de saída da rede treinada e o valor de saída esperado, ou *target*. O processo de treinamento consiste em calcular a sensibilidade da

rede o que significa calcular as derivadas parciais da função de custo (HAYKIN, 2008). Simplificando o apresentado em (WILLIAM e PENG, 1990), pode-se enunciar o EBPTT como:

1. Uma passagem para frente (*forward)* dos dados através da rede é executada. O registro completo dos dados de entrada, o estado da rede (isto é, os pesos sinápticos da rede) e as respostas desejadas durante esse intervalo são salvos.
2. É executado um passo para trás (*backward*) para calcular os valores dos gradientes locais. Esse cálculo se repete passo a passo para o número de amostras da época.
3. Uma vez que o cálculo da propagação reversa foi executado um ajuste é aplicado aos pesos sinápticos.

4.6. Redes NAR, NARX e NIO

Os métodos estatísticos convencionais como o *Autoregressive Conditional Heteroskedasticity* (ARCH), o *Autoregressive Integrated Moving Average* (ARIMA), o Box-Jenkins ou o *Smooth Transition Autoregressive* (STAR) são usados para predizer séries temporais, mas os processos envolvidos nessas técnicas não lidam adequadamente com o ruído e com a não linearidade dos dados (LAHMIRI, 2011). Um sistema ou modelo é considerado linear se todas as funções objetivas e restrições do sistema forem representadas por equações lineares. Caso contrário, é considerado como sendo um modelo não linear (LJUNG e GLAD, 1994).

A *Nonlinear Autoregressive Network with Exogenous Inputs* (NARX), a *Nonlinear Autoregressive* (NAR), e a *Nonlinear Input-Output* (NIO) são redes neurais empregadas no *forecasting*. As RNN dinâmicas, NAR, NARX e NIO podem ser úteis em diversas situações, mas têm méritos e deméritos distintos dependendo do tipo de aplicação na qual é utilizada.

4.6.1. Rede NAR

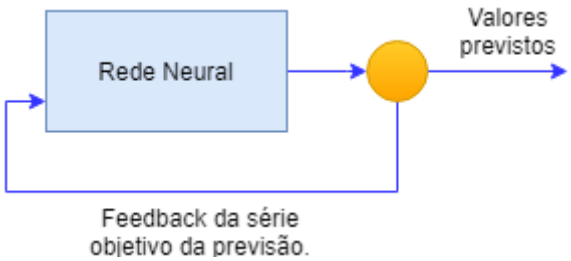
A rede *Nonlinear Autoregressive* (NAR) consiste na ideia simples do que foi apresentado até o momento em relação ao *forecasting*, ou seja, utilizar os valores passados de uma série temporal para prever o seu valor futuro. Nesse modelo, os valores da série já conhecidos são realimentados (*feedback*) na rede neural, que é capaz então de prever os valores futuros.

Matematicamente a rede NAR pode ser modelada conforme a Equação (3) e esquematizada conforme a Figura 10.

$$y(\mathrm{t}) = f(y(t-1), \dots, y(t-\mathrm{n})) \tag{3}$$

Onde os termos $y(t-1), \dots, y(t-n)$ representam a contribuição dos valores passados da série que é objetivo da predição (y).

Figura 10 – Esquemático de uma rede NAR



Fonte: Autoria própria

4.6.2. Rede NARX

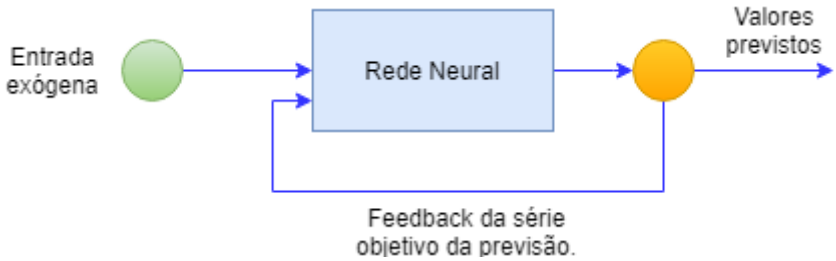
A rede *Nonlinear Autoregressive Network with Exogenous Inputs* (NARX) é uma rede dinâmica recorrente, com conexões de *feedback* abrangendo várias camadas da rede (Caswell, 2014). Na prática, a rede NARX é capaz de prever uma série temporal tendo-se os valores passados da própria série e os dados de uma ou várias outras séries, conhecidas como séries exógenas. Essa série exógena tem influência na série que se espera prever. Por exemplo, ao tentar prever a próxima falha de um equipamento elétrico, além do histórico de falhas deste equipamento outras informações como o histórico de manutenções preventivas e troca de óleo podem auxiliar na predição.

Matematicamente, a rede NARX pode ser modelada conforme a Equação (4) e esquematizada conforme a Figura 11.

$$y(\mathrm{t}) = f(x(t-1), \dots, x(t-n), y(t-1), \dots, y(t-n)) \quad (4)$$

onde os termos $y(t-1), \dots, y(t-n)$ representam a contribuição dos valores passados da série que é objetivo da predição (y) e os termos $x(t-1), \dots, x(t-n)$ representam a contribuição dos valores exógenos.

Figura 11 - Esquemático de uma rede NARX



Fonte: Autoria própria

4.6.3. Rede NIO

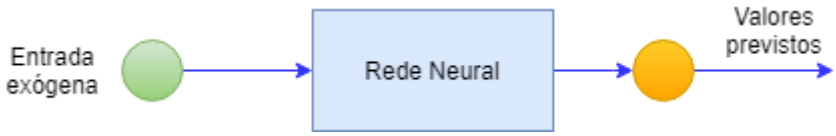
A rede *Nonlinear Input- Output* (NIO), diferente do que se espera e do que ocorre na NAR e na NARX, não utiliza os valores passados da série objetivo da predição para realizar o *forecasting*. Esse comportamento a priori pode parecer estranho, mas esse modelo é extremamente útil quando o processo está no início da aquisição de dados, ou seja, não existe uma série histórica. Então, para realizar o *forecasting* são utilizados valores de outras séries temporais que têm ligação com a série temporal objetivo da predição. Assim como para a rede NARX, essa série é denominada série exógena.

Matematicamente, a rede NARX pode ser modelada conforme a Equação (5) e esquematizada conforme a Figura 12.

$$y(t) = f(x(t-1), \dots, x(t-n)) \quad (5)$$

onde os termos $x(t-1), \dots, x(t-n)$ representam a contribuição das variáveis exógenas.

Figura 12 – Esquemático de uma rede NIO



Fonte: Autoria própria

4.7. Taxa de amostragem

Quando se deseja processar digitalmente um sinal, existe a necessidade de realizar a conversão de valores reais analógicos para valores digitais discretos. Esse procedimento é denominado amostragem do sinal.

O procedimento de amostragem, de maneira geral, consiste na captação de valores temporalmente espaçados da série analógica proveniente do processo de onde estão sendo captados os dados. Esse espaçamento contínuo de aquisição dos valores é denominado taxa de amostragem w_s e a escolha correta desse parâmetro influência na posterior reconstrução adequada do sinal.

Por exemplo, considere um sinal senoidal bem comportado, se a amostragem for realizada com um intervalo de tempo equivalente ao período da senoide, obtém-se unicamente a informação capaz de recuperar o comportamento de uma constante, pois não será possível ter a percepção da forma original do sinal principalmente porque conforme Haykin (2001):

> "*As amostras não nos dizem nada a respeito do comportamento do sinal entre os tempos de amostras*".

A taxa mínima a ser adotada para garantir a recomposição adequada do sinal é denominada taxa de amostragem de Nyquist. Nyquist diz que a frequência de amostragem do sinal deve ser no mínimo duas vezes maior do que a maior frequência medida da amostra, ou seja, $w_s > 2w_m$. Na prática, espera-se utilizar uma taxa de amostragem superior ao mínimo estabelecido por Nyquist, quando esse mínimo não é alcançado os dados são ditos subamostrados.

4.8. Métricas de desempenho

No contexto de previsões de séries temporais, as métricas de desempenho são indicadores que buscam quantificar a divergência entre o valor previsto e o valor real, ou seja, quantificar o erro entre os dados previstos e os dados separados para teste (*test data)*. Existem diversas métricas de desempenho, como a Média Absoluta do Erro (*Mean Absolute Error - MAE*), a Média Quadrática do Erro (*Mean Squared Error - MSE*), a R-quadrado (*R Squared* ou *R^2*), *a* Raiz Quadrática do Erro Médio (*Root Mean Squared Error* - RMSE) e o Erro Percentual Absoluto Médio (*Mean Absolute Percentage Error* - MAPE).

Devido a sua simplicidade, interpretação física clara, boa representatividade para problemas de otimização e desejáveis aplicações para estatísticas e estimativas (WANG E BOVIK, 2009), será utilizada a métrica MSE. A métricas MSE é representada matematicamente, segundo (WANG E BOVIK, 2009), pela Equação (6).

$$\mathrm{MSE}(x, y) = \frac{1}{N}\sum_{i=1}^{N}(x_i - y_i)^2 \tag{6}$$

Onde N representa o número de amostras, x_i representa as amostras da série prevista e y_i representa as amostras do conjunto de teste (*test data*).

5. METODOLOGIA E ETAPAS DE DESENVOLVIMENTO

Para atender aos objetivos propostos, foi realizado um conjunto de experimentos para o levantamento de conclusões em relação a influência da taxa de amostragem e do ruído na predição de séries temporais. Optou-se pelo uso do *software* Matlab 2017 © que dispõe de métodos bem definidos e uma interface para o tratamento de redes neurais clássicas, como o NNTOOL.

O experimento consistiu em efetuar a predição utilizando:

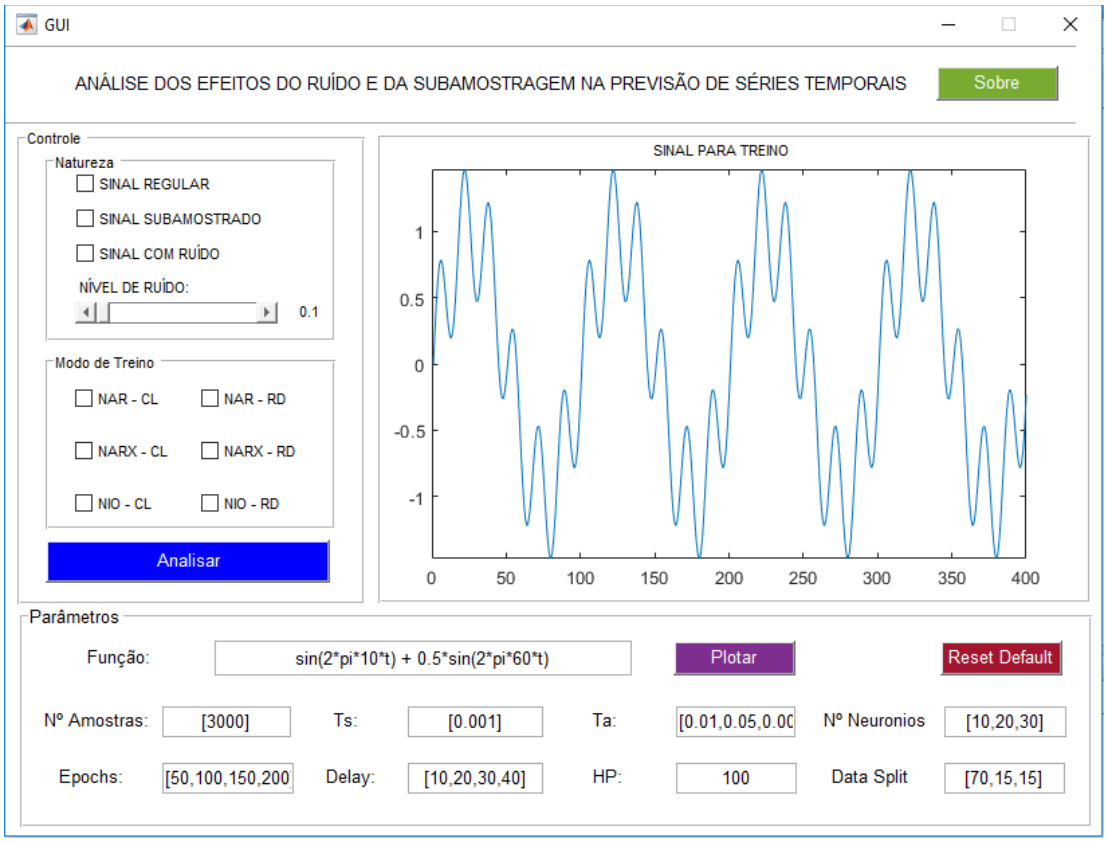
- **Série 1:** Uma série temporal constituída por equacionamento matemático, com características bem controladas e corretas, sem a presença de ruído e subamostragem;
- **Série 2:** Uma série temporal constituída por equacionamento matemático, com características bem controladas e corretas, com a presença de ruído gaussiano branco;
- **Série 3:** Uma série temporal constituída por equacionamento matemático, com características bem controladas e corretas, porém, subamostrada.

Para as três séries supracitadas, além da avaliação dos efeitos da introdução do ruído gaussiano branco e da subamostragem, buscou-se enriquecer a análise aplicando as arquiteturas de rede NAR, NARX e NIO para cada uma das situações, trazendo a possibilidade de verificar se uma das situações propostas se adequa melhor para uma arquitetura do que para outra.

5.1. A Interface

Devido à grande quantidade de situações possíveis e da necessidade de ajuste dos hiperparâmetros para cada situação e função proposta, optou-se pela construção de uma interface gráfica que busca possibilitar ao usuário configurar de maneira facilitada a variabilidade desejada dos hiperparametos. A interface busca prover, além dos hiperparâmetros, a possibilidade de o usuário ajustar tanto a porcentagem de ruído gaussiano branco que será aplicado à serie quanto a taxa de amostragem. A interface desenvolvida é apresentada na Figura 13.

Figura 13 – Interface Desenvolvida



Fonte: Autoria própria

A interface apresenta dois menus de controle, um menu de parâmetros (que contém parâmetros e hiperparâmetros) e um gráfico onde é plotado o sinal que será utilizado no experimento. O primeiro menu é o denominado Natureza, neste menu é possível selecionar como será a natureza do sinal entre as opções REGULAR, SUBAMOSTRADO e RUÍDO e também inserir através do *slider* denominado Nível de Ruído a intensidade de ruído gaussiano branco que incidirá sobre o sinal. Após essa escolha, no menu Modo de Treino existe a possibilidade de selecionar a arquitetura da rede entre as opções NAR/CL, NAR/RD, NARX/CL, NARX/RD, NIO/CL e NIO/CD, onde os as siglas CL e RD referem-se ao método prático utilizado pelo Matlab para realizar a realimentação, tornando a rede neural recorrente, sendo *Closed Loop* (CL) ou *Remove Delay* (RD). Mais detalhes sobre essa implementação serão abordados nas seções seguintes.

O menu Parâmetros apresenta hiperparâmetros e parâmetros para o treinamento das redes escolhidas no Controle**,** além disso, há um campo para a escolha da função que será utilizada no experimento. Note que a função utilizada pode ser plotada através do botão Plotar sendo então visualizada na janela de plotagem.

Resumidamente, os parâmetros disponíveis na interface são:

- Função – Equacionamento matemático que resultará na série a ser utilizada;
- Nº Amostras – Número de amostras da função que será utilizada;
- *Epochs* – Número de épocas que serão utilizadas no treinamento;
- Ts – Taxa de amostragem;
- *Delay* – Hiperparametro *Delay*, utilizado na camada de entrada da rede para o treinamento;
- HP – Número de amostras que se espera prever (HP igual a 100 indica que serão previstos 100 valores);
- Nº Neurônios – Número de neurônios utilizado no treinamento;
- *Data Split* – Conjunto de três parâmetros que indica a separação dos dados, em porcentagem, em treino teste e validação.

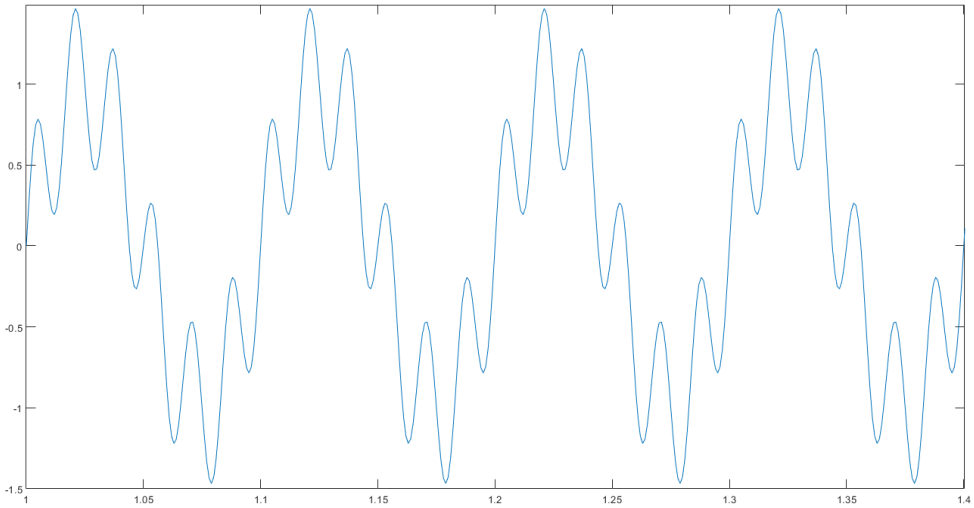
Todos os campos com a presença dos colchetes indicam a possibilidade de o usuário informar um vetor de valores para o parâmetro em questão. No caso do parâmetro Nº Neurônio, há, no exemplo da Figura 13, a entrada [10,20,30] indicando a passagem de um vetor com os valores 10, 20 e 30 o que significa que o algoritmo vai executar o treinamento para 10 neurônios, 20 neurônios e 30 neurônios na camada oculta. Essa variabilidade dos hiperparâmetros será utilizada para realizar uma série de treinamentos onde ocorre uma combinação de todos os hiperparâmetros. Essa variabilidade tem o objetivo de garantir que a análise dos efeitos do ruído e da subamostragem sejam feitos para casos otimizados de predição e com os hiperparâmetros adequados para cada caso.

5.2. A função utilizada

A função determinada para a realização dos experimentos está expressa na Equação 7 e consiste em uma soma de senoides no tempo. O vetor tempo é por padrão amostrado a uma taxa (ts) igual a 0,001 Hz. O gráfico gerado por essa função pode ser melhor analisado na Figura 14.

$$y(t) = \sin(2\pi * 10t) + 0.5 * \sin(2\pi * 60t) \tag{7}$$

Figura 14 – Função utilizada como base para os experimentos



Fonte: Autoria própria

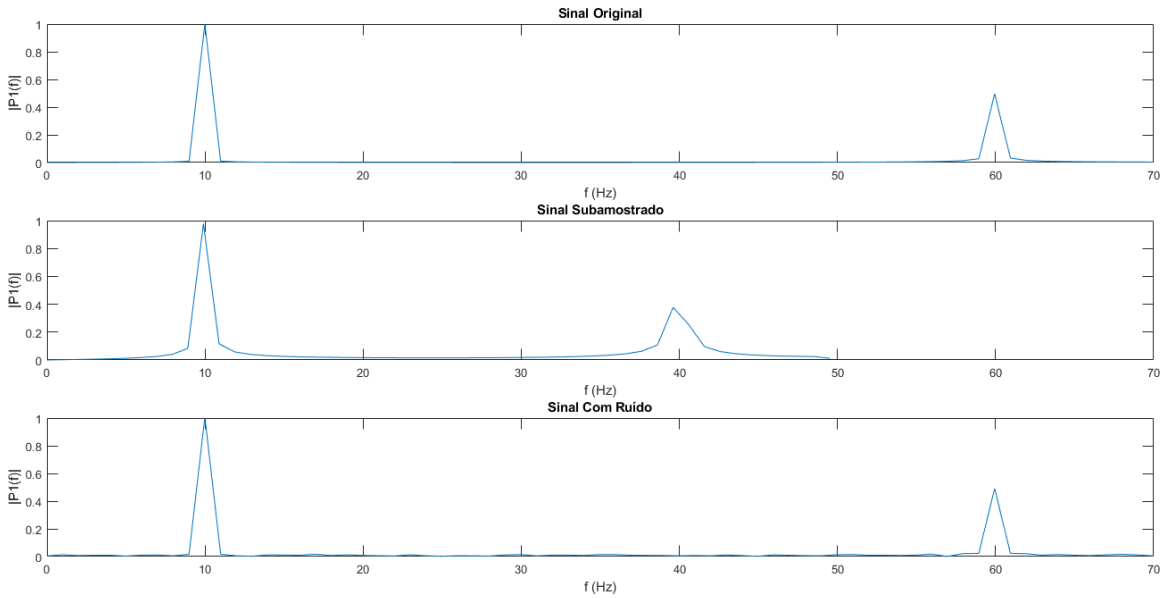
5.3. Adição do ruído e da subamostragem

Para ser avaliada de acordo com o escopo do trabalho, a equação 7 sofreu duas alterações, a adição de ruído e a subamostragem. Para a subamostragem, bastou a adaptação do vetor de tempo (t) da Equação 7 de modo que a frequência de amostragem ficasse n vezes menor. Já para o caso do ruído, a expressão utilizada no *Matlab* se encontrada na Equação 8, onde $y(t)$ é dado pela Equação 7, *randn*() é uma função nativa do Matlab que gera um número aleatório normalmente distribuído, *numel*() retorna o tamanho do vetor y e a simbologia ".*" indica uma multiplicação matricial ponto a ponto. O escalar 0,20 é responsável por dizer a intensidade do ruído.

$$y'(t) = y(t) + 0.20 * randn\big(1, numel(y)\big).* y(t) \tag{8}$$

Com o equacionamento definido para os três casos, gerou-se a Transformada de Fourier para entender melhor a natureza do sinal das Equações 7 e 8 e da subamostragem realizada na Equação 7, para exemplificação, optou-se por uma taxa de amostragem de 0,01 Hz (Não obedecendo a taxa mínima de Nyquist – 0,0083 Hz). O resultado pode ser observado na Figura 15.

Figura 15 – Transformada de Fourier (FFT) para os casos de estudo



Fonte: Autoria própria

Observa-se na Figura 15 que a subamostragem interfere na localização das componentes frequenciais do sinal, principalmente nas componetes de maior frequência. Nota-se que a componente de frequência mais baixa (10 Hz) não sofreu grandes alterações, pois a subamostragem realizada não foi suficiente para interferir nessa componente, o que pode ocorrer com taxas de amostragem inferiores. Isso indica um perigo, pois pode-se tirar conclusões precipitadas em relação a aquisição de dados se não for conhecida a Transformada de Fourier do estado original do sinal, pois olhando apenas para a componente de 10 Hz, não se pode dizer a princípio se o sinal está subamostrado, com a presença de ruídos ou em seu estado original.

5.4. Variabilidade de hiperparâmetros e arquiteturas

Idealmente, quanto maior for o *range* dos vetores de hiperparâmetros, maior será a possibilidade de o algoritmo encontrar resultados otimizados, porém também será maior o tempo de treinamento devido ao alto custo computacional. Ao optar-se por 4 valores de cada hiperparâmetros, o problema a ser resolvido resulta em 256 combinações de hiperparâmetros, em um teste hipotético onde deseja-se avaliar as opções Regular, Subamostrado e Ruído e as 6 opções de arquitetura (NAR/CL, NAR/RD, NARX/CL, NARX/RD, NIO/CL e NIO/CD), tem-se 4608 combinações.

Definiu-se os valores presentes na Tabela 1 como os valores padrão para a realização dos experimentos para as situações propostas, pois esses valores apresentaram soluções com baixos valores de MSE para o equacionamento matemático proposto. Para uma função diferente, recomenda-se que sejam realizados treinos prévios com menos opções de hiperparâmetros de maneira a otimizar os *ranges* escolhidos para a realização do experimento.

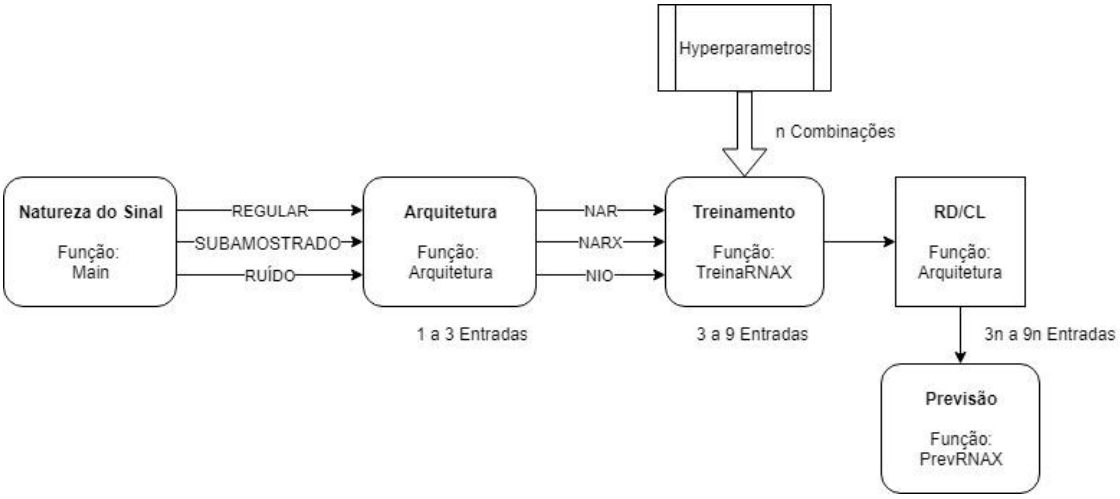
Tabela 1 – *Range* de hiperparâmetros utilizados

Hiperparametro	**Valores**
Número de Épocas	50, 100, 150, 200
Delay	10, 20, 30, 40
Learning Rate	0.01, 0.05, 0.001, 0.005
Número de Neurônios	10, 20, 30

Fonte: Autoria própria

Após a coleta de informações pela interface, ocorre uma fase de interpretação dos controles selecionados pelo usuário, essa etapa tem como finalidade realizar apenas os treinamentos e previsões escolhidas pelo usuário. Conforme descrito, essa é uma questão importante uma vez que o custo computacional na utilização de redes neurais é alto, e a realização todos os testes de uma única vez pode resultar em muitas combinações. A Figura 16 apresenta o diagrama de controle lógico do algoritmo desenvolvido. Pode-se observar em cada etapa a função no código responsável pelo gerenciamento do controle e a quantidade de entradas possíveis em cada etapa do processo.

Figura 16 – Diagrama de controle lógico e entradas



Fonte: Autoria própria

5.5. Treinamento

Com os controles bem definidos, inicia-se o processo de treinamento. Nessa etapa são configurados na rede (*net*) os hiperparâmetros definidos pelo usuário na interface. A Figura 17 apresenta a sobrescrita de alguns valores *default* do Matlab como o número de épocas e a métrica de desempenho.

Figura 17 – Definição de Hiperparametros

```
% ajusta parâmetros da rede
net.TrainParam.epochs = Epocas;
net.divideFcn = 'divideblock';
net.divideParam.trainRatio = 70/100;
net.divideParam.valRatio = 15/100;
net.divideParam.testRatio =15/100;
net.trainParam.max_fail = 0.15*Epocas; % Early Stop
net.trainParam.goal = 0;
net.trainParam.min_grad = 0;
net.trainParam.lr = LearningRate;
net.divideMode = 'time';  % Divide up every sample
net.performFcn = 'mse';  % Mean Squared Error
```

Fonte: Autoria própria

Após o treinamento da rede, pode-se realizar a predição utilizando o método *Closed Loop* (CL) ou método *Remove Delay* (RD). Para o uso do método *Closed Loop* foi necessário a utilização da função closeloop() ao passo que, para realizar o método *Remove Delay* bastou um reajuste do vetor para a remoção do *Delay*. Essa situação pode ser observada na Figura 18.

Figura 18 – Predição RD ou CL.

```
if strcmp(TP,'RD')   % Remove Delay
    net = removedelay(net,HP);
    ys = net(x,xi,ai);
    MP = [M(:,1:net.numInputDelays+HP) cell2mat(ys)];

else    % Close Loop
    [y1,xfo,afo] = net(x,xi,ai);
    [netc,xic,aic] = closeloop(net,xfo,afo);
    [y2,~,~] = netc(cell(0,HP),xic,aic);
    MP = [M(:,1:net.numInputDelays) cell2mat(y1) cell2mat(y2)];
```

Fonte: Autoria própria

Na Figure 18 é possível observar que a implementação Remove Delay, consiste na utilização do comando do Matlab *removedelay*() que é responsável por remover o atraso da rede, o resultado desse comando é um rede idêntica à anterior, mas com os eventos acontecendo sem o

efeito do atraso. Já a implementação do Close Loop ocorre de maneira similar, mas nesse caso tem-se a adição de y_1 proveniente da rede e y_2 da função *netc*(), responsável por realizar uma predição de malha fechada e múltiplos passos a partir de condições iniciais.

Após a realização da predição foi implementado uma avaliação quantitativa do desempenho da rede através do uso da métrica MSE. Essa avaliação tem como objetivo identificar, entre as diversas opções geradas, qual apresenta um resultado superior sobre o ponto de vista dessa métrica de desempenho. São gerados também em conjunto com a métrica um gráfico de desempenho, um de Autocorrelação do Erro e um de Regressão Linear para cada entrada do sistema de predição (3n a 9n, onde n é o número de combinações dos hiperparâmetros).

5.6. Avaliação do resultado

Para cada natureza de sinal, é avaliado o desempenho da rede treinada seguindo a métrica do MSE. Para evitar que sejam salvos e gerados um número excessivo de resultados, apenas os cinco melhores resultados para cada um dos três casos (Regular, Subamostrado e Ruído) são guardados, os casos guardados apresentam o resultado independentemente do tipo de arquitetura utilizada, então os cinco melhores resultados podem ser todos da mesma arquitetura se essa apresentar um resultado superior em termos de MSE do que as outras.

Além do algorótimo salvar os gráficos gerados, ele também salva, para cada teste, o *workspace.* O *workspace*, para o Matlab, é o conjunto de variáveis alocadas na memória em um determinado momento, ou seja, salvar o *workspace* significa salvar o valor das variáveis na memória no disco rígido, de maneira que esses valores podem ser novamente carregados na memória posteriormente. Essa funcionalidade foi implementada para o caso em que seja necessário verificar as condições em que o resultado foi gerado ou extrair alguma informação adicional. A função desenvolvida chamada Avaliação é responsável por essa verificação e pode ser visualizada na Figura 19.

Figura 19 – Função Avaliação

```
function [MSE,saveWork,ii,vecMSE] = Avaliacao(net,x,t,vecMSE)
ii=1;
y= net(x);
saveWork = 0; % Bool para determinar se o workspace será salvo.
MSE =(perform(net, t, y)); % Calcula MSE da rede

% Ordenação descendente para sempre substituir o maior valor.
sort(vecMSE,'descend')

% Verifica se está entre os 5 melhores
while (ii<=numel(vecMSE))
if(MSE<vecMSE(ii))
    vecMSE(ii)=MSE;
    saveWork = 1;
    break
end
ii = ii+1
```

Fonte: Autoria própria

Além do gráfico principal que será gerado contendo a curva do valor real e dos valores previstos alguns outros gráficos serão necessários para a realização de uma interpretação com maior acurácia dos resultados encontrados. Esses gráficos estão disponíveis no Matlab, são eles: Autocorrelação do Erro, Regressão Linear e Desempenho. Além de poderem ser manualmente gerados pela interface, podem ser obtidos através do código da Figura 20.

Figura 20 – Geração dos gráficos auxiliares.

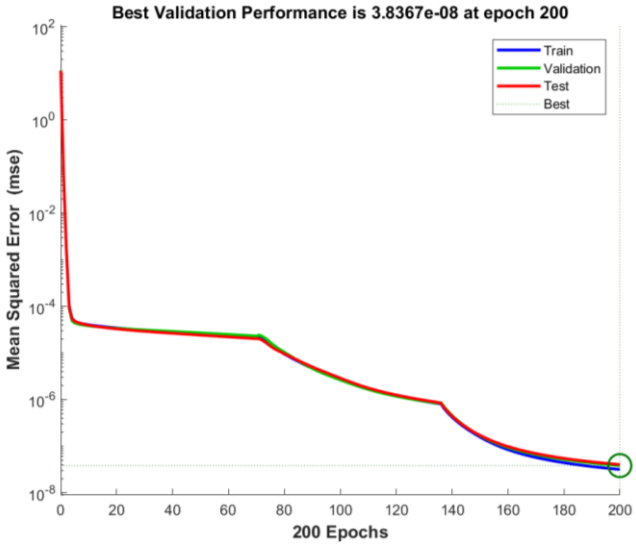
```
function []=pgraphs(net,tr,x,xi,ai,t,Nio,strsave)
y= net(x);
perf = num2str(perform(net, t, y));

% Desempenho
plotperform(tr);

% Autocorrelação
Y = net(x,xi,ai);
E = gsubtract(t,Y);
ploterrcorr(E);

% Regressao
plotregression(t,y,'Regression')

end
```

Fonte: Autoria própria

A Figura 21 traz o gráfico de Desempenho gerado pelo código da Figura 20. O gráfico apresenta o comportamento do treinamento em relação aos *datasets* (*Train Data*, *Validation Data* e *Test Data)*. Em seu eixo vertical exibe a métrica de desempenho escolhida (MSE), e em seu eixo horizontal o número de épocas determinadas para o treinamento. O valor mínimo do eixo horizontal representa o menor MSE encontrado no treinamento, para o caso do exemplo da Figura 21, pode-se dizer que o treino obteve o seu melhor resultado na época 200 de um total de 200 épocas. Entretanto, esse comportamento não pode ser tomado como regra, ao longo do trabalho será possível observar alguns casos em que o menor valor encontrado ocorre em uma época anterior ao número máximo de épocas.

É possível afirmar ainda, para esse caso, que caso o número de épocas fosse maior do que 200 provavelmente ocorreria a obtenção de um resultado com um valor menor de MSE devido a característica descendente da curva de desempenho, apesar de que um valor menor de MSE não significa necessariamente um melhor resultado de predição, devido a problemas como o *overfitting*.

Figura 21 – Gráfico de Desempenho

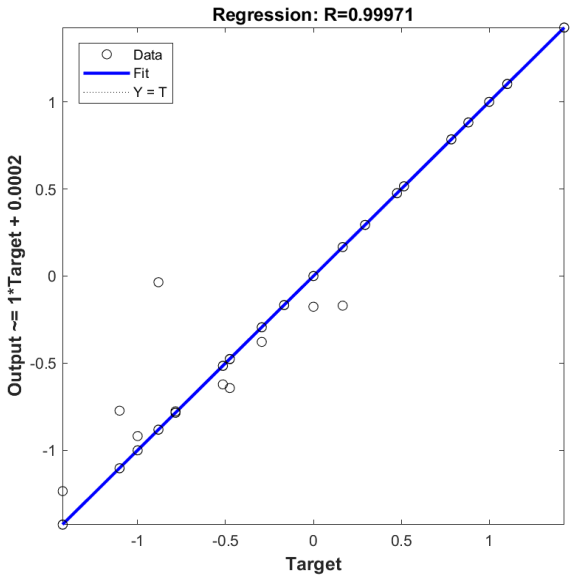


Fonte: Autoria própria

A Figura 22 apresenta o gráfico de Regressão Linear do treinamento. Para a regressão do exemplo, grande parte dos valores se encontram sobre a reta, indicando que a maioria dos valores apresentou um resultado próximo ao esperado. Note que quanto mais próximo de R =1, melhor o resultado obtido.

O ponto de atenção nesse gráfico é que nem sempre a obtenção de valores bem mapeados de entrada e saída, que ocorre quando a maioria dos valores estão sobre a reta, representa um bom resultado. Essa situação de bom mapeamento pode ocorrer no caso de a rede neural ter decorado (*overffitting*), para tanto, a análise do gráfico de Autocorrelação em conjunto com o gráfico de Regressão enriquece a análise e permite tirar conclusões mais precisas.

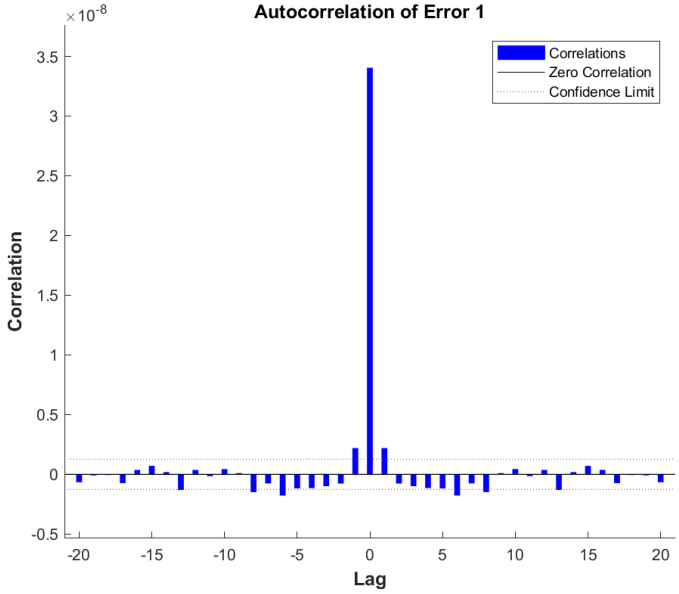
Figura 22 – Gráfico de Regressão Linear



Fonte: Autoria própria

A Figura 23 apresenta o gráfico de Autocorrelação do erro, esse gráfico mostra a correlação do treinamento com o atraso (*Lag*) e indica se a rede neural aprendeu o comportamento esperado. A linha pontilhada vermelha apresenta o limite de confiança da autocorrelação, o Matlab, por padrão, trata o intervalor de confiança como os valores que estão dentro do intervalo de dois desvios padrões (2σ) o que significa 95,45% de confiança se for considerado que os dados seguem uma distribuição normal. A autocorreção para atraso zero é igual ao valor do MSE e apresenta a autocorreção do valor predito com ele mesmo, por isso o seu valor é alto comparado ao normal para os outros atrasos.

Figura 23 – Gráfico de Autocorrelação do erro



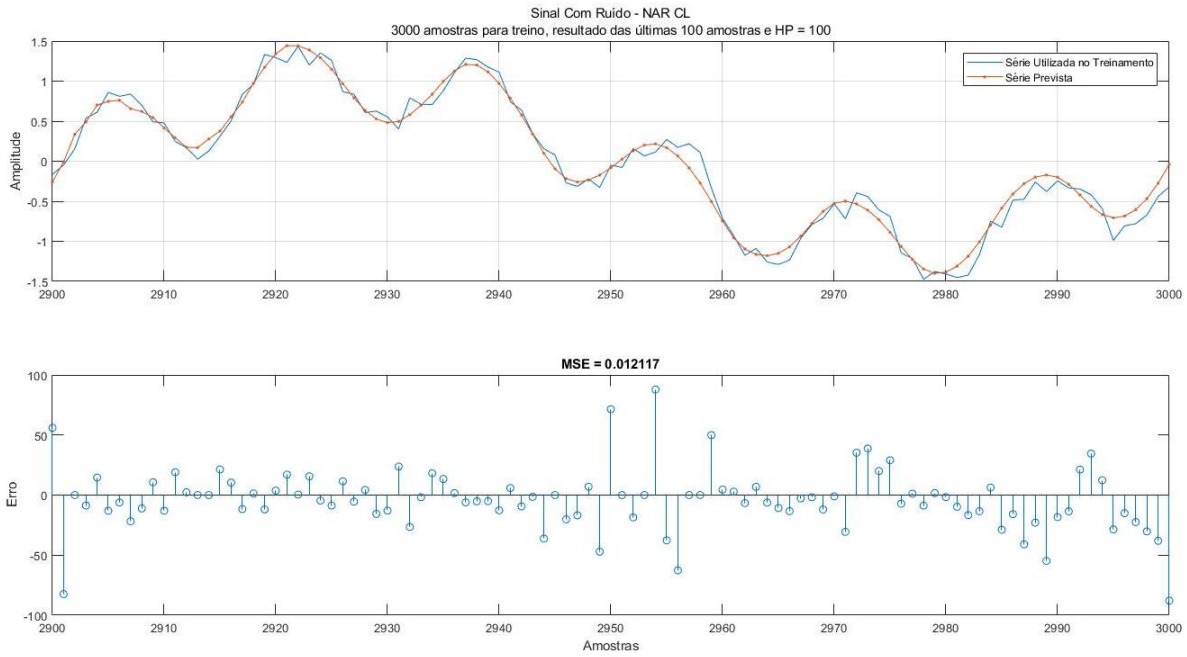
Fonte: Autoria própria

6. RESULTADOS E DISCUSSÃO

Com os resultados gerados de acordo com a metodologia apresentada e os cinco melhores resultados para a métrica do MSE devidamente selecionados é possível analisar alguns casos gerados. Vale ressaltar que a análise realizada é específica para o equacionamento matemático (Equação 7 apresentada na seção 5.2), entretanto a metodologia de análise e os problemas decorrentes dos efeitos podem ser generalizados para outras funções similares. Contudo, espera-se que a interface desenvolvida seja utilizada para analisar individualmente cada função para obtenção de resultados mais fidedignos a realidade do usuário.

O gráfico padrão para o demonstrativo dos resultados encontrados e para a análise das previsões é apresentado na Figura 24. Na parte superior do gráfico, observa-se a série prevista em vermelho, em comparação com a série original em azul. Na parte inferior, observa-se o erro percentual relativo a cada amostra, também é possível observar o MSE do resultado, para o caso da Figura 24, o valor 0.012. O título do gráfico apresenta a descrição do experimento e a arquitetura utilizada, no caso do exemplo, *Sinal Com Ruído* e arquitetura NAR CL.

Figura 24 – Exemplo de resultado



Fonte: Autoria própria

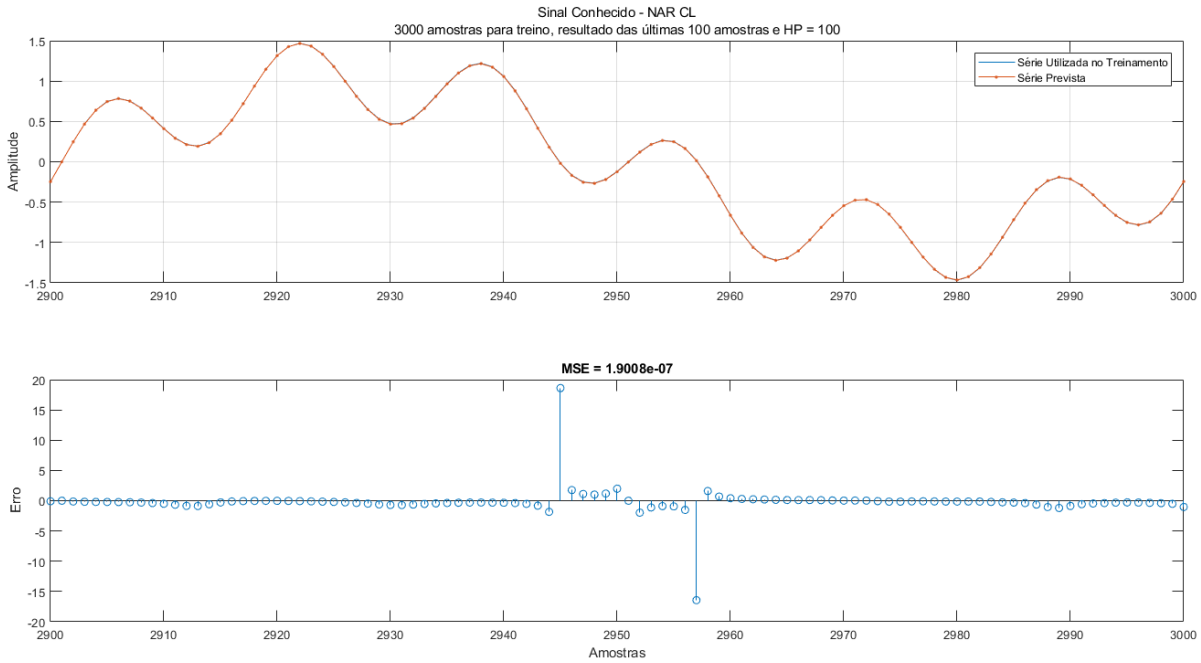
6.1. Melhores resultados encontrados

Essa seção apresenta os melhores resultados numéricos encontrados para um dos testes experimentais realizados. Para cada caso, será apresentado o melhor resultado do ponto de vista da métrica escolhida. O objetivo principal dessa seção é tratar algumas situações que podem surgir, enriquecendo o conhecimento do usuário da interface em relação a análise dos resultados para que ele possa utilizar a interface e interpretar as situações em relação aos efeitos do ruído e da subamostragem.

6.1.1. Melhor resultado para Sinal Conhecido.

O primeiro experimento tratou do Sinal Conhecido, o resultado pode ser observado na Figura 25. Observa-se um MSE muito baixo ($1.9e^{-7}$), o que significa que os valores encontrados aproximam-se significativamente dos valores originais. Essa questão também pode ser notada quando compara-se a série prevista com a treinada, onde observa-se visualmente uma diferença praticamente nula. Isso indica que a predição atendeu as expectativas do ponto de vista do erro e que a metodologia utilizada para encontrar uma boa solução apresenta um resultado consistente quando o objetivo é diminuir o erro de treinamento.

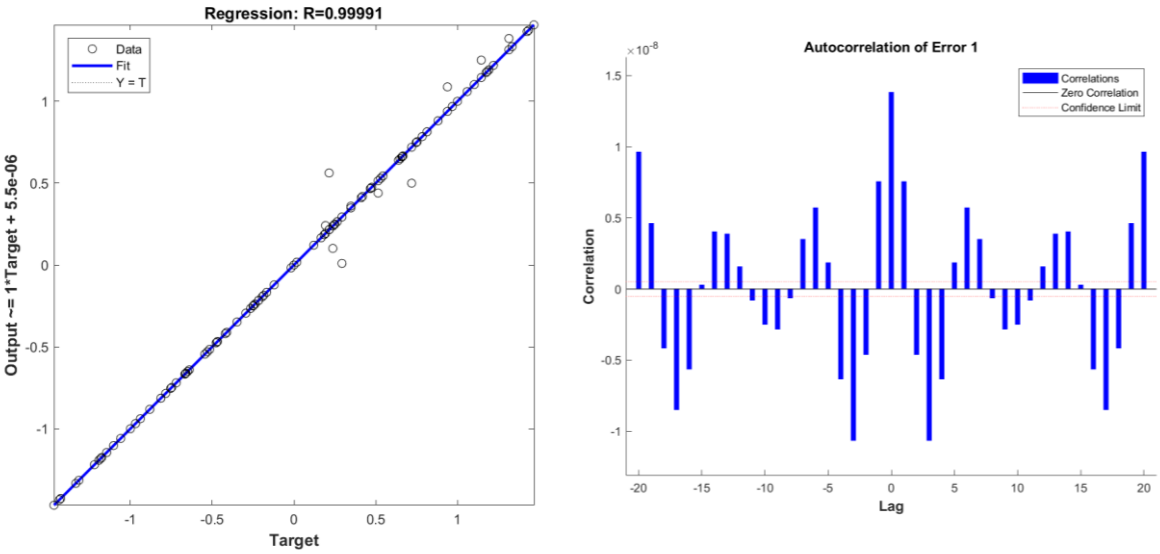
Figura 25 – Resultado com menor MSE para sinal conhecido



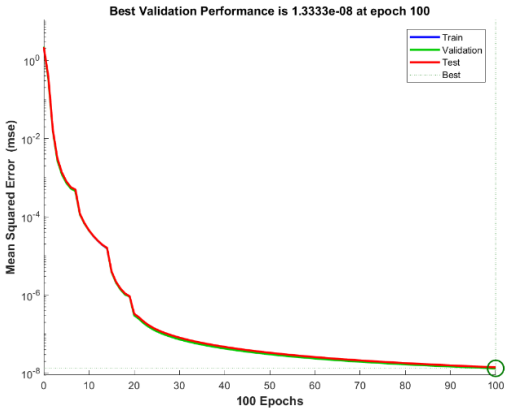
Fonte: Autoria própria

Quando analisa-se a regressão presente na Figura 26 (a), observa-se que os valores apresentam uma aderência considerável a reta de Regressão Linear, isso significa que o treinamento obteve uma boa relação, ou mapeamento, entre a entrada e a saída. Também observa-se que o melhor resultado foi obtido na época 100, sendo que grande parte do avanço ocorreu nas 20 primeiras épocas. Entretanto, o gráfico da autocorrelação do erro indica que a o resultado com menor MSE obtido não apresentou características de aprendizado. Por esses pontos e pelas características da na Figura 26 (c) é possível concluir que o treinamento sofreu *overfitting*.

Figura 26 – Gráficos auxiliares para o melhor resultado sinal conhecido.



(a) (b)



(c)

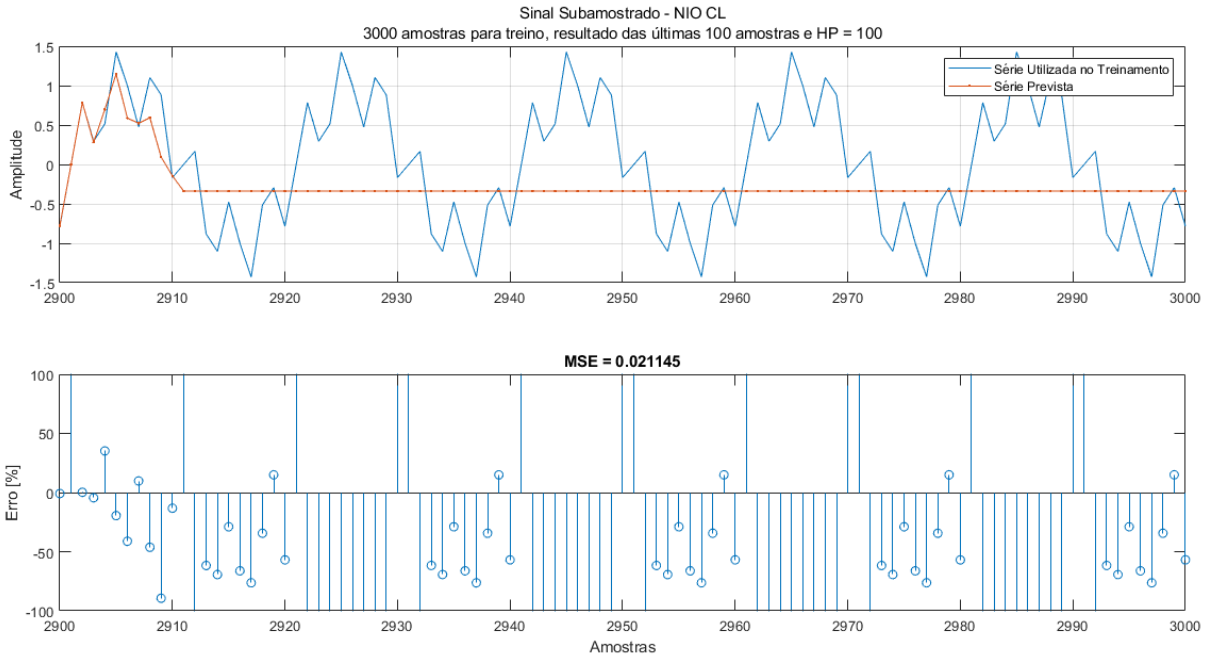
Fonte: Autoria própria

6.1.2. resultado para sinal subamostrado

Para o segundo caso, analisando o sinal com o efeito da subamostragem, utilizou-se no experimento uma taxa de amostragem de 0.005 Hz, o que significa uma amostragem com o tempo cinco vezes maior do que o utilizado para os casos anteriores. O resultado é apresentado na Figura 27 e nota-se que a predição obteve resultados expressivos para as 10 primeiras previsões, e após isso não conseguiu acompanhar a curva original.

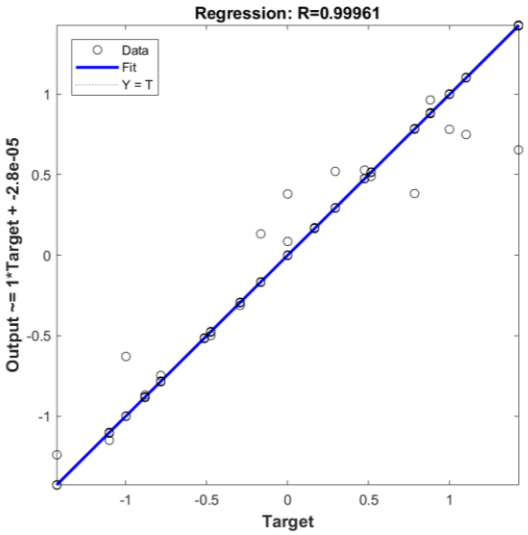
Observa-se ainda que, apesar das regiões de alta energia, a predição não apresentou uma suavização da curva que a princípio poderia ocorrer pela dificuldade de redes neurais preverem variações muito bruscas no comportamento de séries temporais. Nesse caso, analisa-se que a subamostragem apresentou um resultado distante do esperado, o que pode ser agravado se considerarmos que a característica que espera-se prever é o da função em seu estado natural, presente na Figura 22. Apesar disso, os gráficos auxiliares encontrados na Figura 28 indicam um mapeamento consistente e um aprendizado aceitável, todavia, o resultado nos mostra que a RNN não é capaz de entender que o sinal foi subamostrado.

Figura 27 – Resultado com menor MSE para sinal subamostrado

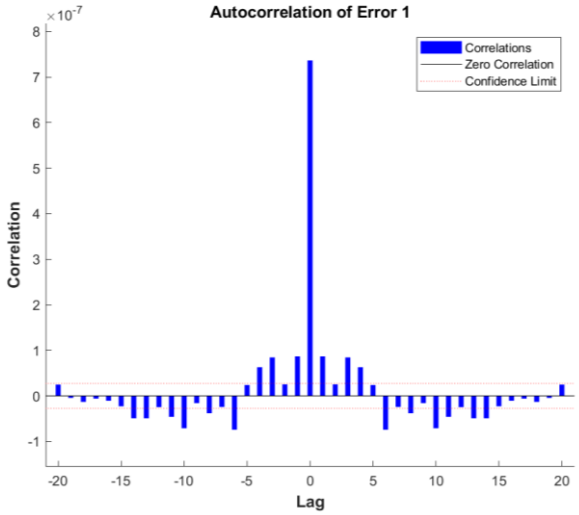


Fonte: Autoria própria

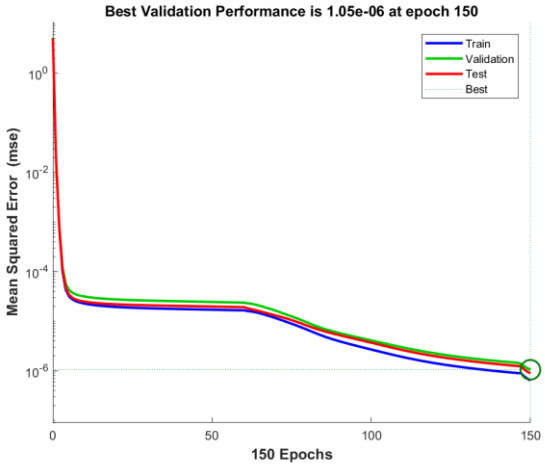
Figura 28 – Gráficos auxiliares para o melhor resultado sinal subamostrado.



(a)



(b)



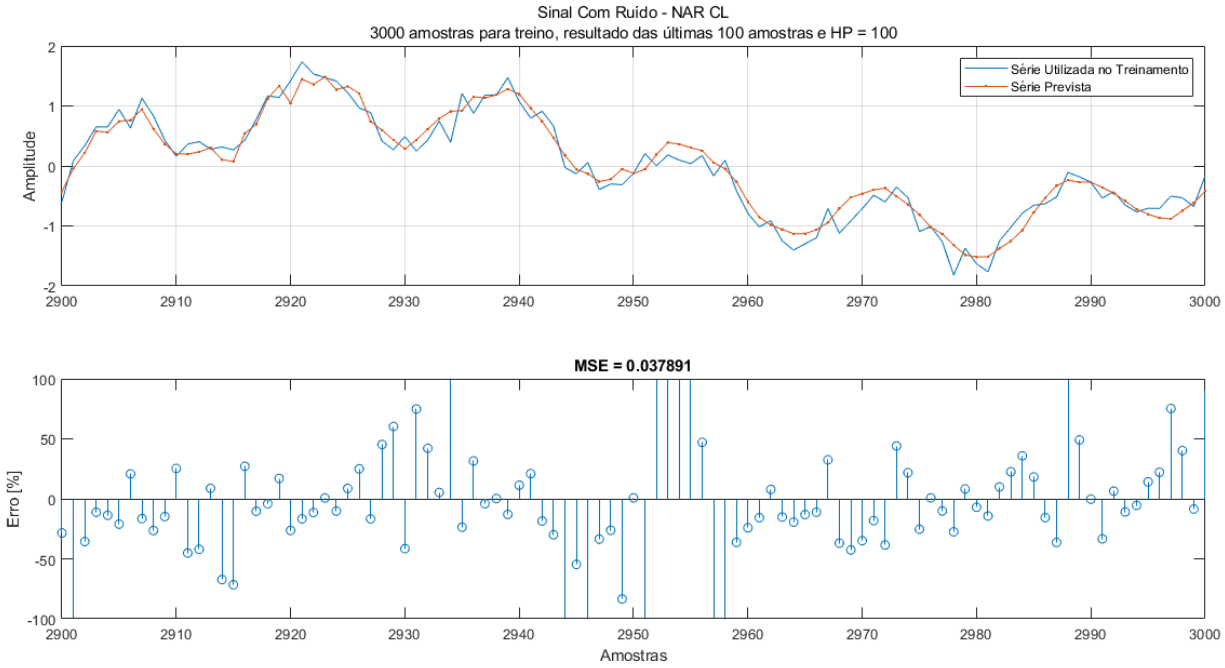
(c)

Fonte: Autoria própria

6.1.3. Melhor resultado para Sinal com Ruído

O melhor resultado para o Sinal com Ruído é apresentado na Figura 29. Observa-se que a curva prevista aproxima-se da série utilizada no treinamento, de maneira geral, ela acompanha a curva original. Apesar disso, nota-se que predição não entende as variações bruscas geradas pelo ruído introduzido, que consiste em 20% em torno do valor do sinal original, e isso acaba tornando a predição próxima da curvatura da série original, sem o efeito do ruído. Essa é uma característica interessante visto que em geral o objetivo esperado é o de prever a série sem o efeito do ruído. Fato explicado pela natureza do neurônio de ser um filtro FIR (*Finite Impulse Response*) sintonizado em torno da portadora do sinal de ruído branco.

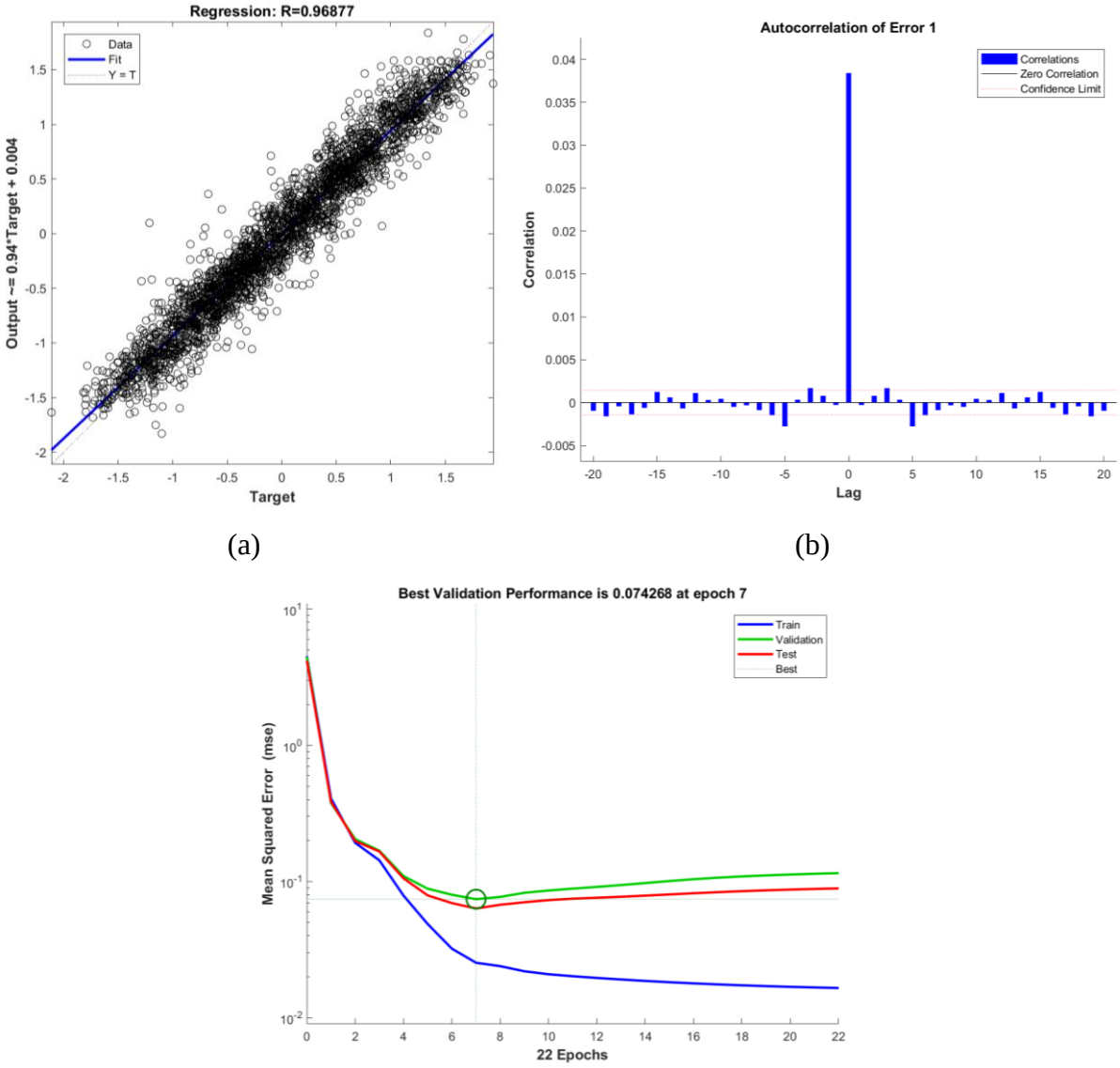
Figura 29 – Resultado com menor MSE para sinal com ruído.



Fonte: Autoria própria

Analisando os gráficos auxiliares, observa-se um mapeamento aderente a tendência linear do gráfico de regressão linear Figura 30(a) entretanto, pode-se destacar que não ocorreu um mapeamento preciso, pois observam-se diversos valores fora da reta de regressão. Essa característica está de acordo com a Figura 31. Apesar disso, a Figura 31(b) confirma que a serie teve um bom aprendizado, pois a maioria dos valores se encontram dentro do intervalo de confiança. Essa constatação é corroborada pela Figura 31(c) que apresenta o *early stop* do treinamento, pois a validação estava começando a divergir e o treinamento possivelmente estava tendendo ao *overfitting*.

Figura 30 – Gráficos auxiliares para o melhor resultado com ruído.



Fonte: Autoria própria

6.2. Resultados por arquitetura

Para ter-se um valor quantitativo dos resultados obtidos por tipo de arquiteturas de redes neural implementada, foi realizada uma contagem de quantas vezes determinada arquitetura esteve presente entre os cinco melhores resultados de cada teste, ou seja, para cada teste realizado, contabilizou-se se uma determinada arquitetura esteve presente entre os cinco melhores resultados. Para isso foram realizados cinco testes completos (com todas as arquiteturas na configuração CL e RD e com o sinal subamostrado e com ruído). Os testes seguiram os parâmetros apresentados na Tabela 1. Os resultados estão apresentados na Tabela 2.

Tabela 2 – Quantitativo dos cinco melhores resultados por arquitetura

	Ruído [%]	Subamostragem [%]	Ruído [Quantidade]	Subamostragem [Quantidade]
NAR CL	52,0%	32,0%	13	8
NAR RD	0,0%	0,0%	0	0
NARX CL	0,0%	0,0%	0	0
NARX CD	0,0%	0,0%	0	0
NIO CL	24,0%	44,0%	6	11
NIO RD	24,0%	24,0%	6	6

Fonte: Autoria própria

O objetivo principal da Tabela 2 é analisarmos se alguma arquitetura se adaptou melhor a um tipo de problema em relação ao outro. Observa-se que a NAR CL obteve melhor resultado para o problema com o ruído e o NIO CL obteve o melhor resultado para a subamostragem. Já as configurações NAR RD, NARX CL e NARX RD não apresentaram nenhum resultado entre os cinco melhores em nenhum dos cinco testes utilizados. Vale notar que, de acordo com a Figura 16, ao todo são gerados 576 resultados para cada teste para cada problema, tendo ao todo 2880 resultados por problema (ruído e subamostragem), sendo assim os 25 melhores resultados encontrados representam apenas 0,87% dos resultados.

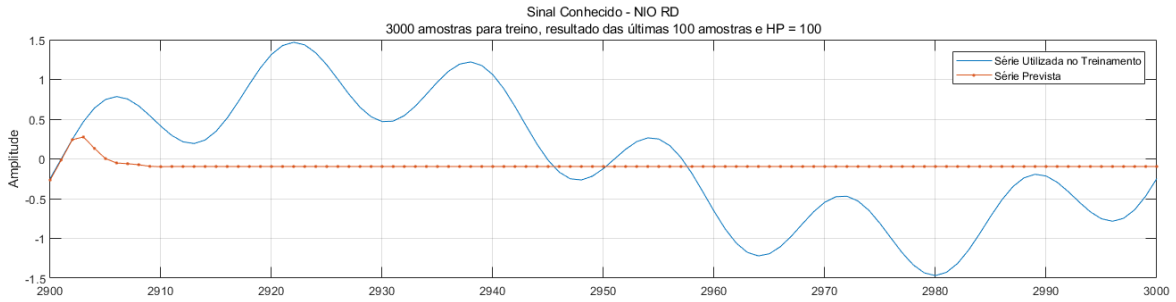
6.3. NIO

Um caso interessante é o que ocorreu com a arquitetura NIO, em diversos testes realizados, a arquitetura NIO apresentou pouca capacidade de prever valores em um grande horizonte de predição. Observa-se na Figura 31(a) que os primeiros três valores foram praticamente sobre a linha original, o quarto valor se aproximou consideravelmente e após isso o resultado se tornou

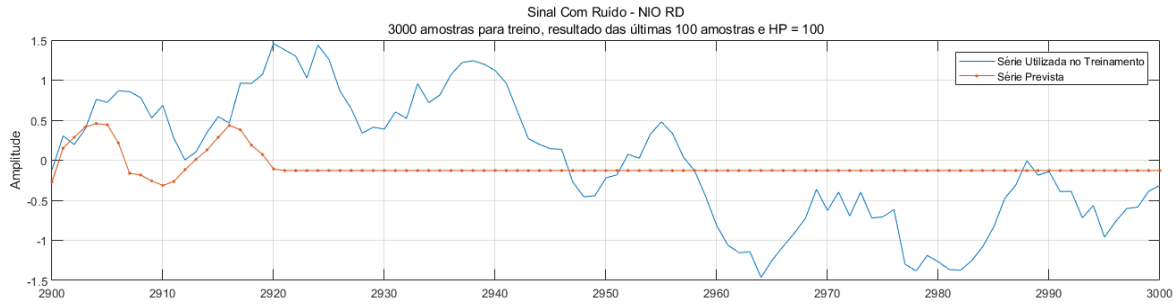
uma linha praticamente contínua e divergente do comportamento da série. Apesar disso, observa-se, na Figura 32, que a série obteve um bom mapeamento e aprendeu consideravelmente bem com os termos de longo atraso, apesar de ter aprendido pouco com os termos de pouco atraso. Essa característica também pode ser notada no melhor resultado para o sinal subamostrado.

O problema é que a rede NIO na configuração *Closed Loop* faz a realimentação com os dados de saída, que não são naturalmente usados no seu treinamento devido a sua característica construtiva. Uma tentativa de predição seria usando a configuração *Remove Delay* (como utilizada no exemplo), entretanto, para grandes horizontes de predição, a rede perde sua capacidade de correlacionar temporalmente o sinal devido à distância entre o horizonte e as últimas amostras reais da série.

Figura 31 – Comportamento das previsões com NIO



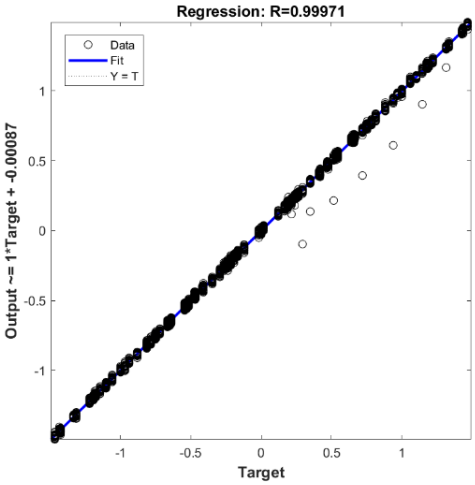
(a)

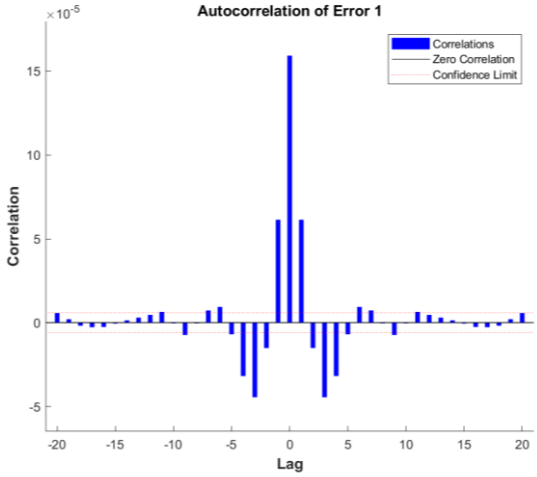


(b)

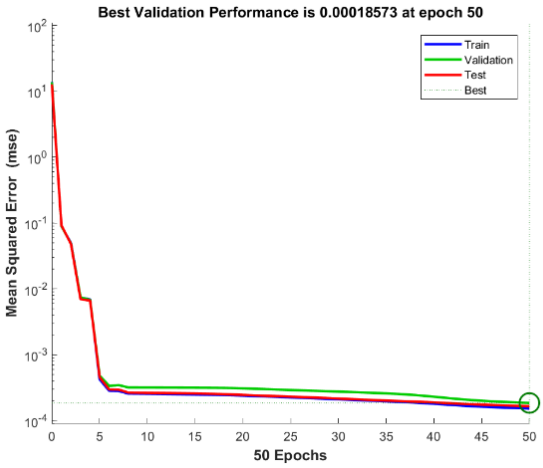
Fonte: Autoria própria

Figura 32 – Gráficos auxiliares da Figura 30





a) (b)



(c)

Fonte: Autoria própria

7. CONCLUSÃO E TRABALHOS FUTUROS.

A partir da interface desenvolvida e dos conceitos abordados, espera-se que o usuário seja capaz de desenvolver as próprias análises em relação aos seus modelos matemáticos específicos, pois sabe-se que não foram abordadas todas as situações possíveis, visto que seriam necessários uma grande quantidade de testes e diferentes equacionamentos matemáticos.

Em relação ao sinal original, notou-se que o algoritmo desenvolvido de otimização tendeu o resultado a situações em que o mapeamento entre a entrada e a saída seja o melhor possível. Entretanto, notou-se que para o sinal bem comportado isso significou, na maioria das vezes, um *overfitting*, devido ao único parâmetro de escolha de melhor resultado ser o menor MSE. Essa característica poderia ser amenizada se fossem utilizados em conjunto várias métricas que favorecessem a escolha do melhor resultado do ponto de vista de aprendizado e não puramente do erro.

Apesar dessa situação observada para o sinal original, essa característica de buscar apenas o menor erro apresentou um resultado significativo quando observado o resultado do sinal com ruído, uma vez que as variações geradas pelo ruído não conseguem ser facilmente mapeadas pela rede, devido sua natureza aleatória. Por isso, o algoritmo apresentou um bom resultado.

Quando tratada a situação com a presença da subamostragem, foi observado que a rede entende apenas a série como a sua componente subamostrada, não conseguindo prever a característica original da série, o que era de se esperar de acordo com o teorema de Nyquist-Shanon. Isso corrobora a necessidade de um sistema que amostre o sinal de maneira em que seja possível realizar a recomposição e predição do sinal original, amostras muito espaçadas apresentam uma dificuldade para a rede em recompor as situações do problema. No caso do desconhecimento da frequência correta de amostragem e do sinal original, pode-se equivocadamente prever a característica de acordo com a subamostragem, não refletindo a situação real do problema alvo da predição.

Consideram-se como possíveis trabalhos futuros que poderiam aprimorar a análise e a interface desenvolvida: A implementação da opção, na interface, do usuário escolher o caminho de salvamento do experimento, visto que o padrão do algoritmo desenvolvido é salvar na pasta

onde encontram-se os códigos do algoritmo e isso pode gerar desordem quando vários experimentos são executados em sequência.

Outra implementação em relação a interface é a adição da opção de inserir uma série de dados em vez do formato atual de digitar uma função matemática, para isso seria necessário adaptar o código da *guide* para ler uma variável específica no *workspace* informado pelo usuário ou abrir um conjunto de dados salvos em uma pasta específica.

Para melhorar a primeira fase de ajuste dos hiperparâmetros, pode-se implementar uma metodologia de *tunning* que busque identificar bons *ranges* de tratamento para o caso escolhido de equacionamento matemático ou variável.

Para tornar a análise mais ampla e incrementar elementos de análise e comparações entre diferentes metodologias, pode-se trabalhar a implementação de mais arquiteturas de rede, incorporando temas como o *Deep Learning.*

8. REFERÊNCIAS BIBLIOGRÁFICAS

ADYA, M.; COLLOPY, F. How effective are neural networks at forecasting and prediction? **International Journal of Forecasting**, pp. 481-495, nov. 1998.

BROCKWELL, P. J.; DAVIS, R. A. **Time Series: Theory and Methods.** 2ed. New York: Springer, 2009.

CRYER, J. D.; CHAN, K.S. **Time Series Analysis: With Applications in R. Springer.** 2ed. New York, Springer, 2010.

GUTIERREZ, G.; SESMERO, M.P.; SANCHIS, A. **Forecasting Time Series by an Ensemble of Artificial Neural Networks based on transforming the Time Series.** 2016 IEEE International Conference on Systems, Man, and Cybernetics, 2016.

HAYKIN, S. O; VEEN, B. V. **Sinais e Sistemas**. 2ed. Porto Alegre: Bookman, 2001.

HAYKIN, S. O. **Neural Networks and Learning Machines**. 3ed. Ontario: Prentice Hall, 2008.

LAHMIRI S. **A comparative study of backpropagation algorithms in financial prediction**, International Journal of Computer Science, Engineering and Applications, 2011.

LJUNG L.; GLAD T. **On Global Identifiability for Arbitrary Model Parameterizations**. Automatica, 30, 265-276, 1994.

MOREIRA, C. **Neurónio**. Revista de Ciência Elementar, 1(01):0006, 2013.

MONTILLET, J.P.; TREGONING, P.; MCCLUSKY, S.; YU, KEGEN. Extracting White Noise Statistics in GPS Coordinate Time Series. **IEEE Geoscience and remote sensing letters**, V.10, p.563-567, maio 2013.

SANG, Y.F.; WANG, D.; WU, J.C. An Improved Wavelet De-noising Method for Time Series Analysis. **Sixth International Conference on Fuzzy Systems and Knowledge Discovery,** p. 517-521, 2009.

SHARMA S. **Epoch vs Batch Size vs Iterations**, Setembro 2017. Disponível em:<https://towardsdatascience.com/epoch-vs-iterations-vs-batch-size-4dfb9c7ce9c9>. Acesso em: 15 de Outubro de 2018.

YAN, W. **Toward Automatic Time-Series Forecasting Using Neural Networks.** IEEE transactions on neural networks and learning systems, V.23 p. 1028-1039, julho 2012.

WANG Z.; BOVIK A. C. **Mean Squared**, IEE, 2009, pp. 98-117, Janeiro 2009.

WERBOS, P. J. **Backpropagation Through Time: What It Does and How to Do It**. IEEE, pp. 1550-1560, Outubro 1990.

WEIGEND A. S.; GERSHENFELD N. A. **Time Series Prediction: Forecasting the Future and Understanding the Past**. 1ed. New Mexico: Westview Press, 1993.

WILLIAM R. J.; PENG J., **An Efficient Gradient-Based Algorithm for On-Line Training of Recurrent Network Trajectories**. Neural Computation, 2, 490-501, 1990.